Automatic Route Setting and Dynamic Rescheduling Following Disturbance

Mouhiyadin Said Ahmed

A Thesis Proposal Submitted to Addis Ababa University

Addis Ababa Institute of Technology

Presented in Fulfillment of the Requirements for the Degree of Master of Science (Electrical and Computer Engineering)

Addis Ababa University

Addis Ababa Institute of Technology

September 2018

Addis Ababa University

Addis Ababa institute of technology

School of Electrical and Computer Engineering

This is to certify that the thesis prepare by Mouhiyadin Said Ahmed, entitled: *Automatic Route Setting and Dynamic Rescheduling Following Disturbance* and submitted in the partial fulfillment of the Requirements for the Degree of Master of Science (Electrical and Computer Engineering) complies with the regulation of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Examiner _____ Signature _____ Date _____

Examiner _____ Signature _____ Date _____

Advisor _____ Signature _____ Date _____

Advisor _____ Signature _____ Date _____

Abstract

Nowadays, development cannot be achieved without modern infrastructure capacity. The new railway project which links Addis Ababa to Djibouti aims to fulfill this task through safety and reliability as major goals. Railway operating near their theoretical capacity is particularly vulnerable to disruption by human behavior or engineering failures that lead to delays or to even cancellations of services. In order to prevent such incidents, the interlocking system is set to automatic or semi-automatic mode and rescheduling is planned by the system to set a new route for the remaining trains.

This study aimed to analyze how the system reacts and respond to disruptive incidents in minimizing the impact of such disturbance. After reviewing relevant literatures regarding most occurring disruptive incidents in train operation and their consequences on the railway network reliability, we defined train rescheduling operation and more precisely the automatic route selection with their characteristics parameters after a disruption such as an accident on a level crossing or faults in electronics device. We introduce C++ programming language to model route setting using spanning tree configuration. A case study on rescheduling operation on a selected British railway area is carried on to evaluate delays cost after disruption. This data was processed to model how delays relevant to a specific disruption affect trains in a railway network using a neural network with a high performance.

Key words: *Automatic route setting, train scheduling, operation disruption, neural network, spanning tree problem,*

Acknowledgment

First, I would like to express my deepest appreciation and sincere gratitude to my advisor Mr. Abebe Teklu for his guidance, support and invaluable contributions throughout the preparations for this thesis.

I would like to thank all the staffs of the Department of Electrical and Computer Engineering and Railway Engineering Center at AAiT and respective officials at Ethiopian Railway Corporation, for providing me all the invaluable materials and helpful pieces of advice.

Finally, I wish to thank all of my friends and family for their persistent support during my thesis work.

Abbreviation

a	Acceleration of train
ANN	Artificial neural network
ARS	Automatic route setting
ATP	Automatic train protection
CDR	Conflict detection and resolution
i/o	input/output
KPI	Key performance indicator
MIP	Mixed integer programming
MLP	Multi-layer perceptron
MSE	Mean square error
T	Time
TD	Total delay
TDP	Total delay penalty
TTP	Train timetabling problem
Vmax	Maximal speed of train

List of Figures

List of Tables

Contents

Chapter One

1. Introduction

Railway transportation system has taken major position in goods and passenger transportations around the world. Technology's progress enhanced safety and reliability with better use of computers and electronics' devices for control purpose. Railway operation before control technology advancements lead to accidents. Most developed countries started using controlled and automatic system to manage train traffic. Railway network became more fluent after optimal control of trains' parameters such as speed, route selection, breaking and others while remaining safe and free from critical errors. However, disruption caused by human behavior or the use of electronics' devices near critical level, can disturb or stop the traffic and by consequence lower the optimal use of network [1, 2].

Railway operating near their theoretical capacity can lead to disruptions in railway operation and can occur for different reasons such as an accident on a level crossing or train technical failures. Managing and scheduling activities keeping in view of competing activities and constraints have always been a challenging. Rescheduling has to be perfectly timed and free from critical error or upon critical errors it must go to the fail-safe state[3,4].Once a delayed train deviates from its original time-distance path, it may affect subsequent trains that are scheduled over the same railway infrastructure or it may create a conflict in passing's or meetings with other trains. Hence, a delayed train may propagate its delay to other trains due to the infrastructure, signaling or timing conflicts. The tactical planning process of scheduling already starts a year before a new annual timetable becomes operational [5].

Operational traffic management is currently mainly directed towards maintaining the tactical day plan and recovering from disruptive events as quickly as possible back to the original timetable. Traffic control is responsible for the train traffic operations in a wider control area. In agreement with the concerned transport operator, a traffic controller may reschedule delayed trains from the next station onwards. Network traffic managers are responsible for general supervision of their part of the network and communications with neighboring control centers. Dispatchers supervise

all train traffic in their control are based on a real-time monitoring of track occupancy and clearance, as well as trains' number record [6 - 8].

The dispatcher may also set routes manually when e.g. a last minute disruption occurs. If an approaching train is delayed and the dispatcher expects a route conflict then she/he may decide to adjust the local route plan and change the route settings orders. In case of large delays, the dispatcher apply predetermined if-then measures in order to determine an alternative path until the next station and keep the (network) traffic control center informed. In practice, major disturbances may influence the off-line plan of operations, thus causing primary (initial) delays, such as train delays or temporary unavailability of some routes, that propagate as secondary (consecutive or knock-on) delays to other trains in the network. In such situations, the timetable no longer provides an optimal use of the infrastructure capacity, and short-term adjustments are worthwhile in order to minimize the negative effects of the disturbances. This process is called real time traffic management, which consists of modifying the timetable such that it becomes compatible with the real traffic situation. Possible traffic control actions include changing dwell time at scheduled stops, changing train speeds along lines, or adjusting train orders at junctions, stations and passing points. Other control actions involve major modifications such as changing train's routes or even cancelling scheduled train journeys [9].

Automatic Route Setting (ARS) is a new control technology which sets automatically the trains routes in a railway network. Its dynamic response makes it perfect in case of incidents and disruptions as trains' new route is rescheduled and set to operate even if one or more incidents occur in the network [10, 11]. Beyond this, it is customary to reschedule the train operation for optimum utilization of the railway system. Therefore, study of automatic route setting and dynamic rescheduling following a certain disturbance is vital and necessary.

1.1 Architecture of real-time railway rescheduling process

In the event of conflict, rescheduling operate by making decisions in order to return the initial running schedule. This has to be achieved on the basis of events that have already happened or are anticipated. The rescheduling operations should produce a fresh, non-conflicting running schedule that takes account of actual circumstances. In general, rescheduling needs to perform the following functions [12-14]:

- Start with a conflict free time table
- Receive information regarding operations
- Detections of conflicts
- Automatic resolutions of conflicts
- Generation of a new conflict free time table
- Data registered for traffic purpose

To solve a conflict, the dispatcher needs to reschedule some trains. The possible modifications are:

- Use of alternative routes
- Extension of a scheduled stop
- Relocation of passing stops
- Additional stops for operational requirements
- Extension of running time
- Cancellation of train over its complete route

Therefore, for effective rescheduling, the goals to be achieved are:

- Minimization of the overall delay of trains
- Minimization of the weighted delay (based on priority for each train)
- Returning as fast and as close to the original timetable

The dispatcher's decisions have to be communicated to the operation center. The dispatcher or the rescheduling system resolves the conflict which has been detected in the dispatching time horizon, such as the 30 next minutes. If the conflict resolution method called "extension of running time" is used, the train drivers have to be informed. This method is used to slow down a second faster train in order to solve the conflicts in the next station where the first slower train will use a siding track. The second train receives information about the optimum speed, to avoid the influence of a distant signal (train movement authority protection).[15-17].

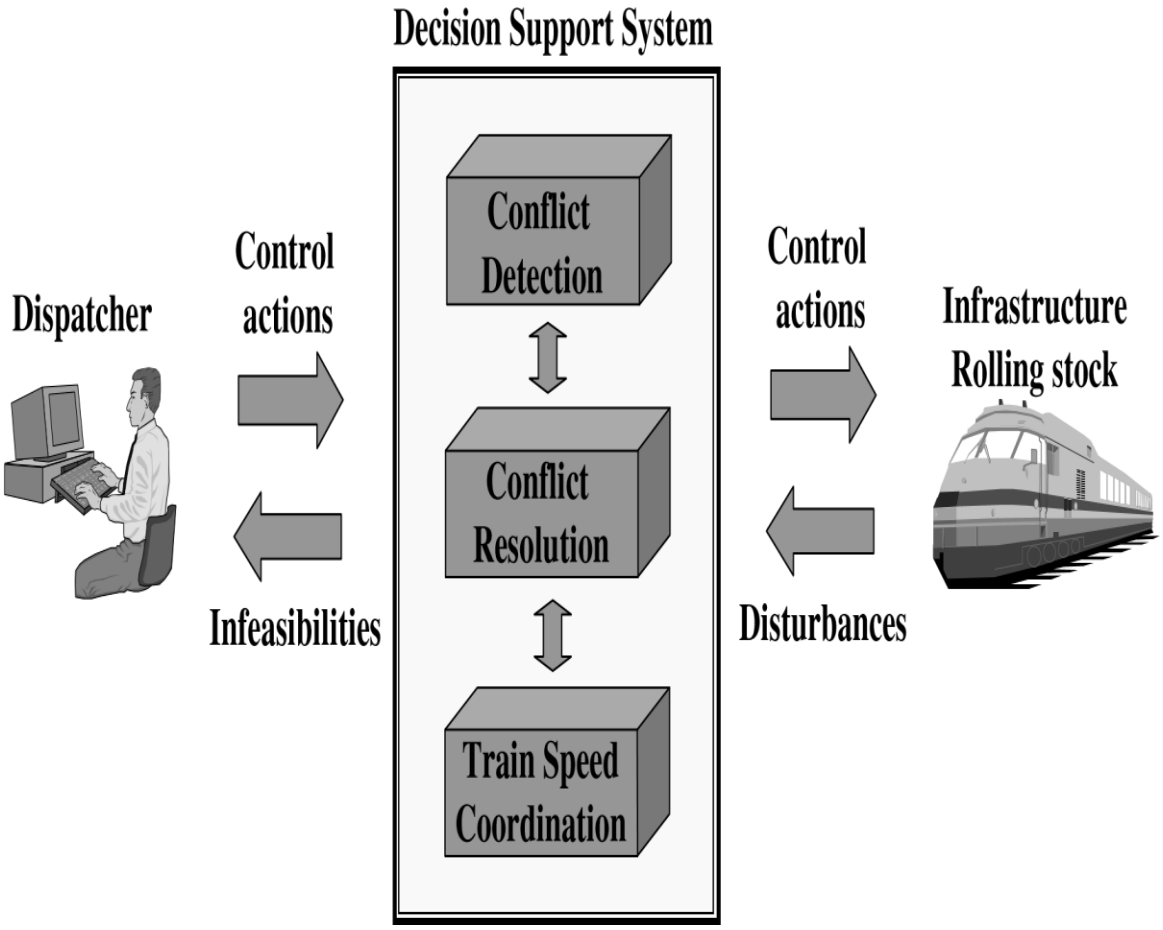


Figure 1 Decision Support System

1.2 Problem Statement

Railway operating near their theoretical capacity can lead to disruptions in railway operation and can occur for different reasons such as an accident on a level crossing or train technical failures. Managing and scheduling activities keeping in view of competing activities and constraints have always been challenging. Rescheduling has to be perfectly timed and free from critical error or upon critical errors it must go to the fail-safe state.

Some disturbances occurring on a simple network (a single track or double track) may lead to disruption of railways operations and re-routing may not be applicable as there is no alternative route to be used. Some other disturbances may be prevented by maintenance. Others will take time to be resolved and rescheduling is possible.

After disruptions, time tabling has to be calculated to plan the delay time and new possible routes have to be set for optimal operations. It's essential to know the factors of time tabling to reschedule safely and lower the delay time. As a new route has to be released, the criteria of the automatic route settings have to be known for optimization purpose. Hence, a delayed train may propagate its delay to other trains due to infrastructure, signaling or timing conflicts. Therefore, study on automatic routing and dynamic rescheduling during disturbance is essential so as to achieve safe and reliable railway transportation system. In this regard, this paper focused on the following issues:

- How the automatic route setting respond to disruptions
- How the systems manage delays and rescheduling
- Case study on rescheduling Operations
- Case study in MATLAB R2009b using a neural network for delay prediction

1.3 Thesis Objectives

1.3.1 General Objective

The general objective of this thesis is to study and analyze the need for automatic route setting and dynamic rescheduling for effective response in case of disturbances in railways operations.

1.3.2 Specific Objective

The specific objectives of this thesis are:

- To conduct a literature survey about automatic routing and dynamic rescheduling
- To develop automatic routing algorithm
- To develop train rescheduling algorithm
- To analyze the impact of delay on the railway system

1.3.3 Scope of the Thesis

Railway network optimization with respect to performance of individual links and nodes is reasonably well understood. Less well developed is the knowledge of the performance of larger dispatching areas and their optimization after the occurrence of unexpected events, such as train

breakdowns, track damages, power failures and bad weather. Therefore, the scope of this thesis is on:

- Understand the criteria for reliable rescheduling operations
- Investigate about automatic route setting and timetabling algorithms
- Railway simulation on a study case for railway rescheduling operations
- Design of a neural network in MATLAB for data processing on a case study on British Rail company

1.3.4 The Research Methodology

The methodology used to achieve the stated objective; we have followed the following research diagram where we start from data collection and literature review. In this section relevant information about automatic route setting and dynamic rescheduling has been intensively investigated and necessary dates were also collected. After having a clear understanding of the two concepts we have developed our system model. The system model includes modeling for route setting, modeling for rescheduling and then delay analysis.

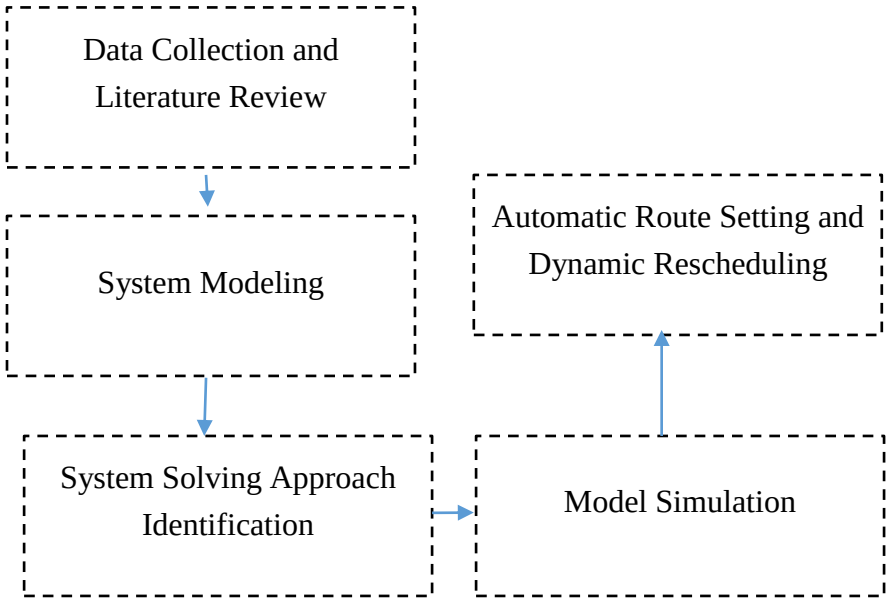


Figure 2 Methodology table

1.3.5 Thesis Limitations

. The rescheduling operation is a large task in railway operation in which routing, scheduling, disruption detection and may other variables have to be taken into consideration. Since the operations of our National railway haven't begun yet, it's hard to collect real data regarding delays and affected train. Hence, we used data provided by NetworkRail. (2010) to simulate and analyze rescheduling Operation using a conflict free timetable provided in Appendix

1.3.6 Thesis Organization

This thesis is organized in five chapters. The first chapter deals about the introduction where a brief introduction of the thesis work is presented and problem statement along with thesis objectives and methodology used is clearly shown. In the second chapter, we first introduce about the literature review on disruptions occurring in most railway operations, then the routing problem in rescheduling operation and finally finish with the time tabling problem which is essential to understand deeply delays.

In chapter 3, we provide the mathematical modeling of rescheduling operations specially the spanning tree problem used for routing and the neural network necessary for our delays analysis.

In chapter 4, a rescheduling operation is simulated using data collected from the British rail company and a simulator. Data collected from the simulation are analyzed using Matlab R2009 rescheduling operation after scenarios of disruptions. On the second hand, we create a neural network on MATLAB capable of learning how delays affect trains. The outputs regarding the learning rate of the neural algorithm also discussed in this chapter.

Chapter 5 settled the conclusions of our work and the recommendation for future works.

Chapter Two

2. Literature Review

2.1 Disruptions in Railway Operations

There are four basic reasons to engage rescheduling operations:[18-20]

- Deviation- the most common type of deviation is a time deviation, specifically exceeding a pre-defined tolerance bandwidth in a production plan (e.g. a train is late or early). Other types of deviation include a train using a different route than planned or operating a different combination of trains. Deviations can be the result of an incident, a disturbance or operational failure
- Disturbance- A disturbance means that due to reduced availability or productivity of a technical component, or of an actor participating in the production, production cannot be continued as planned. After the disturbance is eliminated (and the system regains productivity), the production plan can be adapted
- Incident- An incident interrupts or delays production on a short term basis. After an incident, all resources are fully available again and production can be continued as planned. Incidents often lead to schedule deviations. Incorrect inputs or either human errors are also classified as incidents.
- Service change- A service change consists of adding or changing trains in the existing schedule (e.g. changes in a service by adding supplementary stops or adding a new freight train). These types of changes may also impact other lines and services and therefore a new schedule is needed.

Therefore a disturbance can be distinguished from an incident or deviation by the fact that after a disturbance, new plan conditions (e.g. new vehicle characteristics or new infrastructure characteristics) must be defined, while in the case of an incident or deviation, in most cases only a change of the time conditions for the next reference points is required

2.2 Routing Problem in Rescheduling Operations

The passage of a train through a particular block section is called an operation. A Route of a train is a sequence of operations to be processed in a track yard during a service (Train run). At any time a route is passable if all its block sections are available and the corresponding block signal is green or yellow, i.e., there are no blocked tracks ahead. The timing of a route specifies the starting time of each operation in the route. Each operation requires a travelling time, called the running time, which depends on the actual speed profile followed by the train while traversing the block section. A speed profile is furthermore constrained by [21,22]:

- The rolling stock characteristics (maximum speed, acceleration and braking rates)
- -physical infrastructure characteristics (maximum allowed speed and signaling system)
- -driver behavior (Coasting, braking and acceleration profiles when approaching variable signals aspects

The running time includes the time needed to accelerate (or decelerate) a train due to a scheduled stop, as well as speed variations between two consecutive speed signs. Furthermore, the running time is known in advance since all trains travel at their scheduled speed, which usually contains some margins for recovery. In yards, the routes for the individual trains need to be setup before entering and cleared after leaving (route release procedure), which takes a certain switching time.

We consider two types of optimization models addressing the routing of trains. The line planning problem is to decide the routes for passenger's trains as well as the types and frequencies of use of each train routes [23]. On the other hand, network routing models address the different problems related to freight train routing. In both cases, train scheduling models are adopted to determine the arrival and departure times of the train at all the relevant passing and/or stopping locations.

If we consider the problem of managing real time-time table perturbations in a regional dispatching area. A mixed integer programing model is adopted to maximize the number of passengers transported and backtracking heuristic techniques are developed to solve train conflicts. A conflict resolution system using their algorithm has been implemented to support dispatchers at a traffic control center of the Spanish National railway company [24].

Method	Advantages	Disadvantages
Infrastructure-based train location	-No additional technical equipment or infrastructure extensions are needed -Very accurate position	-No information in a case of delay until a location point is passed. -Signification loss of time until deviation is detected
Periodic Train location	- Periodic Information about train state and position is available. - Additional information is easily transmittable	- Investments for train locations (e.g. GPS) and communication are needed. - Detection is not possible everywhere
Participant Train location	- Detailed information about delay reason is transmittable - Is possible in addition to other detection strategy	-Immense time lag until delay is transmitted to rescheduling system

Table 1 Comparison between Railway rescheduling approaches

2.3 Timetable Problem in Rescheduling Operations

Network dispatchers regulate the railway traffic by sequencing the train movements and setting the routes with the aim of ensuring smooth train movements and limiting as much as possible

existing delays. Due to the strict time limit available for computing a new timetable, which so far is rather infeasible by using existing tools, operators usually restrict themselves mostly to a few manual modifications of the timetable. Experienced dispatchers have developed strategies allowing them simply to foresee possible disruptions well in advance and to take compensatory control actions based on local information. In general, minor changes are preferred instead of extensive rescheduling in order to alter as little as possible the original timetable.

Many knock-on delays, however, could be prevented if the traffic was pro-actively managed [25], i.e , dispatchers can spend their time on preventing traffic disturbances instead of only solving them when they have already happened. Based on accurate monitoring of the actual train positions and speeds, the potential conflicting routes can be predicted in advance and might be resolved in real time. The adjusted targets (location-time-speed) would be communicated to the relevant trains by which the informed drivers could be able to anticipate better on the changed traffic conditions. Such a traffic management system would be effective in coordinating the speed of successive trains on open tracks, securing time windows at junctions/ crossings, or synchronizing the arrival of the trains at stations in case of delays and expected route conflicts.

The problem is solved using an iterative two-level process. The upper level handles the order of meets and overtakes of trains on the tracks sections, using simulated annealing and tabu search, while the lower level determines the start and end times for each train and the sections it will occupy. Experiments on passenger trains running on a Swedish railway are presented to compare the proposed approaches with the optimal solutions [25].

Authors propose an algorithm for train dispatching with a train rescheduling pattern language processing system. In case of severe train traffic disruptions, caused by accidents that may require the suspension of some train line, the algorithm is helpful for the preparation of practical rescheduling plans. Applying actual train schedule data, the authors claim that their approach works satisfactorily. However, they apply local modifications of the original schedule that result in sub-optimal rescheduling decisions [26].

A delay may occur when a train reaches the end of a block section and the subsequent block section is still occupied by another train. The running time of a train on a block section starts when its head (the first axle) enters the block section. Safety regulations impose a minimum distance separation among the trains, which translates into a minimum setup up time (time

headway)between the exit of a train from a block section and the entrance of the subsequent train into the same block section. This time takes into account the time between the entrance of the train head in a block section and the exit of its tail (the last axle) from the previous one, plus additional time margins to release the occupied route and sighting distance [27].

Previous works differ significantly regarding the system topology. Some papers are based on railway lines, while others on general on networks studies. A railway line is a set of consecutive stations and segments limited by two extreme stations. On the other hand, a network is a set of railway lines, where one station can be connected directly with two or more stations. Another important aspect is the type of tracks and platforms: directional/ unidirectional and single/ parallel tracks. These aspects constitute a very important factor of complexity, as a consequence, in most cases, the representation is simplified. It should be noted that, in general, the most complex models cannot deal efficiently with more than 30 trains.

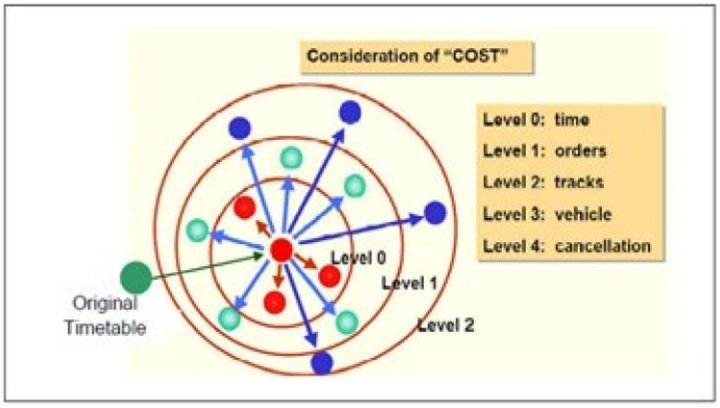


Figure 3 Cost of rescheduling different action level

The robustness of the railway system indicates that the dynamic variations found in railway operation leads to an asymptotically stable behavior of the railway system and the initial schedule plan/ service pattern is restored as shown in figure 3

Table 2 shows the differences of four definitions. For each definition, the delays it can handle, the results after the recovery, and recovery strategies are represented and compared.

Definitions	**Delay situations**	**Results**	**Recovery strategies**
Stability	Perturbations	The original schedule/ service pattern.	No recovery actions.
Robustness	Minor disruptions	The original schedule/ service pattern.	Traffic management strategies.
Recoverable robustness	Disruptions	A new schedule/service pattern with the same volume of traffic as the original solution.	Traffic management strategies.

Table 2 Recovery Strategies

Chapter Three

3. Rescheduling Operations Mathematical Modeling

3.1 New Routes Generating Algorithms

Sophisticated decision support tools based on mathematical programming techniques are needed to help dispatchers to control the railway traffic under sever disturbances, ie., large entrance delays, blockage of some tracks and failures of rolling stock. To this end, centralized decision making focuses on the point-to-point routing and scheduling of trains in a railway network by re-optimizing the use of infrastructure capacity and minimizing the propagation of train delays. A point-to-point train refers to freight or passenger trains where in the origin (A) and destination (B) stations are fixed a priori but local routes and meet-pass decisions between A and B may be ascertained dynamically. This corresponds to a compound rescheduling and local rerouting problem which is very difficult to solve in real time for large networks and only some heuristics have been proposed. We thus address the conflict detection and resolution (CDR) problem, which is practiced every day by traffic controllers to adapt the timetable to delays and other unpredictable events occurring during operations.

A local search algorithm

In general, a local search technique starts from an incumbent solution (IncSol) and then iteratively moves to a neighbor solution. This is typically an incomplete algorithm, as the search may stop even if the best solution found (BestSol) by the algorithm is not optimal. This can happen even if termination is due to the impossibility of improving the current local best solution (LocalBestSol), as the optimal solution (OptSol) can lie far from the neighborhood of the solutions crossed by the algorithm.

The neighborhood of a solution G(F,S) contains all the solutions $G^0(F^0,S^0)$ in which F^0 is obtained from F by changing a single route on the ramified critical path. If the route is contained in the critical path, this can be replaced only with a shorter route, as stated by

Given a rerouting option ω, let u and v be its first and last nodes, and l(ω) be the length of route ω in G(F,S). We estimate the potential of a new local rerouting option ω as the quantity $\Pi(\omega) = l^{F,S(F)}(0,v) - l^{F,S(F)}(0,u) - l(\omega)$, since the length $l^{F,S(F)}(0,v)$ might decrease to $l^{F,S(F)}(0,u)+l(\omega)$ when changing route. We restrict the neighborhood to the ψ routes with highest potential, where ψ is a parameter of our local search procedure. If none of the ψ routes improves the solution, we evaluate the next ψ routes with highest potential. The procedure continues as long as an improvement is possible or no rerouting option is available or the time limit is reached.

For each of the ψ best routes we define a neighbor F^0 by replacing an old route of F from node u to node v with a new route, including all the fixed arcs required to implement the new route and its railway constraints, and then execute the train rescheduling procedure. Among the ψ new solutions $G^0(F^0,S^0)$, we choose the one having the shortest critical path, i.e., the one minimizing the maximum consecutive delay. In case of tie, we choose the solution with minimum average consecutive delay

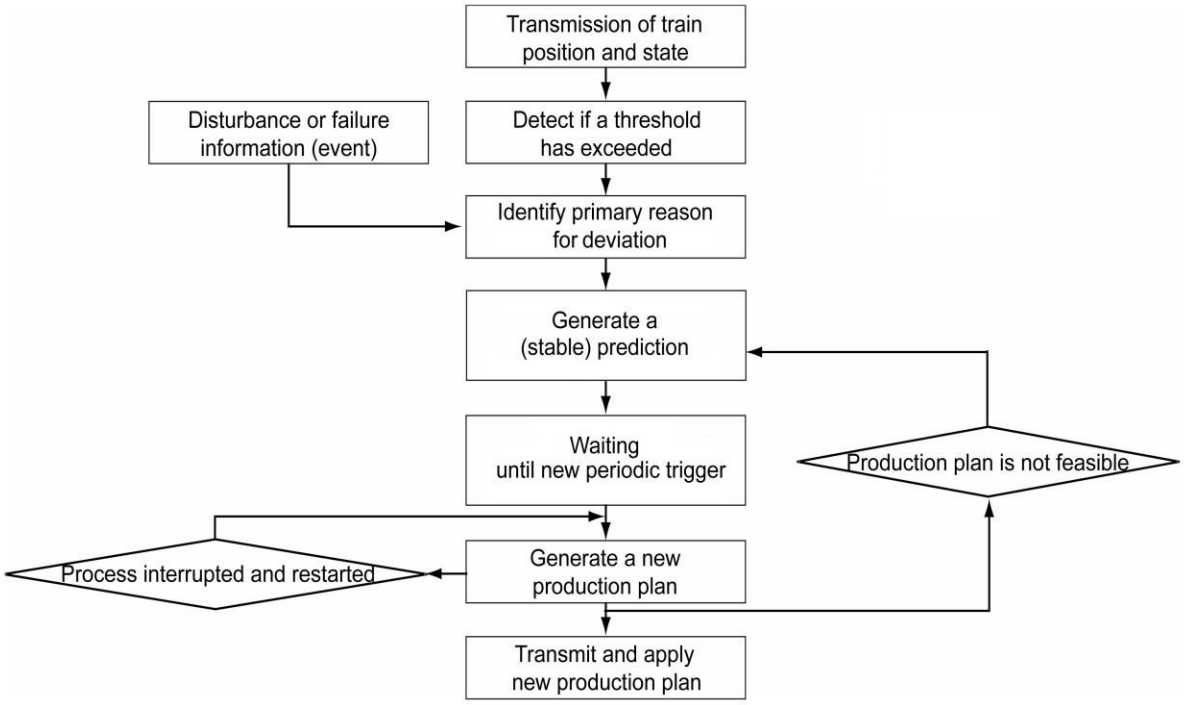


Figure 4 Rescheduling plan used in simulation

Pseudo code for routing with Prim algorithm in c++

```
struct node
{
   int fr, to, cost;
}p[6];
int c = 0, temp1 = 0, temp = 0;
void prims(int *a, int b[][7], int i, int j)
{
   a[i] = 1;
   while (c < 6)
   {
      int min = 999;
      for (int i = 0; i < 7; i++)
      {
         if (a[i] == 1)
         {
            for (int j = 0; j < 7; )
            {
               if (b[i][j] >= min || b[i][j] == 0)
               {
                  j++;
               }
               else if (b[i][j] < min)
               {
                  min = b[i][j];
                  temp = i;
                  temp1 = j;
               } }
         }
      }
      a[temp1] = 1;
      p[c].fr = temp;
      p[c].to = temp1;
      p[c].cost = min;
      c++;
      b[temp][temp1] = b[temp1][temp]=1000;
   }
   for (int k = 0; k < 6; k++)
{
      cout<<"source node:"<<p[k].fr<<endl;
cout<<"destination node:"<<p[k].to<<endl;
```

```
        cout<<"weight of node"<<p[k].cost<<endl;
    }
}
int main()
{
    int a[7];
    for (int i = 0; i < 7; i++)
    {
        a[i] = 0;
    }
    int b[7][7];
    for (int i = 0; i < 7; i++)
    {
        cout<<"enter values for "<<(i+1)<<" row"<<endl;
        for (int j = 0; j < 7; j++)
        {
            cin>>b[i][j];
        }
    }
    prims(a, b, 0, 0);
getch();
```

Approximation used in our algorithm

A graph (directed or undirected) is an ordered pair of sets G=(V,E) . V is a non-empty and finite set of elements called vertices. E is a set of edges or lines, which are 2-element subsets of V (an edge is related with two vertices, and the relation is represented as an unordered/ordered pair of the vertices with respect to the particular edge). [29]

The elements of E are unordered in the case of undirected graphs and the unordered pair of vertices x and yare written as [x, y]. In the case of directed graphs, the pairs in the E set are ordered. The pair of vertices x and y, written (x, y); x is called initial extremity of edge (x, y) and y is called final extremity of edge (x, y).

If an edge with extremities x and y exists, then vertices x and y are adjacent; each extremity of an edge is considered incident with the respective edge. We will consider that the extremities of every edge are distinct (the graph does not contain any loops). Between any given two vertices of the graph there can be at most only one edge.

The information associated with a graph can be as complex as required, but, in order to simplify the problem, we will consider that the vertices are labeled as numbers from 1 to n (where n is the number of vertices in the graph). This labeling is not a restriction (in the following chapters, the vertex number will represent the position, within an array, of the information associated to the vertex).

A route (within a directed graph) is a sequence of vertices $(x_1,x_2,...,x_n)$ in which for any pair of consecutive vertices x_i and x_{i+1} there is an edge (x_i,x_{i+1}). The route is elementary if any vertex is encountered only once in the route. A route is simple if any edge is encountered only once in the route.

The first step required is the identification of track elements involved in setting the route, assigning a unique identifier for each of the elements (the label for the corresponding graph vertex) and the definition of the distance matrix.

In the case that a railway traffic signal or a point machine does not validate the availability requirements, it cannot be used in the route searching procedure. In this case, the matrix elements (2) related to that particular traffic signal or point machine are cancelled (e.g. $a_{r,j}=0$ and $a_{i,r}=0$ for track element r, where i,j=1,…,n).

Similarly, if one of the track sections is not available, the edge corresponding to its start and end vertices is cancelled (e.g. if the start and end vertices are p and q, $a_{p,q}=0$).

The availability matrix allows the generation of the route list based only on the availability of the track elements, disregarding the route distance.

If we wish to generate a route list based on the route distance, as well as the track element availability, based on rules (1) and (2) we can define another square matrix $T=(t_{i,j})$, where $t_{i,j}=m_{i,j}*a_{i,j}$, i,j=1,…,n (3).

3.2 New Schedules Generating Algorithms

We first present four dynamic implication rules. The first two rules are specific for the alternative graph model [30] while the two latter rules were designed [31] for the disjunctive graph but apply as well to the alternative graph model. In what follows, tails q_i, heads r_i, processing times p_i and the values $l^S(0,i)$ and $l^S(i,n)$ are computed for all nodes $i \in N(F)$. [30]

Proposition 5.1.1: Given a selection S, if ((i,j),(h,k)) is an unselected pair of alternative arcs and $l^S(k,h) + a_{hk} > 0$, then arc (h,k) is forbidden and arc (i,j) is implied by S.

Proposition 5.1.2: Given a selection S, if ((i,j),(h,k)) is an unselected pair of alternative arcs and $l^S(i,j) \geq a_{ij}$, then arc (i,j) is called redundant and this is implied by S.

Proposition 5.1.3: Given a selection S, if ((i,j),(h,k)) is an unselected pair of alternative arcs and $r_h + a_{hk} + q_k \geq UB$, then arc (h,k) is forbidden and arc (i,j) is implied by S.

Proposition 5.1.4: If J is an ascendant set of o_c, all arcs $(\sigma(c),j)$, $\forall o_j \in J$, are forbidden. If J is a descendant set of o_c, all arcs $(\sigma(j),c)$, $\forall o_j \in J$, are forbidden.

The following propositions are introduced to establish a link between the selection of arcs from different alternative pairs.[32]

Proposition 5.1.5: Consider a selection S and two unselected alternative pairs ((a,b),(c,d)) and ((i,j),(h,k)). If $a_{ab}+l^S(b,i)+a_{ij}+l^S(j,a) \geq 0$, then arc (h,k) is implied by selection $S \cup \{(a,b)\}$ and arc (c,d) is implied by selection $S \cup \{(i,j)\}$.
Proposition 5.1.5 can be applied particularly with the empty selection $S^{\emptyset}$, when the graph $G(\emptyset)$ is composed only of fixed arcs associated with the train routes. During the execution of the solution procedure, the selection of an arc causes the selection of the entire set of arcs implied by this arc. The next proposition expresses rule 5.1.5 in terms of the railway physical configuration.

Proposition 5.1.6: Consider two alternative pairs ((a,b),(c,d)) and ((i,j),(h,k)). Then, $a_{ab} + l^{\emptyset}(b,i) + a_{ij} + l^{\emptyset}(j,a) \geq 0$ if the following conditions hold.

1. Nodes b and i are associated with a train T_A and are connected by a directed path of fixed arcs, i.e., T_A executes o_b before o_i.

2. Nodes j and a are associated with a train T_B and are connected by a directed path of fixed arcs, i.e., T_B executes o_j before o_a.

T_A and T_B pass through the block sections ((a,b),(c,d)) and ((i,j),(h,k)) refer to

Method used to solve the mathematical modeling

Branch and bound search has been known for a long time and has been widely used in solving a variety of problems, such as integer linear programming and combinatorial optimization problems. In general, the goal in such problems is to systematically search a very large space to find a globally optimal solution. Since the search space is intractably large, some implicit method is required in order to rule out regions of the search space that contain no interesting solutions. Branch and bound algorithms work by the divide and conquer principle: the search space (enumeration tree) is subdivided into smaller subregions (this subdivision is referred to as branching), and bounds are found on all the solutions contained in each subregion under consideration. The strength of the branch and bound approach comes when bounds on a large sub-region show that it contains only inferior solutions, and so the entire subregion can be discarded (pruned) without further examination. Ideally the procedure stops when all the search space is explored. At that point, all non-pruned subregions will have their bounds equal to the global minimum of the function (proven optimum). In practice the procedure is often terminated after a given time (near-optimum).

Running time calculation

The trapezoidal speed trajectory becomes an S-curve position trajectory having three distinct segments corresponding to the acceleration, deceleration, and constant speed portions of the trajectory, and since the calculation varies for each of the segments, it is necessary to determine the time (or position) at which one segment ends and the following segment begins. The minimum energy and best case scenario for moving a load through a position trajectory is when all three times T1, T2 and T3 are equal, but in some applications this is not always practical.

The diagram below shows the simple and familiar case of a trapezoidal speed trajectory with symmetrical acceleration and deceleration times and a sustained constant speed motion segment

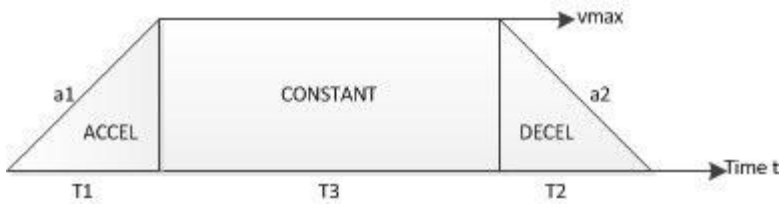


Figure 5 Train speed profile

Approximation of the trapezoidal diagram

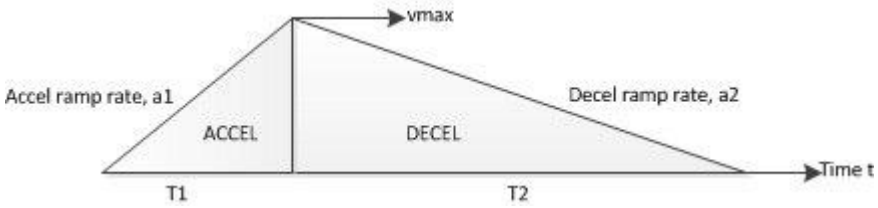


Figure 6 Approximation speed profile of train

In cases when the distance trajectory is short compared to either one or both of the ramp times, the trajectory becomes triangular, and calculating the ramp times T1 and T2 necessary to generate the trajectory becomes a bit more complicated. In the figure above, the acceleration and deceleration rates are different, and the top speed vmax may be less then the vmax speed in the previous trapezoidal trajectory; this presents a more generalized trajectory case, which can be of practical importance when, for example, the motor is driving a high inertial load and requires a slower stop ramp to help dissipate the inertial energy.

The total distance D travelled during a triangular trajectory is the time integral of the acceleration and deceleration speed ramps over their respective durations T1 and T2. Visualize the total distance as the combined areas under the acceleration and deceleration triangles. This relationship leads to equation 1 below, which is simply the area under the two triangles:

$$D = \frac{1}{2} a_1 T_1^2 + \frac{1}{2} a_2 T_2^2 \qquad \text{Eq.1}$$

At the peak speed, the maximum acceleration ramp speed equals the maximum deceleration ramp speed. So equating these peak speeds leads to equation 2 below

$$.a_1 T_1 = a_2 T_2 \qquad \text{Eq. 2}$$

This equation is re-organized to Eq. 3 below. This describes the fact that the acceleration and deceleration ramp times are proportional based on the ratio of the acceleration and deceleration rates: the faster the acceleration ramp, the shorter the time allotted to the acceleration.

$$T_1 = \frac{a2}{a1} T_2 \qquad \text{Eq. 3}$$

The following steps substitutes the relationship between the acceleration and deceleration periods expressed in Eq. 3, into the distance relationship in Eq. 1, then solves for the deceleration time T2 in Eq. 4.

$$D = \frac{1}{2} T_2^2 \left(\frac{a_2^2}{a_1} + a_2\right) \qquad \text{Simplified to} \quad T_2^2 = 2\mathrm{D}/\left(\frac{a_2^2}{a_1} + a_2\right)$$

$$T_2 = \sqrt{2\mathrm{D}/\left(\frac{a_2^2}{a_1} + a_2\right)} \qquad \text{Eq. 4}$$

$$T_1 = \frac{a2}{a1} T_2 \qquad \text{Eq. 3}$$

3.3 Spanning Tree in routing problem

One of the most important algorithms in graph theory is the creation of networks with the property that have minimal length of links but with the property that all nodes still remain connected to the network. This is known as a minimal spanning tree. There are many algorithms for creating a minimal spanning tree but the simplest uses an approach known as the ‘’ greedy algorithm ‘’. Essentially this means choosing a node that is closest to an existing node and making that a link in the spanning tree. The algorithm proceeds by adding the next closest link in the network to the existing minimal spanning network. It is sometimes known as Prim’s Algorithm. The uses of term ‘’tree’’ arises because there are no circuits or loops in a minimal spanning tree. It can be shown that this simple algorithm does indeed produce the minimum connecting set of links for a network. In that sense it produces an optimal outcome in terms of minimizing connecting set of links for network- a rather valuable property given the cost of transportation on rail lines [33].

The number of connections a node has with other nodes is defined as the degree of the node. Most nodes in this network have single tracks but some nodes have more links. The relationship between the number of nodes and their degree has been observed by numerous studies to follow a power distribution with most nodes having a few links and a few nodes having many links. Scale free networks are so termed because the structure of the network remains the same no matter how closely one examines the network. It has ‘’ fractal’’ properties in that there is self similarity independent of the scale of observation of the network. This property is important because it can be shown that this means the network is quite robust to disruptions. If some links are broken, for whatever reason, a high degree of connectivity remains in the network.

Now suppose the edges of the graph have weights or lengths. The weight of a tree is just the sum of weights of its edges. Obviously, different trees have different lengths. The problem: how to find the minimum length spanning tree?

This problem can be solved by many different algorithms. It is the topic of some very recent research. There are several "best" algorithms, depending on the assumptions you make:

- A randomized algorithm can solve it in linear expected time. [34]
- It can be solved in linear worst case time if the weights are small integers. [35]
- Otherwise, the best solution is very close to linear but not exactly linear. The exact bound is O(m log beta(m,n)) where the beta function has a complicated definition: the smallest i such that log(log(log(...log(n)...))) is less than m/n, where the logs are nested i times. [36]

These algorithms are all quite complicated, and probably not that great in practice unless you're looking at really huge graphs

For simplicity, we assume that there is a unique minimum spanning tree. You can get ideas like this to work without this assumption but it becomes harder to state your theorems or write your algorithms precisely.

Assumptions. we adopt the following conventions:

- The graph is connected. The spanning-tree condition in our definition implies that the graph must be connected for an MST to exist. If a graph is not connected, we can adapt our algorithms to compute the MSTs of each of its connected components, collectively known as a minimum spanning forest.

- The edge weights are not necessarily distances. Geometric intuition is sometimes beneficial, but the edge weights can be arbitrary.
- The edge weights may be zero or negative. If the edge weights are all positive, it suffices to define the MST as the subgraph with minimal total weight that connects all the vertices.
- The edge weights are all different. If edges can have equal weights, the minimum spanning tree may not be unique. Making this assumption simplifies some of our proofs, but all of our our algorithms work properly even in the presence of equal weights.

The c++ algorithm on the spanning tree is designed to motivate the modern definition of a "tree" found in books covering graph theory, and then offer several applications of trees as well as one of the first algorithms for finding a minimal spanning tree. The source code are shown on Appendix B.

The output is shown on figure 7. The number of edges is 7. We assign the initial weights. The algorithm is capable of assigning the minimum weights to travel from source number i to destination j.We can see the output revealing the minimum weights between each source and each destination. as follows

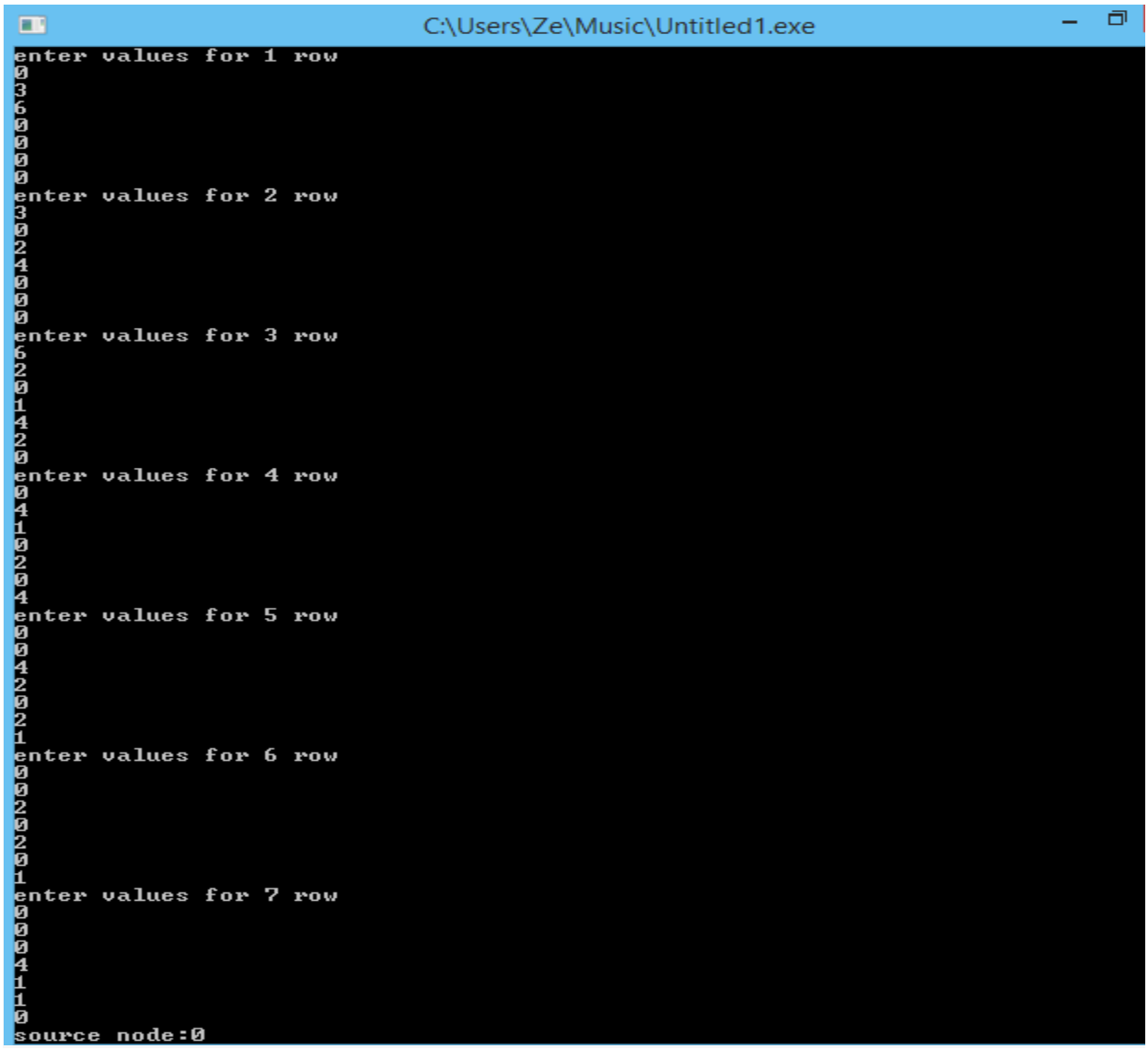


Figure 7 Input of our C++ program using Prims algorithm

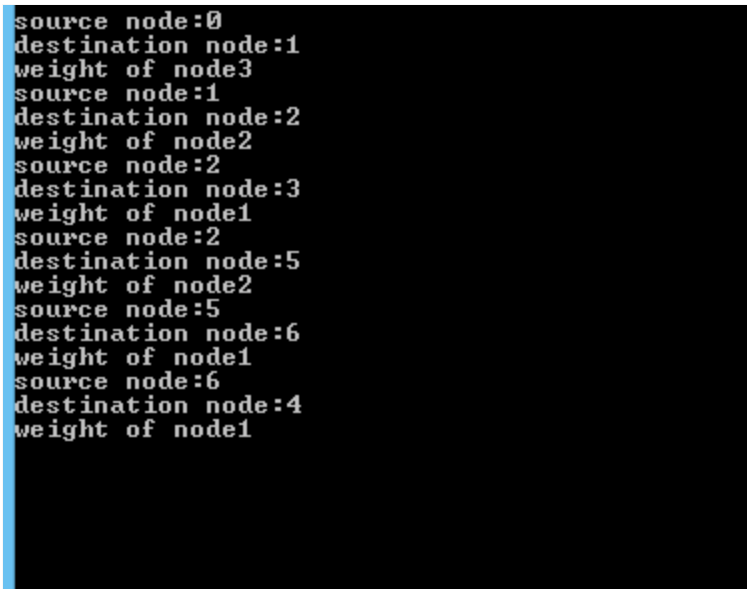


Figure 8 Output of our C++ program using Prims algorithm

In our results we developed a rational approach to construct the routing of trains in a edged graph railway network. In particular, we proposed an optimal path searching algorithm which combined the Prim's algorithm to provide users with more regular and reliable rail system by optimizing the route among stations. Here, we make use of Prims Algorithm. The results show how to optimize the routes among stations and present the user with the shortest route. Here shortest route is in terms of distance between the edges and not the time taken to travel from one station to another .

3.4 Neural Network Use for Delay Analysis

MLP has been inspired by biological neural connections in the human brain. The architecture of an MLP is significantly more simplified than that of a biological brain system. An MLP normally consists of an input layer, (at least) an intermediate layer (i.e., the hidden layer), and an output layer, which shows the performance output of the system. [37]

Let $x_r$represent the input values in a three-layer perceptron. The output of the hidden layer $y_s$and that of the output layer are respectively expressed as follows:

$$y_c = G_c\left(\left(\sum_{r=1}^{R} w_r^r\, x_r\right) + b_c\right), r \in R \quad c \in C$$

$$y_d = G_{dc}\left(\left(\sum_{c=1}^{C} w_{cr}^{dr}\, y_c\right) + b_d\right)$$

where w_r^c is the connection weights between the input and hidden layer, b_c is the bias of the hidden layer, G_c is the activation function of the hidden layer; w_c^d is the connection weights between the hidden and output layer, b_d is the bias of the output layer, and G_d is the activation function of the output layer.In the system, the hidden layer enhances the adaptive learning in MLP, which is the ability to learn how to do tasks on the basis of the data given for training or initial experience.

Let (x_d, y_d) be a training pair in training data D. Then, for arbitrary hidden layer neuron, error E in the weight space can be interpreted as follows:(x_d, y_d)

$$E = \frac{1}{2}(t_d - y_d)^2 = \frac{1}{2}\sum_{d\,\in D}(t_d - y_d)^2$$

where

t_d is the targeted (desired) output of the MLP,y_d is the actual (calculated) output of the MLP, d is the d^{th}input for the MLP, and D is the training data of the MLP. 5 [18].

- Single layer can be used only for simple problems. However its computation time is very fast.
- Multi-layer are most of the neural networks expect deep learning. it uses one or two hidden layers . The main advantage is they can be used for difficult to complex problems. However, they need long training time sometimes.
- Recurrent neural networks are used for dynamic systems mainly and for control systems since it is time-based and information might flow in any direction. You can use them for time based systems where you have a delay and a feedback.
- Another great class is the radial basis and generalized regression neural network. They are very good for approximation .GRNN(General regression neural network) is very quick single pass .

Chapter four

4. Simulation and result analysis

4.1 Simulation environment

A railway network with mixed services, was used as the basis of the model, and shown in Figure 8. The entire network has been allocated to three junctions and rescheduling is applied after scenario 1 and 2 occurs. The simulation is run using OPenRail 1-2. Stations and signals can be selected from a menu and set as timing points. In addition to timing points, the arrival and departure times at stations and passing times at selected signals can be recorded for all trains. From this information, details of the propagation of delays in the system can be analyzed by the visualization method using matlab R2009b

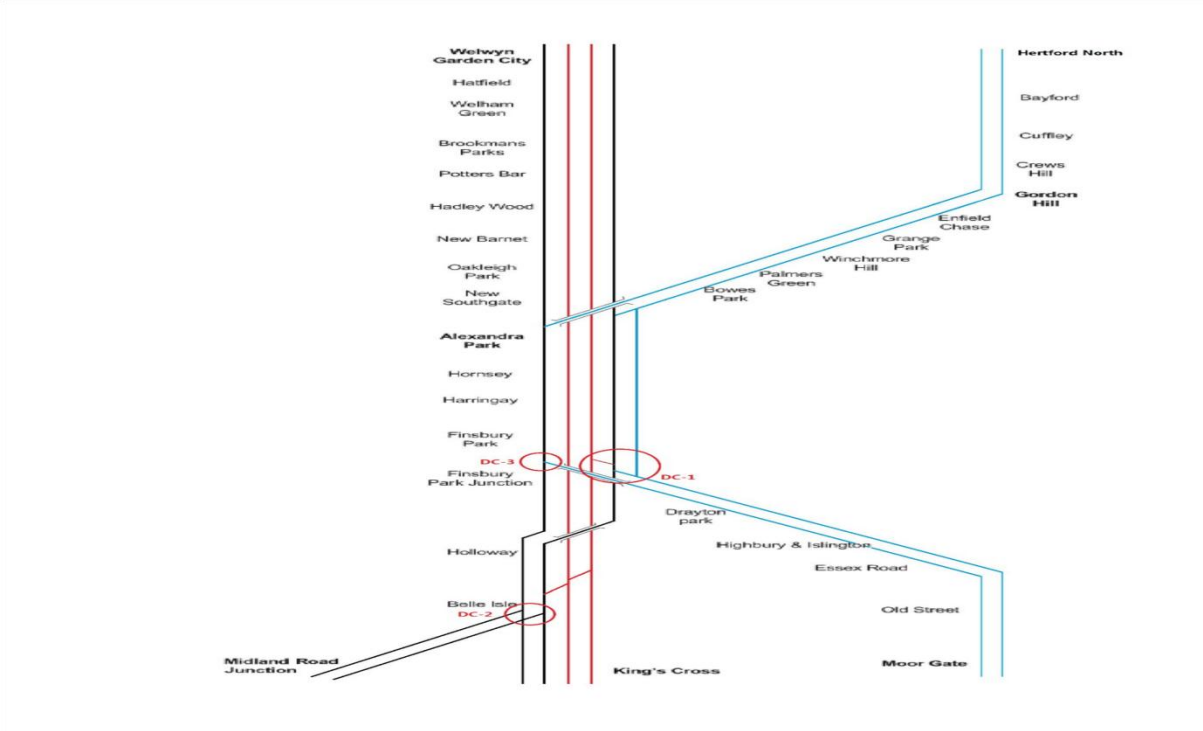


Figure 8 area of simulation

The simulation is carried out with a simplified timetable with 37 services running from 7:00 am to 9:00am based on a timetable published by Network Rail [39] (see Appendix C). These

services operates with four types of vehicles, which are Class 91, Class 317, Class 313 and TL39. The characteristics of each type are presented in Table 3, including the maximum speed of the train, the maximum braking and acceleration rates, and the train-specific delay penalties. All this information is used to simulate train running and rescheduling cost in delays

Type	**Class 91**	**Class 317**	**Class 313**	**TL39**
Maximum speed (km/h)	201	160	121	160
Maximum braking rate (ms^{-2})	0.67	0.78	0.78	0.78
Max acceleration (ms^{-2})	0.588	0.588	0.588	1
Total mass (tonnes)	394	264	220	332
Train length (m)	253.9	158.6	118	161.99
Number of seats	400	400	231	231
Penalty (£s^{-1})	0.5	0.2	0.1	0.1

Table 3 Trains specifications

Introduction to the delay scenarios

Scenario 1: Train E1 is delayed at Potters Bar Station for 930 seconds and at Finsbury Park for 300 seconds. For this scenario, one train is delayed before passing through DC-1, and rescheduling strategy is applied.

Scenario 2: Train E1 is delayed at Potters Bar Station for 930 seconds and at Finsbury Park for 300 seconds. Train J2 is delayed at Old Street Station for 510 seconds

4.2 Results

To evaluate the results, the delay propagations of different trains are presented. Considering the cost function to minimize the total delay penalty, the total weighted delay is used to help visualize and compare the results. Delays are weighted according to the train type: the weight for the passenger trains (vehicle types are Class 313 and TL39) is 1, for the premium passenger trains (vehicle types are Class 317) it is 2, and for high speed trains (vehicle types are Class 91) it is set to 5.

Table 4 and Table 5 show the results obtained by applying our rescheduling approach, including the times to active rescheduling, places of rescheduling, rescheduling approaches and the planned order in which the trains pass the junction area.

Simulation Result for Scenario 1

07:00:00	DC-1	Order = [S11, S16]
07:00:00	DC-2	Order = [S17, S20, S33]
07:00:00	DC-3	Order = [S17, S20, S18, S19, S34]
07:33:00	DC-1	Order = [S12, S24, S15, S13, S21, S14, S22, S28]
07:33:53	DC-2	Order = [30]
07:35:35	DC-3	Order = [S33, S30, S31, S32]
07:46:08	DC-1	Order = [S27, S13, S21, S14, S22, S28, S25, S23, S26]
07:50:46	DC-2	Order = [S15, S14]
07:56:27	DC-1	Order = [S28, S22, S25, S26, S23]
08:00:56	DC-3	Order = [S45]
08:01:52	DC-2	Order = [S28, S41]
08:03:53	DC-2	Order = [S42]
08:05:35	DC-3	Order = [S41, S42, S44, S46]
08:11:08	DC-1	Order = [S37, S26, S39, S23, S40, S35, S36, S38]
08:18:49	DC-2	Order = [S26, S40, S38]

Table 4 Results of rescheduling strategy for scenario 1

In Table 4, the application of our rescheduling plan applies at each time to the order of services have entered the junction area, which means, the trains out of the selected area are deleted from the original timetable order. There is a significant influx in the total weighted delay at time t = 7.8 h because of the fact that the disruption on E1 at Finsbury Park Station delays the trains behind it.

Simulation Result for Scenario 2

Rescheduling time	Junction area	Order of the trains
07:00:00	DC-1	Order = [S11, S16]
07:00:00	DC-2	Order = [S17, S20, S15, S33, S30, S14, S28, S41, S42, S26, S40, S38]
07:00:00	DC-3	Order = [S17, S20, S18, S19, S34, S33, S30, S31, S32, S45, S41, S42, S44, S46]
07:33:00	DC-1	Order = [S12, S24, S15, S13, S21, S14, S22, S28]
07:46:08	DC-1	Order = [S27, S13, S21, S14, S22, S28, S25, S23, S26]
07:56:27	DC-1	Order = [S28, S22, S25, S26, S23]
08:11:08	DC-1	Order = [S37, S26, S39, S23, S40, S35, S36, S38]

Table 5 Results of rescheduling after scenario 2

The result from the rescheduling cost function (i.e. total delay penalty) and the three KPIs of the delay propagation - maximum lateness, time to recover and integral of delay of each scenario are collected and summarized in Table 6 and Table 7. It can also be given as a proportion of the maximum possible integral of delay as: Integral of delay proportion = Integral of delay/ (Maximum lateness* Time to recover). This value gives an indication of how severe the delays were relative to the maximum lateness throughout the period of delay.

Total delay penalty (£)	Maximum weighted Lateness (s)	Time to recover (s)	Integral Delay (s^2) (normalized value (%))	Proportion of Integral Delay (%)
674.3	7165	3055	7943800 (100)	36.29

Table 6 Summary of delay propagation in scenario 1

Table 6 shows total delay penalty and takes 3055 s to recover from the disturbed situation in scenario 1. Table 7 shows total delay penalty and takes 3102 s to recover from the disturbed situation in scenario 2

Total delay penalty (£)	Maximum weighted Lateness (s)	Time to recover (s)	Integral Delay (s^2) (normalized value (%))	Proportion of Integral Delay (%)
738.1	8056	3102	10597535 (100)	42.41

Table 7 Summary of delay propagation in scenario 2

Maximum lateness indicates the maximum total lateness of all journeys running in the network within the time window T; it indicates the worst delay situation. Time to recover measures how quickly the system returns back to a situation with no delays. The third main measure,

Integral of delay, measures the area under the lateness curve. Combined with the other two KPIs, this measure gives a quantitative summary of the delayed situation.

It can also be given as a proportion of the maximum possible integral of delay as: Integral of Delay proportion = Integral of delay/ (Maximum lateness* Time to recover). This value gives an indication of how severe the delays were relative to the maximum lateness throughout the period of delay.

The whole network has been divided into different junctions and each DC indicates a junction area along the network. These junctions work individually and independently, but the network database are shared by all of them. The components of this architecture are detailed and illustrated with this case study.

The advantages are that this method may increase the efficiency of rescheduling in a large scale network because all DCs are processing in parallel and asynchronously, and this method achieves coordination between each DC. The rescheduling results are proven to be feasible and efficient for rescheduling problem. The disadvantage is that global optimization cannot be guaranteed because of the distributed architecture, and boundaries of the DCs should be carefully defined, which will influence the final results

Neural network use for delay Analysis

The data has been collected from Britain railway. It represents the data delays collected and their impact on number of trains in a single area after in type of disruption. We will show how the

learning rate of such neural network can be used to predict future impact of the same disruptions type.

For our experiments, we have chosen a window size of 1 year where data regarding delays and the trains they affect are collected .We will implement on matlab R2009 a neural architecture of type Multilayer Perceptron.

In the experiment, the samples deployed in the model principally consist of two subsets, namely, training and test samples. Sample size is defined as the number of samples in one subset. The training subset data are used to recognize and analyze the potential structure of the connection weights by gradient descent in training phases. Meanwhile, the samples in the test subset are not used during training but are employed to verify these weights. The input features of a neural network are the input variables derived from a dataset that are fed into the neural network. The target will represent the number of trains affected by the delay for one type of disruption.

We began with the intention of analysis and modelling the delay impact on trains on the British rail network. We showed that our neural structure reach the performance target of 98 percent as shown on figure 9

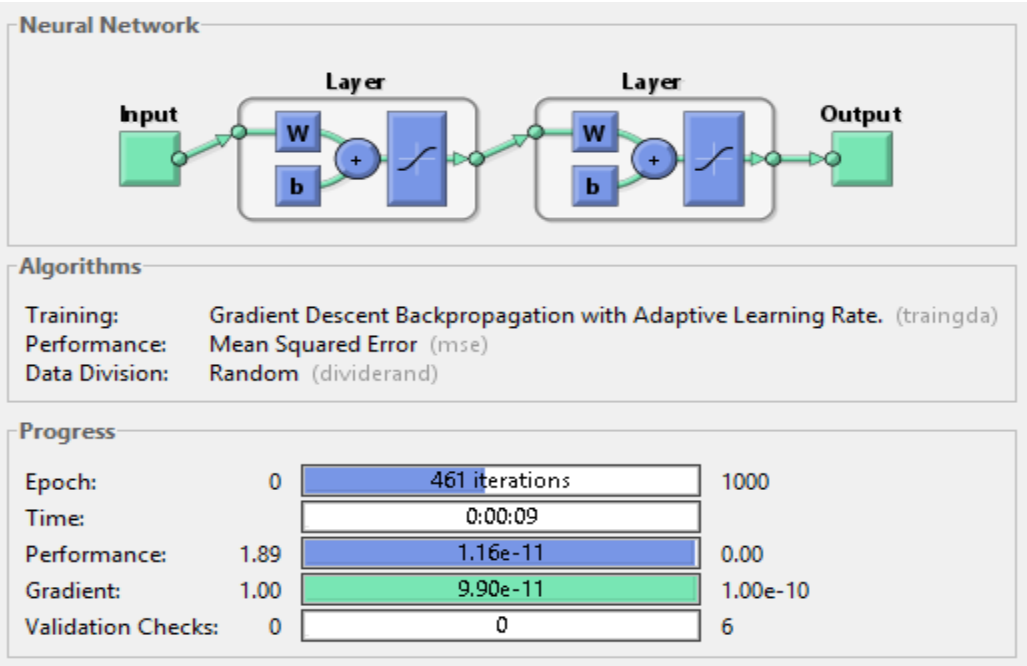


Figure 9 Architecture of neural network used

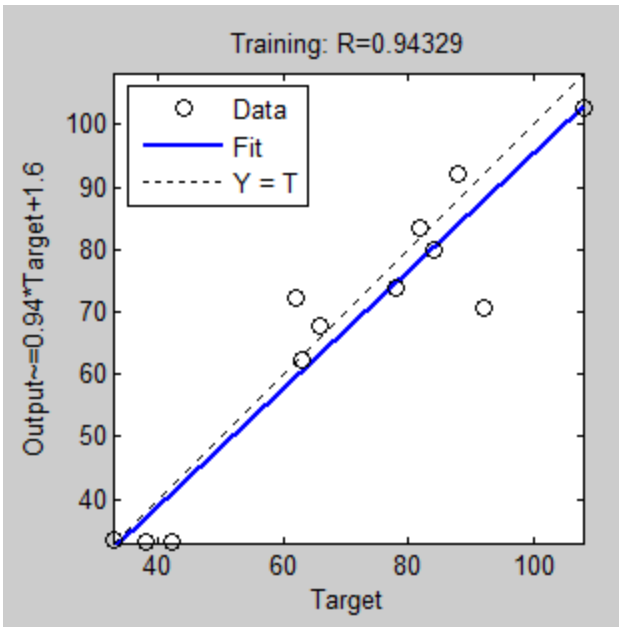


Figure 90 showing Training coefficient R

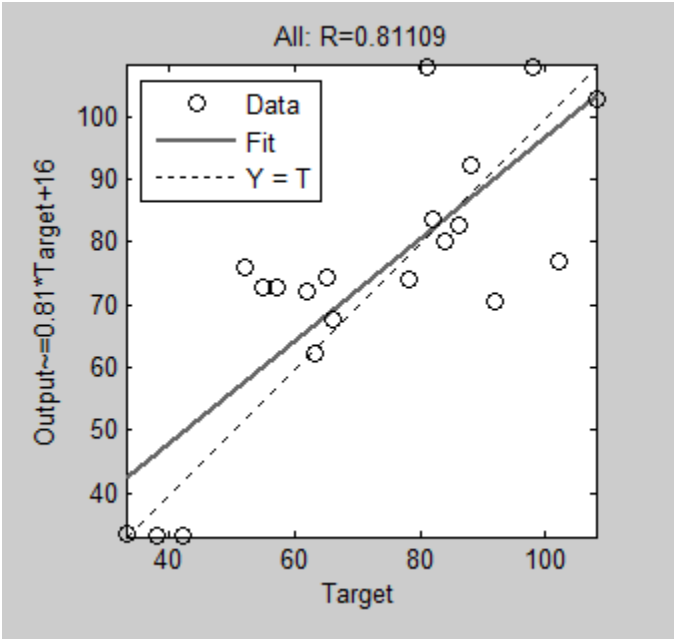


Figure 11 of Adjusted weights of the coefficient R

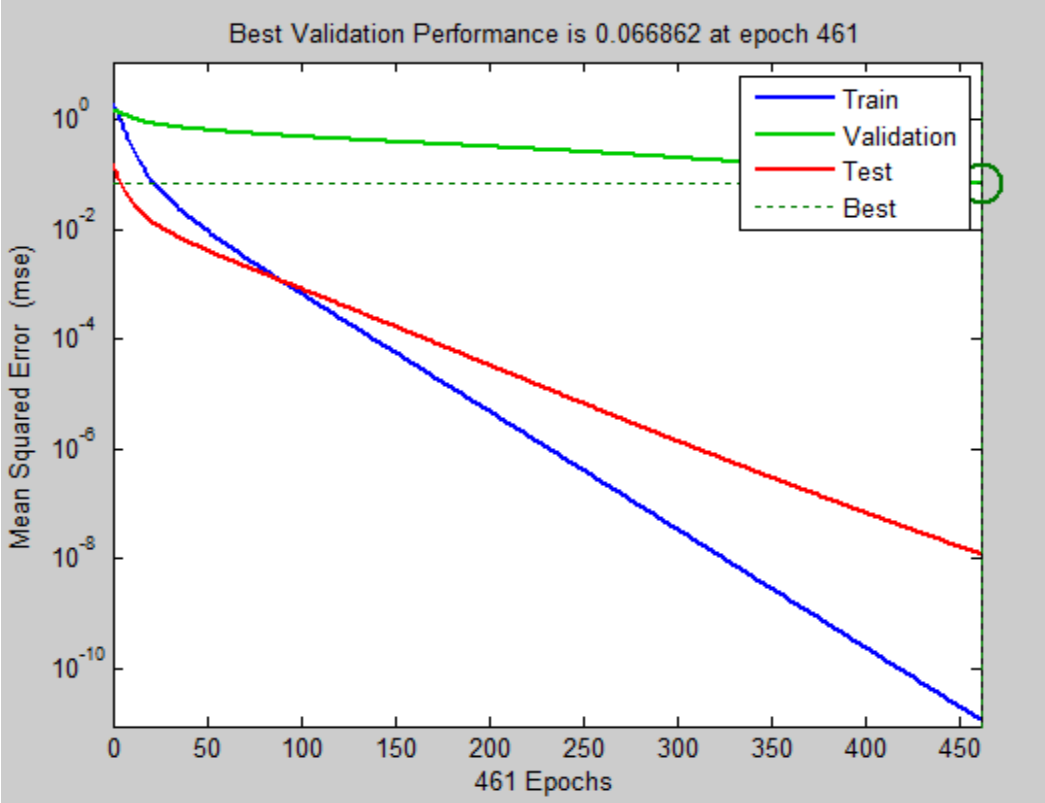


Figure 12 MSE evolution with number of iterations (Epochs)

The process of training a neural network involves tuning the values of the weights and biases of the network to optimize network performance. The default performance function is mean square error (MSE)the average squared error between the network outputs a and the target outputs t. During training, the progress is constantly updated in the training window. Of most interest are the performance, the magnitude of the gradient of performance and the number of validation checks. The magnitude of the gradient and the number of validation checks are used to terminate the training. The gradient will become very small as the training reaches a minimum of the performance. Each time a neural network is trained, can result in a different solution due to different initial weight and bias values and different divisions of data into training, validation, and test sets. As a result, different neural networks trained on the same problem can give different outputs for the same input. To ensure that a neural network of good accuracy has been found, we retrain several times.

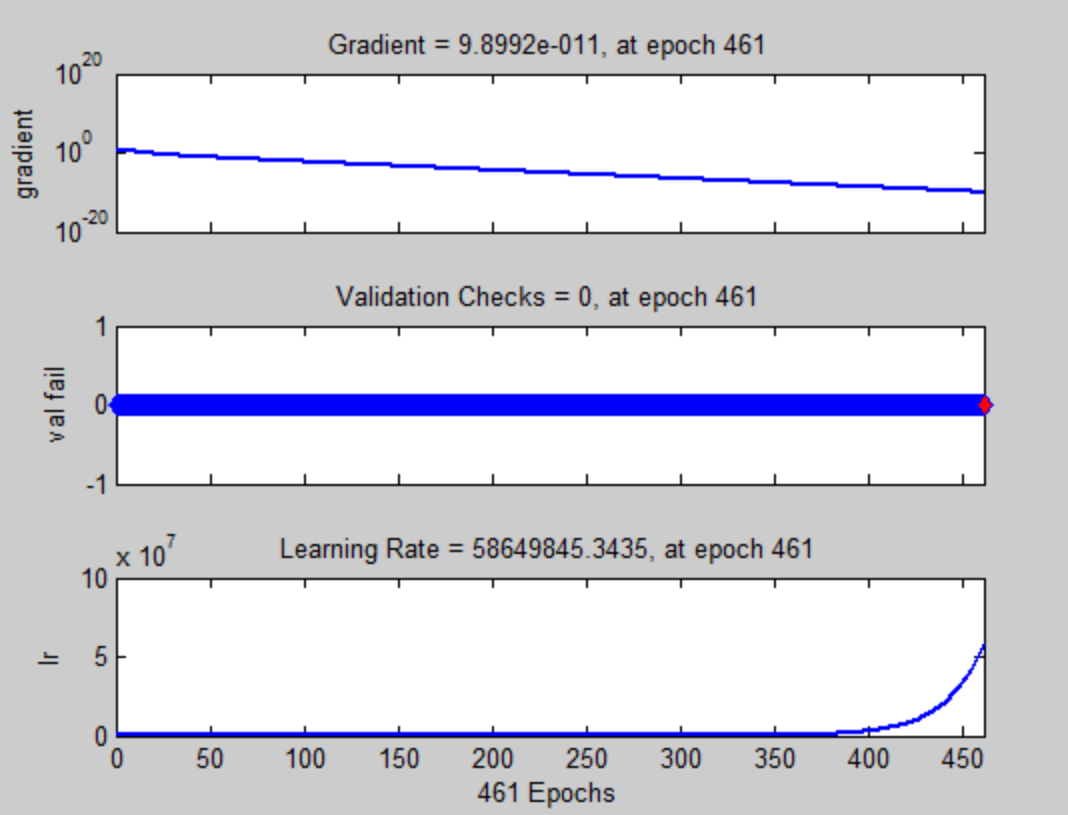


Figure 13 Summary on Gradient and learning rate evolution during learning process

With the operational data and delay available, delay combinations can be easily formed for each primary delay. Trains that have suffered a knock-on delay are allocated to the delay combination of their respective primary delay. We set up a regression plot, which shows the relationship between the outputs of the network and the targets. If the training were perfect, the network outputs and the targets would be exactly equal, but the relationship is rarely perfect as in this case study. The trained network extracts the outputs and targets that belong to the training, validation and test subsets. The final command creates three regression plots for training, testing and validation.

The three plots represent the training, validation, and testing data summarized in Figure 12. The dashed line in each plot represents the perfect result – outputs = targets. The solid line represents the best fit linear regression line between outputs and targets. The R value is an indication of the relationship between the outputs and targets. If R = 1, this indicates that there is an exact linear relationship between outputs and targets. If R is close to zero, then there is no linear relationship between outputs and targets. The performance curve looks good. Training, cross validation and

test sets are not deviating from each other to any serious extent. Although more training could probably continue in order to lower the MSE, training here stopped after 461 iterations. Next is the regression plot in figure 13. It also looks good, to some extent. This graph indicates how close to regression this neural network is.

For this study, the training data indicates a good fit. The validation and test results also show R values that greater than 0.8(See figure 10 and 11). The next step would be to investigate this data point to determine if it represents extrapolation (i.e., is it outside of the training data set). If so, then it should be included in the training set, and additional data should be collected to be used in the test set.

This simulation shows the application of neural network architecture to extract information on the condition of railway vehicle systems from diagnostic event data and to use the extracted patterns for predicting occurring operational disruption events with a learning rate of 98 percent.

Delays above average value are caused by delays of other railway companies or by extraordinary events (e.g.: snowing, problems at switches due to extremely cold weather)

In light of the aforementioned findings we can state that this kind of examination can and should be done, in order to achieve a deeper understanding of the system and delay impact. With this understanding, one is able to conclude both short- and long-term development and management activities. In short-term activity, as in real-time train dispatching, the information can offer answers to questions such as how to mitigate the emergence of secondary delays, how to prioritize trains, and how to recognize possible patterns to predict how many trains could be affected by such delays.

Chapter Five

5. Conclusions and Recommendation

5.1 Conclusion

The occurrence of disruptions leads to a chain of knock-on delays in the railway network due to the growth of traffic demands and limited resources. In this thesis, rescheduling approaches are studied to minimize the effect of these delays. The purpose of real-time train rescheduling is to find a conflict-free operating schedule for the services in a given area and a given time horizon during operations, providing a solution that is compatible with the actual traffic conditions and infrastructure constraints. The main objective of this thesis is to provide the dispatchers with better decision support by considering the delay propagation of the whole railway network after disruptive events.

A time-based delay propagation graph is developed to visualize and evaluate the delay recovery performance. Rescheduling is applied in two scenarios for minimizing the delays. The local rescheduling systems generate reschedule plans applied on different junctions

A case study is presented to verify the feasibility and efficiency of this control architecture to the train rescheduling problem. The advantages are that this method may increase the efficiency of rescheduling in a large scale network because all DCs are processing in parallel and asynchronously, and this method achieves coordination between each because of acceptable delays penalty recorded.

In the second part of the simulation, we have shown the significance of processing delays especially in railway traffic systems with a high proportion of single tracks. We discovered the current ways to analyze delays based on the trains affected by disruptions and found that data mining could offer a novel approach for understanding the delays propagation. We designed a neural network that at least process the most explicit delay chains and identify how many trains are affected with a learning rate of 98 percent. We achieved a high learning rate because the volume of data processed is little compared to the actual volume of delays our railway system provided. The last requires high computational power to run a consequent MLP

Correspondingly, in long-term planning, the information can help to develop the structure of the timetable in the form of adding buffer times, removing critical correlations and so on, in order to gain a more robust railway system.

Since the main contribution of this paper was to understand the rescheduling operations, the design of a perceptron for delay processing is a good case study and shows how neural networks an option for railway data processing. Hence, it can be said that the study achieves both scientific and pragmatic significance, for this study takes the first step to examine delay propagation within railway traffic system and suggests a useful process to end-users for analyzing the actual data regarding rescheduling operations.

5.2 Recommendation

Artificial neural networks have some interesting properties that made these families of machine learning algorithms very appealing when confronting difficult patter-discovery tasks.

From the proposed artificial neural network, and the case study applied to the British Rail network, several recommendations can be done regarding a follow-up research, data management in railway operations and similar machine learning applications.

A focus towards relational integrated data management instead of using the traditional table-based format would greatly increase the possibilities for other studies such as this one, and the general versatility of the data model. As proven in this study, the use of a relational data model such as a graph model enables for more in-depth studies into the dynamics of railway operations, and more general combinatorial analysis.

The current practice to store all information in separate systems with minimal horizontal integration makes the execution of a similar study harder to start up. The possibilities for big data analysis that open up from utilizing a more integrated data model would have enormous potential for issues in asset management, maintenance planning, traffic control and long-term strategic planning.

With the correct data and machine learning methods, the theoretical possibilities would supersede the current performance of micro-simulation in both accuracy and applicability, given

that machine learning methods keep advancing at the current rate. The best performing artificial neural networks can have millions of connections and weight parameters. Theoretically the performance of artificial neural networks using more layers and more connections should improve significantly, suggesting that this method can ideally reach a better accuracy

Reference

[1]. B. Adenso-Diaz, M. Oliva-Gonzalez, and P. Gonzalez-Torre,On-line timetable re-scheduling in regional train services. Transportation Research Part B: Methodological 33, 1999, 387–398.

[2]. A. Bauer, 2008. SNCF-RFF, le couple infernal du rail (in French). Les Echos. 2008-11-12 (http://archives.lesechos.fr/archives/2008/LesEchos/20298-43-ECH.htm).

[3]. M. Kuhn, Automatic route setting integrated in a modern traffic management system. Developments in Mass Transit Systems conference n453, 1998, pp. 140-145.

[4]. [4] S. Sharples, The Impact of Automation in Rail Signalling Operations. Proc. of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit, 225(2), pp. 179-191, 2011.

[5]. RT.Marler and J.S. Arora."Survey of Multi objective Optimization Methods for Engineering, 2004,pp 369–395.

[6]. A. Svendsen et al., Formalizing Train Control Language: Automating Analysis of Train Stations. Proc. of the 12th Int. Conf. on Computer System Design and Operation in Railways and Other Transit Systems, 2010, pp. 245-256.

[7]. P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. SIAM J. on Discrete Mathematics, 1989,pp 550–581.

[8]. Open Journal of Social SciencesVol.04 No.11(2016), Article ID:72341, 10.4236/jss.2016.411005

[9]. European Commission Eurostat (2015) Energy, Transport and Environment Indicators—2015 Edition. European Commission Eurostat, Luxemburg.

[10]. M. Finger, Governance of Competition and Performance in European Railways: An Analysis of Five Cases. Utilities Policy 31, 2014,pp 278-288.

[11]. T.Dollevoet, F. Corman, A. D'Ariano, and D. Huisman, An Iterative Optimization Framework for Delay Management and Train Scheduling. Flexible Services and Manufacturing Journal, 26, 2014,490-515

[12]. A.Caris, C. Macharis, and G.K. Janssensa, Decision Support in Intermodal Transport: A New Research Agenda. Computers in Industry 64, 2013,pp 105-112.

[13]. Michiels, W. and S. Niculescu, Stability, Control, and Computation for Time-Delay Systems: An Eigenvalue-Based Approach. 2nd Edition, SIAM-Society for Industrial and Applied Mathematics, Philadelphia, 2014.

[14]. M. Srinivasan, D. Mukherjee, and A.S Gaur, Buyer-Supplier Partnership Quality and Supply Chain Performance: Moderating Role of Risks, and Environmental Uncertainty,European of Management Journal, 29, 2011,260-271.

[15]. P.Murali, , Dessouky, M., Ordó, F. and Palmer, K. A Delay Estimation Technique for Single and Double-Track Railroads. Transportation Research Part E, 46, 2010,pp 483-495.

[16]. V.S. Rodrigues, D.Stantchev, A.Potter, , M.Naim, and A.Whiteing, Establishing a Transport Operation Focused uncertainty Model for the Supply Chain. International Journal of Physical Distribution & Logistics Management, 38, 2008,pp 388-411.

[17]. Res, V., Fabian Meier, J. Pace and R. Palacin, Rail and Multi-Modal Transport. Research in Transportation Economics, 41,2013,pp 17-30.

[18]. R.M.Goverde, A Delay Propagation Algorithm for Large-Scale Railway Traffic Networks. Transportation Research Part C, 18, 2010, pp 269-287.

[19]. L.Häme. and H.Hakula, Dynamic Journeying under Uncertainty. European Journal of Operational Research, 225, 2013,pp 455-471.

[20]. J.Barta, A.E.Rizzoli, M. Salani, L.M. Gambardella, Statistical Modelling of Delays in a Rail Freight Transportation Network. Proceedings of the 2012 Winter Simulation Conference, Switzerland, 2012.

[21]. E. Kayacan, B.Ulutas, and O.Kaynak ,Grey System Theory-Based Models in Time Series Prediction. Expert Systems with Applications, 37, 2010,pp 1784-1789.

[22]. M.K. Hasan, A Framework for Intelligent Decision Support System for Traffic Congestion Management System. Scientific Research, 2,2010, 270-289.

[23]. J.Ludvigsen, and R.Klaebe, Extreme Weather Impacts in Freight Railways in Europe. Natural Hazards, 70, 2014, pp 767-787.

[24]. D.Graupe. ,Principles of Artificial Neural Networks, 3rd Ed, Advanced Series in Circuits and Systems, World Scientific Publishing Co. Pte. Ltd., 2013.

[25]. Senties, O.B., Azzaro-Pantel, C., Pibouleau, L. and Domenech, S, A Neural Network and a Ge-netic Algorithm for Multiobjective Scheduling of Semiconductor Manufacturing Plant. Industrial & Engineering Chemistry Research, 2009.

[26]. S.Haykin ,Neural Networks and Learning Machines, International Edition ed., Pearson Education, Inc., New Jersey, 2009.

[27]. Hagan, M.T. and Demuth, H.B. Neural Network Design. 2. Ed., PWS Publishing Co., Boston, 2014.

[28]. Roy Chatterjee, S., Mandal, R. and Chakraborty, M. (2013) A Comparative Analysis of Several Back Propagation Algorithms in Wireless Channel for ANN-Based Mobile Radio Signal Detector. International Journal of Science and Modern Engineering, 1.

[29]. Ghoddousi, P., Eshtehardian, E., Jooybanpour, S. and Javanmardi, A. ,When the Maximal Iteration Number for the Generation Is Reached, the Training Process for the Network Must Be Terminated, and the Established MLP Is Considered as a Well-Trained Network. Automation in Construction, 30, 2013 , pp216-227.

[30]. Auber D., Chiricota Y., Jourdan F., MelanconG.: Multi-scale visualization of small world networks. In Proceedings of the 2003 IEEE Symposium on Information Visualization, 2003, pp. 75–81.

[31]. Anthonisse J.: The rush in a directed graph, Technical Report BN 9/71. Tech. rep., Stichting Mathe- matisch Centrum, Amsterdam, 1971.

[32]. Brandes U., Pich C.: Centrality estimation in large networks. International Journal of Bifurcation and Chaos 17, 2007, pp2303–2318.

[33]. Chen C., Morris S.: Visualizing evolving net- works: Minimum spanning trees versus pathfinder net- works. In Proceedings of the 2003 IEEE Symposium on Information Visualization , 2003 pp. 67–74.

[34]. Freeman L.: A set of measures of centrality based on betweenness. Sociometry 40, 1997, pp 35–41.

[35]. Ghoniem M., Fekete J.-D., Castagliola P.: A comparison of the readability of graphs using node- link and matrix-based representations. In Proceedings of the 2004 IEEE Symposium on Information Visualization, 2004, pp. 17–24.

[36]. Hachul and Junger ,An experimental com- parison of fast algorithms for drawing large general graphs". In Proceedings Graph Drawing 2005 (GD'05), 2005, pp. 235–250.

[37]. Karger, Klein, and Tarjan, A randomized linear-time algorithm to find minimum spanning trees, J. ACM, vol. 42, 1995, pp. 321-328.

[38]. NetworkRail. (2010). "Route Plans 2010 - Route Plan G East Coast & North East."

Appendix A

```
/*
* C++ Program to Find MST(Minimum Spanning Tree) using Kruskal's Algorithm
*/
#include<iostream>
#include<conio.h>
using namespace std;
int flag = 0, v[7];
struct node_info
{
   int no;
}*q = NULL, *r = NULL;
struct node
{
   node_info *pt;
   node *next;
}*top = NULL, *p = NULL, *np = NULL;
void push(node_info *ptr)
{
   np = new node;
   np->pt = ptr;
   np->next = NULL;
   if (top == NULL)
   {
      top = np;
   }
   else
   {
      np->next = top;
      top = np;
```

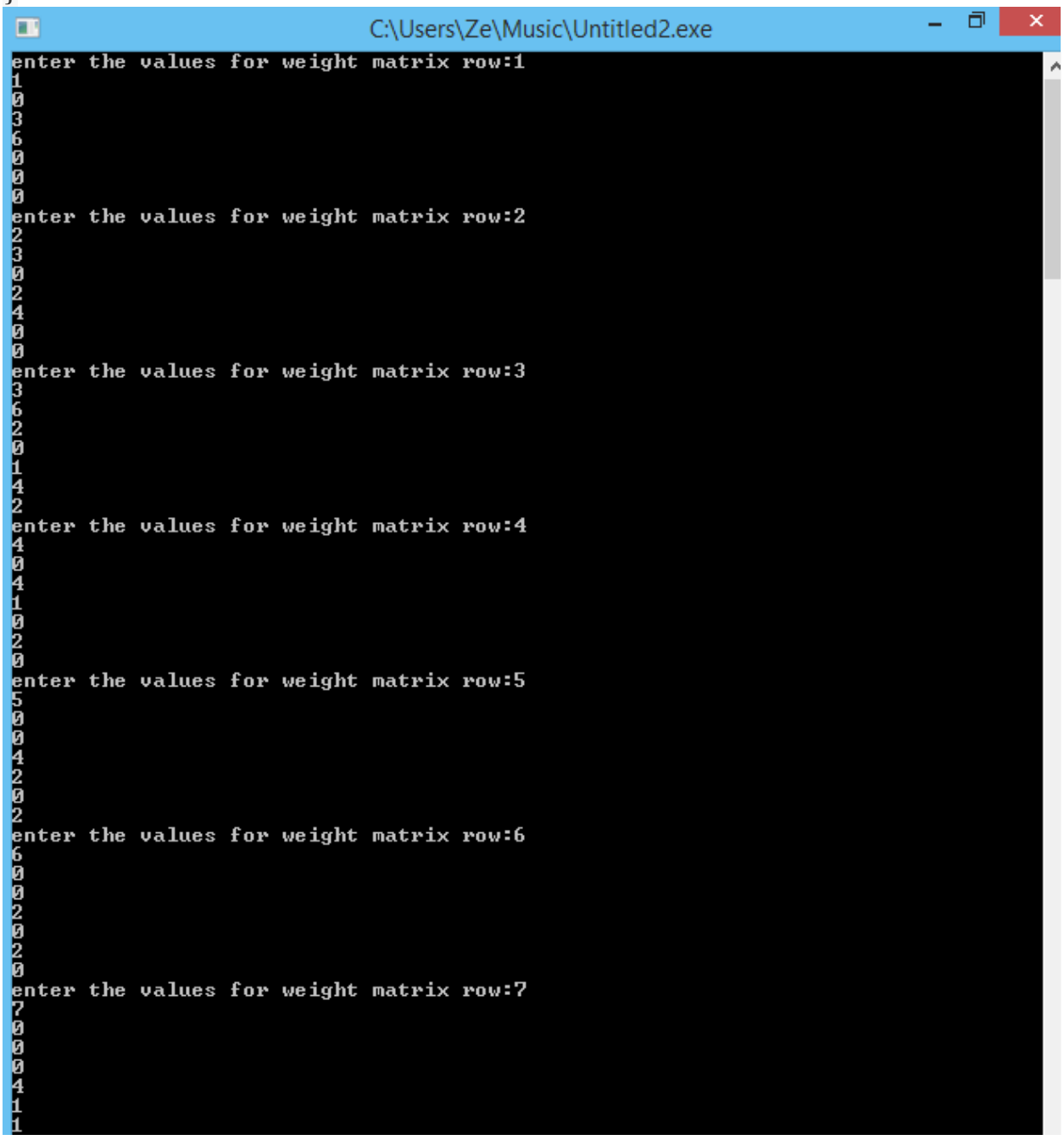
```
    }
}
node_info *pop()
{
    if (top == NULL)
    {
        cout<<"underflow\n";
    }
    else
    {
        p = top;
        top = top->next;
        return(p->pt);
        delete(p);
    }
}
int back_edges(int *v,int am[][7],int i,int k)
{
    q = new node_info;
    q->no = i;
    push(q);
    v[i] = 1;
    for (int j = 0; j < 7; j++)
    {
        if (am[i][j] == 1 && v[j] == 0)
        {
            back_edges(v, am, j, i);
        }
        else if (am[i][j] == 0)
            continue;
        else if ((j == k) && (am[i][k] == 1 && v[j] == 1))
```

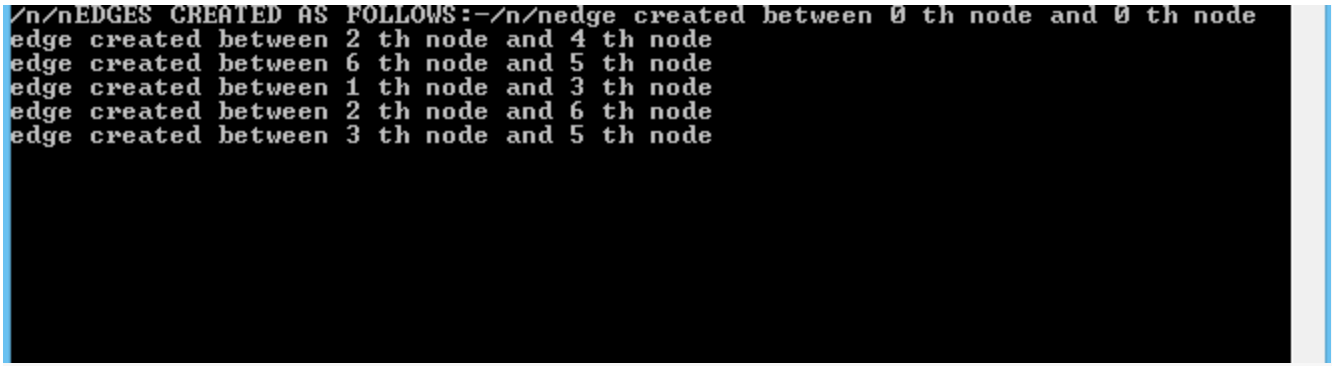
```
            continue;
        else
        {
            flag = -1;
        }
    }
    r = pop();
    return(flag);
}
void init()
{
    for (int i = 0; i < 7; i++)
        v[i] = 0;
    while (top != NULL)
    {
        pop();
    }
}
void kruskals(int am[][7], int wm[][7])
{
    int ve = 1, min, temp, temp1;
cout<<"/n/nEDGES CREATED AS FOLLOWS:-/n/n";
    while (ve <= 6)
    {
        min = 999, temp = 0, temp1 = 0;
        for (int i = 0; i < 7; i++)
        {
            for (int j = 0; j < 7; j++)
            {
                if ((wm[i][j] < min) && wm[i][j]!=0)
                {
```

```
                min = wm[i][j];
                temp = i;
                temp1 = j;
            }
            else if (wm[i][j] == 0)
                continue;
        }
    }
    wm[temp][temp1]=wm[temp1][temp] = 999;
    am[temp][temp1]=am[temp1][temp] = 1;
    init();
    if (back_edges(v, am, temp, 0) < 0)
    {
        am[temp][temp1]=am[temp1][temp] = 0;
        flag = 0;
        continue;
    }
    else
    {
        cout<<"edge created between "<<temp<<" th node and "<<temp1<<" th node"<<endl;
        ve++;
    }
  }
}
int main()
{
  int am[7][7], wm[7][7];
  for (int i = 0; i < 7; i++)
    v[i] = 0;
  for (int i = 0; i < 7; i++)
  {
```

```
    for(int j = 0; j < 7; j++)
    {
        am[i][j] = 0;
    }
}
for (int i = 0; i < 7; i++)
{
    cout<<"enter the values for weight matrix row:"<<i + 1<<endl;
    for(int j = 0; j < 7; j++)
    {
        cin>>wm[i][j];
    }
}
kruskals(am,wm);
getch();
```

```
}
```


C:\Users\Ze\Music\Untitled2.exe
enter the values for weight matrix row:1
1
0
3
6
0
0
0
enter the values for weight matrix row:2
2
3
0
2
4
0
0
enter the values for weight matrix row:3
3
6
2
0
1
4
2
enter the values for weight matrix row:4
4
0
4
1
0
2
0
enter the values for weight matrix row:5
5
0
0
4
2
0
2
enter the values for weight matrix row:6
6
0
0
2
0
2
0
enter the values for weight matrix row:7
7
0
0
0
4
1
1


/n/nEDGES CREATED AS FOLLOWS:-/n/nedge created between 0 th node and 0 th node
edge created between 2 th node and 4 th node
edge created between 6 th node and 5 th node
edge created between 1 th node and 3 th node
edge created between 2 th node and 6 th node
edge created between 3 th node and 5 th node

Appendix B

```
/*
* C++ Program to find MST(Minimum Spanning Tree) using
* Prim's Algorithm
*/
#include <iostream>
#include <conio.h>
using namespace std;
struct node
{
   int fr, to, cost;
}p[6];
int c = 0, temp1 = 0, temp = 0;
void prims(int *a, int b[][7], int i, int j)
{
   a[i] = 1;
   while (c < 6)
   {
      int min = 999;
      for (int i = 0; i < 7; i++)
      {
         if (a[i] == 1)
         {
            for (int j = 0; j < 7; )
            {
               if (b[i][j] >= min || b[i][j] == 0)
               {
                  j++;
               }
               else if (b[i][j] < min)
               {
                  min = b[i][j];
                  temp = i;
                  temp1 = j;
               } }
         }
      }
      a[temp1] = 1;
```

```
        p[c].fr = temp;
        p[c].to = temp1;
        p[c].cost = min;
        c++;
        b[temp][temp1] = b[temp1][temp]=1000;
    }
    for (int k = 0; k < 6; k++)
{
        cout<<"source node:"<<p[k].fr<<endl;
cout<<"destination node:"<<p[k].to<<endl;
        cout<<"weight of node"<<p[k].cost<<endl;
    }
}
int main()
{
    int a[7];
    for (int i = 0; i < 7; i++)
    {
        a[i] = 0;
    }
    int b[7][7];
    for (int i = 0; i < 7; i++)
    {
        cout<<"enter values for "<<(i+1)<<" row"<<endl;
        for (int j = 0; j < 7; j++)
        {
            cin>>b[i][j];
        }
    }
    prims(a, b, 0, 0);
getch();
}


[
```

Appendix C

NOMINAL TIMETABLE OF THE CASE STUDY

The simulation in Chapter 4 is carried out with a simplified timetable with 37 services running from 7:00 am to 9:00am based on a practical timetable published by Network Rail [39]. The following tables give the information of these 37 services, including the Train numbers, Service IDs/Train IDs, Types of vehicles, Departure time at the stations, Minimum and Maximum dwell time at the stations and Arrival time at the termination.

Timetable 1: Hertford to Moor Gate

	1	2	3	4	5
	S12/A1	S21/A2	S22/A3	S23/A4	S35/A5
	Class 313	Class 313	Class 313	Class 313	Class 313
Hertford	07:00:00	07:15:00	07:30:00	07:45:00	08:00:00
Bayford	07:04:21 30s~30s	07:19:21 30s~30s	07:34:21 30s~30s	07:49:21 30s~30s	08:04:21 30s~30s
Cuffley	07:09:00 30s~30s	07:24:00 30s~30s	07:39:00 30s~30s	07:54:00 30s~30s	08:09:00 30s~30s
Crews Hill	07:12:00 30s~30s	07:27:00 30s~30s	07:42:10 30s~30s	07:57:10 30s~30s	08:12:10 30s~30s
Gordon Hill	07:16:00 60s~60s	07:31:00 60s~60s	07:46:00 60s~60s	08:01:00 60s~60s	08:16:00 60s~60s
Enfield Chase	07:18:30 60s~60s	07:33:30 60s~60s	07:48:30 60s~60s	08:03:30 60s~60s	08:18:30 60s~60s
Grange Park	07:20:30 30s~30s	07:35:30 30s~30s	07:50:20 30s~30s	08:05:20 30s~30s	08:20:20 30s~30s

Winchmore Hill	07:22:30 30s~30s	07:37:30 30s~30s	07:52:30 30s~30s	08:07:30 30s~30s	08:22:30 30s~30s
Palmers Green	07:25:00 30s~30s	07:40:00 30s~30s	07:55:00 30s~30s	08:10:00 30s~30s	08:25:00 30s~30s
Bowes Park	07:27:30 30s~30s	07:42:30 30s~30s	07:57:30 30s~30s	08:12:30 30s~30s	08:27:30 30s~30s
Alexandra Palace	07:30:30 60s~60s	07:45:30 60s~60s	08:00:23 60s~60s	08:15:23 60s~60s	08:30:23 60s~60s
Hornsey	…	…	…	…	…
Harringay	…	…	…	…	…
Finsbury park	07:35:30 60s~75s	07:50:30 60s~75s	08:05:32 60s~75s	08:20:32 60s~75s	08:35:32 60s~75s
Drayton Park	07:38:30 60s~60s	07:53:30 60s~60s	08:08:33 60s~60s	08:23:33 60s~60s	08:38:33 60s~60s
Highbury & Islington	…	…	…	…	…
Essex Road	07:41:22 30s~30s	07:56:22 30s~30s	08:11:22 30s~30s	08:26:22 30s~30s	08:41:22 30s~30s
Old Street	07:45:30 60s~60s	08:00:30 60s~60s	08:15:30 60s~60s	08:30:30 60s~60s	08:45:30 60s~60s
Moor Gate (Arrive Time)	07:48:00	08:03:00	08:18:00	08:33:00	08:48:00

Timetable 2: Welwyn Garden City to Moorgate

	6	7	8	9	10	11
	S11/C1	**S24/C2**	**S37/C3**	**S13/B1**	**S25/B2**	**S36/B3**
	Class 91	Class 91	Class 91	Class 313	Class 313	Class 313
Welwyn Garden City	07:00:00	07:30:00	08:00:00	07:06:00	07:36:00	08:06:00
Hatfield	…	…	…	07:10:36 30s~30s	07:40:36 30s~30s	08:10:36 30s~30s
Welham Green	…	…	…	07:14:00 30s~30s	07:44:00 30s~30s	08:14:00 30s~30s
Brookmans Parks	…	…	…	07:16:30 30s~30s	07:46:30 30s~30s	08:16:30 30s~30s
Potters Bar	…	…	…	07:20:30 75s~90s	07:50:30 75s~90s	08:20:30 75s~90s
Hadley Wood	…	…	…	07:24:00 30s~30s	07:54:00 30s~30s	08:24:00 30s~30s
New Barnet	…	…	…	07:27:00 30s~60s	07:57:00 30s~60s	08:27:00 30s~60s
Oakleigh Park	…	…	…	07:29:30 30s~60s	07:59:30 30s~60s	08:29:30 30s~60s
New Southgate	…	…	…	07:32:45 30s~30s	08:02:45 30s~30s	08:32:45 30s~30s
Alexandra Palace	…	…	…	07:36:00 30s~30s	08:06:00 30s~30s	08:36:00 30s~30s
Hornsey	…	…	…	07:38:00 30s~30s	08:08:00 30s~30s	08:38:00 30s~30s
Harringay	…	…	…	07:40:15 30s~30s	08:10:15 30s~30s	08:40:15 30s~30s
Finsbury park	…	…	…	07:43:00 30s~60s	08:13:00 30s~60s	08:43:00 30s~60s

Drayton Park	…	…	…	07:45:30 30s~60s	08:15:30 30s~60s	08:45:30 30s~60s
Highbury & Islington	…	…	…	…	…	…
Essex Road	…	…	…	07:49:30 30s~30s	08:19:30 30s~30s	08:49:30 30s~30s
Old Street	…	…	…	07:53:00 30s~60s	08:23:00 30s~60s	08:53:00 30s~60s
Moor Gate	07:20:00	07:50:00	08:20:00	07:56:01	08:26:01	08:56:01

Timetable 3: Welwyn Garden City to Kings Cross

	12	**13**	**14**	**15**	**16**	**17**
	S16/F1	**S27/F2**	**S39/F3**	**S14/D1**	**S26/D2**	**S38/D3**
	Class 91	Class 91	Class 91	Class 317	Class 317	Class 317
Welwyn Garden City	07:05:00	07:35:00	08:05:00	07:10:00	07:40:00	08:10:00
Hatfield	…	…	…	07:15:00 30s~60s	07:45:00 30s~60s	08:15:00 30s~60s
Welham Green	…	…	…	…	…	…
Brookmans Parks	…	…	…	…	…	…
Potters Bar	…	…	…	07:23:15 30s~60s	07:53:15 30s~60s	08:23:15 30s~60s
Hadley Wood	…	…	…	…	…	…

New Barnet	…	…	…	…	…	…
Oakleigh Park	…	…	…	…	…	…
New Southgate	…	…	…	…	…	…
Alexandra Palace	…	…	…	07:38:30 30s~45s	08:08:30 30s~45s	08:38:30 30s~45s
Hornsey	…	…	…	…	…	…
Harringay	…	…	…	…	…	…
Finsbury park	…	…	…	07:45:10 30s~60s	08:15:10 30s~60s	08:45:10 30s~60s
Holloway	…	…	…	…	…	…
Belle Isle	…	…	…	07:49:30 60s~60s	08:19:30 60s~60s	08:49:30 60s~60s
Kings Cross	07:21:00	07:51:00	08:21:00	07:52:30	08:22:30	08:52:30

Timetable 4: Welwyn Garden City to Midland Road Junction

	18	**19**	**20**
	S15/E1	**S28/E2**	**S40/E3**
	TL 39	TL 39	TL 39
Welwyn Garden City	07:01:00	07:31:00	08:01:00
Hatfield	07:05:30 30s~30s	07:35:30 30s~30s	08:05:30 30s~30s
Welham Green	…	…	…
Brookmans Parks	…	…	…

Potters Bar	07:11:30 30s~60s	07:41:30 30s~60s	08:11:30 30s~60s
Hadley Wood	07:15:00 30s~30s	07:45:00 30s~30s	08:15:00 30s~30s
New Barnet	07:18:00 30s~60s	07:48:00 30s~60s	08:18:00 30s~60s
Oakleigh Park	07:20:00 30s~30s	07:50:00 30s~30s	08:20:00 30s~30s
New Southgate	07:23:00 30s~30s	07:53:00 30s~30s	08:23:00 30s~30s
Alexandra Palace	…	…	…
Hornsey	…	…	…
Harringay	…	…	…
Finsbury park	07:30:00 30s~60s	08:00:00 30s~60s	08:30:00 30s~60s
Holloway	…	…	…
Belle Isle	…	…	…
Midland Road Junction	07:34:00	08:04:00	08:34:00

Timetable 5: Kings Cross to Welwyn Garden City

	21	**22**	**23**	**24**	**25**	**26**
	S942/K1	**S29/K2**	**S43/K3**	**S20/H1**	**S30/H2**	**S42/H3**
	Class 91	Class 91	Class 91	Class 317	Class 317	Class 317
Kings Cross	07:00:00	07:30:00	08:00:00	07:03:00	07:33:00	08:03:00

Belle Isle	…	…	…	…	…	…
Holloway	…	…	…	…	…	…
Finsbury park	…	…	…	07:13:00 75s~90s	07:43:00 75s~90s	08:13:00 75s~90s
Harringay	…	…	…	…	…	…
Hornsey	…	…	…	…	…	…
Alexandra Palace	…	…	…	07:22:00 75s~90s	07:52:00 75s~90s	08:22:00 75s~90s
New Southgate	…	…	…	…	…	…
Oakleigh Park	…	…	…	…	…	…
New Barnet	…	…	…	…	…	…
Hadley Wood	…	…	…	…	…	…
Potters Bar	…	…	…	07:42:30 30s~60s	08:12:30 30s~60s	08:42:30 30s~60s
Brookmans Parks	…	…	…	…	…	…
Welham Green	...	...	...	…	…	…
Hatfield	…	…	…	07:52:30 30s~60s	08:22:30 30s~60s	08:52:30 30s~60s
Welwyn Garden City	07:18:00	07:48:00	08:18:00	07:56:00	08:26:00	08:56:00

Timetable 6: Moor Gate to Welwyn Garden City

	27	**28**	**29**

	S18/I1	S31/I2	S44/I3
	Class 313	Class 313	Class 313
Moor Gate	07:00:00	07:30:00	08:00:00
Old Street	07:02:30 30s~30s	07:32:30 30s~30s	08:02:30 30s~30s
Essex Road	07:06:00 30s~30s	07:36:00 30s~30s	08:06:00 30s~30s
Highbury & Islington	…	…	…
Drayton Park	07:09:00 30s~60s	07:39:00 30s~60s	08:09:00 30s~60s
Finsbury park	07:12:00 75s~90s	07:42:00 75s~90s	08:12:00 75s~90s
Harringay	07:15:00 30s~30s	07:45:00 30s~30s	08:15:00 30s~30s
Hornsey	07:16:45 30s~30s	07:46:47 30s~30s	08:16:47 30s~30s
Alexandra Palace	07:19:00 30s~60s	07:49:00 30s~60s	08:19:00 30s~60s
New Southgate	07:27:20 30s~30s	07:57:20 30s~30s	08:27:20 30s~30s
Oakleigh Park	07:31:00 30s~30s	08:01:00 30s~30s	08:31:00 30s~30s
New Barnet	07:33:00 30s~30s	08:03:00 30s~30s	08:33:00 30s~30s
Hadley Wood	07:36:00 30s~30s	08:06:00 30s~30s	08:36:00 30s~30s
Potters Bar	07:40:00 30s~60s	08:10:00 30s~60s	08:40:00 30s~60s

Brookmans Parks	07:43:00 30s~30s	08:13:00 30s~30s	08:43:00 30s~30s
Welham Green	07:45:30 30s~30s	08:15:30 30s~30s	08:45:30 30s~30s
Hatfield	07:49:30 30s~60s	08:19:30 30s~60s	08:49:30 30s~60s
Welwyn Garden City	07:53:00	08:23:01	08:53:01

Timetable 7: Moor Gate to Hertford

	30	**31**	**32**	**33**	**34**
	S19/J1	**S34/J2**	**S32/J3**	**S45/J4**	**S46/J5**
	Class 313	Class 313	Class 313	Class 313	Class 313
Moor Gate	07:05:00	07:20:00	07:35:00	07:50:00	08:05:00
Old Street	07:07:30 30s~30s	07:22:30 30s~30s	07:37:30 30s~30s	07:52:30 30s~30s	08:07:30 30s~30s
Essex Road	07:11:00 30s~30s	07:26:00 30s~30s	07:41:00 30s~30s	07:56:00 30s~30s	08:11:00 30s~30s
Highbury & Islington	…	…	…	…	…
Drayton Park	07:14:30 60s~60s	07:29:30 60s~60s	07:44:30 60s~60s	07:59:30 60s~60s	08:14:30 60s~60s
Finsbury park	07:18:00 75s~90s	07:33:00 75s~90s	07:48:00 75s~90s	08:03:00 75s~90s	08:18:00 75s~90s
Harringay	07:20:50 30s~30s	07:35:50 30s~30s	07:50:50 30s~30s	08:05:50 30s~30s	08:20:50 30s~30s
Hornsey	07:22:00 30s~30s	07:37:00 30s~30s	07:52:00 30s~30s	08:07:00 30s~30s	08:22:00 30s~30s

Alexandra Palace	07:25:10 30s~30s	07:40:10 30s~30s	07:55:10 30s~30s	08:10:10 30s~30s	08:25:10 30s~30s
Bowes Park	07:27:30 30s~30s	07:42:30 30s~30s	07:57:30 30s~30s	08:12:30 30s~30s	08:27:30 30s~30s
Palmers Green	07:29:40 30s~30s	07:44:40 30s~30s	07:59:40 30s~30s	08:14:40 30s~30s	08:29:40 30s~30s
Winchmore Hill	07:32:20 30s~30s	07:47:20 30s~30s	08:02:20 30s~30s	08:17:20 30s~30s	08:32:20 30s~30s
Grange Park	07:34:30	07:49:30	08:04:30	08:19:30	08:34:30
	30s~30s	30s~30s	30s~30s	30s~30s	30s~30s
Enfield Chase	07:37:00 60s~60s	07:52:00 60s~60s	08:07:00 60s~60s	08:22:00 60s~60s	08:37:00 60s~60s
Gordon Hill	07:40:30 60s~60s	07:55:30 60s~60s	08:10:30 60s~60s	08:25:30 60s~60s	08:40:30 60s~60s
Crews Hill	07:42:30 30s~30s	07:57:30 30s~30s	08:12:30 30s~30s	08:27:30 30s~30s	08:42:30 30s~30s
Cuffley	07:45:30 30s~30s	08:00:30 30s~30s	08:15:30 30s~30s	08:30:30 30s~30s	08:45:30 30s~30s
Bayford	07:51:30 30s~30s	08:06:30 30s~30s	08:21:30 30s~30s	08:36:30 30s~30s	08:51:30 30s~30s
Hertford	07:55:00	08:10:01	08:25:01	08:40:01	08:55:01

Timetable 8: Midland Road Junction to Welwyn Garden City

	35	**36**	**37**
	S17/G1	**S33/G2**	**S41/G3**
	TL 39	TL 39	TL 39
Midland Road Junction	07:02:00	07:32:00	08:02:00

Belle Isle	…	…	…
Holloway	…	…	…
Finsbury park	07:09:30 60s~210s	07:39:30 60s~210s	08:09:30 60s~210s
Harringay	…	…	…
Hornsey	…	…	…
Alexandra Palace	…	…	…
New Southgate	07:15:00 30s~30s	07:45:00 30s~30s	08:15:00 30s~30s
Oakleigh Park	07:18:00 30s~30s	07:48:00 30s~30s	08:18:00 30s~30s
New Barnet	07:20:30 60s~60s	07:50:30 60s~60s	08:20:30 60s~60s
Hadley Wood	07:23:10 30s~30s	07:53:10 30s~30s	08:23:10 30s~30s
Potters Bar	07:27:00 60s~60s	07:57:00 60s~60s	08:27:00 60s~60s
Brookmans Parks	…	…	…
Welham Green	…	…	…
Hatfield	07:33:30 60s~60s	08:03:30 60s~60s	08:33:30 60s~60s
Welwyn Garden City	07:37:00	08:07:00	08:37:00