DEVELOPMENT OF SIMULATION MODEL FOR EVALUATING OPERATIONAL PERFORMANCE OF RAILROAD NETWORKS

by

Saad Syed

Bachelor of Engineering, Ryerson University, Toronto, 2008

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Civil Engineering

Toronto, Ontario, Canada, 2011

DECLARATION

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

_____

Saad Syed

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Saad Syed

Development of Simulation Model for Evaluating

Operational Performance of Railroad Networks

By

Saad Syed

Master of Applied Science in Civil Engineering, 2011

Department of Civil Engineering, Ryerson University

ABSTRACT

Railroads move freight traffic on their network based on an overall operations plan that includes blocking, train formation, and train scheduling plans. The optimization of these operations over the entire network is integral to maximizing efficiency and minimizing costs. This thesis develops a simulation model for analyzing various operation plans of a railroad network along with guidelines for establishing a comprehensive operations plan. The objective is to move all freight on the network with minimal cost. With the model simulation and comparison of several operation plans can be performed to determine the 'best case' plan. The model implements a discrete state, deterministic simulation approach. The user-friendly software for implementation of the model was programmed in VBA and Excel. Application of the model is demonstrated using a hypothetical railroad network. The results show that the model is an effective tool in evaluating various scenarios and helping in determining the best plan.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Said Easa, for his guidance and support in developing this thesis. I would also like to thank the other members of the examining committee, Dr. Bhagwant Persaud and Dr. Kaamran Raahemifar for taking the time to review the thesis, as well as Dr. Ahmed El-Rabbany for chairing the examination.

I am grateful to Mr. Paul Kerry, Mr. Ray Dai, and Mr. Marc Waver of Canadian Pacific Railway for providing me with their thoughts, experience and details about railroads during my research. I would also like to express my gratitude to Miss Raji Sextus and Mr. Bernard James for their thoughts and comments on my thesis document.

In addition, I would like to thank the Transport Association of Canada for awarding me their Foundation Scholarship and a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada for the partial funding of my research.

DEDICATION

I would like to dedicate this thesis to my parents, grandparents, and siblings.

TABLE OF CONTENTS

Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Tables xi

List of Figures xiii

1. INTRODUCTION 1

    1.1. Background 1

    1.2. Problem Statement 5

    1.3. Objectives 6

    1.4. Thesis Outline 8

2. LITERATURE REVIEW 11

    2.1. General Freight Rail Information 11

        2.1.1. Freight Rail Transportation 11

        2.1.2. Freight Reliability and Flexibility 13

        2.1.3. Train Dispatching 17

    2.2. Capacity of Railroad Networks 18

        2.2.1. Rail Network Operations and Modeling 18

        2.2.2. Rail Network Planning 22

    2.3. Classification Yards 24

        2.3.1. Flat Yards 24

2.3.2. Hump Yards 27
2.3.3. Typical Rail Yard Operations 28
2.3.4. Modelling Classification Yards 30
2.4. Classification Methods 36
2.4.1. First In First Out Method 37
2.4.2. Priority Based Method 38
2.4.3. Terminal Operations Optimization 42
2.5. Integration of Yard Operations and Main Line Operations 42
2.6. Summary 44
3. RAILROAD NETWORK SYSTEMS 45
3.1. Railroad Network Models 46
3.1.1. Mathematical Optimization Models 46
3.1.2. Simulation Models 47
3.1.3. Comparison of Models 48
3.2. Classification Yards and Yard Operations 48
3.2.1. Individual Yard Capacity 50
3.2.2. System Effects on Yard Capacity 50
3.3. Mainline (Linehaul) Operations 51
3.3.1. Origin-Destination Travel Demands 52
3.3.2. Train Composition (Pull lists) 52
3.3.3. Train Routing 53
3.4. Network Element Interactions 53
3.4.1. System Elements 55

3.4.2. External Elements 55
3.5. Typical Model Components 56
3.5.1. Model Constraints 56
3.5.2. Model Objective Function(s) 57
3.5.3. Model Input Data 59
3.5.4. Model Output Data 61
3.5.5. Existing Routing Methods 62
3.6. Summary 63
4. MODEL DEVELOPMENT 65
4.1. Model and Guidelines Overview 65
4.2. Model Data Inputs 67
4.2.1. Example Network 68
4.2.2. Network Data 68
4.2.3. Yard Data 71
4.2.4. Route Data 72
4.2.5. Origin-Destination Traffic Data 73
4.2.6. Blocking Plan Data 73
4.3. Guidelines for Train Scheduling 75
4.3.1. Train Schedules in the Model 78
4.4. Proposed Block Assignment Method 80
4.4.1. Block to Train Assignment in the Model 80
4.5. Proposed Train Routing Method 84
4.6. Model Outputs 88

4.6.1. O-D Tables 88
4.6.2. Train Statistics 89
4.6.3. Yard Queue Determination 93
4.6.4. Yard Statistics 95
4.7. Analysis of Model Results 97
4.8. Model Capabilities 100
4.9. Model Limitations 101
4.10. Summary 102
5. MODEL VERIFICATION AND APPLICATION 104
5.1. Model Verification 104
5.1.1. Data Inputs 104
5.1.2. Outputs 107
5.1.3. Model Verification Summary 107
5.2. Model Application: Single Plan 109
5.2.1. Initial Process and Model Setup 110
5.2.2. User Intervention and Subsequent Iterations 115
5.2.3. Results and Analysis 115
5.3. Model Application: Multiple Plans 120
5.4. Summary 121
6. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS 122
6.1. Summary 122
6.2. Conclusions 123
6.3. Recommendations 124

REFERENCES 126

Appendix 1 – List of Definitions and Abbreviations 131

Appendix 2 – Data Inputs for Example Model Application 134

Appendix 3 – Model Application Results – First Iteration 138

Appendix 4 – Model Application Results – Final Iteration 144

Appendix 5 – Software Code 148

LIST OF TABLES

Table 2.1 Cost and Effect of Freight Rail Capital Investments (AASHTO, 2009) 13

Table 2.2 Car Cycle Times From Kwon et al (1995) 14

Table 2.3 Trip Time and Reliability Performance of Different Trains (Kwon et al. 1995) 15

Table 4.1 Example of Network Inputs 70

Table 4.2 Example of Yard Data Inputs 72

Table 4.3 Example of Route Inputs 74

Table 4.4 Example O-D Table for Daily Operations 74

Table 4.5 Example O-D Table for Weekly Operations 74

Table 4.6 Example of Blocking Plan 75

Table 4.7 Example of Train Schedule Input 79

Table 4.8 Example of Block to Train Assignments 84

Table 4.9 Example of Train Schedule Input 87

Table 4.10 Example Preliminary O-D Table 87

Table 4.11 Example Final O-D Table 87

Table 4.12 Example Post Simulation O-D Table 89

Table 4.13 Example of Train Statistics 94

Table 4.14 Example of Yard Statistics 96

Table 4.15 Example of Queuing at Yards 96

Table 4.16 Analysis Table 99

Table 5.1 Yard Inputs Settings (Model Verification) 105

Table 5.2 Available Routes (Model Verification) 105

Table 5.3 Original O-D Tables (Model Verification) 106

Table 5.4 Train Schedule (Model Verification) 106

Table 5.5 Static Blocks Input (Model Verification) 108

Table 5.6 Final O-D Tables (Model Verification) 108

Table 5.7 Yard Specific Outputs (Model Verification) 109

Table 5.8 Train Specific Outputs (Model Verification) 109

Table 5.9 Yard Data for Application 112

Table 5.10 Shortest Path Matrix 112

Table 5.11 Train Schedule (Model Application) 114

Table 5.12 Original O-D Table 117

Table 5.13 O-D Table – No User Intervention 117

Table 5.14 O-D Table – Post User Intervention 117

Table 5.15 Train Specific Results 118

Table 5.16 Yard Statistics 119

Table 5.17 Multiple Plan Assessment 120

LIST OF FIGURES

Figure 1.1 Overview of Thesis Structure 10

Figure 2.1 Flat Switching Diagram (Dirnberger J., 2006) 25

Figure 2.2 Kicking Cars Diagram (Dirnberger J., 2006) 26

Figure 2.3 Typical Operations' at a Classification Yard 28

Figure 2.4 Yard Processing Capacity VS Train Length (Kraft, 2002d) 33

Figure 2.5 Dwell Time VS Block Departures (Kraft, 2002d) 34

Figure 2.6 Dwell time VS Processing Capacity (Kraft, 2002d) 34

Figure 2.7 Three Links in Series (Kraft, 1998) 44

Figure 2.8 Three Links in Parallel (Kraft, 1998) 44

Figure 3.1 Example Network Railroad – Martinelli and Teng (1995) 54

Figure 3.2 Train Itineraries for Physical Route (1, 2, 4, 6) – Martinelli and Teng (1995) 54

Figure 4.1 Model Overview Flowchart 66

Figure 4.2 Example Railroad Network Map (http://www.cpr.ca) 69

Figure 4.3 Example Network 69

Figure 4.4 Process for Train Scheduling 76

Figure 4.5 Step 1 - Assignment of Direct and Pure Blocks 81

Figure 4.6 Step 2 - Assignment of Direct and Impure Blocks 82

Figure 4.7 Step 3 - Assignment of Unassigned Blocks 83

Figure 5.1 Network Configuration for Verification of Model 105

Figure 5.2 Network Topology and Yard Location 112

1. INTRODUCTION

Railroads move freight from location to location using fixed tracks and long trains to maximize economies of scale and efficiency. As can be expected, the overall costs of the infrastructure are quite high, whereas, comparatively, the operational costs are relatively low. Though this is the case, there is still always room for improvement in the efficiency and cost effectiveness of railroad operations since this may reduce costly infrastructure investments.

Railroads operate with four (4) major operating plans in place. These include the blocking plan, the train formation plan, the train schedule and the empty car distribution plan. This thesis is focused on three (3) of the major operating plans, which include the blocking plan, the train formation plan, and the train schedule. The blocking plan regulates the contents and the number of blocks (set of cars) whereas the train formation plan regulates which blocks make up each train and how, consequently, the traffic flows over the network. The train schedules and empty car distribution plans are typically created after the fact and often do not affect blocking or formation plans (Martinelli, 1996).

This thesis is focused on railroad operations and maximizing their efficiency by modeling and analyzing railroad operations including blocking, train formation and scheduling. Ultimately, the goal is to move all the freight on the network with the minimal costs averaged over the entire operation.

1.1. Background

The cost efficient and reliable movement of goods has always been a challenge in the freight industry. In terms of freight transportation there are a few major modes available for business in

the current market. In the United States of America, the American Association of State Highway and Transportation Officials (AASHTO) has standing committees for five of the following freight transportation modes: Standing Committee on Rail, Standing Committee on Water, Standing Committee on Aviation, Subcommittee on Highway Transport, and Special Committee on Intermodal Transportation and Economic Expansion. Additionally there is also the mode of pipeline freight transportation, but that has a very limited application.

Rail transportation competes primarily with highway transportation (short haul and long haul trucking) and marine transportation. The advantages of highway transportation which rail transportation does not often have, include door to door service, high reliability, and more economic short haul service (i.e. within a city).

Rail transportation is the movement of goods or people from point A to point B in trains which travel along a set path on a railway track. A train consists of a locomotive, individual cargo or passenger cars, and a caboose coupled together. The coupling process usually happens in rail yards. When it comes to freight trains, these rail yards are called Classification Yards (discussed in detail in Section 2.3).

A railroad network consists of railway tracks and various classification and intermodal yards. Each train travelling along a section of track is travelling from yard to yard in order to reach its final destination. Because a railway track is a fixed asset and trains may only travel in one of two directions along the track and there is no way to magically route trains to specific addresses or venues like one can with land based vehicles, it is very important for freight shippers (rail companies) to transport merchandise as close as possible to where it must be picked up (origin)

and dropped off (destination). This is often completed via the customer dropping their cargo off and picking it up at predetermined locations such as rail yards.

These yards are the hubs where railcars are directed or redirected after each leg of their journey in order for a specific piece of freight to get from its origin to its destination. In this operation the individual rail cars are often transferred from train to train in order to reach its final destination. This is analogous to how passengers travel from airport to airport often having to switch planes and depart on connecting flights to reach their final destinations (Liu, Ahuja, & Sahin, 2008). Similar to how aircraft passengers must obtain their own transportation to and from the airport, rail freight must also be transported to and from classification yards (or to specific pick up points).

It is common knowledge that railcars spend the majority of their time during trips at intermediate (between origin and destination) classification yards. This is because classification yards can only handle a certain number of cars for any given period of time and rely heavily on the skills and experience of their yardmasters (Innovative Scheduling, 2005). As such, it is important not only to limit the number of yards a railcar travels through, but it is even more important to make that rail yard more efficient and capable to handling higher loads.

A railway system consists of three essential elements (Pachl, 2002):

1. Infrastructure (tracks, signalling equipment, stations, and yards/terminals);
2. Rolling Stock (locomotives and cars); and
3. Operating rules and procedures.

The combination and the interaction of these three elements is what we know today as a railway system. The study of one or all of these elements with a goal to improve the efficiency of the

system will bring us closer to realizing the most efficient use of rail systems. This thesis will focus on operating rules and procedures.

A railway track is a structure which consists of parallel lines of rail which are held together by railway ties (sleepers) resting on a crushed stone ballast which helps to diffuse the weight of the above trains onto the subgrade below. The rails are often prefabricated in sections and tied together with either bolted or welded rail joints or butt welds between the rails. The main function of a rail is to provide a rigid surface and direction to the trains passing on the track above. Due to the immense weight and the lengths of the trains the tracks must provide very gradual horizontal and vertical adjustments throughout the length of the track. The rails also provide a medium through which electric currents travel for signalling purposes (Mundrey, 2005).

Railway tracks and their respective right of ways are generally owned by railroads such as Canadian Pacific Railway or Canadian National Railway. Since individual railroads own, operate and maintain their own sets of track there is typically no direct government agency involvement as there is in the maintenance of roads and highways. Passenger rail companies such as Via Rail Canada and Amtrak often use the existing infrastructure provided by the freight shippers and many times freight shippers use each other's infrastructure in order to meet customer needs. This is done through trackage rights agreements between the various corporations.

Rolling stock, as with tracks, are owned and operated by individual railroads and even individual shippers. More rolling stock means more flexibility but more operating costs in labour, maintenance, and storage. Operating rules and procedures vary between railroad corporations,

countries and continents all have various different rules and regulations which determine how they will operate a railroad.

There are three principal factors that contribute to railroad freight transportation efficiency (Dirnberger J. , 2006) and (Armstrong, 1990):

1. The low coefficient of friction between the steel wheel and rail means low rolling resistance which allows locomotives to power not only themselves but also additional loads in the form of cars;
2. A fixed guide-way for the movement of trains by a single operating crew such that the restriction of moving freight in single vehicles was removed; and
3. Infrastructure strong enough to support heavy loads over vast geographic areas in order to permit economies of scale.

The combination of these three factors allows railroads to spread throughout the country and create a cost effective means to transport large amounts of freight over the surface. Using trains, as opposed to single vehicles, to move goods along a fixed track is also very important in that it minimizes the need for large right of ways which waste enormous amounts of real estate (Armstrong, 1990).

1.2. Problem Statement

In order to be competitive in the freight transportation market, railroads need to maximize their economies of scale in a way that will allow them to pass on savings to their customers. In addition to cost, there are additional issues of reliability and flexibility which must also be addressed by railroads. These issues must be addressed by railroads in their operating plans.

Operating plans are used to guide railroad business at any given time by giving railroad workers a framework and a guideline for performing tasks. This means that train schedules, train contents and train movements must all be controlled by such a plan. In order to create a plan that is effective, efficient and reliable, a railroad has to look at many factors such as delivery time, service time for trains and cars, crew availability, business trends, etc. Once a plan has been created, it is not enough to say that it works, but each railroad must do what it can to optimize its operations such that "best case" plan is implemented and maximum economies of scale are achieved. Guidelines for the assessment of multiple, integrated operations plans in a common way are required in order to determine which plan is the best.

Existing railroads currently rely heavily on the skills and experience of veteran employees, who make decisions about train schedules, product development (blocking plans) and how they split up cars into the bowl (classification tracks). Additionally, current practices of railroad companies prevent priority traffic from getting special treatment and all traffic is treated in the same manner. Each train / car is served on a FIFO (first in-first out) basis at every yard it reaches.

When running simulations, railroads supply their software with various data which includes static blocking plans, train schedules, traffic files (OD data, car types, etc.) as well as network topology and geography. A shortest path algorithm is applied in order to route each individual car to its destination. The software will then assign each car to a specific predetermined block or set of blocks which are destined to travel from node to node. The blocks are, then, assigned to individual trains which carry them to their destinations.

For this system to work, different sections and groups within the railroad come up with the different strategies which make up the various operations plans. This means that the blocking

plan is created separately from the train schedule and the block to train assignment. With the model developed in this thesis, the back and forth of two separate groups working on individual plans is taken away. Instead, one analyst can develop, test and analyze an integrated operations plan with only traffic, network, and yard data, thus, taking away the lengthy back and forth required between multiple departments. This thesis addresses these issues with the model and guidelines developed.

1.3. Objectives

There are three main objectives for this thesis:

1. To create a model to assist in testing and analysing the operation plan(s);
2. To create guidelines for building integral portions of the operations plan; and
3. To create a user friendly software application to implement the model.

The first objective, to create a model to assist in the testing and analysis of operation plans, is accomplished by creating a simulation model which takes into account the various factors associated with railroads on a network level. The many inputs and outputs are focused on what a railroad analyst will actually use/need in order to make sound decisions with respect to the quality of an operating plan.

The second objective of this thesis is to create a set of guidelines for preparing data to be entered into the model. It is not enough just to have traffic and network data because train schedules, block to train assignments and blocking within each yard play major roles in the transportation of goods. This thesis will prepare step by step guidelines and/or provide examples of the aforementioned operations.

The third objective is to create a user friendly software application by which the model can be applied. This will be accomplished using Microsoft Excel and VBA as a programming platform. The software will make it simple to enter the various data requirements in an organized fashion and will provide the results of the model in a simple and easy to read manner. This will make analysis of an operations plan quite simple for the user.

1.4. Thesis Outline

Figure 1.1 provides a general overview for the thesis structure and research activities. This thesis is divided into six (6) main chapters as follows:

- **Chapter 1 - Introduction**: This chapter provides a brief introduction to freight rail and movement of goods. It also serves to represent the scope of research in this thesis.
- **Chapter 2 - Literature Review**: This chapter provides a comprehensive literature review of various railroad network systems and various types of models. The research conducted prior to selection of the proposed model type and the proposed model development is also represented here.
- **Chapter 3 –Railroad Network Systems:** This chapter provides a comprehensive review of specific railroad characteristics and various model objectives in existing research. It also provides an overview of model types used in railroad operations planning and provides a brief overview of existing model applications used in industry.
- **Chapter 4 – Model Development:** This chapter outlines the proposed model including inputs, outputs, and most importantly methodology. It also creates guidelines for how to use the proposed model. Model capabilities and limitations are presented here.

- **Chapter 5 – Software Application:** This chapter provides an overview of the software application which allows the user to easily apply the model. The model is validated using hypothetical data and then an example application of the model using hypothetical data is presented.
- **Chapter 6 – Summary, Conclusions and Recommendations:** This chapter provides a summary, conclusions and recommendations based on the research in this thesis. The recommendations proposed future study in this field applies to possible extensions of the model.

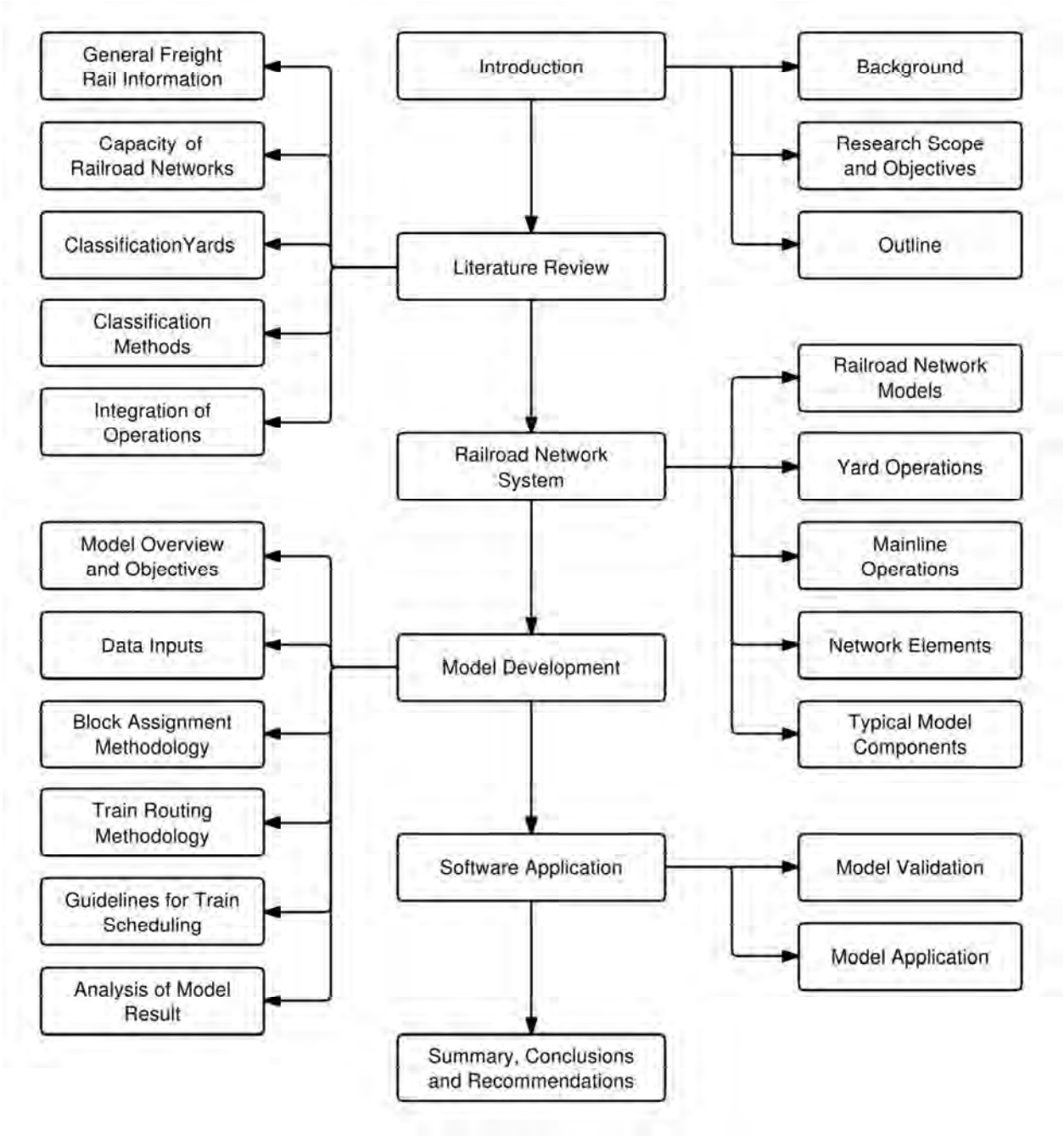

Figure 1.1 Overview of Thesis Structure

2. LITERATURE REVIEW

A literature review was conducted on the existing research in freight rail networks and classification yards. Peer reviewed journal papers, magazine articles, text books, government reports, and project reports were reviewed for this purpose. These sources have been grouped into several main categories and are summarized in this section.

2.1. General Freight Rail Information

This section discusses general freight rail characteristics and data with respect to Freight Rail Transportation, Reliability and Flexibility, and finally Train Dispatching. Freight Rail Transportation is discussed in terms of benefits, operational statistics and future potential investments. Reliability and Flexibility are defined and discussed in terms of railcar movements on class 1 railroads. Finally, Train Dispatching methods are discussed.

2.1.1. Freight Rail Transportation

There are significant benefits to Freight rail, not only to consumers but also the general public. These benefits include Transportation System Capacity and Highway Cost Savings, Economic Development and Productivity, International Trade Competitiveness, Environmental Health and Safety, and Emergency Response Capabilities (AASHTO, 2009). According to AASHTO and their 2009 Freight Rail report, there are currently only seven class 1 railroads in operation today (BNSF Railway, CSX Railroad, Grand Trunk Corporation, Kansas City Southern Railway, Norfolk Southern Railroad and Subsidiaries, Canadian Pacific Railroad, and Canadian National

Railroad). This is due to various consolidations and mergers which have occurred since 1980 when the railroads in America were deregulated.

AASHTO has also shown statistics regarding the operations of class 1 railroads (all data reflects the year of 2000). AASHTO points out (since deregulation) rail networks have been downsized significantly to only the "core" network but productivity has improved dramatically. Having said that, the costs to consumers has gone down and is lower than all other general freight transportation services such as marine or trucking. This is a significant improvement over the 1980's when the costs for freight rail were higher than all other surface/marine based operations. Even with all of these productivity gains lower consumer costs, the railroad freight transportation services are not able to rebuild their market share (lost to long haul trucking and short haul trucking).

According to the Railway Association of Canada in their 2009 publication of Railway Trends, over the past 10 years freight revenues per tonne have been steadily increasing. Although there has been a slight decline in carloads originated since 2005, the freight rail industry is still farther ahead of where it was in 1991 (The Railway Association of Canada, 2001). Though the freight industry is continuing to gain higher revenues per tonne, they are losing the overall fight against trucking in the freight transportation business. In having said that, it is also important to note that should issues such as service reliability and speed of delivery be addressed, rail may seem like a more fruitful mode of freight transportation to its customers.

AASHTO has also conducted a study of what the future of freight rail will be if there is no growth (limited investment), moderate growth (constrained investment), paced growth (moderate investment) and aggressive growth (significant investment). This is a study of what 20 years

(between 2000 and 2020) of investment into rail services would cost and what the impacts on shippers, highway users and highway costs would be, please refer to Table 2.1 to see the consolidated effects as provided in the AASHTO report. Because the onus to provide the bulk of the funding would be on railway corporations it is increasingly more important to find ways to raise capital for any future growth. This can be done in a few ways such as by borrowing money from banks, selling additional stocks, selling assets, optimizing the infrastructure in order to reap the additional benefits, or any combination of the list.

Many papers have discussed the analysis and the optimization of class 1 railroads by either improving the existing infrastructure, optimizing network flows, optimizing intermodal or regional yards or even by significant capital infrastructure growth.

2.1.2. Freight Reliability and Flexibility

Reliability in freight transportation is an extremely important concept as it is in any freight transportation network. This is the idea where freight is delivered to the appropriate delivery

Table 2.1 Cost and Effect of Freight Rail Capital Investments (AASHTO, 2009)

Scenario	Costs (In Billions of Dollars – $USD)			Impacts (In Billions of Dollars – $USD)			
	Total	Private Share	Public Share	Shippers	Highway Users	Highway Needs	Total
No Growth	105-156	82-102	23	326	492	21	839
Moderate Growth	145-165	122-142	23	162	238	10	410
Paced Growth	175-195	122-142	53	0	0	0	0
Aggressive Growth	205-225	122-142	83	-239	-397	-17	-653

point at the appropriate time (both agreed to upon signing the delivery contract between the shipper and the railroad company). It has been readily identified that reliability in rail is a very important issue by Kwon et al (1995) and Kraft (2002a) just to name a couple. This is even more important because of the high standards (98% reliability – as reported in Kraft (2002a)) set by the trucking industry.

Reliability of a freight rail in North America was studied by Kwon et al (1995) using class 1 railroad data collected from the Association of American Railroads (AAR's) Car Cycle Analysis System (CCAS). In this study the authors selected a sample of Rail Origin-Rail Destination (O-D) pairs within three major rail freight groups and typical car types: general merchandise train service (general boxcars), unit train service (covered hopper cars), and intermodal train service (double stack cars). The car cycle as described by the paper consisted of the time the first car is loaded until the time the empty car is returned to the original loader. This includes the aggregated total time where the car is being loaded, moved (loaded) on the mainline, sitting (loaded) in terminals, emptied by the receiver (consignee), moved (empty) on the mainline, sitting (empty) in terminals, and finally arrives empty at the shippers. Table 2.2 shows the results of the car cycle time for the three different types of cars/services – all time is measured in days.

Table 2.2 Car Cycle Times From Kwon et al (1995)

Note: (all data measured in days)	Boxcar	Covered Hopper	Double Stack
Loading Time	2.15	1.92	0.73
Loaded Time	8.77	5.33	3.21
Unloading Time	1.48	1.27	0.22
Empty Time	14.48	6.76	1.99
Total Cycle Time	26.88	15.27	6.15

In order to show trip time and reliability Kwon et al showed the mean trip time, standard deviation, and two other measures known as the *n*-day-percent and the maximum *n*-day percent. The *n*-day-percent is the measure of the probability that a car will arrive within a time window of *n*/2 days of the mean trip time. Because there is a skew in the trim time distributions, the maximum *n*-day-percent is the measure of the maximum probability that the car will arrive based on *n* days but not focused in the mean (i.e. the window of *n* days can move along the distribution in order to find the max). The results of the reliability study are presented in Table 2.3. It should be noted that since the lowest reliability is that of the general merchandise box car and coupled with the fact that unit trains do not typically spend much time in classification terminals, we can deduce an important link between terminal efficiency and rail car reliability. As reported in Kwon et al (1995), majority of trip time either empty or loaded is spent in terminals and that terminal and train delays account for more than 40% of all shipment delays. This is important because unreliable train service can be mitigated by improving the reliability and the efficiency of these operations.

Table 2.3 Trip Time and Reliability Performance of Different Trains (Kwon et al. 1995)

Elements	Box Car	Hopper Car	Double Stack
Number of O-D Pairs	477	102	20
Number of Railroads	2.11	1.47	n/a
Distance (miles)	788.1	831.0	n/a
Mean Trip Time(days)	7.16	5.25	2.53
Std Dev of Trip Time (days)	2.62	2.04	0.50
Maximum 1-day-%	32.42	41.90	89.2
Maximum 2-day-%	48.56	60.95	n/a
Maximum 3-day-%	61.07	73.21	n/a

Another very important function of a transportation network which affects reliability is known as flexibility. Flexibility is defined as:

> The ability of a system to adapt to external changes, while maintaining satisfactory system performance. System performance is characterized by parameters such as level of service, maintainability, and profitability. (Morlok & and Chang, 2004, p. 406).

External, uncontrollable, factors or variability could be as simple as poor weather conditions or something as complicated and devastating as a massive earthquake. Other factors could include traffic volume surges (based on economic factors), traffic flows (based on shifts in directional flows), mainline outages based on motor vehicle accidents, car damages, etc. (Morlok & and Chang, 2004) and (Dirnberger J. , 2006). Internal factors or variability should also be considered in this definition and these can include poor management of resources such as mismanaging crew allocation, misroutes and incorrect sorting at yards, rework, processing times at terminal operations, worker experience variations, etc. (Dirnberger J. , 2006). These factors all affect reliability and thus managing these factors appropriately by adapting to them while maintaining satisfactory system performance (the definition of maintaining flexibility) will help to maintain reliability for the network.

In addition to the above factors, there are also cost implications to having or not having certain flexibilities in a railroad operation. The following excerpt emphasizes this importance.

> Few railroads have fully embraced the concept of scheduled operations. While they recognise the value of adhering to the plan, they still wish to retain some of flexibility in their operations … Scheduling everything can lead to an increase in train starts and crew expense. (Kraft, 2002e, p. 19).

The above quote makes mention of the concept by which railroads have the flexibility built into their systems to be able to cancel or consolidate trains in an effort to save money. This is very important because exploiting this kind of flexibility is one way a railroad can offer better service to its investors, who also have a vested interest in not only the reliability but the overall economy of their railroads.

2.1.3. Train Dispatching

Freight trains are dispatched either on a tonnage-based system or a scheduled system. The tonnage based approach is an age old classic in which railroads hold all trains until they have enough freight to fill the train to capacity. In this format, railroads maximize their train utilization. Often times, using this approach many trains are either delayed or even cancelled and thus minimizes the total number of trains operated. Serious drawbacks for the tonnage-based approach include (Ireland P. e., 2004):

1. The yards cannot fine-tune their operations based on a repetitive schedule, and they require more railcars and greater storage capacity to cope with the traffic variability;
2. Demands for crew and locomotive resources may increase along with the costs for repositioning crews and equipment; and
3. Most importantly, customers suffer from unreliable service because the railroad gives train-operation economics priority over customer needs.

A very different approach, the scheduled option, requires a disciplined corporate structure and high levels of pre-planning. Serious drawbacks for the scheduled-based approach include (Ireland P. e., 2004):

1. They require operating trains with low tonnage when customer demand is below expectations;
2. They depend on railways' systematically forecasting traffic levels by the day of the week, and quickly adjusting the plan;
3. They require a granular, actionable understanding of each customer's requirements in each corridor; and
4. The needed schedule-based models require sophisticated operations research software to conduct comprehensive and timely analyses of different alternatives.

In selecting which operating style a railroad will use, the above drawback factors for either the tonnage-based or scheduled-based approach must be taken into account.

2.2. Capacity of Railroad Networks

The capacity of rail networks can be determined by design of said networks and the operation practices on them. The following is a review of some of the previous research in modeling, planning and operations of a railway network.

2.2.1. Rail Network Operations and Modeling

Looking at the metro line capacity (number of passengers) in Athens, Greece, Ballis et al (2004) used a simulation model which worked in four separate phases. Phase one involved the assumptions about train characteristics (power, shape, number of cars, car capacity, etc.). Phase two involved description and assumptions of the stations and track geometry; this allowed the model to trace train movements in reference to a zero point (a start point). Phase three involved calculations of the train driving diagrams which used kinetic equations and the coasting method,

whereby a train accelerates and once at a specific velocity the train continues to move, coasting, until required to stop. The kinetic equations take into account the general velocity equations which include concessions for weight and also include concessions for the track geometry (horizontal and vertical curves). Phase four involved the actual simulation and graphical representation of the model in hopes of creating a timetable for trains through various blocks (sets of tracks between sets of stations). It was in phase four where boarding and alighting times for passengers and turnaround times for trains were also incorporated into the model and the model was created to allow user intervention and adjustment to things such as interruptions, slow down and speed up trains or even immobilize or reactivate lines to see the effect of such behaviour on the capacity of the lines.

Although this model cannot specifically be applied to freight railways, it does show a very simple framework which can be applied to specific rail lines in order to determine the maximum capacity of a line. This could be based on number of cars (shipments), number of locomotives, number of yards/hubs and track geometry in order to determine how many freight trains can be accommodated. This is a far more difficult task than with a metro line, where number of passengers can be somewhat accurately determined, while freight varies not only in how much is being shipped regularly but also how much can be processed in any given yard. There are already specific analysis models (discussed later in this section) which show how many cars a yard can accommodate based on various inputs and yard types.

Another railroad capacity model by Ramsey et al (1986) used computer software to draw space-time diagrams in a simulation which would plot the diagrams in front of the user. The simulation could be stopped at any time by the user in order to backtrack and readjust the decisions made by the computer model to determine different results. Though this model is very simplistic, it uses

three modules: a data entry module, a train performance calculator, and a straight rail capacity model which helps to provide a fairly accurate and timely simulation of rail line capacity. This model was comparable to existing, more complex models at the time.

A recent mathematical programming model by Higgins et al (1995) was created to identify train schedules on single lines using a branch and bound procedure. This model took into account train velocities, train priorities, line segment lengths with overtaking points (such as double track sections and siding locations), and times of scheduled stops as required. The objective function of this model was to minimize the delay of trains arriving at a destination as well as the operating costs of each train (i.e. costs for fuel, crew, power etc.). Since single line rail schedule modeling has been fairly extensively studied in the past, this paper primarily works to improve the solution time of a single line train scheduling problem.

Another way to look at the capacity of railroads is not to focus on the number of trains and cars that pass through a single line or a terminal, but to look at the amount of freight which can be carried, in terms of overall system capacity. This means that instead of analyzing at specific segments of rail links or terminals/yards, a model which addresses the operational measure of system capacity as a whole had to be created. This was done by Morlok and Riddle (1999) where they used a linear mathematical programming model to determine an estimate of the overall system capacity based on origin-destination. This model included the limitations of capacity at facilities, limited resources in the vehicle fleet, labour, and fuel, as well as environmental regulations and even management structure. This model, called MAXCAP, makes it possible to estimate overall system capacity based on tonnes per year or similar units. A variation of this model, called ADDCAP, can also determine where capacity can be added in the various components of the model, such as fleet size or terminal capacities. The authors also suggest that

this model could be used to assess the effects of regional trade agreements where origin-destination pairs for various trips vary, the effect of technological changes in transportation, and even the effect of natural disasters or other catastrophes.

Morlok and Chang (2004) expanded on the capabilities of MAXCAP by assessing both reserve capacity and by assessing the sensitivity of a transportation system to changes in traffic patterns using a new model deemed ADDVOL. The former was a conservative method of reserve capacity estimation, whereas the latter allowed the authors to more accurately determine the flexibility of demand variations on transportation systems. Flexibility is discussed later in this text. The MAXCAP model is used to determine reserve capacity of a system, but the application is limited to the use of only base traffic patterns (fixed traffic patterns). The ADDVOL model uses adjusted traffic patterns which allow for a more flexible system that gives a more robust estimate. Flexibility is a very important concept and is required in order to recover after any given event and more importantly, flexibility is more important than speed in terms of overall reliability (Judge, 2002).

In their 2006 study titled Estimating Freight Transportation System Capacity, Flexibility, and Degraded-Condition Performance, Sun et al expanded on Morlok's original model (MAXCAP) by applying general characteristics of existing transportation network models to MAXCAP. The authors pursued an idea that the absolute physical capacity of a network may not be reached because the practical limits of the network due to the degradation of service quality (for example level of service) becomes unacceptable. The authors complete their objective by adding three main functions to the model:

1. Reflecting uncertainty in traffic patterns by use of Dirichlet distribution;

2. Incorporating volume-delay functions and service quality constraints; and
3. Assigning flows to paths using stochastic equilibrium models thus replacing the predefined paths as set in MAXCAP originally.

This enhanced model becomes a nonlinear programming problem which must be solved using a heuristic step-sized search method as developed by the authors. This is different from the simple linear model originally proposed by Morlok, but it does prove to increase the overall levels of confidence of the models results.

2.2.2. Rail Network Planning

The most important part of the paper by Higgins et al (1995) was finding a solution to the optimal siding location problem while considering variable train velocities and non-uniform departure times. The idea is not to change existing sidings or specific fixed sidings (fixed due to various operational or regulation reasons), but to determine where best to place sidings in order to resolve potential train conflicts (such as crossings or overtaking). This model takes into account the cost parameters of delayed trains, operating costs, and upper speed limits at fixed intervals on the track. The model assumes a weekly cyclical train schedule in which scheduled stops are only permitted on fixed sidings. It is necessary for the model to solve for three sets of variables which include track segment lengths, arrival and departure times and binary conflict resolution. The model is solved by decomposing the original problem into two sub-problems (one which solves the track segment lengths and arrival and departure time variables, and the other which solves the binary conflict resolution variable) which are then solved using an iterative process until there is no more improvement for either. The paper also shows an example

of how using optimal siding locations can help to minimize train delays due to conflicts on single lines.

Mathematical programming was used by Liu, Ahuja & Sahin (2008) in figuring out a very top level scientific approach to optimal network configuration (i.e. yard locations, line directions etc.). This paper is of importance because it addresses what the newly formed and consolidated class 1 railroads may be able to do in the future to optimize and expand their lines based on a mathematical optimization using the greedy approach. In this study Liu et al looked at which yards could be shut down with minimal impacts, if and where new hubs should be opened, and where specific line or hub capacity expansions should be created. This study also looked at the costs associated with the theorized improvements.

The study looked at the yard and the capacity problems separately creating optimization problems for both and solving them using real data from an existing class 1 railway. In the yard optimization model they included drop, add, and a pair-exchange algorithms which allowed for the specific what-if analysis of existing facilities. The capacity expansion problem included three separate items, the blocking capacity, the car handling capacity and the line capacity. If certain yards would be added or removed or certain capacity was to be increased then overall costs could go down according to the study, and as such, the optimization of a network is increased. This is a very capital intensive program however, so the reality of such implementation is quite scarce, but this top level scientific assessment is a great tool for railway companies to review when deciding on how to conduct their operations in the future.

2.3. Classification Yards

The objective of a freight rail classification yard is to sort or classify cars from an inbound train into blocks (single length of cars which all have similar characteristics such as destination, type of cars, or even type of consist) which are then assembled into various outbound trains. This is an important and necessary step because trains typically contain various cars moving from different start points to different endpoints. It is rare to see a train which starts and ends at a given location without the need to be split apart or added to in order to maximize train capacity (with the exception of unit trains which are typically able to bypass the classification process entirely).

There are two general types of classification yard: flat yards and hump yards. They both provide the same function of classifying cars from inbound trains into new outbound trains but how they get from inbound to outbound are two entirely different processes.

2.3.1. Flat Yards

A flat yard is a very easy to design and construct because it can be done on a level grade and only requires an inbound line, an outbound line, and a classification area with a set of classification tracks. This makes a flat yard very inexpensive to construct and operate. In the case of a flat yard, switching locomotives take cars from the receiving track and then push and/or pull them to their various classification tracks. The process is explained with the assistance of Figure 2.1 (A, B, and C):

2.1.A. A brakeman typically rides the leading car to ensure there are no obstacles to line switches as the movement proceeds;

2.1.B. The brakeman steps down near the point where cars are about to be uncoupled. When the cars have been placed and the train stops, the brakeman uncouples the cars; and

2.1.C. The Switch engine reverses, leaving the car(s) on the desired classification track.

The actual process is conducted by a method called Drilling. A switch engine begins by accelerating quickly pushing a cut of cars. The engine then hits the brakes quickly, thereby "kicking," the car(s) toward their respective classification tracks where the car(s) either stop due to friction or by hitting the existing cars already standing on the track. The process for kicking cars is described using Figure 2.2 (A, B, and C).

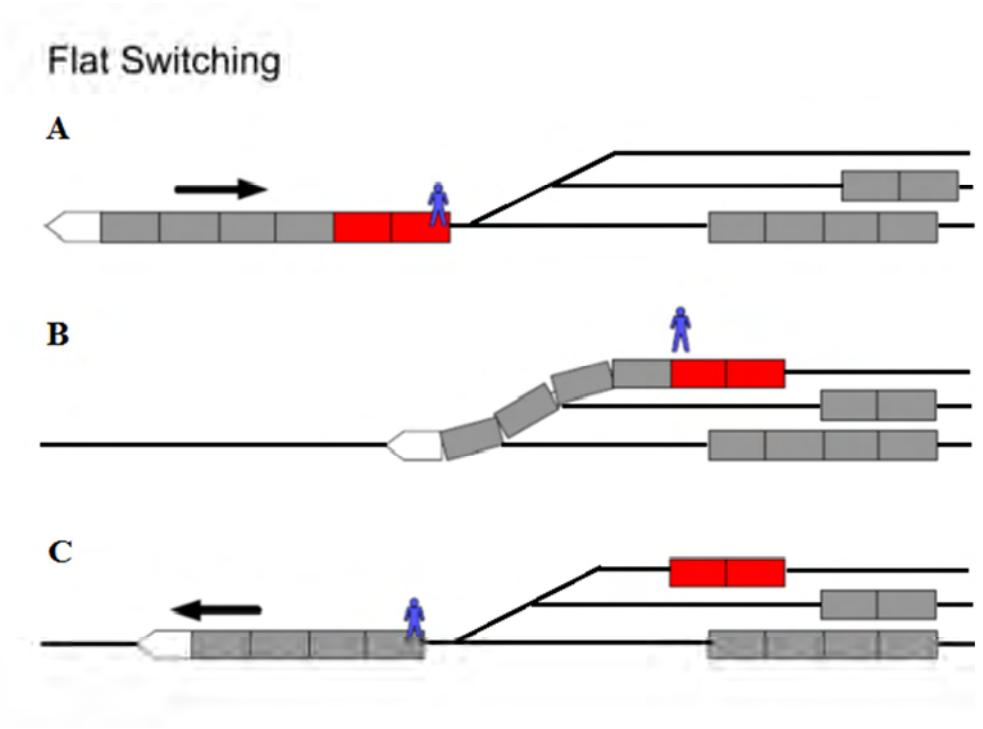

Figure 2.1 Flat Switching Diagram (Dirnberger J., 2006)

2.2.A. First the air brakes are released from all cars. The locomotive pushes the cars towards the classification tracks while the brakeman lifts the uncoupling lever and signals the engineer to “kick” the cars;

2.2.B. The engine accelerates rapidly while the brakeman runs beside cars holding the uncoupling lever until the desired speed is reached. Once this occurs, the brakeman signals the engineer to apply the locomotive brakes leaving the uncoupled cars to continue rolling forwards toward the class tracks. A second brakeman lines the switches to route cars onto the desired class tracks.

2.2.C. This process continues until the locomotive and attached cars have run out of space on the lead track at which time the locomotive must pull the cars back to have enough room to continue “kicking” cars.

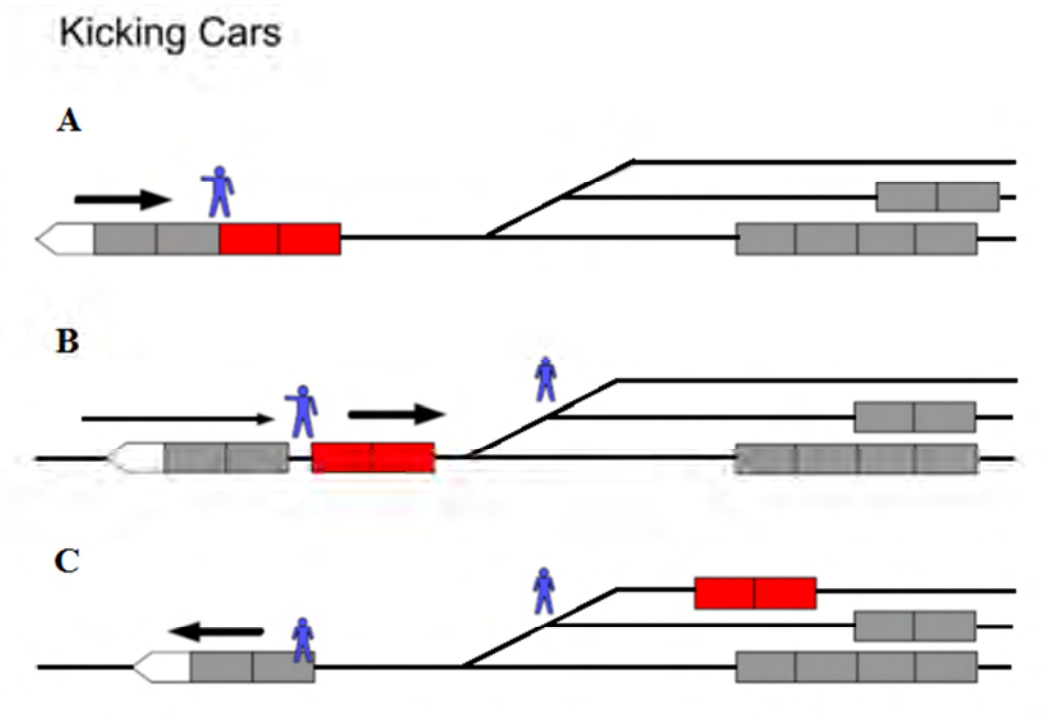

Figure 2.2 Kicking Cars Diagram (Dirnberger J., 2006)

Due to the large amount of manual labour involved, if a flat yard was classifying single cars all the time, it would be a very cumbersome, inefficient and labour intensive process. Therefore flat yards are best used when switching or classifying cuts of cars or blocks of cars from trains which stay together throughout the switching process. It should also be noted that cars are subject to additional handling in flat yards, especially if they are closer to the engine because the engine periodically has to back up and build up track space between it and the classification tracks such that the kicking process may proceed (Dirnberger J. , 2006). The process is also very damaging to the cars and the cargo because of all the impacts which occur due to the kicking and consequently the smashing into each other.

2.3.2. Hump Yards

Hump yards differ from flat yards because they use gravity, instead of drilling in order to help classify individual cars. Hump yards are typically used by class 1 railroads for their major classification hubs; such is the case for Canadian Pacific Railroads (CPR) Toronto Yard in the Agincourt Borough of Toronto, Ontario. This is because hump yards are largely more efficient than typical flat yards at classifying large amounts of individual cars. This efficiency is extremely important when there are a large number of incoming and outbound trains with multidiscipline cars with varying destinations. In a hump yard a hump locomotive will push a set of cars over a hump, releasing each car individually at the crest of the hump and allowing gravity to let the car slide down to the bowl. The cars are slowed down by mechanical retarders (brakes) that prevent the cars from smashing into standing cars on the classification tracks at high speeds, thus minimizing damage to couplers, cars and cargo. This is vastly more efficient than flat yards not only in terms of energy use but also staffing since no additional personnel are required for braking and uncoupling cars individually while they go over the hump.

2.3.3. Typical Rail Yard Operations

In a process similar to both types of yards, once all of the cars are in their respective tracks, they are ready to be released for outbound trains. This is where (in both types of yards) trim engines conduct the pull-down process where they take all blocks for an outbound train and string them together to create said outbound train. There are four major operations at any given classification yard. These operations happen sequentially and the next operation depends on the prior operation. Typical operations at a classification yard can be grouped into the following (also see Figure 2.3):

1. Train Receiving and Inbound Inspection;
2. Classification or Sorting of Cars onto Classification Tracks;
3. Train Marshalling or Assembly; and
4. Outbound Inspection and Train Departure.

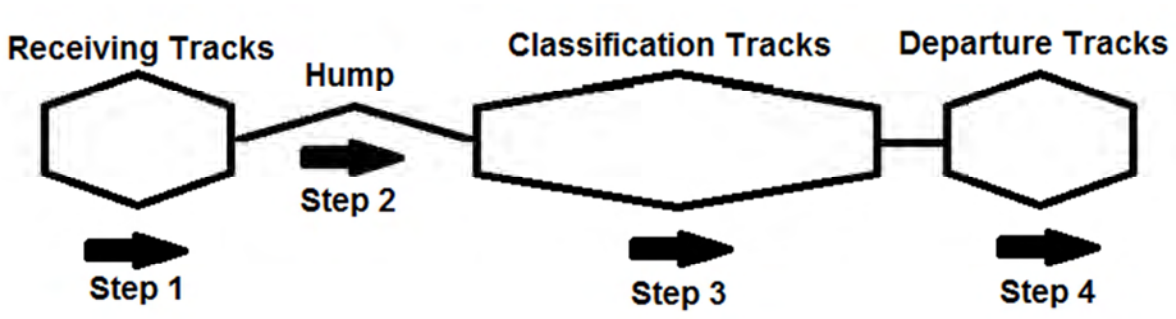

Figure 2.3 Typical Operations' at a Classification Yard

In addition to the above noted tasks, there are many tangentially relevant facilities within a typical rail yard which do not directly affect classification (although they can indirectly) such as intermodal facilities, shop facilities, short term storage facilities, etc. (Innovative Scheduling, 2005).

During the receiving the inbound inspection is comprised of checking the rail cars for structural integrity, safety, as well as the consist of each car. During this time period in a hump yard the yard crews will turn off the air brakes so that cars become free rolling (this way they can be pushed or pulled as required) and the cars will then be put into queue so that they can be classified. In a flat yard, the inbound cars will be uncoupled and split up only at the cut limits and the air brakes will only be bled if individual cars are being classified.

The cars are then sorted onto various classification tracks which act as temporary storage for use while building a new train or a portion of a new train. This sorting, or classification, is conducted in one of two ways: by use gravity such as in hump yards or by use of switching locomotives in flat yards as described above.

Once the cars have been sorted onto the classification tracks into their new yard assignments and the cut off time for the departing train has come (time at which the train must be built for departure) the cars are built into an outbound train at the trim end of the yard, inspected as they were during receiving, and released to their next destination after a brake test. During this step, the trim locomotives also perform a process where they pull all of the cars (those which are not going to leave on the outbound train) on the classification track to the very end of the classification tracks thus leaving room behind them for additional classification of new cars.

2.3.4. Modelling Classification Yards

Because a railcar in a yard goes through four major operations, one after the other, they are naturally thought of as cars in a queue. Thus significant works in the past by authors such as Petersen (1977), Turnquist and Daskin (1982), Martland (1982), and Kraft (1990's – 2000's) have been completed in order to understand the delays in yards.

In 1977, Petersen used queuing theory to create an analytic model of the operations inside a classification yard. The model was described in two papers Railyard Modeling: Part I - Prediction of Put-Through Time and Railyard Modeling: Part II - The effect of Yard Facilities on Congestion. The model determines the probability distribution of the time which any given rail car will spend inside a yard. This model found difficulty in determining service time distributions for the classification and the assembly operations of the yard but found that exponential ones worked. The model further simplified the operations within the yard by assuming the inter-departure times (the time between scheduled departing trains) for each train, constant. According to the results of this simplified model, the major delays inside a yard are caused by the classification and assembly processes. This makes sense due to the setup of most yards where there are far fewer inbound/outbound tracks than there are classification tracks. The inbound trains wait to be classified and the outbound trains wait to be assembled, while these trains wait, the classification and assembly processes typically continue (either one after the other or in parallel). Though this model did not work to optimize the work being completed in the yard, it was one of the first to help understand the delays and the processes in a yard.

Turnquist and Daskin (1982) analyse yard operations from the rail cars point of view instead of a train's point of view. They build directly on the Petersen models and in this research they found

that possible causes for delays in yards are often conflicting thus a balance of different train dispatching strategies must be taken into account to create a best case scenario. Through study of this model Turnquist and Daskin determined that variance in train length causes significant classification delays thus implying that constant train lengths may be required to minimize classification delays. However, their connection delay model postulates that running trains on anything but a strict schedule would cause variable delays because trains would just sit there and wait until they were full prior to leaving. These provide conflicting views, because dispatching trains only when they reach a specific number of cars will greatly change the scheduled train departures or vice versa, departing trains at only the scheduled times will mean that a train may have a random number of cars on it at the time of departure. It was through the study of this model that Turnquist and Daskin determined that the solution to optimizing the yard delay problem lay somewhere in between steady scheduled trains and constant trains lengths. The major benefit of this model was that this information could be found without the need to tiresome simulation work as was conducted in the past. Again, because the models were simplified significantly in order to make them work efficiently, the authors urge that results of the models be used only during preliminary analysis and as a guidance tool.

The PMAKE model (Martland, 1982) was created over a 10 year period by MIT in conjunction with the Federal Railroad Administration. After a review of both simulation and queuing models the author claimed that the PMAKE model is cheaper and easier to use than simulation or queuing models because it is faster and more realistic. In addition, after a review of both the Petersen and Turnquist models (above) Martland states:

> Queuing models have not yet been used to any great extent by railroads. Part of the reason for this ... may be that the basic assumptions of the queuing approach do not apply

> in rail yard operations. Arrivals at a yard are not random, but exhibit daily, weekly, and seasonal cycles that are well known to railroad operating officials. The service time distribution is not constant, but varies as a railroad changes the number of switch crews working at each part of the yard, adjusts maintenance activities, adjusts the shifts operated, and changes the relative priorities of switching activities. The system, therefore, is not in a steady state as required for the use of steady state queuing models. The number of servers and the service rate are both adjusted continuously to reflect the build up of queues at various parts of the yard. At most, the system is in a steady state only to the extent that the expected queue length, arrival distribution, and service rate follow a weekly cycle: the expected situation a week from now is the same as the expected situation N weeks from now at the same time of day. (Martland, 1982).

In order to move away from the steady state assumptions made by the queuing models and the cumbersome approach of simulation models, PMAKE aimed to determine the probability of cars making their connections in the time available. This is done so by taking cut-off times (inbound, outbound, or both) for trains into account. PMAKE is able to predict the impact of schedule changes on yard performance in terms of yard times and potential delays. PMAKE can also take other variables into account such as connection times, traffic priority, yard traffic volume, pattern of car flows in a yard, the pattern or reliability of train arrivals in a day, and power availability. PMAKE functions as developed by MIT are relatively easy to calibrate (can be done so using various approaches) and use for analysis of yard performance. Thus, PMAKE is a system which has been more commonly used than other models such as queuing or simulation in operations and service planning by railroads (Martland, 1982).

In his 2002 Trains magazine article (Kraft 2002c, 2002d), Kraft describes how well a yard operates is dependant primarily on a railroads operating strategy. He also notes that there are three major descriptive formulas and plotted graphs which help to describe how well a yard processes cars, what affects the dwell time in a yard and what affects the efficiency of a yard.

For yard processing capacity, the number of trains a yard may originate and the length of such trains are the key factors. He states “As train lengths increases, train departures will decrease drastically unless the yards processing capacity increases at a faster rate. Train departure frequency in turn determines how long cars sit in yards.” Refer to Figure 2.4 for the graph and the Equation 2.1 which corresponds to the graph.

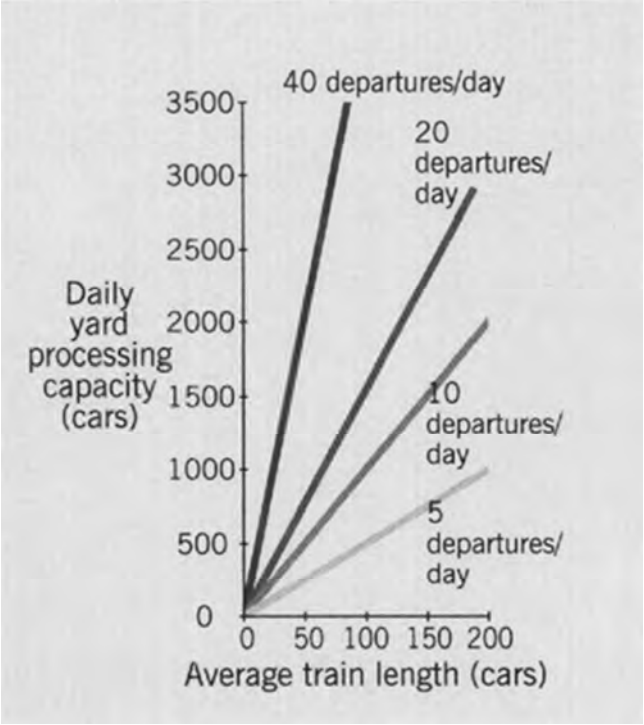

Figure 2.4 Yard Processing Capacity VS Train Length (Kraft, 2002d)

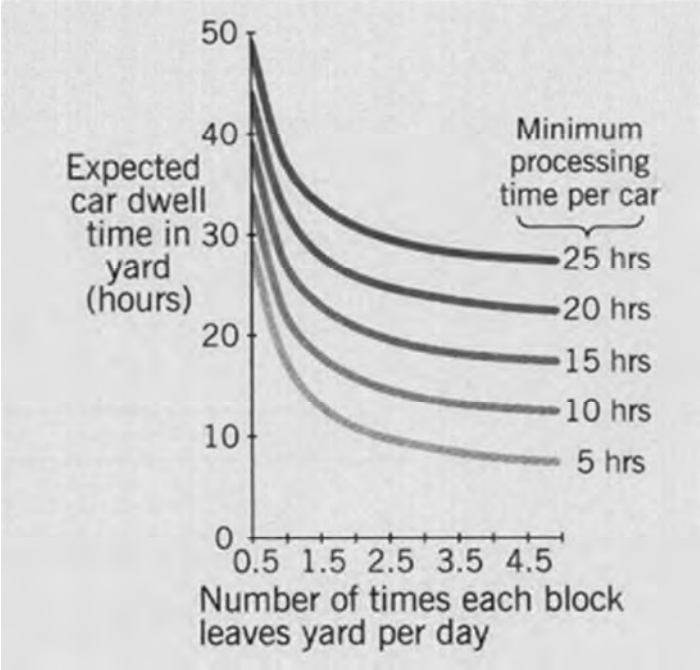

Figure 2.5 Dwell Time VS Block Departures (Kraft, 2002d)

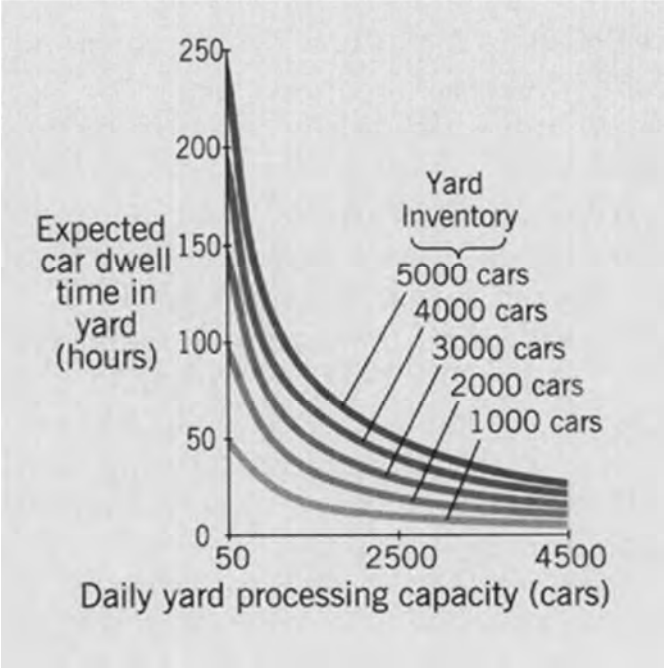

Figure 2.6 Dwell time VS Processing Capacity (Kraft, 2002d)

$$NUMTrains = CAPYard / LTrain \quad (2.1)$$

where,

NUM_{Trains} = (Train departures per day) (# of Trains)

CAP_{Yard} = Daily yard processing capacity (# of Cars)

L_{Train} = Average train length (# of Cars)

For dwell time in yards, the processing times and the frequency of departures per day are the most important factors. Kraft states “Once block departure frequency increases much past 1.5 times per day, its effect on car dwell time approaches zero.” Thus, only a faster yard processing time (which includes inbound inspection, classification, and assembly of outbound trains) could have a hope in lowering dwell time. The graph presented in Figure 2.5 takes a best case scenario where no additional delays (such as limited yard resources, overall yard capacity, directional processing, etc.) are taken into consideration. Equation 2.2 represents the graph.

$$T_{Dwell} = (T_{Process} + 12) / BLOCK_{Departures} \quad (2.2)$$

where,

T_{Dwell} = Yard Dwell Time (hours)

$T_{Process}$ = Minimum processing time per car (hours)

$BLOCK_{Departures}$ = # of times each block departs per day (#)

Another problem proposed by Kraft is that dwell time is farther aggravated by destroying its own processing efficiency. This means that with more cars dwelling in the yards inventory the more congestion there is, thus creating a slowdown for the processing of inbound trains. This is a

viscous cycle which eventually could lead to the shutting down of the classification process until all cars are removed from the bowl and dispatched on outbound trains. This phenomenon is represented in Figure 2.6 where longer dwell times drastically decrease the yard capacity and by Equation 2.3 below.

$$T_{Dwell} = NUM_{Cars} / PROC_{Cars} \quad (2.3)$$

where,

NUM_{Cars} = Yard inventory (# of cars)

$PROC_{Cars}$ = Cars processed per day (# of cars).

2.4. Classification Methods

As Kraft explains (The Yard: Railroading's Hiddent Half (Part 1), 2002c), there are two extremes of blocking practices in railroading, Extreme Deferment and Extreme Anticipation. Extreme Deferment is a “bump-along” system where no classification of a car occurs until the car is nearest its final destination. Extreme Anticipation is the opposite, where cars are classified at the first possible yard in order to create and arrange blocks in the order of where they are going. There are also varying possibilities for yard classification practices in between the two extremes. Using hump yards and flat yards together, railroads are now able to conduct what is called support blocking. This is a scenario where pre-sorting and pre-blocking of individual cars can be completed using a hump yard such that when the blocks reach a flat yard, there is no individual car to classify, only blocks. This practice allows railroads to realize cost savings by utilizing each type of yard to its most efficient manner.

2.4.1. First In First Out Method

Various strategies exist for classifying rail cars within a yard. There is the common first in, first out (FIFO) method which allows the cars on the first inbound train to be classified into their new respective blocks and shipped out on the first outbound train. This is an example of an extreme anticipation style method which classifies the cars as soon as they enter each yard. A few other methods have been proposed, such as the priority based classification method and the dynamic classification system. Both are great improvements over FIFO because they take train length and capacity or shippers schedules and delivery times into account prior to sorting cars.

Currently, rail yards build car trip plans in two steps. The first step involves assigning a "yard block" or classification code to each car based typically on its ultimate destination and other various car characteristics. This is done by a computer which uses lookup tables to determine which yard the car must go to next in order to reach its final destination (Kraft, 2000b). Step two would involve determining which trains can carry the classified cars to their next destination. Though car scheduling systems can accomplish this task, it is often disregarded due to schedule fluctuations in the trains and thus cars are usually shipped out on the first available outbound train in the order the cars were classified in. This is a flawed system in that it does not consider car delivery times or specific train capacity and train schedules. If there are cars which have a significant amount of slack time in their delivery deadline, this is not an issue. However, if there are cars which have very sensitive delivery times, or their delivery times are time critical, they must be "cherry picked" in order to be put on the next outbound train. This means that a crew must manually extract the car from the bowl after classification by backing the switching engine into the classification yards, uncoupling the cars ahead of it and removing the priority car onto a different siding. Then they must return the unwanted cars back on to the tracks and continue with

their outbound train assembly. This whole process is very costly in both time and resources, but should this method not be utilized, there are potential delivery delay losses which can also occur hurting both the freight shipper and the rail corporation.

2.4.2. Priority Based Method

An excellent strategy to help mitigate the need to cherry pick high priority cars is called priority–based classification as discussed in Kraft's 2002 paper *Priority-Based Classification for Improving Connection Reliability in Railroad Yards- Part I: Integration with Car Scheduling.* It is very important, from a business standpoint, for carriers to provide reliable, on time services to their clients. In this front, the rail companies have failed and thus have lost a great deal of their market share in freight shipping to short haul and long haul trucking. Kraft mentions that when trains are run on a scheduled system, they help in producing an environment where terminal efficiency can be improved (because the inbound and outbound trains, theoretically, will be scheduled to arrive or depart from the terminal in a way that mitigates overcrowding), but this cannot change the terminal operations by itself. As has been described already, most car scheduling systems create what is known as a "yard block," which shows where each car will end up along its journey. This does not take into account which train the cars will go on, and if that were taken into account along with the yard information, this would be known as a "train block." The idea behind the priority based switching is that cars, instead of being cherry picked, should be scheduled onto specific trains as well as yards based on priority. The cars can be then classified onto different tracks based on: 1) Car Delivery Priority; 2) Train block; and 3) Yard block. This would mitigate any need to cherry pick at the trim end of the yard and thus be a more economical solution to creating reliable delivery service. In order to complete this task, low

priority cars would have to be rescheduled onto later trains thus maintaining minimum capacity for the higher priority cars on the next outbound train.

In his 2000 paper, Kraft wrote about the Terminal Priority Movement Planner. This was a hybrid model which combined the following:

1. Maximize the number of cars making scheduled connections; and
2. Minimize outbound train delay waiting for connections.

This model is a non-linear mathematical programming problem, when solved was able to show management when there were any late connections, thus allowing humans, not computers to make decisions on which trains to delay or which connections to disregard. In addition, this problem is solved using a breadth-first branch and bound search algorithm which uses the FIFO sequence of classification to set its upper bound (since FIFO is often not the optimized solution). In his paper, Kraft outlines the process by which the mathematical formulation works and is as follows:

1. Calculate the earliest time each outbound train can be "set" (the time when all the cars scheduled to said train have been sorted onto the classification tracks);
2. Subtract the projected "set" time from the target "set" time (the time at which the train should be "set" in order to meet its scheduled train departure) in order to define the difference in hours;
3. Use the exponential function, e^x to assign a solution "cost" for each train (higher the difference time in step 2, the higher the function "cost");
4. Sum the scores of each train to determine an aggregate score; and
5. Propose the hump sequence which results in the minimum score.

This algorithm was programmed in C and tests showed after 100 iterations there were few improvements. Having said that, the program was cut off after 100 iterations and took between 15-20 seconds to solve realistic one day problems (according to the author). In his 2002 paper, Kraft expanded on the original hump-sequencing model by adding a few steps:

1. Automatically reschedule cars which cannot be processed in time (onto later trains);
2. Advance any cars which are expected to be processed in time for earlier connections (provided there is room on the train – this also helps to maximise train capacity utilization as well as keeping the terminal from becoming over congested with extra cars); and
3. If any outbound trains have more cars scheduled than it can handle, determine which cars to bump to the next train.

It is important to note that all car scheduling occurs always and up to 36 hours prior to the actual hump procedure. This pre-determining of where cars will go allow a yard to prioritize their hump sequence of cars and place all priority cars onto specific classification tracks in their specific train blocks. This is very important because it allows yards to start a separate block, or classification track for those cars which are not of high priority and will not be taken on the first out train due to capacity restrictions. Though this process requires an additional track, it is important to note that once the initial train block is humped and classified, another train block may start behind it, filling up the remainder of the classification track length. Another way to mitigate (especially in smaller yards) yard congestion due to use of additional classification tracks for low priority cars is by the use of a rehump track, where lower priority cars would be sent in order to be rehumped later, after the priority train departs. This method is simple enough

to be completed manually (as noted by the Kraft, 2002). These two strategies, with similar outcomes, would certainly help to create a more reliable service, where the higher yard costs are mitigated by the overall service improvements and lowered requirements to run additional trains due to a large number of missed connections.

Another part of the priority–based classification system, after the hump sequencing and train block determination, is the block to track assignment sequence. This is an iterative process, based on a set of heuristic rules which is not a mathematically optimization based solution due to the complexity of the system, although Kraft poses that a better solution can be achieved through mathematical programming. The rules for iterations and the process are presented (Kraft, 2002b). In making the block to track assignments, the iterative process has to manage rehump cars by making sure that no priority cars go to the rehump track unless there is time for them to make their connection after the next rehump clearance. The typical rehump track in the Kraft paper was quoted to be complete every 8 hours or so. The system then determines the number of active blocks and determines whether or not they will fit in the tracks. This is done through the next set of steps which include coercing previously assigned blocks (making sure that once a block to track assignment has started then it remains in place unless the block is closed or the track is full), assigning and splitting very large blocks to multiple tracks (this reduces the numbers of blocks which can be actively assigned), and greedily assigning track space to small blocks (into the spaces behind larger, closed out blocks). Due to the different lengths of rail cars, when assigning blocks to the tracks a margin of safety in terms of block length was incorporated into the system. Afterwards, the system looks to see whether all of the train blocks have been assigned, if not then the system runs a loop lowering the number of active blocks by one in attempts to see if the next iteration will fit.

2.4.3. Terminal Operations Optimization

A very different way of looking at optimizing terminal is an approach which looks at the classification yard as a production system (Dirnberger J. a., 2007). This method looks at using factory physics, lean, Theory of Constraints, and Statistical Process Control in order to improve terminal operations. This research identified the pull-down process (actually the train assembly process) as the bottleneck in a rail terminal. In order to improve operations at the pull-down area this research proposes various improvement options including adding additional switching engines and using the hump engine to help switch cars. More importantly, the research created what is known as the Bowl Condition Manager (BCM) which helps managers determine the quality of sorting after the hump process. This is important because sorting the cars properly the first time allows the yard to focus on the pull-down process without being distracted by cherry picking out of place cars or conducting what Dirnberger calls rework at the trim end of the yard.

2.5. Integration of Yard Operations and Main Line Operations

A combined solution, known as a dynamic car scheduling system, is one which integrates both yard blocking and train dispatching models in a way to maximize train capacity and reliability using a mathematical optimization. In his 2000 paper, Implementation Strategies for Railroad Dynamic Freight Car Scheduling, Kraft notes that current difficulties with scheduling systems can be "overcome by adopting an optimization based scheduling algorithm, which would couple this decision making process by taking train capacities into account, and would allow for dynamic routing of shipments based on delivery commitments and train capacity." Dynamic scheduling allows for changes to the current lookup table system, which statically assigns a single path to each car based on its destination without taking other system factors into account.

These changes to the current system include identifying and utilizing all the paths available to a car instead of only the original one, thus, creating a flexible car routing option. This “multi-pathing” system, as Kraft calls it, is more reliable, mathematically, than a single-pathing system in making the best use of all available train capacity and schedule slack time. This again is a concept of flexibility, which as discussed earlier, is a major factor in reliability for railroads. This concept can be theoretically established using simple probability calculations as shown by Kraft, the single-pathing scenario of three links in a series which has a combined reliability of 51.2% (0.8 x 0.8 x 0.8), whereas the multi-pathing scenario of three links in parallel which as a combined reliability of 99.2% (1 - (0.2 x 0.2 x 0.2)). Refer to Figure 2.7 and Figure 2.8 respectively. These links and nodes can be thought of as mainlines and yards respectively.

Kraft looks at various options to assign cars to outbound trains based on the Dynamic Car Scheduling process (DCS) but with respect to current railroad practices. The original DCS as proposed by Kraft would require an overhaul of current practices in that very high data integrity and accuracy would be required as well as a strictly adhered to schedule that to switch from the full FIFO style of classification into a more priority based one. Kraft concluded in his paper a decoupled, two step process for classification and train assembly:

1. At the yard: classify cars by yard block only, following the recommendation from the trip plan but ignore the train information; and
2. At train assembly: count the number of cars assigned to each train and pull them from the classification track irrespective of their priority, and if additional capacity remains on the train, fill it with additional cars on the classification track.

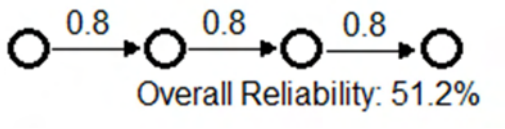

Figure 2.7 Three Links in Series (Kraft, 1998)

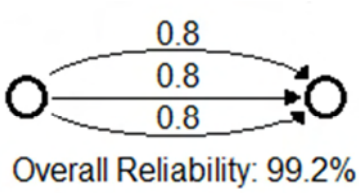

Figure 2.8 Three Links in Parallel (Kraft, 1998)

From the notes above, this is a good way to implement a dynamic car scheduling system without overhauling data collection and terminal processes while still being able to maximize train capacity utilization so that all trains run at maximum capacity, even if car trip plans do not necessarily require them to.

2.6. Summary

This chapter represents a broad, comprehensive literature review of existing research in freight rail operations, classification yard operations and general railroad information. From the research conducted, details of how a railroad network is set up, how it functions and how it can be optimized can be determined. It is important to note that much of the existing literature has been focused on mathematical optimization modeling and this leads to a lot of focus on single elements of a larger network resulting in slightly unrealistic results.

3. RAILROAD NETWORK SYSTEMS

This chapter reviews the major components of the physical railroad network as well as the operating rules and regulations by which railroads function. Additionally, this chapter discusses various model components as reviewed in multiple existing models. The elements discussed here are very general in nature.

A railroad network consists of a set of stations (or yards) connected by rails upon which trains of rail cars powered by locomotives travel. This travel is the movement of goods from the origin (the starting) yard to the destination (the finishing) yard. The operations of a railroad can typically be split into two separate categories: yard operations and linehaul operations. Yard operations include classification, inspection and maintenance of the individual rail cars whereas linehaul operations involve the actual movement of the cars from one location to another.

It is important to note that railroads operate with four (4) major operating plans in place. These include the blocking plan, the train formation plan, the train schedule and the empty car distribution plan. The blocking plan regulates the contents and the number of blocks whereas the train formation plan regulates which blocks make up each train and how the traffic will flow over the network. The train schedules and empty car distribution plans are typically created after and often do not affect blocking or formation plans (Martinelli, 1996). This thesis is focused on three (3) of the major components of the operations plan including the blocking plan, the train formation plan, and the train schedule.

3.1. Railroad Network Models

Rail network routing models can typically be classified into two types: optimization models and simulation models. This section discusses these two types of models. These models sometimes combine and integrate yard operations with mainline operations in order to provide a simultaneous routing of traffic and workload output at yards.

3.1.1. Mathematical Optimization Models

Mathematical optimization is simply the minimization or maximization of a single objective function given a set of constraints by which the system is governed. In doing this, analysts often only get “optimal” results with respect to one aspect of the railroad network such as number of car switches or overall distance of travel for cars. No literature reviewed in the course of this study utilised multi-objective function optimization to determine an optimized solution. Mathematical optimization can run specific algorithms and determine various things individually, such as shortest path, lowest cost or minimum time in the network. Though this may be an effective way to solve for a minimum in that one specific category this is not realistic in that railroads have many criteria which must be satisfied simultaneously.

Since a rail network is so complicated and has so many different individual systems involved, it is easy to see that single function mathematical programming solutions would never be globally optimized. For example, minimizing the number of switches for individual cars can lead to long waits in order to create specific unit trains which carry only one type of O-D traffic. Though this will certainly make sure that the individual car is only switched at its Origin yard and then at its

Destination yard, it will create havoc for the railroad and the consumer in terms of delay, wasted storage space and lost revenues.

3.1.2. Simulation Models

Simulation is the imitation of something real (a process, a system, etc.) and can be represented as a physical model or as a mathematical model. Simulation models are defined as "abstractions of a real system which retain the essential aspects of a system" (Papacostas & Prevedouros, 2001). In saying that, it is important to note that since simulation models are abstractions of reality, they are not exactly the same as real life. This means that some simplifications are necessary, but the major system components are always included.

Simulation is often most typically used when a system is so large and so complex that an analytical model would be inefficient and costly. Such an example of a large, complicated system is a railroad network. The railroad system is one with many individual components which work together in order to move traffic from origin to destination. Though there are physical models of railroads (model train sets) these are different from the large scale mathematical railroad network models used today in industry. The reason large scale mathematical models are used is because the cost of implementing various train strategies in a test scenario in real life would be very costly, time consuming, and aggravating to customers.

Mathematical Simulation models can be either deterministic or probabilistic in nature; however, railroad network models are typically deterministic in nature. This is because railroads are, on a whole, typically steady state systems with relatively low fluctuations in traffic patterns on a day to day basis (Interview with Ray Dai of CPR, 2010).

3.1.3. Comparison of Models

Often optimization results are very general and they are not easy to implement in real life and, thus, a simulation model can prove to be more effective in creating a more real world scenario. Simulation can provide many individual details (outputs) at the same time given the same inputs as an optimization model, but not require running multiple individual models independently of one another. In addition, simulation is more realistic, existing conditions can be easily modeled whereas an optimization model is more general and may not consider day to day events such as track closures or routine maintenance operations without excessive programming. With simulation, a mathematically optimized solution is not typically found. Instead, results are provided for review and through completing multiple iterations of the analysis with varying data or criteria, using human intelligence, a "best case" integrated solution can be determined.

For the reasons stated above, this research will focus on a simulation model to accomplish its goals. The mathematical simulation model developed is a discrete-state, and deterministic simulation model.

3.2. Classification Yards and Yard Operations

Yards are the hubs for all originating and terminating traffic. In a railroad network, yards are where cars go to get "classified" (sorted) into their respective "blocks" (groups of cars with similar properties such as destination or car type) after which they get built into trains to be dispatched. These cars can be originating traffic stemming from the yard or they could be cars which have been transported from other yards and have to be reclassified.

There are four major operations at any given classification yard. Although for any one car, these operations happen sequentially and the next operation depends on the prior operation. Large yards typically continue to run 24 hours a day and all tasks happen simultaneously. Typical operations at a classification yard can be grouped into the following (also refer to Figure 2.3):

1. Train Receiving and Inbound Inspection;
2. Classification or Sorting of Cars onto Classification Tracks;
3. Train Marshalling or Assembly; and
4. Outbound Inspection and Train Departure.

During the receiving, the inbound inspection is comprised of checking the rail cars for structural integrity, safety, as well as the consist of each car. During this time period in a hump yard, the yard crews will turn off the air brakes so that cars become free rolling (this way they can be pushed or pulled as required) and the cars will then be put into queue so that they can be classified. In a flat yard, the inbound cars will be uncoupled and split up only at the cut limits and the air brakes will only be bled if individual cars are being classified.

The cars are then sorted onto various classification tracks which act as temporary storage for use while building a new train or a portion of a new train. This sorting, or classification, is conducted in one of two ways: by using gravity such as in hump yards or by using switching locomotives in flat yards as described above. Once the cars have been sorted onto the classification tracks into their new yard assignments and the cut off time for the departing train has come (time at which the train must be built for departure) the cars are built into an outbound train at the trim end of the yard, inspected as they were during receiving, and released to their next destination after a brake test.

3.2.1. Individual Yard Capacity

Yards can typically be assigned average capacities as is done in various papers such as Martinelli and Teng (1996), Assad (1980) and as per the case study in Troup et al (1977). In these situations, the yard is looked at as a black box where cars go in and cars go out (there needs to be a conservation of flow here) but the specific yard operations are not typically modeled in train routing/makeup models. In essence the yard can be looked at as a black box where incoming trains arrive and are processed, and outgoing trains depart. In spite of this, yard resources affect how many cars can be classified (the capacity) and how quickly. Factors affecting yard capacity can include sorting techniques and strategies, system loads, and yard management.

3.2.2. System Effects on Yard Capacity

There are various system factors which influence how individual yards perform on a daily basis. The following is a list of the major factors which affect individual yards adopted from Troup et al (1977) pp176.

a. Composition and delivery times of cars from other yards (system, interchange or even industrial);
b. Time allowances for delivery of cars to their destinations (i.e. contracts – this can often be precluded when modeling routing of trains or blocks and agreements can be readjusted between the industry and the railroad with notice based on schedule changes from the railroad);
c. Constraints on sizes of trains (based on yard sizes, main line track infrastructure strengths, and motive power allocation);

d. Availability and reliability of advanced information (train formation, consist information, etc.) of incoming train traffic;
e. Rules and regulations for crews (unionized members);
f. Blocking and formation strategies followed in other system yards (i.e. yards working together).

These above factors are just the broad spectrum of what can affect a yard outside of the yards own management and resources. These are all things which are not within the control of individual yards and, therefore, must be looked at and treated on a macro-network system level in order to reap the benefits of synergy in the network. Making the entire system work together in order to optimize the flow of traffic can lead to far greater benefits than the optimization of individual yards.

3.3. Mainline (Linehaul) Operations

Track segments between yards can be single lines or multiple parallel lines which move between yards in the network. Trains can typically move in opposite directions on the lines so long as there are prebuilt sidings (short train length segments of track) where one train can wait while another one passes. These sidings not only allow trains to pass each other in the opposite direction but also in the same direction such that trains with higher priority and speed can overtake slower or lower priority trains. The geometry of the main lines can affect how fast trains can move on the lines and how heavy the trains can be. The important part of linehaul operations involve decisions about which trains move which blocks over a given set of tracks in order to get from one yard (origin) to another (destination).

3.3.1. Origin-Destination Travel Demands

Origin-Destination (O-D) demands are represented by the number of rail cars destined to arrive at a given yard from an origin yard. This is similar to other forms of transportation such as automobile. O-D demands on any given network are based very much on the economy and industries located along rail routes. Though this is the case, railroads can still accurately predict the travel demands on their network on, at the very least, a seasonal basis with some room for fluctuations, and design an operating plan to justify running various trains in order to serve the travel demand.

3.3.2. Train Composition (Pull lists)

Trains themselves are a composite of locomotive power (1 or more engines) and sets of blocks of cars traveling in the same direction. A train can carry any car or set of cars (blocks) as required by the railroad. The makeup of the train, the pull list (list of which blocks to take on its journey – also known as block assignment), is a very contentious issue because if done correctly it can help to prevent delays and minimize costs, but if predicted incorrectly the effects can be devastating and crippling to a railroad. These lists are developed in an attempt to create minimal impacts on the overall system.

Depending on the train route, different blocks of cars traveling to various destinations (either final or intermediate) can be pulled. This means that trains do not have to pull blocks going to the same destination as the train. They can pull additional traffic (preferably moving in the same direction overall) in order to bring the traffic closer to its final destination yard. This can help in keeping traffic moving and prevent traffic jams at yards.

3.3.3. Train Routing

The physical infrastructure, e.g. the tracks and yards, are typically set and not often the subject of tactical planning studies. The actual use of existing physical systems is what is often being planned, and this is done through creating itineraries based on physical routes and transportation demands. Figure 3.1 shows an example of a simple network of 6 yards (the black dots) connected by 10 sets of railroad tracks (lines). If for example there is a travel demand for traffic to travel from yard 1 to yard 6, there are four different physical routes possible: (1, 2, 4, 6), (1, 2, 5, 6), (1, 3, 4, 6), and (1, 3, 5, 6). Along physical route (1, 2, 4, 6), there are four specific itineraries possible and this is shown in Figure 3.2. The itineraries show how a block of cars which are to travel from yard 1 can travel to yard 6 based on the (1, 2, 4, 6) physical route. The itineraries (denoted by I#) show where a given train will travel from yard 1 to yard X on the way to its final destination, each straight line between the arrows shows a different train. The most direct route is one where a train travels from yard 1 to yard 6 without stopping at yards 2 or 4 but this can be expensive and the demand may not justify running a straight train in this manner, therefore, itineraries for each block of cars is required. The decision of which route to take can be affected by the various factors suggested in prior sections but can also be adjusted based on travel demands (Origin-Destination).

3.4. Network Element Interactions

This section presents how various elements of a railroad network interact with each other. The types of elements which can affect a network can be categorized into two specific sections: system elements and external elements.

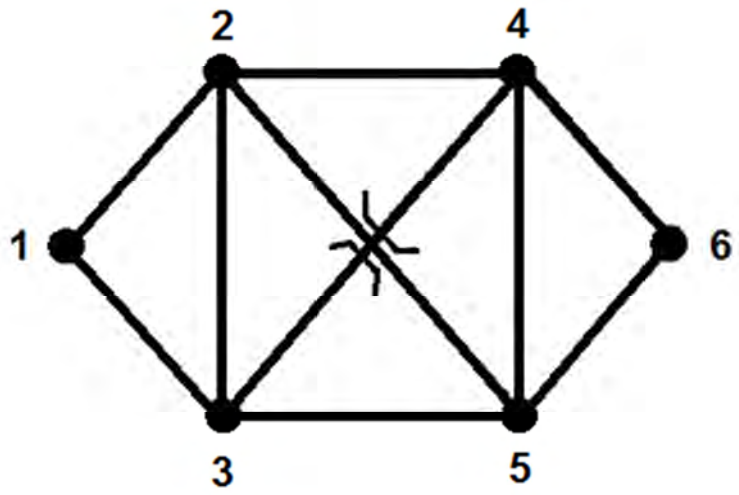

Figure 3.1 Example Network Railroad – Martinelli and Teng (1995)

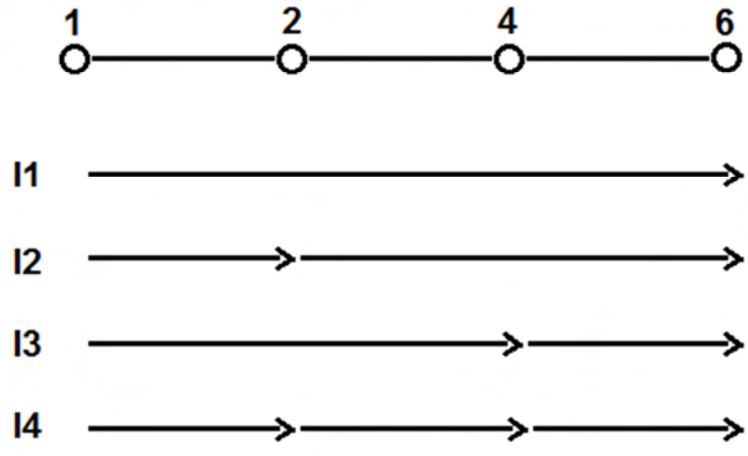

Figure 3.2 Train Itineraries for Physical Route (1, 2, 4, 6) – Martinelli and Teng (1995)

3.4.1. System Elements

System yards collect railcars from local industries (origin) and dispatch railcars to other yards (destinations) in order to move traffic from one place to another. The network of yards is a system where cars from one origin to another destination must travel in order to get as close as possible to their final destination. The cars at system yards do not typically generate and terminate exactly at those yards, but somewhere else local to those system yards.

For the purposes of modeling however, cars generated at local yards can often be considered as being generated at system yards. Cars at the originating system yard must be sorted into blocks which are moved on road trains from yard to yard until reaching its final yard destination, being resorted or classified along the way at intermediate yards as required. In order to allow the system to run smoothly and efficiently the car, block and train loads must be balanced in such a way that it takes into account the traffic demands, yard capabilities, and main line capabilities. This must be done in order to maximise profits and reliability while minimizing delays and operating costs in order to satisfy the systems needs.

3.4.2. External Elements

Within the system there are other factors such as crew resources and unions which can affect what, how and when work takes place, and there are government and regulatory restrictions which can affect the performance of trains. Often these elements are not modelled. Crew management can be excluded because it is a separate topic and it is typically considered that the crews will be available for the operations that are being modelled. Regulatory elements such as car inspections and safety do not need to be included because they are typically incorporated into the train speeds or yard processes and are, therefore, often not modelled in a network sense.

Other elements such as regular maintenance and fuelling of locomotives, car repairs, are all typically built into the schedules while yard work is being completed (e.g. while cars are being classified, others are being fixed, engines are being maintained, etc.). This way they are not the controlling operation and, therefore, become secondary focuses which are not often modelled in existing routing models which have been reviewed as a part of this study.

3.5. Typical Model Components

A model is comprised of multiple segments which, together, create the whole picture. These components include model objectives (objective functions), model constraints (model formulations and calculations), model input data, and model output data. These are explained below with examples of typical elements provided.

3.5.1. Model Constraints

There are various constraints and factors which must be considered when building a computer simulation model for a railroad network. These constraints are what make the model realistic and provide some resistance to the actual routing of cars throughout the network.

The following is a list of constraints upon which various previous models have been based:

a. Physical Constraints in the Network (such as yard and link data);
b. Individual Yard Capacities;
c. Yard Dwell Times;
d. Maximum and Minimum Train Lengths;
e. Maximum and Minimum Block Sizes;

f. Flow Conservation in intermediate yards (what goes in must come out);
g. All Cars must be Routed from Originating Yards to Specific Destination Yards irrespective of how many intermediate yards they go through;
h. Cost constraints on specific movements (i.e. resistance on certain routes);
i. Time constraints based on delivery commitments (i.e. an exponential cost function in order to mitigate delays or rerouting).

Though not each of the above noted constraints are considered in every individual model, the above list goes to prove that many factors must be considered in the preparation of any model. These constraints are what make the model fit into a system and define the characteristics of said system. They also provide insight as to how the system works and how each element interacts with one another within the system in reality. For example, if a constraint is put on the maximum speed of a train on the network, the train will only be able to move that fast throughout the course of its journey. An example of such a constraint is presented in Equation 3.1.

$$S_t \leq S_{max} \tag{3.1}$$

where,

S_t = the speed of the train (km/h); and

S_{max} = is the maximum allowable speed of travel on the entire network (km/h).

3.5.2. Model Objective Function(s)

When looking at optimization models, each model has a specific objective function. This is the one individual item that a model is trying to optimize (either minimize or maximize) in order to

determine the “optimum” configuration of a given set of parameters in a specific network. This objective function takes into consideration the various constraints considered in the model and prints a specific output. The following list shows some general objective functions which have been studied in past literature of mathematical optimization models:

a. Minimize car dwell time in yards (blocking optimization);
b. Minimize car costs (based on time spent on the network);
c. Minimize time spent on the network (while attached to a road train – e.g. transit time);
d. Minimize total number of car handlings/classifications (blocking optimization);
e. Minimize operating and delay costs (blocking or transit time optimization); and
f. Maximize number of cars which can run through a network.

The downfall of this method is that it looks only at one specific objective function and does not consider all of the intricate realities of the entire network. Even with many constraints built into the system there are often other considerations which are missed. Adding too many additional constraints to the model can possibly render it computationally impossible to solve. An example of this type of objective function is presented by Martinelli and Teng in Equation 3.2. This equation represents a minimization problem which focuses on the overall operating time for a given train/cars on a train over the course of travel time over a specific route and arrival operations at the destination yard. This focus on minimizing the time does not particularly consider the number of cars per yard or the total number of car handlings in a network.

$$Minimize\ Z = \left(\sum_{j=1} t_j * X_j\right) + \left(\sum_{j=1}(\delta * X_j * c_j + v_j * X_j)\right) \quad (3.2)$$

where,

t_j = average travel time for train j (hours)

X_j = number of cars on train j (# of Cars)

δ is a 0-1 function which determines whether the constant cost is applied or not

c_j = constant operating time at the destination station (hours)

v_j = variable operating time at the destination station (hours)

The small scope of optimization functions is not a problem in simulation models as a simulation can provide various results on any or all of the factors mentioned above. Though the system may not be mathematically optimized for any individual function, the model can present various outputs simultaneously. These outputs can be analyzed individually or together, providing a more complete system overview as opposed to a narrow individual function. Optimizing individual criteria may lead to losses in other sectors of the system, and a whole network synergy may not be realized. The downfall of a simulation however involves running multiple analyses and manually reviewing results in order to find a best solution as opposed to a mathematically optimized solution. With simulation there is no "optimized" solution and thus an analyst may revise and re-run simulations may be present in order to get a "best case" plan. This can prove be quite time consuming should the analyst be unfamiliar with a specific model, however, the overall result of a best solution can prove to be a more globally refined operations plan.

3.5.3. Model Input Data

In any given model there must be inputs which are entered by the user in order to feed information to the model. Within railroad networks there are quite a few sets of data which are

required prior to even beginning a simulation or optimization in order for the model to run realistically. The following is a list of typical input data used in existing models (based on literature review). The list also includes whether the inputs are fixed (always the same when the model is used) or variable (can vary between model uses):

a. Physical constraints of the network (number of yards, yard locations, lengths of tracks, train running speeds) – these are typically fixed, however speed can be adjustable;
b. Motive power (number of locomotives) – this could be variable, fixed or assumed to be infinite if not explicitly included in the model (i.e. railroads can acquire more power as needed);
c. Crew requirements – this could be variable, fixed or assumed to be infinite if not explicitly included in the model (i.e. railroads can acquire more crews as needed);
d. Travel demands (O-D) – variable (though they vary over time, these values are known or estimated from past data);
e. Physical or variable costs for car movements of delays – variable (though they vary over time, these values are known or estimated from past data);
f. Possible physical routes for trains – fixed (though the model can either determine routes or they can be manually supplied);
g. Possible itineraries for all physical routes – fixed (though the model can either determine itineraries or they can be manually entered);
h. Operating times for routes and itineraries – variable (known or estimated based on congestion, speed restrictions, noise laws, etc.);
i. Typical Yard capacities (if the yards are looked at as a black box as suggested in Troup et al (1977)) – these are typically fixed (estimated from past); and

j. Individual Yard operating times (if the yards are looked at as a black box as suggested in Troup et al (1977)) – these are typically fixed (estimated from past).

The data inputs will all be entered into the model prior to or during the course of running the model and will be used to determine the individual or various outputs which the model produces.

3.5.4. Model Output Data

The output(s) of a model are what the analyst requires in order to help make decisions about the system and how to optimise performance. Optimization models often only provide a single optimized output with some various outputs on "how" to get there, such as train configurations or blocking movements. Simulation models on the other hand can often provide multiple and various outputs which can be analyzed all at once, though not mathematically optimized, they can be iteratively optimized by the user and running multiple simulations. The simulation approach allows a more comprehensive and integrated system optimization rather than a single function optimization.

The following is a list of various output(s) of existing models:

a. Physical routes and itineraries for given car(s)/block(s)/train(s);
b. If it is a train formation model – number and sizes of trains to be created (given a blocking plan);
c. If it is a blocking model – number and sizes of blocks to be created;
d. Number of times each car is handled/switched;
e. Flow through each yard;
f. Number of cars in each train or in each block;

g. Number of trains required on a timely basis (and in turn crew requirements, overall costs, etc.); and
h. Various or Overall costs (based on cost functions in the model).

The results of the above outputs will not only help to satisfy the objective function(s), but also show what a railroad should do to accomplish the goals set out by the model and a means of measuring the changes. These outputs must be analyzed by the user to determine whether they are realistic and employable as well as if they actually make the system perform better than in its current state.

3.5.5. Existing Routing Methods

The process of routing cars, blocks and trains, is an iterative and complicated process. Various operating plans (the blocking plan, the train formation plan, the train schedule and the empty car distribution plan) must be combined by the railroad to create a final operating plan which is not only functional but also "optimal" / "best case." It is important to note that in general, railroad traffic is fairly steady state. There are some seasonal variances and some economic factors which can affect railroad traffic but in general it is assumed that traffic at least on a monthly or seasonal level is steady state. For this reason railroads are able to create general train schedules and static blocking plans with reasonable confidence.

When running simulations of their networks, railroads supply their current software with various data which includes static blocking plans, train schedules, traffic files (OD data, car types, etc.) as well as network topology and geography. They, then, use the shortest path algorithm such as Dijkstra's algorithm in order to determine the shortest path for each individual car moving from

one node to another. They often combine a shortest path algorithm with one which takes resistance factors into account, thus, preventing cars from being assigned undesirable paths. After this, the software will assign each car to a specific predetermined block or set of blocks which are destined to travel from node to node. The blocks are, then, assigned to individual trains which carry them to their destinations. The existing models also provide various system and network statistical data by which users can review overall system performance (Oliver Wyman Group, 2010).

With this system, different sections and groups within the railroad come up with the different strategies which make up an integral operations plan. This means that if there is a train schedule is causing the blocking plan to be less efficient because trains are not scheduled at the time blocks are ready, the blocking coordinator would have to go to the scheduling coordinator and the two would have to go back and forth, running a long and time consuming simulation each time. With the model developed here, the back and forth of two separate groups working on individual plans is taken away. Instead, with the model and guidelines proposed, one analyst can develop, test and analyze an integrated operations plan with only traffic, network, and yard data.

3.6. Summary

This chapter has reviewed the optimization and simulation models in order to determine which one is the most suitable method for this thesis. This chapter also reviewed the major components of the physical railroad network as well as the operating rules and regulations by which railroads function. It has provided the basis for setting up and creating a model which will assist in optimizing railroad networks on a whole. From the discussion in this chapter, selective criteria

are carried forward into the model development which address the major areas of concern with respect to performance enhancement and cost minimization.

4. MODEL DEVELOPMENT

This chapter outlines the development of the simulation model and the methodology. This chapter also discusses inputs and outputs to the model, as well as the guidelines which assist in developing operation plans. Finally, model capabilities and limitations are discussed.

4.1. Model and Guidelines Overview

Railroads move freight traffic on their network based on an overall operations plan that includes blocking, train formation, and train scheduling plans. The optimization of these operations over the entire network is integral to maximizing efficiency and minimizing costs. The model developed here uses a discrete state, deterministic simulation approach for analyzing various operation plans of a railroad network. Along with the model, various guidelines for establishing a comprehensive operations plan are developed. The objective of the operations plan is to move all the freight on the railroad network reliably and with a minimal cost. In order to maintain reliable service with a minimal cost, various factors must be reviewed such that they can be compared amongst several alternative options. By analyzing and comparing several options using a straightforward and consistent criteria will help in providing a 'best case' operations plan with the aforementioned goals in mind. These factors for comparing multiple operation plans are described, in detail, further in this chapter.

The general overview of the model is shown in Figure 4.1. The major components include general network data entry, operation plan formation (various route restrictions, train schedules, and blocking strategies) and data entry, block to train assignments, train and O-D pair routing, as

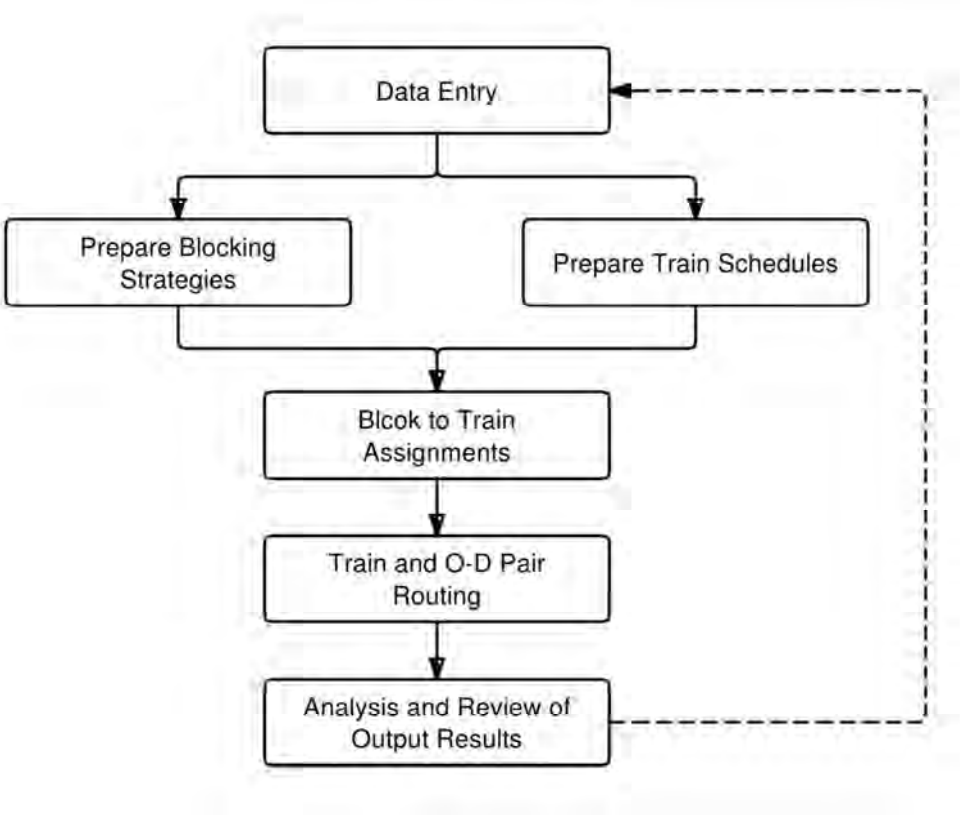

Figure 4.1 Model Overview Flowchart

well as analysis of operation plan and model results. To accomplish the data requirements of the model, the analyst must enter some general network data including number of yards (variable), number of trains (variable), and number of routes (variable), as well as minimum and maximum train sizes. The data entered here affects what data will be required for each yard, various routes, O-D pairs and train schedules. Typically, the yard, route and O-D data are fairly standard, or given, for a particular network. The train schedules should be prepared using the methodology and guidelines developed in this thesis, which are both straight forward and easy to use.

Once this data entry is complete, the model tentatively assigns blocks to trains using the 3-step process developed here and allows the analyst to review the assignments for inconsistencies.

Since the analyst can review the block to train assignments, Braess Paradox may be avoided. This paradox occurs because the model selects the shortest possible route for each train, without consideration for the entire networks' movements. Once the block assignment is complete, the model then prioritizes trains based on build times and special handling requests. This allows the model to build a queue list for each yard and provide the analyst with details of when each train is using yard resources. The model, then, initiates the train routing algorithm simulating the operation of each individual train and transporting each individual car in its O-D pair.

The model can output various individual train statistics, route statistics, and yard statistics including when each yard is being used (for building or receiving trains). In these results, the analyst can retrieve much needed data required to make a decision on which operations plan is the 'best.' Though it is not always clear which plan is best, using the guidelines developed in this thesis, a used can determining the "best case" plan based on various economic, reliability and environmental factors. The model can also be used to test various train plans, blocking plans, and yard plans (including resizing/closing yards). Alternatively, the model can also be used to determine what effects traffic pattern changes would have on the network. More details about the model components are presented in the following sections of this chapter.

4.2. Model Data Inputs

Various data requirements are present for the model to be effective in providing a realistic simulation. Data input into the model are broken down into several distinct sections:

1. Network Data;
2. Yard Data;

3. Route Data;
4. Origin-Destination Traffic Data; and
5. Blocking Plan Data.

These inputs and their functions are described in the following subsections.

4.2.1. Example Network

Figure 4.2 shows an example of a large and comprehensive railroad network in Canada and the northern United States owned by the Canadian Pacific Railroad (CPR) Company. The tracks shown are the major lines connecting all of CPR's major nodes of operation and City Centers. Each of these Cities has a large yard on the CPR Network. Although additional yards of various sizes exist on the network, this map only covers the major City Centers.

For the purposes of this thesis and providing a detailed explanation with examples of the model inputs, a small hypothetical subset of a large network, similar to that of CPR's, has been provided as shown in Figure 4.3. The examples in all of Sections 4.2.X were created in reference with this network diagram.

4.2.2. Network Data

Network inputs include the number of yards, number of trains, maximum and minimum train sizes, and number of routes. An example of the network data as it is input into the model is provided in Table 4.1. The number of yards represents how many yards are in the network being analyzed and is typically a static number. The number of trains is typically a variable number which varies over the course of the analysis and can be changed as often as the analyst chooses to do so. The minimum and maximum size for each train is measured by number of cars.

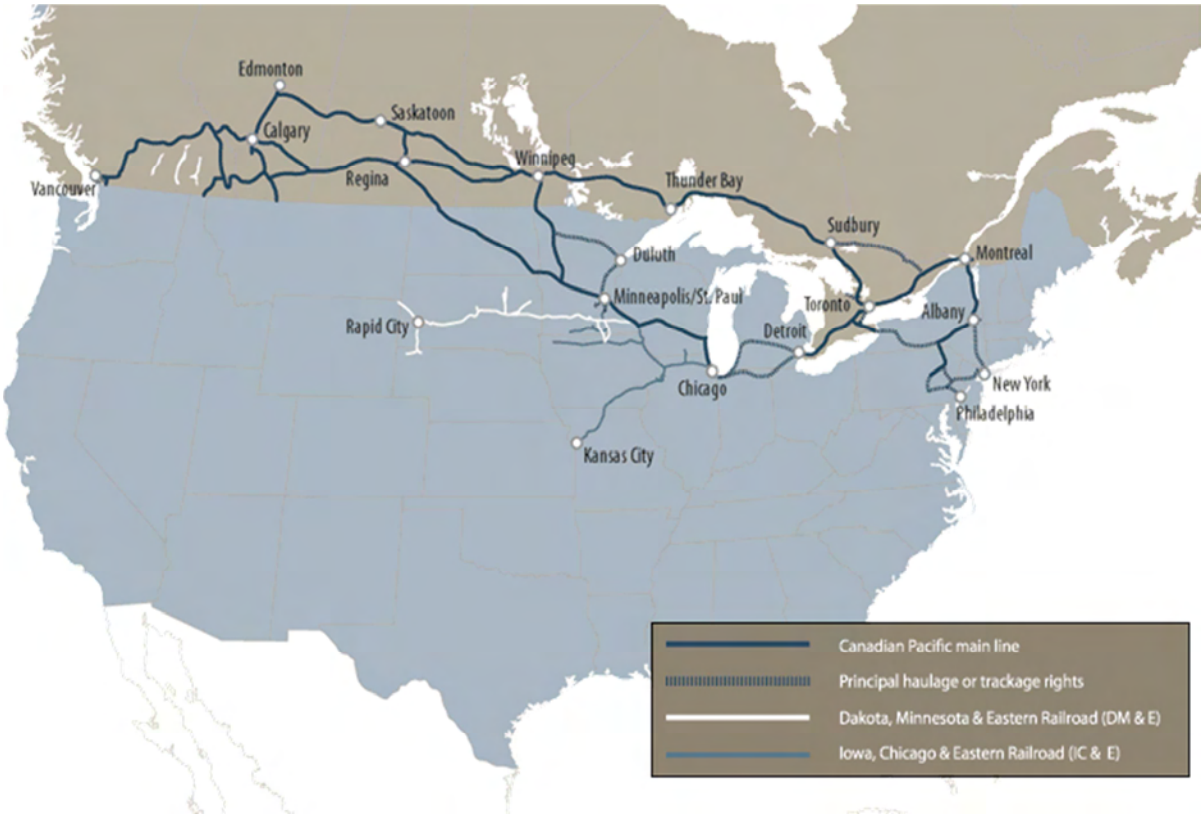

Figure 4.2 Example Railroad Network Map (http://www.cpr.ca)

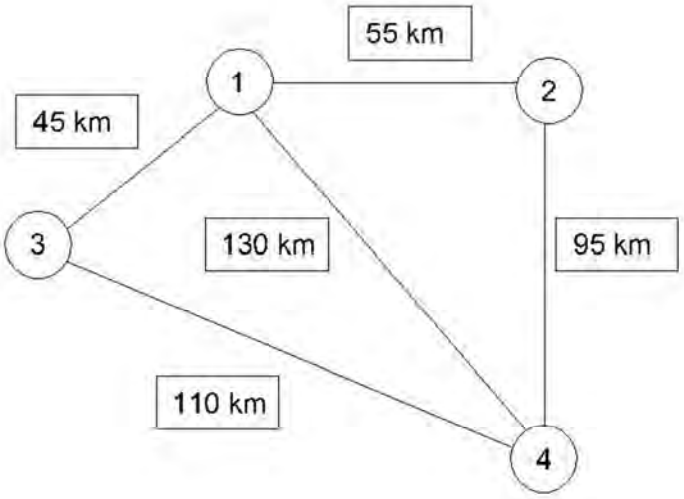

Figure 4.3 Example Network

Table 4.1 Example of Network Inputs

Total Number of Yards
5
Max Train Size
999
Min Train Size
0
of Trains
15
of Routes
25

This is variable and can depend on, but is not limited to, the following: how the network is setup and the available locomotive power or size of sidings on mainline tracks. The number of routes is dependent on how many yards exist in the network, their configuration, and the links between them. The number of yards and number of routes can also change based on the analysis being completed. For example, if a sensitivity study on the closure of one yard was being conducted, removing a single yard from the network would have effects on the number of routes (as all routes going through that said yard would be negated).

Though there is no limit to how many yards the model and software application can work with, the software needs to be reconfigured manually when the number of yards changes. As with the yards, there is no actual limit to the number of routes or trains that the model can handle, but some software manipulation may be required if the number exceeds 50 and 35 respectively. It is important to note that additional yards, additional routes, and additional trains cause slower simulations. There are no software limits on the number of cars a train can carry. As the analyst adds data into the model, the number of yards, trains, and routes can change during the

assessment of a network. Though it is quite easy to change the number of trains and routes, up to 35 and 50 respectively, as one would just need to add or subtract routes and trains, it is slightly more difficult with yards. The yard data, discussed below, is simple to adjust, but the Origin-Destination tables discussed below, require some more adjustments because removing or adding a yard can dramatically change the O-D's and the changes must be reflected within the model.

4.2.3. Yard Data

Yard data consists of yard identification (ID), the type of yard, yard input time and yard output time. The Yard ID is a sequential numbering system starting at 1. This is a simplistic measure for the model to identify each yard. For example if there were 5 yards in the network, each yard would be given an ID number from 1 to 5. The daily blocking capability of a yard is limited to the maximum number of yards in the network. This is due to a software limitation which does not take into account more than one type of O-D traffic pairs. If other classification characteristics for O-D traffic were to be incorporated then additional blocks would need to be formed.

The Average Yard Input (Incoming Train) Time includes Train Receiving, Inbound Inspection and preparation for the Classification of cars. The switching time per car is the amount of time it takes, on average, for one car to be classified. This average time is how long it takes a car to move from the control of the switching engine to its respective classification track. The Average Yard Output (Yard Service) Time includes Train Assembly, Outbound Inspection and Train Departure. An example of the yard data as it is entered into the model is shown in Table 4.2. The Type of Yard can be a Hump Yard or a Flat Yard. The Type of Yard and the Minimum Pure Block Size are for information purposes only.

Table 4.2 Example of Yard Data Inputs

Yard Number	Type of Yard	Daily Blocking Capacities	Minimum Pure Block Size	Incoming Train Prep Time (Minutes)	Switching Time Per car (Minutes)	Yard Service Time (Per Train - Minutes)
1	Hump	4	10	00:15:00	00:00:30	00:30:00
2	Flat	4	10	00:25:00	00:00:45	00:30:00
3	Hump	4	10	00:20:00	00:00:30	00:25:00

4.2.4. Route Data

Route inputs involve the user manually entering the various routes between each yard (so long as they are connected). This is dependant also on whether the user decides to allow a specific route to be used between two yards. If a route is not entered into the model, the model assumes that it does not exist. An example of the route inputs into the model is provided in Table 4.3. The user must enter the origin yard, the destination yard, path between the yards (i.e. which yards the train would pass as it travels) and the overall distance between the two yards along that specific path. The model determines which of the various routes/paths is the shortest and, then, assigns that specific route/path to each specific train accounting for any connections present.

In the example, Route #4 and Route #5 both originate and terminate at yards 2 and 3, respectively. If the model were presented a choice between the two routes, it would choose Route #5 because the total route length is less than that of Route #4. If the user for some reason did not want Route #5 to be used by the model, it should be left out of the Route Inputs.

4.2.5. Origin-Destination Traffic Data

Origin-Destination Traffic data are entered into the model in a tabular format which specifies how many cars are generated at each origin yard destined for each destination yard. The model can be setup to interpret the O-D data such that it is split up into 4 hour segments for a total duration of a single day or duration of seven individual weekdays. Each of these segments is considered to be a specific timeslot and this is used by the model when building trains. The time period adjustment requires some programming modifications, but the methodology of the model and formulations stay the same. Any O-D traffic data in the tables can be set to "Special Handle" or Priority processing by formatting the text in the O-D table to be bold. This means that the bold O-D traffic will always be handled (classified or assembled) first, prior to other similar traffic on the train. Examples of O-D table can be seen in Table 4.4 and Table 4.5. An example of non-priority and priority cars can be seen in Table 4.4 with O-D 1-5 at 08:00hrs and 12:00hrs respectively. Other O-D's can be interpreted similarly.

4.2.6. Blocking Plan Data

The blocking plan is set up by the user and determines which O-D pairs will be transported on which blocks. The model determines whether the block is mixed or pure by realizing whether there are multiple O-D pairs assigned to the same block or not. The number of blocks at any given yard is limited to a maximum of the number of yards in a network less one. This is because the model realizes only one type of car. There is no consideration for multiple classification characteristics of O-D traffic. Pure blocks are denoted by a 1 and mixed blocks are denoted by a 0. This is determined by the model and is important because with train assignments, the mixed

blocks will be picked up by the first available train. This is discussed in more detail in Section 4.4.1.

Table 4.3 Example of Route Inputs

Route #	Origin Yard	Destination Yard	Route Path (Yard to Yard)	Route Length (km)
1	**1**	**2**	12	**55**
2	**1**	**3**	13	**45**
3	**1**	**4**	14	**130**
4	**2**	**3**	243	**205**
5	**2**	**3**	213	**100**

Table 4.4 Example O-D Table for Daily Operations

From	To	04:00	08:00	12:00	16:00	20:00	23:59	Next Day	Total
1	**1**	1	2	2	0	1	0	0	**6**
1	**2**	3	3	16	9	27	0	0	**58**
1	**3**	1	2	11	5	6	0	0	**25**
1	**4**	0	0	0	0	0	0	0	**0**
1	**5**	2	5	**34**	**21**	40	0	0	**102**
Total		**7**	**12**	**63**	**35**	**74**	**0**	**0**	**191**

Table 4.5 Example O-D Table for Weekly Operations

From	To	Mon	Tues	Weds	Thurs	Fri	Sat	Sun	Total
1	**1**	41	46	41	48	0	0	0	176
1	**2**	0	3	0	16	0	0	0	19
1	**3**	0	0	0	0	0	0	0	0
1	**4**	0	0	0	0	0	0	0	0
1	**5**	0	8	0	24	0	0	0	32
Total		41	57	41	88	0	0	0	227

Table 4.6 shows an example of a blocking plan at Yard 1 where only three (3) blocks are being utilized. The Block number associated with each O-D pair is given and the model determines whether the block is pure or mixed. Each O-D pair where the O is equal to the D is automatically given a Block # of 0 because this traffic is deemed unnecessary to move from the yard. The remainder are numbered from 1 to *n*-1 (where *n* is the number of yards) in any order determined by the user. The example also shows the blocking plan at Yard 2 where each O-D pair has its own traffic block.

4.3. Guidelines for Train Scheduling

This section describes the process by which a user can create or adjust a proposed train schedule in order to clear the O-D tables (move each O-D pair from its Origin Yard to its Destination Yard) for the analysis period. A train schedule should be based on historical traffic data from

Table 4.6 Example of Blocking Plan

Yard 1		
Destination Yard	**Block #**	**Pure or Mixed**
1	*0*	1
2	*1*	1
3	*2*	1
4	*3*	1
5	*3*	0
Yard 2		
Destination Yard	**Destination Yard**	**Destination Yard**
1	1	1
2	0	1
3	2	1
4	3	1
5	4	1

railroads. Traffic is typically steady state over the whole of the railroad network. This means that once a train schedule is set, it is often in place for a long period of time. Figure 4.4 shows the general steps with respect to creating a preliminary train schedule. This should be created in conjunction with the blocking plan to ensure the optimal coordination.

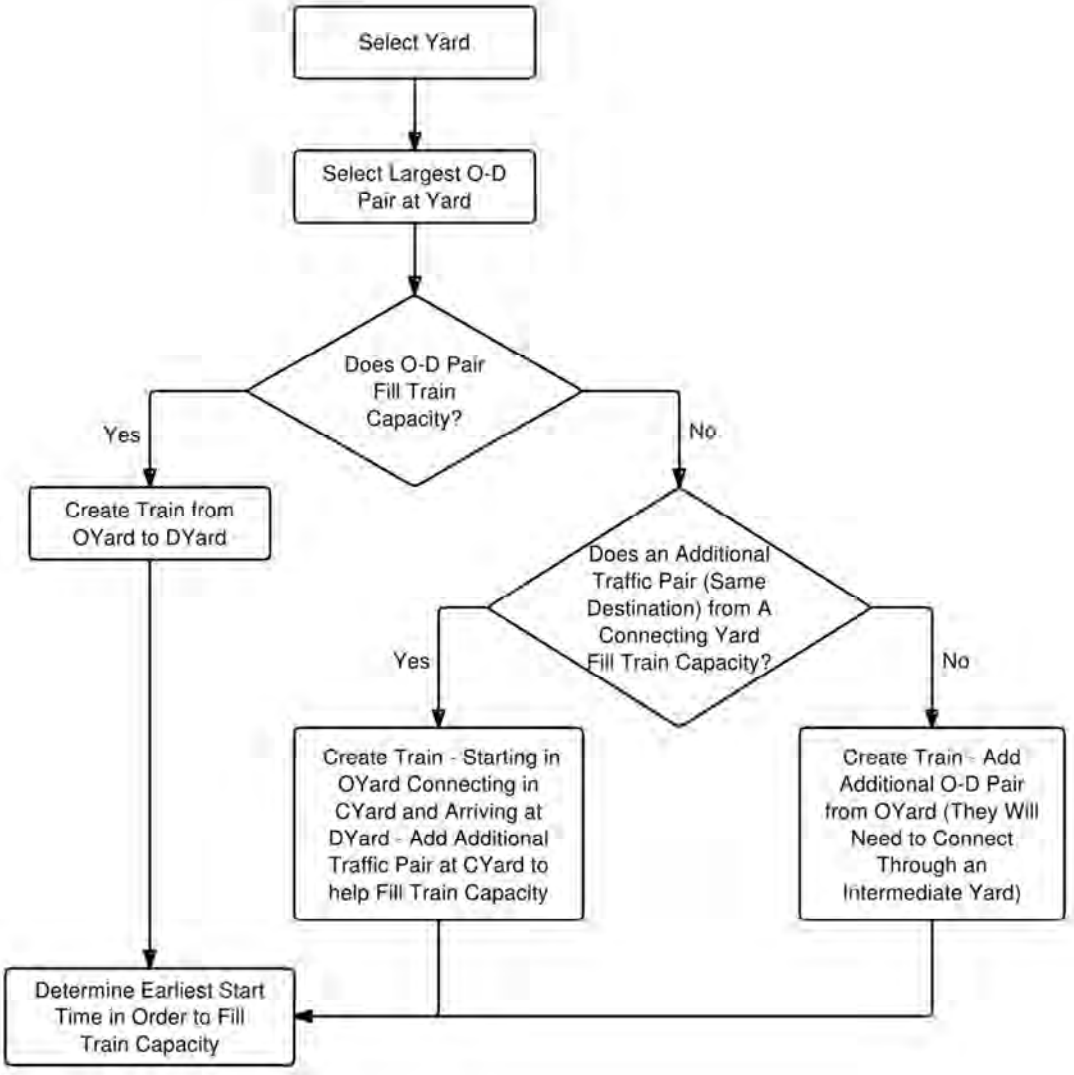

Figure 4.4 Process for Train Scheduling

`The process should be repeated for each yard at least once, in order to assure that there is one train leaving each yard. If there is no originating train at a single yard, at a minimum, a connecting train should be provided, otherwise, all traffic will get stuck at the yard with no way out. Additional trains at each yard should be created on an as needed basis. The process can be repeated bearing in mind the largest O-D demand should be substituted by the remaining largest in a yard, if trains have already been created originating from that yard.

The following provides some insight for the analyst when selecting yards and creating trains. The train plan should start by determining which yard the most O-D traffic originates from and which yard it is destined to. It must be determined whether there are enough cars during the analysis period to fill an entire train or if additional cars may be required. It is important that at least one train terminate at each yard and on at least one train pick-up cars at each yard by originating or by connection at that yard. This will ensure that all cars will be given an opportunity to get transported.

Scheduled times for each train must be determined by the user. This should be done in a way that when a train is built, the cars are readily available in the yard. For example, if 30 cars are available at 1200hrs and an additional 45 cars are not available until 1600hrs then it would be best to build the train at or after 1600hrs. There is a balance of how many cars are available at any given time and how long cars must wait inside a yard at any given time for a train to be built. The sooner traffic is moved from the yard it originates from, the faster it can reach its destination. At the same time, if a train moves with a less than full load of cars then the economies of scale on crew and locomotive costs may not be realized. If there are enough cars for one single O-D pair to fill an entire train, the absolute earliest time for train departure must be assessed in order to limit car waiting time at yards.

Once all of the direct O-D trains (if any) have been established then the remainder of the trains can be established. These remaining trains will carry multiple O-D pairs from one yard to another. These are not necessarily the same yards as the O-D demand, but the progress of each car should be towards its final destination. This is done by looking at the yards which have the majority of the traffic and determining where the traffic is required to go. For example, one can look at Yard 1 which may have O-D pairs of traffic going from Yard 1 to Yard 2 (1-2) and from Yard 1 to Yard 3 (1-3). Assuming that the 1-2 pair has the majority of the cars, but not enough to warrant a 1-2 only train and that the 1-3 pair has some cars but not enough to warrant a 1-3 train, one can build a train which moves the 1-2 and 1-3 O-D pairs from Yard 1 to Yard 2. This way if there are pairs of 2-3 traffic which must be transported from Yard 2 to Yard 3, the 1-3 traffic pair will be combined with the 2-3 pair once it arrives at Yard 2. Yard 2 can now process and forward the 2-3 pair (which is combined with the 1-3 pair that arrived from Yard 1) to Yard 3, its final destination. Additionally, trains which are assigned connecting yards can be very helpful in filling excess train capacity so that it does not go to waste. For example if there O-D pairs for yards 1-3 and yards 2-3 exist with a combined traffic load sufficient to justify a single train, then it makes sense to originate a train at Yard 1, connecting in Yard 2 and terminating at yard 3. On its route it can pick up traffic from Yard 1 (originating) and Yard 2 (connecting).

4.3.1. Train Schedules in the Model

The Train Schedule is entered by the user with the six following characteristics: train number, build time, origin yard, connecting yard, destination yard, and average train speed. The train number is a sequentially numbered system by which the model identifies each train. The number of trains here must match the number of trains identified by the user in the network inputs stage (though this is something which can be revised by the user quite easily). The build time is the

time of day based on a 24-hour clock (or 168-hour clock for a seven day model) and for each train it does not need to be in any specific order. The model determines when each train is built automatically and, then, runs them in order based on a linear progression of time (not train ID number). The origin yard, connecting yard, and destination yard are the yards which a train will travel from, pick up additional cars from, and arrive at respectively. This is assuming that there are routes which connect the origin yard to the destination yard.

If no routes are present, the model informs the user and stops the simulation for that specific train. The final input is the (average) train speed by which the model calculates travel and arrival times for each individual train. For each train the user is also required to enter whether special handling (or priority) is required (this is a 0-1 variable). If the special handling variable is triggered it directs the software to give priority (in any queue conflict situation) at yards, for example, during the time of building a train or at the arrival and classification of cars.

An example of the input data for a train can be found in Table 4.7. In this example there are two trains which run at 0800hrs and 1200hrs respectively. The first of the two trains starts in Yard 1 and ends in Yard 3. The speed is averaged to be 60 km/h and there is no special priority or handling for this train required so the priority section is left as 0. Alternatively train 2 leaves Yard 2, connects in Yard 4, and arrives at Yard 3 travelling at a speed of 70 km/h. Since this train has priority for a reason determined by the user, the priority section is changed to 1. In this manner, the entire train schedule can be entered into the model.

Table 4.7 Example of Train Schedule Input

Train #	Scheduled Build Time	OYard	CYard	DYard	Speed	Priority
1	08:00	1	-	3	60	0
2	12:00	2	1	3	70	1

4.4. Proposed Block Assignment Method

With respect to this thesis and the model it presents, the following routing methodology is proposed. Users will be required to enter train schedule, static blocking, traffic O-D tables and network geography data for the model. The model will suggest block to train assignments (provide multiple options in some cases) for blocks from each yard to trains that are scheduled to depart from each yard. The model also takes into account trains which pass through a yard and conduct pick-up operations for specific blocks at a predetermined yard by assigning pick-up blocks at connection yards. This operation, then, allows the user to intervene in the model operation and adjust the block assignments as desired (one of multiple ways to conduct a sensitivity analysis of the network).

A block can be assigned to one or more trains leaving from the same origin yard. This is important because various trains will travel at various times of the day or week. This means that multiple pickups for a block throughout the day can help to alleviate yard congestion and train capacity overruns by moving freight more frequently. The following flow charts (Figure 4.4, Figure 4.5, and Figure 4.6) show an overview and the step-by-step procedure of the routing algorithm used in this model.

4.4.1. Block to Train Assignment in the Model

Block to Train Assignments are the backbone of the overall train movement simulation. Once a train is scheduled, it needs a pull-list of which cars it is to take on its route. The block to train assignment is the list of which blocks will be attached to each train. The model determines each direct block assignment and each indirect assignment using the methodology presented here.

Although the Block to Train Assignments are recommended by the model, they still need to be manually inspected and adjusted by the user as required. The user may adjust these recommendations in order to represent a more realistic routing pattern or just to test additional routing patterns as desired.

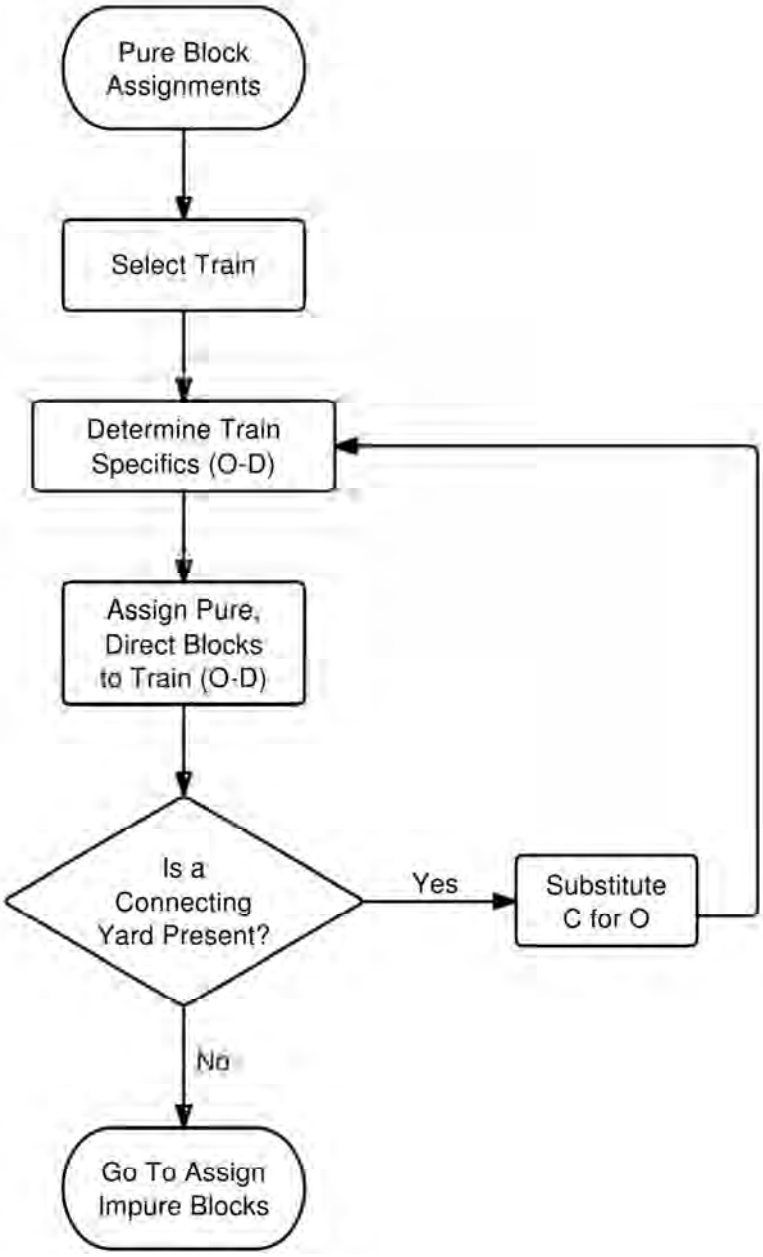

Figure 4.5 Step 1 - Assignment of Direct and Pure Blocks

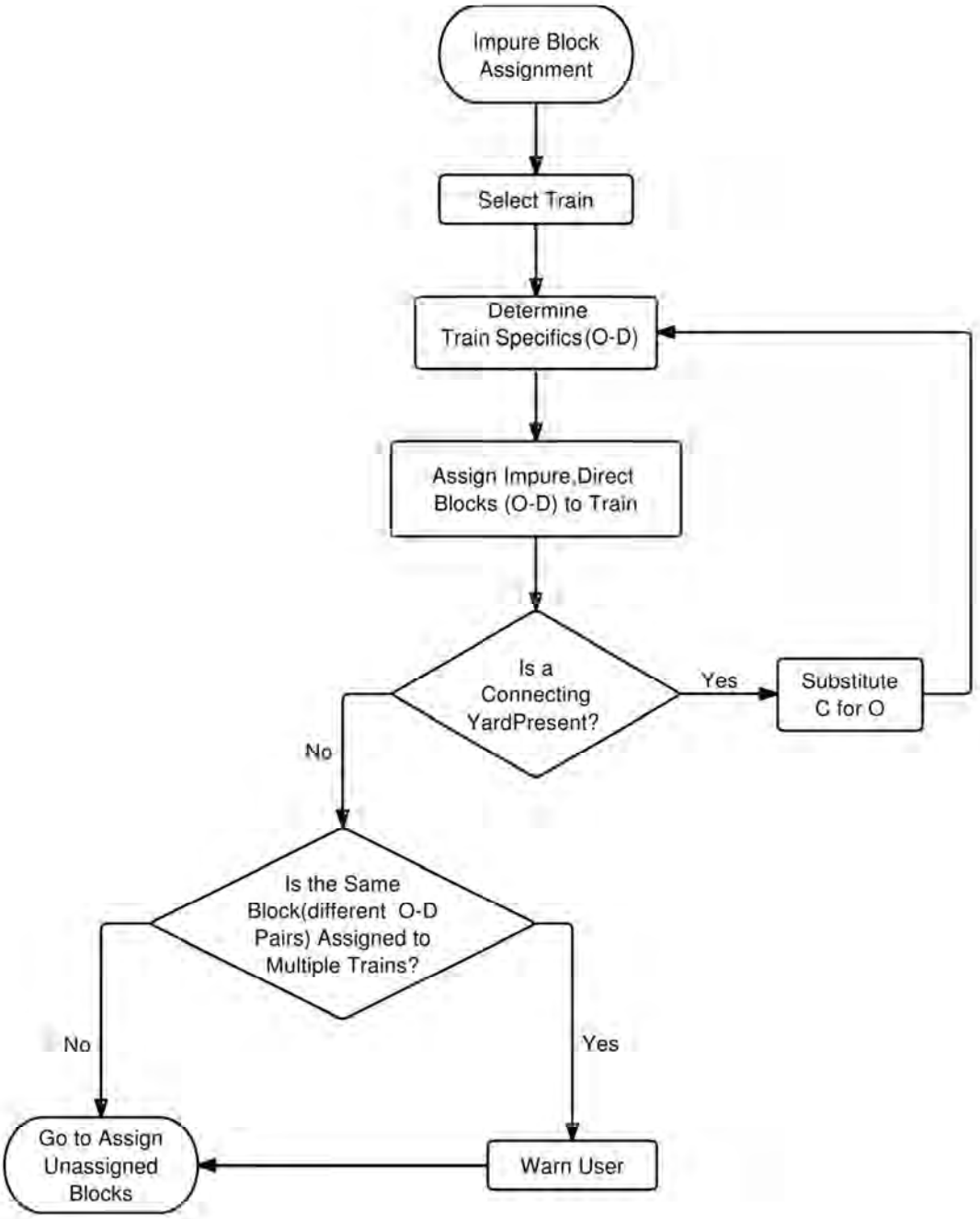

Figure 4.6 Step 2 - Assignment of Direct and Impure Blocks

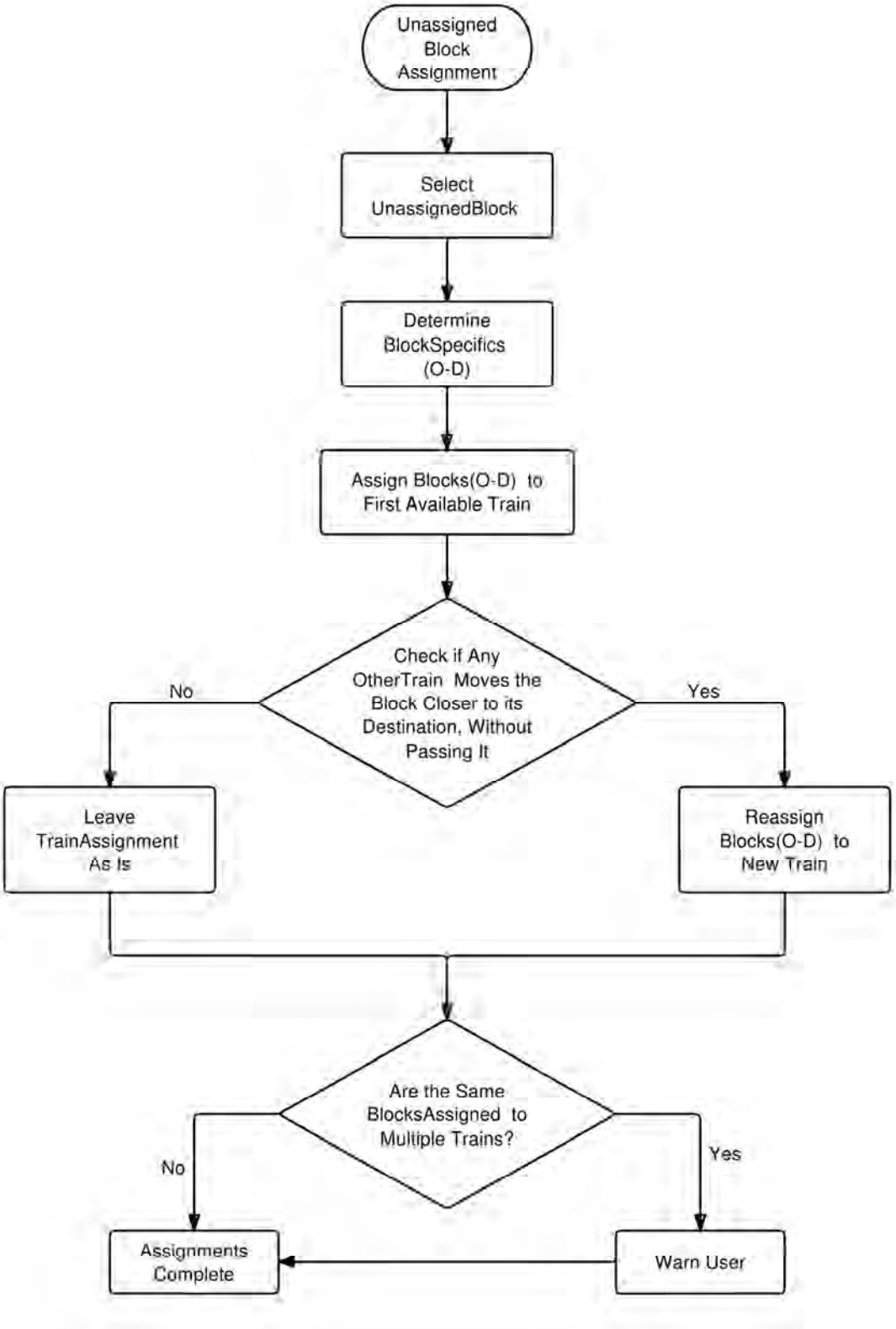

Figure 4.7 Step 3 - Assignment of Unassigned Blocks

Table 4.8 Example of Block to Train Assignments

Train #	OYard	CYard	DYard	Block Assignment	
1	1	-	3	2	-
2	2	1	3	2	2

The example in Table 4.8 shows the block to track assignment as it is applied to Train #1 and Train #2 scheduled as per Table 4.7 and using the Blocking Plan from Table 4.6. As per the methodology developed in this thesis, it can be seen that Block #2 (assigned to O-D 1-3) is assigned to Train #1. This is what is known as a direct, pure blocking assignment since the block is not mixed and the train on which it is assigned is moving directly to the O-D pairs intended Destination. For each connecting yard the block assignment is highlighted in the input cell.

4.5. Proposed Train Routing Method

The model takes the existing route data provided by the user and applies it to the train schedule in order to determine train routes. This is done by determining the shortest direct path between two nodes (bypassing all other yards in between if necessary) and only connecting to one yard if specified by the user. This bypass method in the model allows trains to skip certain yards along a route and thus skip additional classification procedures for cars which do not need it. Building trains which bypass certain yards is a technique railroads have used in order to minimize unnecessary classifications in the past, though it is often only applied to what are called unit trains.

Once routes have been selected, the model calculates the desired arrival times for each train based on length of route and average train speed. The desired build times and arrival times may

have conflicts (i.e. arrival of more than one train at the same time) at each individual yard and, thus, the model calculates where these conflicts occur. If there are conflicts and the trains have the same priority, the later train will get deferred until the earlier train has been processed. If one train has priority over the other, it will be the first one processed irrespective of which train arrived first (with respect to the time it takes to process each individual train at each individual yard). This means the model can anticipate whether it should hold a low priority train in the sidings while a higher priority train is on route. Using this methodology, adjusted arrival and build times are calculated for each train (refer to Section 4.6.2).

At this point the model starts to process the trains in chronological order by build time (irrespective of train ID number). Each block of traffic assigned to the train from the routing process is attached to the train in order of the timeslot of when it is available in the O-D table. This means that cars in the 08:00 hrs timeslot will be attached to the train prior to the cars in the 12:00 hrs timeslot. The only exception to this is if an O-D pair is bolded and deemed priority. When there is a priority O-D pair, it is attached to a train prior to any others. When there are multiple priority O-D pairs, they are attached to the train in chronological order. This is done until the train reaches maximum capacity. Only cars which are ready prior to or at the build time are attached to the train. This process is repeated for each train. If for any train the prescribed minimum number of cars is not reached, the model does not run that train and alerts the user.

Once each train simulation is complete, the cars are added to the post simulation O-D table into their respective timeslot. The model assumes that cars from a train are available in the destination yard based on the actual arrival time plus the switching time (see Section 4.6.2 for more discussion on this). The model moves the cars on the trains and updates the O-D tables, printing the results on the screen in real time (as the individual trains are run). Once the overall

network simulation is complete, the user can determine if all the O-D demands have been achieved or if the user must return to the planning stages and tweak the routing and/or the train schedules to facilitate the full movements of the train. For guidelines on creating and tweaking the train schedule, refer to Section 4.3. An illustrated example of this process is described below.

The following example of train routing helps to illustrate how the model conducts its routing algorithm. From Table 4.9 it can be seen that Train 1 is scheduled to build at Yard 1 at 1200 hrs and depart for Yard 2 whereas Train 2 is scheduled to be built at 1700hrs at Yard 2. The trains will be assigned the shortest routes between the two yards (only two yards since there are no connecting yards) of 55 km (determined from Figure 4.3). Assuming that Train 1 will take only 1-2 O-D traffic, it can be seen that all traffic from 1-2-1 (21 cars), 1-2-2 (33 cars), and 1-2-3 (11 cars) for a total of 65 cars will be taken. Similarly, Train 2 will take a total of 72 cars from 2-1-1, 2-1-2, 2-1-3, and 2-1-4, similarly. Since cars can only be taken from timeslots which are ready at or before the build time of a train, cars from the 2-1-5 timeslot (2000hrs) will not be attached to Train 2. O-D demand 2-1-3 is shown in bold meaning that it is of special priority. This means that it will be handled first when it comes time to build the train. After this O-D is attached, the remainder will be attached to the train and then the train will be ready to depart the station.

Since at least one O-D pair attached to Train 2 is of high priority, the arriving traffic from Train 2 will also be of high priority and is shown as bold text in the O-D tables below. Because each train is travelling a short distance at a reasonable speed, it will reach its destination within the time alloted to the next available timeslot. The cars from Train 1 are slotted to Yard 2 at 1600hrs and the cars from Train 2 are sloted to Yard 1 at 2000hrs. Table 4.10 shows the original O-D data whereas Table 4.11 shows the final results after the trains have been simulated.

Table 4.9 Example of Train Schedule Input

Train #	Scheduled Build Time	OYard	CYard	DYard	Speed	Priority
1	12:00	1	-	2	60	0
2	17:00	2	-	1	70	0

Table 4.10 Example Preliminary O-D Table

From	To	04:00	08:00	12:00	16:00	20:00	23:59	Next Day	Total
1	**1**	3	2	22	1	0	0	0	**28**
1	**2**	21	33	11	42	0	0	0	**107**
Total		**24**	**35**	**33**	**43**	**0**	**0**	**0**	**136**
2	**1**	12	21	**25**	14	32	12	0	**126**
2	**2**	6	3	8	8	3	0	0	**28**
Total		**18**	**24**	**33**	**22**	**35**	**12**	**0**	**154**

Table 4.11 Example Final O-D Table

From	To	04:00	08:00	12:00	16:00	20:00	23:59	Next Day	Total
1	**1**	3	2	22	1	**72**	0	0	**100**
1	**2**	0	0	0	42	0	0	0	**42**
Total		**3**	**2**	**22**	**43**	**72**	**0**	**0**	**142**
2	**1**	0	0	0	0	32	12	0	**44**
2	**2**	6	3	8	73	3	0	0	**93**
Total		**6**	**3**	**8**	**72**	**35**	**12**	**0**	**147**

4.6. Model Outputs

The model outputs various O-D tables, individual train statistics, route use statistics, and individual yard statistics. This section provides an overview and examples of said outputs.

4.6.1. O-D Tables

The most important output of the model is the post simulation O-D table which provides valuable information to the overall function of the simulated operating plan. The tables are represented in place of the original tables as entered by the user. The new tables provide a post simulation view on where cars are located within the network. The end sum of all the cars in the network will be the same as the beginning sum (when the user entered the data) due to conservation of flow.

The operational plan can be judged on whether all of the cars have moved from their origin yards to their destination yards. If there are O-D pairs which have been left behind, then, the Operating Plan has failed to some degree and some additional refinement may be necessary prior to operational implementation. An example of a post simulation O-D table is provided in Table 4.12. In this example the table shows all the traffic to be received at Yard 1 totalling 117 cars through the day. The yard 1 results are desirable, however, the table also shows 27 cars left over in the 1-2 (20:00hrs) O-D. The 27 cars presented here have not been moved to their intended destination and this is not a desirable result. As such, the user would review the operation plan and determine the appropriate adjustment to the plan to fulfill the movement requirements. This can be done by adjusting the train schedule/speed, blocking plan or even the original O-D tables.

Table 4.12 Example Post Simulation O-D Table

From	To	04:00	08:00	12:00	16:00	20:00	23:59	Next Day	Total
1	**1**	1	2	2	0	1	111	0	**117**
1	**2**	0	0	0	0	27	0	0	**27**
1	**3**	0	0	0	0	0	0	0	**0**
1	**4**	0	0	0	0	0	0	0	**0**
1	**5**	0	0	0	0	0	0	0	**0**
Total		**1**	**2**	**2**	**0**	**28**	**111**	**0**	**144**

4.6.2. Train Statistics

Train statistics provide the following: Specific Route; Time for Route; Number of Cars Transported; Amount of Time Required for Switching Cars; and Actual Build, Connection and Arrival Times.

Specific Route

This is the route of travel for the specific train selected by the model based on the shortest route of all the available routes entered by the user. If there is a connecting yard, the model selects two separate routes, one for each leg of the journey between the Origin and Connection yards and the Connection and Destination yards. This can be quite important if, for example, there are residential communities along the given route and noise restrictions are in place. The analyst then would have to readjust the routing plan, or train schedules in order to mitigate any noise restrictions or by-laws in the area.

Time for Route

The model also shows the amount of time it will take a train to traverse a specific route as

selected by the model. This time is calculated using the following equation:

$$T_{train(n)} = S_{train} / L_{route(n)} \quad (4.1)$$

where,

$T_{train(n)}$ = Time required for a specific train to travel the length of the route – for each leg of the route (hours)

S_{train} = Speed of the specific train (km/h)

$L_{route(n)}$ = Length of a specific leg of the route (km)

In the formula above, *n* represents the leg of the route being assessed. Leg 1 (e.g. if $n = 1$) represents the origin yard and/or the route between the origin yard and the next yard (could be connection yard if there is one, or the destination yard otherwise). Leg 2 (e.g. if $n = 2$) represents the connection yard and/or the route between the connection yard and the destination yard.

Number of Cars Transported

This is the total number of cars transported on the specific train after the simulation has been completed. This is calculated by the following equation:

$$NUM_{\boldsymbol{cars}} = {}'NUM_{cars} + O\text{-}D_{cars(i,j,k)} \quad (4.2)$$

where,

NUM_{cars} = the number of cars attached to a specific train in the current iteration/total

$'NUM_{cars}$ = the number of cars attached to a specific train in the previous iteration

$O\text{-}D_{cars(i,j,k)}$ = number of cars in an O-D pair (# of Cars)

i – Origin Yard (yard where the train is being built) (Yard #)

j – Destination Yard (yard where the cars are destined to go – not necessarily the same as the train destination) (Yard #)

k – Timeslot for when the group of cars is being picked up (corresponds to the O-D-time table as described in Section 4.2.5) (Timeslot #)

This is an iterative process by which the overall number of cars is calculated by running the calculation multiple times until all of the required blocks have been attached to the train.

Amount of Time Required for Switching Cars

This is the time required for classification of all cars on a train once they arrive at the train destination yard. This is calculated by:

$$T_{totalswitch} = T_{Prep} + NUM_{Cars} * T_{Switch} \quad (4.3)$$

where,

$T_{Totalswitch}$ = Total time required for switching an individual train at a yard (hours)

T_{Prep} = Preparation time for switching each train (hours)

NUM_{Cars} = Number of cars on a specific train (# of Cars)

T_{Switch} = Time for switching each individual car as input (hours)

Actual Build, Connection and Arrival Times

The actual build time is calculated based on the desired build time entered by the user and the various queues at the respective Origin Yard for each train. If there are no other trains being built at the same time, the actual build time should equal the desired build time. Alternatively, the actual build time for the current train will be delayed based on the length of time required to build for another train which has already begun. The model calculates the actual build time based on a table of trains being built at that given time at that same yard. Similarly, this is done at connections and at arrivals for each yard individually.

The *actual connection time* is calculated using the following equation:

$$T_{connection} = T_{build} + T_{service(1)} + T_{train(1)} \quad (4.4)$$

where,

$T_{connection}$ = Actual arrival time of train into connection yard (hours)

T_{build} = Actual train build time (hours)

$T_{service(n)}$ = Amount of time to build train as per user input for the respective yard (hours)

The *actual arrival time* of a train at a specific yard is calculated using the following equation:

$$T_{arrival} = T_{build} + T_{service(1)} + T_{service(2)} / 2 + T_{train(1)} + T_{train(2)} \quad (4.5)$$

where,

$T_{arrival}$ = Actual arrival time of train into destination yard (hours)

T_{build} = Actual train build time (hours)

$T_{service}$ = Amount of time to build train as per user input (hours)

The $T_{service(2)}$ is divided by 2 to cut the time down in half. This is because this model assumes that the time required at a connection yard to attach cars to a train is not the same as the build time required at the origin yard, since all of the inspection work required with the locomotive and power required is shortened. This model also assumes that the cars are ready prior to the train arriving at the connection yard, thus speeding up the connection process.

The time at which cars arrive in the actual class bowl and are attached to a specific timeslot in the O-D table is calculated using the following equation:

$$T_{arrivalfinal} = T_{build} + T_{service} + T_{train(1)} + T_{train(2)} + T_{totalswitch} \quad (4.6)$$

where,

$T_{arrivalfinal}$ = Actual arrival time of cars into the classification bowl of destination yard such that they are ready to be incorporated into the next train (hours)

The timeslot where the cars are added to at the destination yard for each train is the next closest ($T_{arrivalfinal}$ rounded up to the nearest timeslot interval) timeslot corresponding to $T_{arrivalfinal}$ in the O-D table at that yard. More information on how the queues are calculated in the model are provided in Section 4.6.3.

An example of the output as it is displayed by the model is provided in Table 4.13. Here, the model displays only the final results, not the processes whereby it gets to the final numbers.

4.6.3. Yard Queue Determination

The model assumes a day has 2880 timeslots available to it over a 24 hour period (each timeslot represents a 30 second period of time). With this, the model can assign a specific timeslot(s) to a

Table 4.13 Example of Train Statistics

Train	Route1	Time	Route2	Time	# of Cars	Switching Time	Calculated Time		
							OYard	CYard	DYard
1	13	00:45	-	-	**87**	01:03:30	08:00	-	08:45
2	21	00:47	13	00:39	**56**	00:48:00	12:00	12:47	14:01

train based on how long it needs each yard's resources. Each timeslot is initially assigned a 0 which is replaced by a specific train number as the model progresses. Each yard has its own 2880 timeslots and each train is assigned to those slots as it uses the specific yards.

The model begins by placing priority trains into their respective timeslots first, followed by regular trains. This process involves assigning one train at a time to the timeslots provided. For each recurring train after the very first one, the model starts to determine whether there are any conflicts with that train at that specific yard. If the starting time for the new train is in conflict with another train (e.g. the timeslot is already full), the model determines the next available slot and fills it with the new train. The model also ensures that the entire time required for the train at the yard is in one single slot (e.g. the timeslots are all sequential) so that there are no stop and starts for any given train. The model processes each train in order of entry (e.g. train 1 is slotted prior to train 2) with exception to priority trains.

This process is completed for each train being built at the origin yard first, then the connecting yard and then finally the destination yard. Since the assigned timeslots determine when the trains will actually be built at the origin yard, new connection and arrival times must be generated for each train. The model recalculates the new times for each train individually using the same methodology as in calculating the desired times. The model, then, continues by calculating the

connecting yard timeslots for building trains and determines the new times based on the results. After this, the model calculates the arrival timeslots for each yard. Since the switching time is dependent upon the number of cars on a train (see Equation 4-6), during the first iteration of the queue determination, the model assumes each train carries 75 cars.

This is done so that a relatively more accurate picture of how many cars will be on each train can be gathered. Since the model runs each train individually, in chronological order, the arrival times of cars into their respective yards' class tracks can affect later trains' lengths. This is the reason the queue determination is run twice, once to determine semi-accurate train times and again to determine more accurate times with train lengths taken into consideration. The second iteration of the queue determination model follows the same processes (with the addition of a determination of how many cars will be on each train).

4.6.4. Yard Statistics

At any given yard, the model provides information of how many cars were initially at the yard at the beginning of the analysis period and how many additional cars are brought in and have been classified in the yard at the end of the analysis period. This number includes cars which are destined to end their trip at the yard and cars which are just connecting through. Additionally the amount of time the classification engines (either hump or flat) are in use is also recorded as a sum of all the individual trains throughout the day. This gives an overall amount of time when the switching was being completed throughout the day.

The number of trains which have originated, connected through, or arrived at each yard is also provided for the analyst. The train numbers can be very useful to an analyst when looking at

staffing levels at a yard and maintaining levels for the number of trains being processed. An example of this set of outputs can be seen in Table 4.14.

Alhough the yard statistics output table shows the amount of time that classification occurred in a day, it does not show the exact times when it occurred. There is another set of outputs, related to yards specifically, which show yard usage at 30 second intervals throughout the day. This is with respect to the schedules of each train and queuing at each yard. The model presents data on the time when each specific train is using any given yard resource (building or switching). This is important because with the information of when the yard is in use and how much traffic a yard must process the analyst can recommend staffing requirements at each yard. This table is also the backbone of how the model develops actual train movement times as opposed to the desired times provided by the user which can often conflict. An example of the yard queuing output is provided in Table 4.15.

Table 4.14 Example of Yard Statistics

Yard	**1**	**2**
Cars (Initial)	191	140
Cars (Additional)	212	71
Classification Time Used	04:00	01:00
# of Train Origins	3	1
# of Train Connections	0	1
# of Train Destinations	4	2

Table 4.15 Example of Queuing at Yards

Yard	**Time**	**Train - Origin Building Process**	**Train - Destination Classification Process**
1	22:33:30	6	11
1	22:34:00	6	11
1	22:34:30	6	11

4.7. Analysis of Model Results

In order to create a straightforward and consistent form of analyzing the results provided by the model, several factors must be considered simultaneously. The major factors, in order of most to least importance, to be reviewed by the user, should include the following:

a. Total number of individual trains per operations plan: This is highly connected with railroad costs due to high start up prices for each individual train.
b. Total number of switches for all yards in the entire network: This is highly correlated with railroad service times and reliability, more switching inadvertently means more intermediate yards and more time spent waiting for trains as opposed to actual movement of goods; and
c. Total car-distance travelled on entire network: This is highly correlated to the environmental impacts trains have in that minimizing the distances traveled will also minimize fuel consumption and emissions created by locomotives.

The total number of individual trains can be retrieved from the train schedule as created using the methodology developed in this thesis. The number of trains should be translated into a per day basis. This means that if a train runs every two days, it is equivalent to one half a train running in one day. The translation is important because each operating plan may have different schedules and they have to be compared at the same level for the analysis to be effective. The total number of switches in the network is the sum of all the additional cars in each yard as provided by the yard statistics output (reference Equation 4.7). The total car-distance travelled on the network is calculated using Equation 4.8. This provides the amount of overall travel in car-km for the entire

network. Minimizing these three individual main criteria will result in an overall minimized cost and minimized impacts for the entire railroad network.

$$Numswitches = \sum CarsAdd_n \quad (4.7)$$

where,

Numswitches = total number of switches on the entire network.

$CarsAdd_n$ = number of additional cars at yard n; and

$$D_{CAR} = \sum [(R_{1x} + R_{2x}) * NumCars_x] \quad (4.8)$$

where,

D_{CAR} = the total distance of travel for all cars involved (car-km);

$(R_1 + R_2)$ = the distance of each specific route for any given train;

NumCars = the number of cars on any given train; and

x = a given the train involved with the operations plan.

This analysis is best completed using a tabular format. Table 4.16 shows an example of how to best implement this. While assessing the three criteria individually amongst each of the operating plans, the user can insert the actual number value for each criterion in the analysis table. After each operating plan has been entered into the table, the user may assign each plan and criterion with a rating from 1 to *n*, where 1 is the best and *n* is the worst (*n* being the number of operations plans under review). In tallying up all of the ratings for each of the plans, the user will be left with an overall score by which each plan can systematically be assessed based on the core

Table 4.16 Analysis Table

Criteria	**Weight**	**Plan 1**		**Plan 2**		**Plan 3**	
		Value	Rating	Value	Rating	Value	Rating
Number of Trains	3	10	3	8	1	9	2
Number of Switches	2	275	3	268	2	268	2
Total Car-Distance	1	1980	3	1895	2	1830	1
Non-Weighted Totals		9		5		5	

criteria aforementioned. It is important to note that some users may find it beneficial to add a weighting to each of the three criteria since the implications associated with each are quite different. A suggested weighting for the criteria as listed above (a, b, and c) is 3, 2, and 1 respectively. This weighting reflects the costs for the most expensive to least expensive criteria. This ensures that the final analysis will be conducted such that all operational costs are minimized.

To review an example of this assessment system, refer to Table 4.16. In this example, three hypothetical plans have been created and compared against each other. Plan 1 has the most points followed by Plans 2 and 3 which both have an equal amount of points when considering the non-weighted assessment criteria. The lower number of points indicates a better operations plan in terms of optimization, therefore, Plans 2 and 3 would tie for the best plan. However, when the weighting is applied it is clearly seen that Plan 2 has (3*1 + 2*2 + 1*2 = 9) points and Plan 3 has (3*2 + 2*2 + 1*1 = 11) points. With this hypothetical scenario, since Plan 2 has the lowest number of points overall, it is easy to see that it is the most economical option and should be regarded as the ‘best alternative’ solution. Other plans can be analyzed similarly. In addition, other factors can be developed in order to help assess the best alternatives but are beyond the scope of this thesis.

4.8. Model Capabilities

The model capabilities are relevant to the creation and optimization of operations plans for a given network. This is a versatile model which has many capabilities including the following:

1. The primary function of this model is to assist the user in creating, testing and analyzing various operating plans on a railroad network. The model can be used to create a preliminary operations plan including blocking, block to train assignments and train schedules as well as train routing. This feature allows the user to create and update plans as information is brought together from outside sources. This means that a user can look at the operations of an existing network or of a brand new network without changing models or software;
2. The model can analyze what-if scenarios with different train schedules and blocking plans by adjusting or recreating new plans to simulate. Doing this, the user can review the various outputs and analyze each one independently or together as one. This will allow the analyst to determine which plan is the 'best option' considering not only train traffic, but also yard and route impacts;
3. In terms of routing, the model automatically chooses the shortest path available to it. This means that all shipments will travel the shortest possible distance as entered into the model. Sometimes there are scenarios where the user may not want the shortest path to be used; this is simply overcome by manually adjusting data in the model.
4. The model can be used to determine what effects traffic pattern changes would have on the network; and

5. The model can assist the analyst determine what impact the closing of a yard or certain routes would have on specific train traffic. This is especially important when testing what-if scenarios with respect to natural disasters such as flooding (like what happened in New Orleans during the aftermath of Hurricane Katrina).

4.9. Model Limitations

This section discusses some of the model's limitations as it is applied in a real work application. Though these limitations are apparent in the model, they can be overcome with some further development of the model and its accompanying software application.

1. Each yard is setup such that it only has one set of hump/classification tracks inbound and only one set of outbound tracks. This means that yards that have multiple leads into their classification tracks or multiple departure tracks are not accurately modelled. In this scenario, the best an analyst can do is to determine a combined average for the car switching and train setup times as entered into the model and to conduct a sensitivity analysis. With varying train setup and car switch times, the user can determine how the whole network will likely respond to an operations plan.
2. Each car is assumed to be similar in that various weights or lengths are not incorporated into the model. In reality, there are many different types of cars which can factor how a yard and a railroad will block their traffic. In reality, blocking is not based solely on final destination of cars but also on what they carry and what style of car they are (such as box car, hopper car, etc.).

More discussion with respect to limitations is provided in the conclusions and recommendations of this thesis.

4.10. Summary

This chapter has reviewed the guidelines used in the creation of railroad operations plan and the model created for the purpose of simulating operation plans. The overall model as well as specific inputs and outputs were discussed, with examples. Most importantly, a guide for assessing various operations plans in a common way was presented. Finally, this chapter has discussed some of the various capabilities and limitations of the model.

From the literature review, it was determined that reliable and cost effective deliveries of goods are the most important characteristics railroads and customers are looking for. Effective operation plans, priority traffic management plans, and management of emergency situations (rerouting traffic, for example) are essential to reaching the aforementioned goals. The main objectives of this thesis were to develop a model and guidelines which would assist in the development, testing, and analysis of operation plans. In turn, the various, integrated, operation plans can be compared against one and other resulting in a “best case” or “optimized” solution which would satisfy reliability and cost effectiveness as mentioned above. Guidelines for this type of analysis along with operation plan development developed in this thesis are easy to comprehend and implement in real life situations.

Existing railroads currently rely heavily on the skills and experience of veteran employees, who make decisions about train schedules, product development (blocking plans) and how they split up cars into the bowl (classification tracks). Since there was not a large amount of literature with

respect to guidelines on how to develop a train schedule, how to develop block to train assignments and how to analyzing multiple operation plans by comparing simulation results , this thesis focused on developing these guidelines. Guidelines with respect to the development of train schedules are logical and intuitive. The easy to comprehend flowchart provides a framework for creating the most logical and economical train schedule. The model is able to simulate railroad networks in hours, days, or weeks and thus is very versatile when it comes to developing train schedules. Guidelines with respect to the assignment of blocks to trains were provided using a three-step system which uses human intelligence to maximise the efficiency of car movements using scheduled trains. This system can be tested by the model and various blocking plans can be created, tested and analysed to determine the best case blocking plan and the block to train assignments. The final guidelines involved the assessment of multiple, integrated operations plans in a common way. This assessment uses a quantitative and qualitative comparison chart and can be customized to the analysts / railroads needs and goals.

Additionally, current practices of railroad companies prevent priority traffic from getting special treatment and all traffic is treated in the same manner. With the exception of unit trains which bypass yards completely, each train / car is served on a FIFO basis at every yard it reaches. In terms of reliability for high paying customers and high priority traffic, this is a great hindrance. This model addresses this issue by allowing for simulation of priority traffic, for both cars and trains, being handled ahead of regular traffic.

5. MODEL VERIFICATION AND APPLICATION

A user friendly software application was created in order to assist in the use of the model developed in this thesis. The data entry, network simulation, train routing and outputs follow the process as described in Chapter 4. This chapter presents a worked example of the model using hypothetical data in order to exhibit the solution process and analysis techniques. This chapter also helps to highlight some of the capabilities of the model which are, then, discussed in detail.

5.1. Model Software Verification

The following section shows the verification of the model software comparing the results of the software analysis versus the manually completed analysis. For the verification of the model and the software, a small and simple network of three yards was used. Each yard was connected to the other in a triangular formation and had traffic destined for each of the other yards. Comparing the model results and outputs to what is expected of the model will help to verify that the model software works correctly.

5.1.1. Data Inputs

The following image (Figure 5.1) shows the yard network setup and distances between yards. For this hypothetical network data were entered into the software. The same data were used for the manual calculations which were used to validate the application. The following tables show the various inputs as required by the model. Descriptions and requirements for the Data are discussed in Section 4.2. Table 5.1 shows the specifics of each yard and Table 5.2 shows the various possible routes as entered into the model.

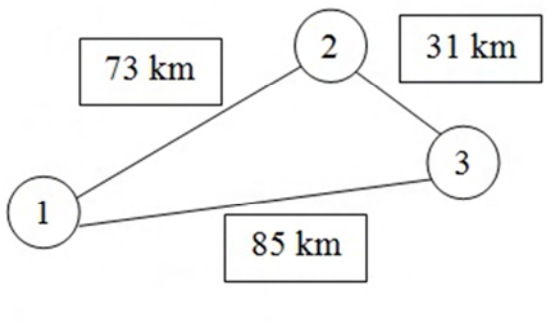

Figure 5.1 Network Configuration for Verification of Model

Table 5.1 Yard Inputs Settings (Model Verification)

Yard Number	Type of Yard	Switching Time (Per Train - Minutes)	Building Time (Per Train - Minutes)
1	Flat	00:30	00:30
2	Hump	00:30	00:30
3	Hump	00:30	00:30

Table 5.2 Available Routes (Model Verification)

Route #	Origin Yard	Destination Yard	Route Path (Yard to Yard)	Route Length (km)
1	1	2	12	73
2	1	3	123	104
3	1	3	13	85
4	2	1	21	73
5	2	3	23	31
6	3	1	321	104
7	3	1	31	85
8	3	2	32	31

Table 5.3 shows the original O-D tables for the network. The goal will be to move all of the O-D demands from their origins to their destinations, regardless of whether the situation is optimal or not. This is just for verification purposes. Using the methodology created in Section 4.3, the train schedule was created (as shown in Table 5.4). After all the data, including the blocking data (as shown in Section 5.1.2), was entered into the model, the model was run and the outputs were reviewed for consistency with manual calculations.

Table 5.3 Original O-D Tables (Model Verification)

From	To	04:00	08:00	12:00	16:00	20:00	24:00	Next Day	Total
1	**1**	0	10	0	0	0	0	0	30
1	**2**	0	10	0	0	0	0	0	0
1	**3**	0	10	0	0	0	0	0	0
Total		**0**	**10**	**0**	**0**	**20**	**0**	**0**	**30**
2	**1**	0	0	10	0	0	0	0	0
2	**2**	0	0	10	0	0		0	30
2	**3**	0	0	10	0	0	0	0	0
Total		**0**	**0**	**20**	**0**	**10**	**0**	**0**	**30**
3	**1**	0	0	0	10	0	0	0	0
3	**2**	0	0	0	10	0	0	0	0
3	**3**	0	0	0	10	0	0	0	30
Total		**0**	**0**	**0**	**30**	**0**	**0**	**0**	**30**

Table 5.4 Train Schedule (Model Verification)

Train #	Scheduled Build Time	OYard	CYard	DYard	Speed	Priority
1	08:00	1	-	2	50	0
2	12:00	2	-	3	50	0
3	16:00	3	2	1	50	0
4	16:30	3	-	2	50	0

5.1.2. Outputs

The model outputs, as described in Section 4.6, were reported by the software application as predicted. The four trains moved all of the cars to each of their respective destination yards. The model assigned blocks to each train as per the blocking plan (shown in Table 5.5) using the methodology provided in chapter 4 earlier. After the trains were simulated, the yard statistics as well the train statistics were calculated and provided in the software interface. As per the train schedule, a total of two (2) trains used facilities at Yard 1, four (4) trains at Yard 2 and three (3) Trains at Yard 3. Each of the trains carried 20 cars except for Train 4, which carried only 10 cars. Table 5.6 presents the final O-D tables of the three yards used for the model verification, Table 5.7 provides the yard specific model outputs for the verification, and Table 5.8 shows the train specific outputs including train arrival times at destination yards.

5.1.3. Model Verification Summary

In order to determine the block assignment algorithm one can review how the model assigned O-D's 1-2 (Block 1) and 1-3 (Block 2). Since there was only one train originating from yard 1, it makes sense that all the outgoing blocks be assigned to train 1. This was in fact the case. If one were to look at Yard 3, where there were two trains originating from the same yard, train 3 was traveling to yard 1 and train 4 was traveling to Yard 2, logically all O-D traffic corresponding to those yards should be on trains 3 and 4 respectively. This was the case when the model was run, the model assigned Block 1 (O-D: 3 -1) and Block 2 (O-D: 3 -2) to trains 3 and 4 respectively.

The O-D demands were met by using only 4 trains which simulated the movement of all cars to their respective yards. There were no carry over cars which remained in either of the yards overnight and no trains which travelled overnight. Because this was a model verification, no

minimums on train size were declared and the maximum was set to 999 in order for the model to assume train capacities negligent.

Table 5.5 Static Blocks Input (Model Verification)

Yard			Pure or	Train
O	D	Block	Mixed	Assignment
1	**1**	0	1	
1	**2**	1	1	1
1	**3**	2	1	1
2	**1**	1	1	3
2	**2**	0	1	
2	**3**	2	1	2
3	**1**	1	1	3
3	**2**	2	1	4
3	**3**	0	1	

Table 5.6 Final O-D Tables (Model Verification)

From	To	04:00	08:00	12:00	16:00	20:00	24:00	Next Day	Total
1	**1**	0	10	0	0	20	0	0	30
1	**2**	0	0	0	0	0	0	0	0
1	**3**	0	0	0	0	0	0	0	0
Total		**0**	**10**	**0**	**0**	**20**	**0**	**0**	**30**
2	**1**	0	0	0	0	0	0	0	0
2	**2**	0	0	20	0	10	0	0	30
2	**3**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**20**	**0**	**10**	**0**	**0**	**30**
3	**1**	0	0	0	0	0	0	0	0
3	**2**	0	0	0	0	0	0	0	0
3	**3**	0	0	0	30	0	0	0	30
Total		**0**	**0**	**0**	**30**	**0**	**0**	**0**	**30**

Table 5.7 Yard Specific Outputs (Model Verification)

Yard	1	2	3
Cars (Initial)	30	30	30
Cars (Additional)	20	30	20
Classification Time Used	00:30:00	01:00:00	00:30:00
# of Train Origins	1	1	2
# of Train Connections	-	1	-
# of Train Destinations	1	2	1

Table 5.8 Train Specific Outputs (Model Verification)

Train #	Route1	Time	Route2	Time	# of Cars	Calculated Time (OYard)	Calculated Time (CYard)	Calculated Time (DYard)
1	12	01:27	-	-	**20**	08:00	-	09:57
2	23	00:37	-	-	**20**	12:00	-	13:07
3	32	00:37	21	01:27	**20**	16:00	17:07	19:04
4	32	00:37	-	-	**10**	16:30	-	17:37

5.2. Model Application: Single Plan

The following section represents an example of model use, and analysis of the results. The network topology as well as traffic data are provided and the analyst's actions are mimicked in order to present the various capabilities of the software. The objective of this specific model will be to create a train schedule and block assignment plan which will clear the O-D tables (move all pairs to their final destination) within the analysis timeframe. Any lagging O-D pairs which have not made their final destination will be addressed also. Since the model is a simulation, multiple iterations must be performed and results compared against one another in order to determine whether the plan is the "best case" scenario.

5.2.1. Initial Process and Model Setup

This section provides a step-by-step process for operating the train model in the software application. This general process will guide the user through the major steps in implementing the model for a given operations plan. The application itself is described through the discussion in the following sections of this chapter.

1. Enter General Data (Network, Yard and Route Data);
2. Click ‘Next’ Button to Continue;
3. Enter O-D Data;
4. Click ‘Create OD Array’ Button to Continue;
5. Formulate and Enter Train Schedule;
6. Click ‘Block Setup’ Button to Continue;
7. Click ‘Assign Blocks’ Button to Continue;
8. Review Block Assignments and Adjust as Required;
9. Click ‘Time Calculations’ Button to Continue;
10. Click ‘Run Trains’ Button to Continue;
11. Review and Analyze Train/Yard Results;
12. Make Adjustments to Various Model Data as Required; and
13. Repeat Steps 1 through 12 as Necessary – and re-run the Model.

Application Data

The data used for the model is hypothetical in nature. This data reflects a network with yards in relatively close proximity of each other (as the time horizon for this study is only one day). Since

the model in this thesis has been developed for a local/regional sized application and analysis, this hypothetical network suits the model well. The network consists of eight (8) yards connected together as shown in Figure 5.2. Each circle represents a yard and each inscribed number represents the yard ID. The distance between yards is listed along each connecting route. The yard data are presented in Table 5.9. For this network, 72 routes were entered manually in the software (see Appendix 2 for List of Routes) based on the network configuration as shown in Figure 5.2. Using the routes list input by the user, the model calculated the shortest distance between each yard (see Table 5.10). The time separated original daily O-D Data can be found in Appendix 2.

Initial Train Schedule Setup

Using the daily O-D data, the train schedule was set up. Using the methodology in Section 4.3, the train schedule was created as shown in Table 5.11. This table also presents the assumed speed of the train and whether the train needs any special handling (or prioritization). For the purpose of this study, it was assumed that the trains would have various speeds depending on which yards they started at. Though this is not necessarily how train speeds are determined in reality (various factors play a role) it will suffice for the purpose of presenting this application.

For the minimum and maximum number of cars per train, 0 and 999 were assumed. This allows the model to run trains no matter what their size is and allows the user to analyze the train schedule based on actual simulated car numbers. This way a user can decide that a train may not be able to run every day, rather it may be possible to run a full train every two or more days instead. If the limits had been set in place the trains that did not meet the minimum or maximum requirements would not have run in the model.

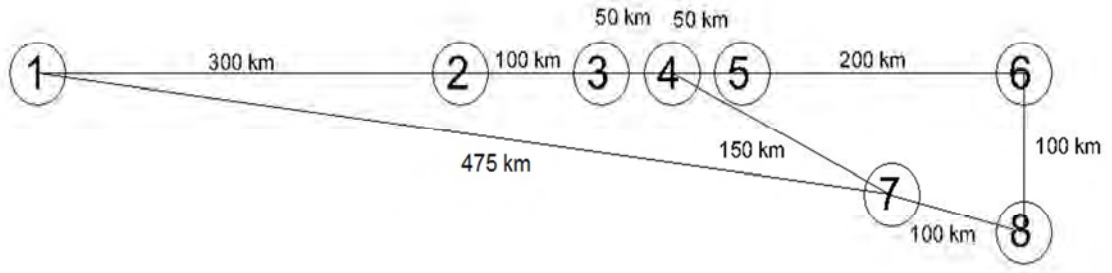

Figure 5.2 Network Topology and Yard Location

Table 5.9 Yard Data for Application

Yard Number	Type of Yard	Switching Prep Time (Per Train - Minutes)	Switching Time Per car (Minutes)	Building Time (Per Train - Minutes)
1	Hump	00:15:00	00:00:30	00:30:00
2	Hump	00:15:00	00:00:30	00:30:00
3	Flat	00:20:00	00:00:45	00:30:00
4	Flat	00:20:00	00:00:45	00:30:00
5	Flat	00:20:00	00:00:45	00:30:00
6	Hump	00:15:00	00:00:30	00:30:00
7	Flat	00:20:00	00:00:45	00:30:00
8	Hump	00:15:00	00:00:30	00:30:00

Table 5.10 Shortest Path Matrix

Yard	1	2	3	4	5	6	7	8
1	0	300	400	450	500	675	475	575
2	300	0	100	150	200	400	300	400
3	400	100	0	50	100	300	200	300
4	450	150	50	0	50	250	150	250
5	500	200	100	50	0	200	200	300
6	675	400	300	250	200	0	200	100
7	475	300	200	150	200	200	0	100
8	575	400	300	250	300	100	100	0

This would mean that all traffic assigned to non-simulated trains (due to minimum car requirements not being met) would be deferred to the next day for movement at the same originating yard thus creating more congestion and longer wait times at each yard. The model determined the most logical route for each train (based on shortest distance) and calculated the actual times based on the desired build time, routes and speed. The model determined data can be found in Appendix 3.

Priority Traffic

Prioritization (special handle) of a train or O-D pair determines whether a train will be built or loaded prior to any others which may have the same desired build time. This provides the analyst an opportunity to assure that time sensitive traffic can be moved without delay. Since the movement of train traffic is often based on large scale contracts (such as with major car manufacturing companies), it is in the best interest of railroads to meet delivery times.

Block Assignment and Movement of Cars

The blocks were split up in the model as per O-D pairs at each yard meaning that there are a total of seven (7) blocks at each yard. The model automatically assigns the block assignments to each train as discussed in Section 4.4. Once this has been completed, the model is ready to run the trains. The model generated block assignments can be found in Appendix 3.

The model, then, runs the trains to determine the movement of cars based on the O-D tables, train schedule, and block assignments. When the model is run for the first time, there may be areas which are flagged to the user, such as cars which have not reached their final destination or

trains which have abnormally high or abnormally low usage of capacity. The results of the initial iteration (O-D tables, yard statistics and train statistics) can be found in Appendix 3.

Table 5.11 Train Schedule (Model Application)

Train ID #	Build Time	Origin Yard	Connection Yard	Destination Yard	Speed (km/h)	Special Handle
1	20:00	1	-	2	50	1
2	20:00	1	-	5	50	0
3	20:00	1	-	6	50	0
4	20:00	1	-	7	50	0
5	20:00	1	7	8	50	0
6	20:00	1	-	3	50	0
7	20:00	2	4	6	70	0
8	20:00	3	-	1	50	0
9	16:00	3	5	6	50	0
10	23:59	4	-	2	70	0
11	12:00	5	-	1	50	0
12	20:00	5	-	1	50	1
13	20:00	5	-	8	50	0
14	12:00	6	-	1	70	0
15	20:00	6	-	1	70	0
16	20:00	6	-	4	70	0
17	20:00	6	-	5	70	0
18	20:00	6	-	8	70	0
19	20:00	7	2	1	50	0
20	08:00	8	-	1	60	0
21	16:00	8	-	1	60	0
22	08:00	8	7	6	60	1
23	16:00	8	7	6	60	0
24	16:00	8	-	7	60	0

5.2.2. User Intervention and Subsequent Iterations

After the initial results of the model are determined, it is important to revisit the initial train schedule, blocking assignments and routing of cars, especially if areas are flagged. This must be done after some analysis of the O-D tables, yard statistics and train statistics has been completed. Any areas which are flagged by the model, or by the user, can be addressed at this stage. The typical areas of concern are improper utilization of train capacity, unmoved or inadequately moved O-D combinations, extended delays for rail cars, and improper utilization of yard capacity. They can be addressed by the user in a variety of ways which involve adjusting original operational plan characteristics and data in the model.

The user can, then, adjust the train schedules such that it includes additional or fewer trains than when they leave. The time at which a train leaves makes a difference in how many cars it can carry and the arrival time at the destined yard because these can affect trains which are being built at the destination yard. Additionally, the user can route non-pertinent traffic away from heavily used yards or toward yards with unused capacity as required. Similarly, adjustments can be made with the block assignments, to address areas of concern with train capacity utilization.

5.2.3. Results and Analysis

This section will provide a summary of the model results and compare the final iteration results with the first iteration results. Table 5.12, Table 5.13 and Table 5.14 show a summary of the original, overall O-D table, the overall table after the first iteration and the overall table after the final iteration; the numbers in bold represent the traffic which was not moved from its original location to its final destination. It can be seen that both the first and the final iterations route the majority of the freight traffic to their final destinations, but not all of it. The differences between

how the traffic moves between the first and final iteration may seem small and inconsequential, but do make quite a significant difference. This is because the routing can affect which yards are in use and at what times, as well as which trains are filled or not filled.

An example of one difference is seen with the 6-2 O-D pair. Since there is no direct train to yard 2 from yard 6, the 6-2 pair must be moved through an intermediate yard. The first iteration which moved the traffic using only the model suggested block assignments would have the 6-2 pair move from yard 6 to yard 1 and then back to yard 2 on a separate train. This means an additional 600 km of movement for 38 cars. When some adjustments are applied to the blocking plan, the 6-2 block can be put on a train going from yard 6 to yard 4. In this manner the cars travel a lesser distance overall, and can be picked up by the next available 4-2 train. This is very beneficial to the 4-2 train, as it helps to maximize its capacity usage in the future since train 4-2 only has 13 cars on it after 24 hours. For a more detailed, time separated, O-D table, refer to Appendix 2, Appendix 3, and Appendix 4 for the original, first iteration, and the final iteration data respectively.

Table 5.15 shows the train specific results (block assignment and # of cars on the train at the end of the analysis period). The block assignments which are highlighted correspond only to the connecting yard. The table also has notes in the right most column regarding how frequently each train should operate. It is assumed that the minimum amount of cars to justify running a single train is a hard 60 with a desirable minimum amount of cars being a soft 80. The assumed hard maximum number of cars is 150. This means that any trains with less than the absolute minimum amount of cars should have some special instructions such as running on different intervals (not daily). If a train is deemed unnecessary, then alternative solutions for moving the

O-D pairs associated with that train should be made and the blocking strategy and block assignments should be adjusted appropriately.

Table 5.12 Original O-D Table

Yard	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	6	58	25	0	102	140	47	63
2	30	0	0	0	28	82	0	0
3	65	0	0	0	0	21	0	25
4	0	13	0	18	0	13	0	0
5	171	0	0	0	0	60	0	140
6	182	38	0	103	23	0	0	121
7	72	0	0	0	0	18	0	45
8	176	19	0	0	32	117	22	33

Table 5.13 O-D Table – No User Intervention

Yard	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	702	**57**	0	0	**24**	0	0	**25**
2	0	71	0	0	0	0	0	0
3	0	0	25	0	0	0	0	0
4	0	0	0	121	0	0	0	0
5	0	0	0	0	133	0	0	0
6	0	0	0	0	**28**	451	0	0
7	0	0	0	0	0	0	69	0
8	0	0	0	0	0	0	0	402

Table 5.14 O-D Table – Post User Intervention

Yards	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	702	**16**	0	0	0	0	0	0
2	0	71	0	0	0	0	0	0
3	0	0	25	0	0	0	0	0
4	0	**38**	0	121	0	0	0	0
5	0	0	0	0	157	0	0	0
6	0	0	0	0	**28**	451	0	**25**
7	0	0	0	0	0	0	69	0
8	0	0	0	0	0	0	0	402

Table 5.15 Train Specific Results

Train ID#	Block Assignment							# of Cars	Analyst Notes
1	1							**61**	Run daily (initially allow a one day lag for accumulation of additional cars from other yards – additional 19 cars per day).
2	4	3						**102**	Run Daily
3	5							**140**	Run Daily
4	6	6						**47**	Run Train every 2 days to allow for internal accumulation of additional cars.
5	7	7						**108**	Run Daily
6	2							**25**	Run Train every 3~4 days to allow for internal accumulation of additional cars.
7	5	5	2	3	4	6	7	**123**	Run Daily
8	1	2	3	4	6			**65**	Run Daily
9	5	5	7					**106**	Run Daily
10	2	1	3	4	6	7		**13**	Run every 2 days to allow accumulation of additional cars both internally (13 cars per day) and cars from other yards (38 cars per day).
11	1	2	3	4	6			**85**	Run Daily
12	1	2	3	4	6			**86**	Run Daily
13	7							**140**	Run Daily
14	1							**88**	Run Daily
15	1							**94**	Run Daily
16	4	3	2					**141**	Run Daily
17	5							**55**	Run Daily
18	7							**121**	Run Daily
19	1	1	2	3	4	5		**102**	Run Daily
20	1	2	3	4				**131**	Run Daily
21	1	2	3	4				**64**	Run Daily
22	6	6	5					**74**	Run Daily
23	6	6	5					**93**	Run Daily
24	7							**22**	Run Train every 3~4 days to allow for internal accumulation of additional cars.

Table 5.16 Yard Statistics

Yard	1	2	3	4	5	6	7	8
Cars (Initial)	441	140	111	44	371	467	135	399
Cars (Additional)	715	74	25	141	157	536	69	369
Classification Time Used	07:58	01:07	00:39	02:06	02:38	05:43	01:32	03:50
# of Train Origins	6	1	2	1	3	5	1	5
# of Train Connections		1		1	1	1	3	
# of Train Destinations	8	2	1	1	2	5	2	3

Table 5.16 shows the yard statistics after the final iteration. From this data one can determine that the largest and most utilised yards are Yard 1, Yard 6, and Yard 8. This is important because the railroad analyst can determine how best to allocate yard resources such as crews or engines. Additionally, if this was a study where the user wanted to determine the relevancy of necessity of a given yard, they could look more closely at Yards 3, 4 and 5. The yards are spaced closely together and there is a very limited amount of traffic travelling to or through the yard. If one were to combine the traffic in those yards, consolidation of trains and resources could potentially be realized. The user would have to re-simulate a new network with adjusted O-D data and train schedules in order to see the differences between the existing condition and the proposed.

The model, as applied to the hypothetical data, has provided an operating plan which uses 24 trains to move the traffic in the network. 19 trains operate daily while the other 5 operate intermittently during the week. The blocking plan has been established and can be followed easily at each yard. The queuing data for each yard can also be retrieved from the software as well as other yard statistics which can then be downloaded to each yardmaster that can in turn use that information for staffing and coordinating yard operations.

5.3. Model Application: Multiple Plans

When assessing two or more plans in attempts to determine the best plan, it is important to note that the plans have to be for the same overall network and traffic data. The plan timeline should also be the same so as to compare each plan on the same level. This example of assessment is completed in accordance with the guidelines presented in Section 4.7.

This example is created using the network and traffic data from Section 5.2. Two plans are being taken into consideration. Though both plans are solved by the model, they have one key difference in that the route between Yard 1 and Yard 7 is not available to Plan 1 but is available in Plan 2. This means that the cars and trains may have to travel farther in order to reach their destinations for Plan 1. For the number of trains running per day, the calculation is made as follows. For each daily train, the number of trains per day is increased by one. For each train running in intervals of more than one day, the number is increased by the one day equivalent. For example, if a train was running every three days, the daily number of trains would increase by a factor of 1/3. The number of switches is calculated by the model and is the sum of additional cars at all of the yards. The results of the plans and the analysis can be found in Table 5.17.

Table 5.17 Multiple Plan Assessment

Criteria	**Weight**	**Plan 1**		**Plan 2**	
		Value	Rating	Value	Rating
Number of Trains	1	21.00	1	21.50	2
Number of Switches	1	2059	1	2086	2
Total Car-Distance	1	982, 400	2	920, 298	1
Totals		4		5	

As predicted, Plan 1 has a higher total Car-Distance, however it has fewer train starts and fewer number of switches. Therefore, overall, Plan 1 seems to be superior to Plan 2 in this assessment. If weighted factors were applied in accordance with section 4.7, Plan 1 would still be the optimal plan as per this analysis.

5.4. Summary

This chapter has provided verification for the model software as well as a detailed example of its use and analysis of an individual, hypothetical, operations plan. This hypothetical application showed one example of creating an operations plan, using the least amount of trains to move the majority of the traffic. Additional examples of creating operation plans considering minimum number of switches, reutilization of yards and other resources, adjustments of routes and blocking criteria can be created similarly and assessed using the criteria set forth in Section 4.7.

Hypothetical data were used in this thesis because real world data (as provided by CPR) was overly detailed. The data were not able to be separated into O-D pairs as per the simplistic characteristics used in this model. The O-D traffic data provided split its' pairs up using car type, car length, car consist, etc. The origins were always the original start point for each car, for example, a warehouse or a small distribution center and the destinations were similar. Though extracting a small subset of yards from a real network would have been possible, the traffic data would have been incomplete at best. The data were more complicated that the model in this thesis is able to handle, thus hypothetical data, similar to that in Troup et al (1977) was used.

6. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

6.1. Summary

Railroads move freight traffic on their network based on an overall operations plan that includes blocking, train formation, and train scheduling plans. The optimization of these operations over the entire network is integral to maximizing efficiency and minimizing costs. This thesis developed and illustrates a simulation model for analyzing various operation plans of a railroad network along with guidelines for establishing a comprehensive operations plan. This model focuses on traffic routing and simulation of traffic flows. The three main objectives for this thesis were:

1. To create a model to assist in testing and analyzing the operation plan(s);
2. To create guidelines for building integral portions of the operations plan; and
3. To create a user friendly software application to implement the model.

Details pertaining to the above objectives have been discussed and can be reviewed in Section 1.3. Through a broad literature review of various railroad and yard operations as well as general railroad information, a comprehensive and realistic model for use on a small to medium sized network has been created. A comprehensive example for the use of the model, the software application and guidelines was also presented. The model and guidelines developed enable a single analyst to create a fully integrated operations plan with only traffic, network, and yard data at their fingertips. Inefficient train schedules, blocking plans, and block to train assignments can be adjusted and revamped in a few simple steps. The model is intuitive and easy to use making this a user friendly

The software application was created using Microsoft Excel and VBA as a programming platform. The computer based simulation model is created in order to assist in designing a block routing plan which is effective and efficient in terms of overall railroad costs and delivery times. The simulations allow the user to determine results of various operating plans and various scenarios within a virtual railroad network. This allows a user to review and analyze data from various network elements and determine a globally acceptable "best case" solution taking many factors into account. With the software, a user can also model and analyze the flexibility and reliability of the system and determine the effects of various planned and unplanned changes in the network. The software is easy to use and examples are provided in Chapter 5.

6.2. Conclusions

From the research conducted, the following conclusions may be drawn:

1. Railroad operations are quite complicated and major operation plans each play an important role. The overall success of an operations plan on a railroad network can only be achieved when all operating plans within the network are integrated. This includes the integration of plans over the entire network by using excess yard and train capacity to augment areas of a network which are overflowing with traffic.
2. With an optimized operations plan in place, railroads can greatly benefit from the economies of scale which made them a force in the freight industry in the first place. The optimized operations plans will not only save the railroad companies money, but also increase the reliability and flexibility of their shipments. These are the most important factors shippers look for in railroads.

3. This model will allow users to review different elements of the railroad network and the impacts of simple or complex changes to each element of the network based on an overall system point of view. Using this model, a user can simulate various elements of a railroad network system as a whole and determine a global optimum solution to an operations plan. This is different from many mathematical optimization models which often focus on one objective and not a whole system wide analysis.
4. Railroads often have some form of priority traffic, whether it is for a high paying customer or a military effort, there are always trains which will need special handling and treatment in order for them to get to their destinations on time. This is addressed by this model which has the ability to give special handling to either O-D pairs or trains individually. This means that an analyst has flexibility in providing special handling to cars prior to train formation and classification of cars, without the cumbersome act of cherry picking at the end of a classification operation. This is one aspect of the model which can save both time and money for railroads in the future, as succeeding in providing special service can often give one railroad a competitive edge over another which does not offer priority shipping.

6.3. Recommendations

The recommendations and proposed areas of further study/additions to the model are as follows:

1. Incorporating additional rail car characteristics such as height, width, length, weight, and car type to the O-D traffic tables could help to really narrow down specific and realistic constraints when creating blocks and trains. The length and weight are of great

importance in building trains as they will determine a more accurate maximum train size and impact the number of locomotives required for each train.

2. Shortest route and least resistance algorithms can be added to the model, which do not require the user to enter the route data manually. This can potentially save a significant amount of time in data entry and network setup within the model itself. The use of resistance methods in the routing algorithm can allow the model to determine appropriate paths in a network, allowing the model to take into account the Braess paradox, thereby alleviating the work from the user.
3. This model can also be augmented by combining additional train movement models which take into account train specific information. This could include dynamic train scheduling algorithms which use optimization techniques to maximize average train size. This is based on time of departure and train movement models which show how trains will actually move along a corridor. The latter of the two is also important when two trains traveling either in the same or opposing directions meet on the same track and have to overtake or pass each other (as this will delay train running time).
4. Full development of the software for the model which incorporates all of the guidelines and assists in automating the analysis should be considered for the future.

REFERENCES

AASHTO. (2009). *Freight Rail - Bottom Line Report.* Washington: AASHTO.

Armstrong, J. (1990). *The Railroad, What it is, What it does: The Introduction to Railroading - 3rd Edition.* Omaha, NE: Simmons-Boardman Books Inc.

Assad, A. (1980). Modelling of Rail Networks: Toward a Routing/Makup Model. *Transportation Research Part B: Methodological , 14* (1-2), 101-114.

Ballis, A., Liberis, K., & Moschovou, T. (2004). Investigating the Capacity of a Metro Line by Means of a Simulation Model. *Journal of Rail and Rapid Transit. 218: Part F*, pp. 67-78. Institute of Mechanical Engineers.

Bodin, L. D., Golden, B., Schuster, A., & and Romig, W. (1980). A Model for Blocking Trains. *Transportation Research Part B: Methodological , 14* (1-2), 115-120.

Bradley, Hax, & Magnanti, a. (1977). Mathematical Programming: An Overview. In Bradley, Hax, & a. Magnanti, *Applied Mathematical Programming* (pp. 1-37). Reading, Massechusettes, United States of America: Addison-Wesley.

Canadian Pacific Railway. (2010, June 12). *Where We Ship*. Retrieved June 12, 2010, from Canadian Pacific Railway Website: http://www8.cpr.ca/cms/English/Customers/New+Customers/Where+We+Ship/default.htm

Cardeau, J., Toth, P., & and Vigo, D. (1998, November). A Survey of Optimization Models for Train Routin and Sorting. *Transportation Science , 32* (4), pp. 380-404.

Daganzo, C. (1987). Dynamic Blocking for Railyards: Part 1. Homogeneous Traffic. *Transportation Research: Part B , 21B* (1), 1-27.

Daganzo, C. (1987). Dynamic Blocking for Railyards: Part 2. Heterogeneous Traffic. *Transportation Research: Part B* (21B), 29-40.

Daganzo, C. (1986). Static Blocking at Railyards: Sorting Implications and Track Requirements. *Transportation Science , 20* (3), 189-199.

Dirnberger, J. a. (2007). Lean railroading: Improving railroad classification terminal performance through bottleneck management methods. *Transportation Research Record - Journal of the Transportation Research Board , 1995*, 52-61.

Dirnberger, J. (2006). *Development and Application of Lean Railroading to Improve Classification Terminal Performance.* Urbana, Illinois: University of Illinois at Urbana-Champaign.

Eiselt, H. a. (2007). *Linear Programming and its Applications.* New York: Springer.

Fernandez, A., Cucala, A., & Cuadra, F. d. (2006). Predictive Traffic Regulation for Metro Look Lines Based on Quadratic Programming. *Journal of Rail and Rapid Transit. 220: Fart F*, pp. 79-89. Institute of Mechanical Engineers.

Gormick, G. (2005, August). How Fluid is Your Yard? *Railway Age , 206* (8), pp. 18-28.

Higgins, A., Ferreira, L., & Kozon, E. (1995). Modeling Single-Line Train Operations. *Transportation Research Record - Journal of the Transportation Research Board , 1489*, 9-16.

Hillier, F. a. (2001). *Introduction to Operations Research* (Seventh Edition ed.). New York: Mcgraw-Hill Higher Education.

Innovative Scheduling. (n.d.). *IHYM: Innovative Hump Yard Manager - Presentation - Innovative Scheduling.* Retrieved December 06, 2009, from http://www.innovativescheduling.com/Files/Presentations/IS_IHYM_Product_Presentation.pdf

Innovative Scheduling. (2005). *IHYM: Innovative Hump Yard Manager - White Paper - Innovative Scheduling.* Retrieved December 06, 2009, from Innovative Scheduling Website: http://www.innovativescheduling.com/Files/WhitePapers/IS_IHYM_WhitePaper.pdf

Ireland, e. a. (2003). *Perfecting the Scheduled Railway: Model-Driven Operating Plan Development.* Princeton, NJ: MultiModal Applied Systems, Inc.

Ireland, P. e. (2004). The Canadian Pacific Railway Transforms Operations by Using Models to Develop its Operating Plans. *Interfaces , 34* (1), 5-15.

Jha, K., Ahuja, R., & and Sahin, G. (2007, November 28). New Approaches for Solving the Block-to-Train Assignment Problem. *Networks* , pp. 48-62.

Judge, T. (2002). Industry Experts Address Railroad Congestion and Capacity. *Railway Age , 203* (9), 20.

Kraft, E. (2000a). A Hump Sequencing Algorithm for Real Time Management of Train Connection Reliability. *Transportation Quarterly* , 95-115.

Kraft, E. (1998). *A Reservations-Based Railway Network Operations Management System - PHd Dissertation.* Philadelphia: University of Pennsylvania.

Kraft, E. (2002e, July). Adding Fexibility to the Scheduled Railraod. *Railway Age , 203* (7), pp. 19-21.

Kraft, E. (2000b). Implementation Strategies for Railroad Dynamic Freight Car Scheduling. *Transportation Quarterly* , 119-137.

Kraft, E. (2002a). Priority-Based Classification for Improving Connection Reliability in Railroad Yards- Part I: Integration with Car Scheduling. *Transportation Quarterly* , 93-105.

Kraft, E. (2002b). Priority-Based Classification for Improving Connection Reliability in Railroad Yards: Part II - Dynamic Block to Track Assignment. *Transportation Quarterly* , 107-119.

Kraft, E. (2002c). The Yard: Railroading's Hiddent Half (Part 1). *Trains , 62* (6), 46-67.

Kraft, E. (2002d). The Yard: Railroading's Hiddent Half (Part 2). *Trains , 62* (7), 36-47.

Kraft, E., Srikar, B., & Phillips, R. (2000c). Revenue Management in Railroad Applications. *Transportation Quarterly* , 157-175.

Kuehn, J. (1999). *Transportation By Design.* Princeton, NJ: MultiModal Applied Systems, Inc.

Kwon, O., Martland, C., Sussman, J., & Little, P. (1995). Origin-to-Destination Trip Times and Reliability of Rail Freight Serices in North American Railroads. *Transportation Research Record - Journal of the Transportation Research Board , 1489*, 1-8.

Liu, J., Ahuja, R., & Sahin, G. (2008). Optimal Network Configuration and Capacity Expansion of Railroads. *Journal of the Operational Research Society* , 911-920.

martinelli, D. a. (1996). Optimization of Railway Operations Using Neural networks. *Transportation Research: Part C , 4* (1), 33-49.

Martinelli, D. a. (1996). Optimization of Railway Operations using Neural Networks. *Transportation Research - Part C , 4* (1), 33-49.

Martland, C. (1982). PMAKE Analysis: Predicting Rail Yard Time Distributions Using Probabalistic Train Connection Standards. *Transportation Science , No. 16* (No. 4), 476-506.

Miller, J. (1985). *A Railroad Terminal Evaluation Methodology.* West Virginia University, College of Engineering. Morgantown, VA: West Virginia University.

Morlok, E., & and Chang, J. (2004). Measuring Capacity Flexibility of a Transportation System. *Transportation Research , 38A* (No. 4), 405-420.

Morlok, E., & Riddle, P. (1999). Estimating the Capacity of Freight Transportation Systems - A Model and Its Application in Transport Planning and Logistics. *Transportation Research Record: Journal of the Transportation Research Board , TRR No. 1653* (Paper No. 99-166), 1-8.

Mundrey, J. S. (2005). *Railway Track Engineering, 3rd Edition.* New Dehli: Tata Mcgraw-Hill.

Niu, H., & Wong, W. (2002). A Train Dispatching Optimization for Classification Yards by Genetic Algorithm. *3rd International Conference on Traffic and Transportation Studies*, (pp. 299-304). Guilin, China.

Oliver Wyman Group. (2010, March 16). *MultiRail : RailPlanning Blog - MultiRail Enterprise Edition.* Retrieved August 13, 2010, from Oliver Wyman Group : http://blog.railplanning.com/multi-rail/

Pachl, J. (2002). *Railway Operation and Control.* Rochester, WA: Gorham Printing.

Papacostas, C., & Prevedouros, P. (2001). *Transportation Engineering & Planning.* Upper Saddle River, New Jersey: Prentice Hall Inc.

Petersen, E. R. (1977a). Railyard Modeling: Part I. Prediction of Put-Through Time. *Transportation Science , 11* (1), 37-49.

Peterson, E. R. (1977b). Railyard Modeling: Part II. The effect of Yard Facilities on Congestion. *Transportation Science , 11* (1), 50-59.

Railway Association of Canada. (2004). *Meet Your Neighbor - The Railway in Your Community.* Retrieved February 1, 2010, from RAC - All About Rail: http://www.railcan.ca/sec_rac/en_rac_allAboutRail.asp

Ramsey, G., Hutchingson, B., & Rilett, L. (1986). Simplified Railroad Capacity Model. *Journal of Transportation Engineering , 112* (4), 358-368.

Randolph, J. (2009). A Guide to Writing the Dissertation Literature Review. *Practical Assessment, Reseach & Evaluation , 14* (13), 1-13.

Sun, Y., Turnquist, M., & and Nizick, L. (2006). Estimating Freight Transportation System Capacity, Flexibility, and Degraded-Condition Performance. *Transportation Research Record: The Journal of the Transportation Research Board , No. 1966*, 80-87.

The Railway Association of Canada. (2001). *Railway Trends.* Ottawa: Railway Association of Canada.

The Railway Association of Canada. (2009). *Railway Trends.* Ottawa: Railway Association of Canada.

Turnquist, M. a. (1982). Queueing Model of Classification and Connection Delays in Railyards. *Transportation Science , 16* (2), 207-230.

Troup *et* a*l (1977). Railroad classification yard technology: an introductory analysis of functions and operations.* United States, Federal Railroad Administration, Office of Research and Development.

Van Dyke, C. a. (1999). *Algorithm-Based Blocking Plan Analysis.* Princeton, JN: MultiModal Applied Systems, Inc.

Appendix 1 – List of Definitions and Abbreviations

Block – is a group of cars that are moved together by one or more trains from a common origin or assembly point to a common destination or disassembly point. For an individual car, the common origin and destination may be either the same as the ultimate origin or destination of the car, or may be intermediate points in the car's route where the car is to be marshalled. (Van Dyke, 1999)

Bowl – a configuration of tracks branching off from a common main track suitable for classifying cars for similar destinations or making blocks (also known as a set of classification tracks) (Miller, 1985)

"Class 1" Railroad – is a railroad which has over $250 million (CAD) in profits.

Classification – the process of grouping or classifying railroad cars for common handling or destination (Miller, 1985).

CPR – Canadian Pacific Railway

Cut – any set of cars that share a common destination track and, by chance or design, are sequenced together in an arriving train (Dirnberger J. , 2006).

Hump – a set of tracks situated on a raised mound located at the entrance to a bowl. By pushing railroad cars over the hump, the cars roll by force of gravity into the bowl for classification. Hence the terms hump yard and gravity classification yard are somewhat synonymous. The term humping therefore implies the processes described above (Miller, 1985).

Power – a term used to indicate one or more locomotives (Miller, 1985).

Receiving/ Departure Tracks – used to receive an inbound train into a yard and/or make up and depart outbound trains from a yard (Miller, 1985).

Rehump – process of humping cars more than once to achieve the desired classifications (Miller, 1985).

Road Train – train operated between major yards for the purpose of hauling cars between yards in different terminal areas (Miller, 1985).

Run-through – a train operated through a gateway or terminal area without being broken apart. Such trains may be inspected and have locomotives serviced and crews changed in a yard (Miller, 1985).

Sluff Tracks –overflow tracks designated to be used when classification tracks are not available or when the anticipated volume of traffic is too small to warrant a separate classification track (Miller, 1985).

Switch – the operation that separates two adjacent sets of cars, and sends the sets to their assigned classification tracks. Although every car must be sorted, not all require switches (Dirnberger J. , 2006).

Terminal – is an assemblage of facilities which are provided for the purposes of assembling, assorting, classifying and relaying trains (Pachl, 2002). A terminal can consist of the minimum of a receiving yard, a classification yard and a departure yard. Additionally terminals can have

additional yards such as transload facilities, mechanical yards, roundhouses, etc (Armstrong, 1990).

Through Blocking – process of grouping or classifying cars into blocks which can be moved through a second classification complex without being separated and reclassified (Miller, 1985).

Through Train – run-through train (Miller, 1985).

Trim Operation – the process of pulling blocks of cars from a bowl and setting the blocks onto a departure track in order to assemble an outbound train (Miller, 1985).

Unit Train – is a train which has only one O-D combination and thus travels directly from one yard to another bypassing (see run-through train) all yards on its way. This type of train is typically reserved for large shipments of coal or grain originating at one single distribution center.

Yard – is an arrangement of sidings (or set of tracks) which run parallel to each other used for making up trains, sorting cars and trains, and similar purposes. (Pachl, 2002)

Appendix 2 – Data Inputs for Example Model Application

List of Routes

Route #	Origin Yard	Destination Yard	Route Path (Yard to Yard)	Route Length (km)
1	1	2	12	300
2	1	3	123	400
3	1	4	1234	450
4	1	5	12345	500
5	1	6	123456	700
6	1	8	1234568	800
7	1	7	12347	600
8	1	8	123478	700
9	2	1	21	300
10	2	3	23	100
11	2	4	234	150
12	2	5	2345	200
13	2	6	23456	400
14	2	8	234568	500
15	2	7	2347	300
16	2	8	23478	400
17	3	1	321	400
18	3	2	32	100
19	3	4	34	50
20	3	5	345	100
21	3	6	3456	300
22	3	8	34568	400
23	3	7	347	200
24	3	8	3478	300
25	4	1	4321	450
26	4	2	432	150
27	4	3	43	50
28	4	5	45	50
29	4	6	456	250
30	4	8	4568	350
31	4	7	47	150
32	4	8	478	250
33	5	1	54321	500
34	5	2	5432	200
35	5	3	543	100

Route #	Origin Yard	Destination Yard	Route Path (Yard to Yard)	Route Length (km)
36	5	4	54	50
37	5	6	56	200
38	5	8	568	300
39	5	7	547	200
40	5	8	5478	300
41	6	1	654321	700
42	6	2	65432	400
43	6	3	6543	300
44	6	4	654	250
45	6	5	65	200
46	6	8	68	100
47	6	7	687	200
48	6	8	65478	500
49	7	1	74321	600
50	7	2	7432	300
51	7	3	743	200
52	7	4	74	150
53	7	5	745	200
54	7	6	7456	400
55	7	8	78	100
56	7	6	786	200
57	8	1	874321	700
58	8	2	87432	400
59	8	3	8743	300
60	8	4	874	250
61	8	5	8745	300
62	8	6	86	100
63	8	7	87	100
64	8	5	87456	500
65	1	7	17	475
66	1	8	178	575
67	7	8	78	100
68	8	7	87	100
69	7	1	71	475
70	8	1	871	575
71	1	6	1786	675
72	6	1	6871	675

Original Daily O-D Table (Broken Down into 4 Hour Section)

From	To	04:00	08:00	12:00	16:00	20:00	23:59	Next Day	Total
1	**1**	1	2	2	0	1	0	0	**6**
1	**2**	3	3	16	9	27	0	0	**58**
1	**3**	1	2	11	5	6	0	0	**25**
1	**4**	0	0	0	0	0	0	0	**0**
1	**5**	2	5	34	21	40	0	0	**102**
1	**6**	17	16	25	34	48	0	0	**140**
1	**7**	2	4	30	7	4	0	0	**47**
1	**8**	4	2	14	27	16	0	0	**63**
Total		**30**	**34**	**132**	**103**	**142**	**0**	**0**	**441**
2	**1**	0	5	10	5	10	0	0	**30**
2	**2**	0	0	0	0	0	0	0	**0**
2	**3**	0	0	0	0	0	0	0	**0**
2	**4**	0	0	0	0	0	0	0	**0**
2	**5**	27	0	0	0	1	0	0	**28**
2	**6**	10	22	14	22	14	0	0	**82**
2	**7**	0	0	0	0	0	0	0	**0**
2	**8**	0	0	0	0	0	0	0	**0**
Total		**37**	**27**	**24**	**27**	**25**	**0**	**0**	**140**
3	**1**	11	11	30	13	0	0	0	**65**
3	**2**	0	0	0	0	0	0	0	**0**
3	**3**	0	0	0	0	0	0	0	**0**
3	**4**	0	0	0	0	0	0	0	**0**
3	**5**	0	0	0	0	0	0	0	**0**
3	**6**	4	4	13	0	0	0	0	**21**
3	**7**	0	0	0	0	0	0	0	**0**
3	**8**	5	5	15	0	0	0	0	**25**
Total		**20**	**20**	**58**	**13**	**0**	**0**	**0**	**111**
4	**1**	0	0	0	0	0	0	0	**0**
4	**2**	7	0	0	6	0	0	0	**13**
4	**3**	0	0	0	0	0	0	0	**0**
4	**4**	11	0	0	7	0	0	0	**18**
4	**5**	0	0	0	0	0	0	0	**0**
4	**6**	8	0	0	5	0	0	0	**13**
4	**7**	0	0	0	0	0	0	0	**0**
4	**8**	0	0	0	0	0	0	0	**0**
Total		**26**	**0**	**0**	**18**	**0**	**0**	**0**	**44**

From	To	04:00	08:00	12:00	16:00	20:00	23:59	Next Day	Total
5	**1**	17	51	17	52	34	0	0	**171**
5	**2**	0	0	0	0	0	0	0	**0**
5	**3**	0	0	0	0	0	0	0	**0**
5	**4**	0	0	0	0	0	0	0	**0**
5	**5**	0	0	0	0	0	0	0	**0**
5	**6**	6	18	6	18	12	0	0	**60**
5	**7**	0	0	0	0	0	0	0	**0**
5	**8**	14	42	14	42	28	0	0	**140**
Total		**37**	**111**	**37**	**112**	**74**	**0**	**0**	**371**
6	**1**	70	18	0	56	38	0	0	**182**
6	**2**	7	15	0	13	3	0	0	**38**
6	**3**	0	0	0	0	0	0	0	**0**
6	**4**	8	88	0	6	1	0	0	**103**
6	**5**	2	13	0	2	6	0	0	**23**
6	**6**	0	0	0	0	0	0	0	**0**
6	**7**	0	0	0	0	0	0	0	**0**
6	**8**	56	9	0	35	21	0	0	**121**
Total		**143**	**143**	**0**	**112**	**69**	**0**	**0**	**467**
7	**1**	24	24	0	24	0	0	0	**72**
7	**2**	0	0	0	0	0	0	0	**0**
7	**3**	0	0	0	0	0	0	0	**0**
7	**4**	0	0	0	0	0	0	0	**0**
7	**5**	0	0	0	0	0	0	0	**0**
7	**6**	6	6	0	6	0	0	0	**18**
7	**7**	0	0	0	0	0	0	0	**0**
7	**8**	15	15	0	15	0	0	0	**45**
Total		**45**	**45**	**0**	**45**	**0**	**0**	**0**	**135**
8	**1**	41	46	41	48	0	0	0	**176**
8	**2**	0	3	0	16	0	0	0	**19**
8	**3**	0	0	0	0	0	0	0	**0**
8	**4**	0	0	0	0	0	0	0	**0**
8	**5**	0	8	0	24	0	0	0	**32**
8	**6**	35	19	14	49	0	0	0	**117**
8	**7**	0	9	0	13	0	0	0	**22**
8	**8**	8	9	8	8	0	0	0	**33**
Total		**84**	**94**	**63**	**158**	**0**	**0**	**0**	**399**

Appendix 3 – Model Application Results – First Iteration

After First Iteration Daily O-D Table (Broken Down into 4 Hour Section)

From	To	04:00	08:00	12:00	16:00	20:00	23:59	Next Day	Total
1	**1**	1	2	2	0	1	213	483	702
1	**2**	0	0	0	0	0	3	54	57
1	**3**	0	0	0	0	0	0	0	0
1	**4**	0	0	0	0	0	0	0	0
1	**5**	0	0	0	0	0	0	24	24
1	**6**	0	0	0	0	0	0	0	0
1	**7**	0	0	0	0	0	0	0	0
1	**8**	0	0	0	0	0	0	25	25
Total		**1**	**2**	**2**	**0**	**1**	**216**	**586**	**808**
2	**1**	0	0	0	0	0	0	0	0
2	**2**	0	0	0	0	0	0	71	71
2	**3**	0	0	0	0	0	0	0	0
2	**4**	0	0	0	0	0	0	0	0
2	**5**	0	0	0	0	0	0	0	0
2	**6**	0	0	0	0	0	0	0	0
2	**7**	0	0	0	0	0	0	0	0
2	**8**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**0**	**0**	**0**	**0**	**71**	**71**
3	**1**	0	0	0	0	0	0	0	0
3	**2**	0	0	0	0	0	0	0	0
3	**3**	0	0	0	0	0	0	25	25
3	**4**	0	0	0	0	0	0	0	0
3	**5**	0	0	0	0	0	0	0	0
3	**6**	0	0	0	0	0	0	0	0
3	**7**	0	0	0	0	0	0	0	0
3	**8**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**0**	**0**	**0**	**0**	**25**	**25**
4	**1**	0	0	0	0	0	0	0	0
4	**2**	0	0	0	0	0	0	0	0
4	**3**	0	0	0	0	0	0	0	0
4	**4**	11	0	0	7	0	0	103	121
4	**5**	0	0	0	0	0	0	0	0
4	**6**	0	0	0	0	0	0	0	0
4	**7**	0	0	0	0	0	0	0	0
4	**8**	0	0	0	0	0	0	0	0
Total		**11**	**0**	**0**	**7**	**0**	**0**	**103**	**121**

From	To	04:00	08:00	12:00	16:00	20:00	23:59	Next Day	Total
5	**1**	0	0	0	0	0	0	0	0
5	**2**	0	0	0	0	0	0	0	0
5	**3**	0	0	0	0	0	0	0	0
5	**4**	0	0	0	0	0	0	0	0
5	**5**	0	0	0	0	0	0	133	133
5	**6**	0	0	0	0	0	0	0	0
5	**7**	0	0	0	0	0	0	0	0
5	**8**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**0**	**0**	**0**	**0**	**133**	**133**
6	**1**	0	0	0	0	0	0	0	0
6	**2**	0	0	0	0	0	0	0	0
6	**3**	0	0	0	0	0	0	0	0
6	**4**	0	0	0	0	0	0	0	0
6	**5**	0	0	0	0	0	0	28	28
6	**6**	0	0	0	66	0	81	304	451
6	**7**	0	0	0	0	0	0	0	0
6	**8**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**0**	**66**	**0**	**81**	**332**	**479**
7	**1**	0	0	0	0	0	0	0	0
7	**2**	0	0	0	0	0	0	0	0
7	**3**	0	0	0	0	0	0	0	0
7	**4**	0	0	0	0	0	0	0	0
7	**5**	0	0	0	0	0	0	0	0
7	**6**	0	0	0	0	0	0	0	0
7	**7**	0	0	0	0	0	22	47	69
7	**8**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**0**	**0**	**0**	**22**	**47**	**69**
8	**1**	0	0	0	0	0	0	0	0
8	**2**	0	0	0	0	0	0	0	0
8	**3**	0	0	0	0	0	0	0	0
8	**4**	0	0	0	0	0	0	0	0
8	**5**	0	0	0	0	0	0	0	0
8	**6**	0	0	0	0	0	0	0	0
8	**7**	0	0	0	0	0	0	0	0
8	**8**	8	9	8	8	0	0	369	402
Total		**8**	**9**	**8**	**8**	**0**	**0**	**369**	**402**

First Iteration – Block to Train Assignments (1)

O Yard	D Yard	Block	Train Assignment
1	**1**	0	
1	**2**	1	1
1	**3**	2	6
1	**4**	3	2
1	**5**	4	2
1	**6**	5	3
1	**7**	6	4
1	**8**	7	5
2	**1**	1	19
2	**2**	0	
2	**3**	2	7
2	**4**	3	7
2	**5**	4	7
2	**6**	5	7
2	**7**	6	7
2	**8**	7	7
3	**1**	1	8
3	**2**	2	8
3	**3**	0	
3	**4**	3	8
3	**5**	4	8
3	**6**	5	9
3	**7**	6	8
3	**8**	7	8
4	**1**	1	10
4	**2**	2	10
4	**3**	3	10
4	**4**	0	
4	**5**	4	10
4	**6**	5	7
4	**7**	6	10
4	**8**	7	10

First Iteration – Block to Train Assignments (2)

O Yard	D Yard	Block	Train Assignment
5	**1**	1	11, 12
5	**2**	2	11, 12
5	**3**	3	11, 12
5	**4**	4	11, 12
5	**5**	0	
5	**6**	5	9
5	**7**	6	11, 12
5	**8**	7	13
6	**1**	1	14, 15
6	**2**	2	14, 15
6	**3**	3	16
6	**4**	4	16
6	**5**	5	17
6	**6**	0	
6	**7**	6	14, 15
6	**8**	7	18
7	**1**	1	19
7	**2**	2	19
7	**3**	3	19
7	**4**	4	19
7	**5**	5	19
7	**6**	6	22, 23
7	**7**	0	
7	**8**	7	5
8	**1**	1	20, 21
8	**2**	2	20, 21
8	**3**	3	20, 21
8	**4**	4	20, 21
8	**5**	5	20, 21
8	**6**	6	22, 23
8	**7**	7	24
8	**8**	0	

List of Model Assigned Block Assignments and Train Data (first iteration)

Train ID#	Proposed Build Time	OYard	CYard	DYard	Block Assignment							SP Hand	# of Cars
1	20:00	1		2	1							1	**58**
2	20:00	1		5	4	3							**110**
3	20:00	1		6	5								**140**
4	20:00	1		7	6								**47**
5	20:00	1	7	8	7	7							**108**
6	20:00	1		3	2								**25**
7	20:00	2	4	6	5	5	2	3	4	6	7		**123**
8	20:00	3		1	1	2	3	4	6	7			**90**
9	16:00	3	5	6	5	5							**81**
10	23:59	4		2	2	1	3	4	6	7			**13**
11	12:00	5		1	1	2	3	4	6				**85**
12	20:00	5		1	1	2	3	4	6			1	**86**
13	20:00	5		8	7								**140**
14	12:00	6		1	1	2	6						**110**
15	20:00	6		1	1	2	6						**110**
16	20:00	6		4	4	3							**103**
17	20:00	6		5	5								**23**
18	20:00	6		8	7								**121**
19	20:00	7	2	1	1	1	2	3	4	5			**102**
20	08:00	8		1	1	2	3	4	5				**139**
21	16:00	8		1	1	2	3	4	5				**88**
22	08:00	8	7	6	6	6						1	**66**
23	16:00	8	7	6	6	6							**69**
24	16:00	8		7	7								**22**

List of Calculated Train Routes and Timings (first iteration)

Train #	Proposed Build Time	OYard	CYard	DYard	Speed	Route 1	Time	Route 2	Time	Calculated Time (OYard)	Calculated Time (CYard)	Calculated Time (DYard)
1	20:00	1		2	50	12	06:00			20:00		26:30
2	20:00	1		5	50	12345	10:00			20:30		31:00
3	20:00	1		6	50	123456	14:00			21:01		35:31
4	20:00	1		7	50	12347	12:00			21:31		34:01
5	20:00	1	7	8	50	12347	12:00	78	02:00	22:02	34:32	37:02
6	20:00	1		3	50	123	08:00			22:32		31:02
7	20:00	2	4	6	50	234	03:00	456	05:00	20:00	23:30	29:00
8	20:00	3		1	50	321	08:00			20:00		28:30
9	16:00	3	5	6	50	345	02:00	56	04:00	16:00	18:30	23:00
10	23:59	4		2	50	432	03:00			23:59		27:29
11	12:00	5		1	50	54321	10:00			12:00		22:30
12	20:00	5		1	50	54321	10:00			20:00		30:30
13	20:00	5		8	50	568	06:00			20:30		27:00
14	12:00	6		1	50	654321	14:00			12:00		26:30
15	20:00	6		1	50	654321	14:00			20:00		34:30
16	20:00	6		4	50	654	05:00			20:30		26:00
17	20:00	6		5	50	65	04:00			21:01		25:31
18	20:00	6		8	50	68	02:00			21:31		24:01
19	20:00	7	2	1	50	7432	06:00	21	06:00	20:00	26:30	33:00
20	08:00	8		1	50	874321	14:00			08:30		23:00
21	16:00	8		1	50	874321	14:00			16:00		31:00
22	08:00	8	7	6	50	87	02:00	786	04:00	08:00	10:30	15:00
23	16:00	8	7	6	50	87	02:00	786	04:00	16:30	19:00	23:30
24	16:00	8		7	50	87	02:00			17:01		19:31

Appendix 4 – Model Application Results – Final Iteration

After Final Iteration Daily O-D Table (Broken Down into 4 Hour Section)

From	To	04:00	08:00	12:00	16:00	20:00	23:59	Next Day	Total
1	**1**	1	2	2	0	1	213	483	702
1	**2**	0	0	0	0	0	3	16	19
1	**3**	0	0	0	0	0	0	0	0
1	**4**	0	0	0	0	0	0	0	0
1	**5**	0	0	0	0	0	0	0	0
1	**6**	0	0	0	0	0	0	0	0
1	**7**	0	0	0	0	0	0	0	0
1	**8**	0	0	0	0	0	0	0	0
Total		**1**	**2**	**2**	**0**	**1**	**216**	**499**	**721**
2	**1**	0	0	0	0	0	0	0	0
2	**2**	0	0	0	0	0	0	71	71
2	**3**	0	0	0	0	0	0	0	0
2	**4**	0	0	0	0	0	0	0	0
2	**5**	0	0	0	0	0	0	0	0
2	**6**	0	0	0	0	0	0	0	0
2	**7**	0	0	0	0	0	0	0	0
2	**8**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**0**	**0**	**0**	**0**	**71**	**71**
3	**1**	0	0	0	0	0	0	0	0
3	**2**	0	0	0	0	0	0	0	0
3	**3**	0	0	0	0	0	0	25	25
3	**4**	0	0	0	0	0	0	0	0
3	**5**	0	0	0	0	0	0	0	0
3	**6**	0	0	0	0	0	0	0	0
3	**7**	0	0	0	0	0	0	0	0
3	**8**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**0**	**0**	**0**	**0**	**25**	**25**
4	**1**	0	0	0	0	0	0	0	0
4	**2**	0	0	0	0	0	0	38	38
4	**3**	0	0	0	0	0	0	0	0
4	**4**	11	0	0	7	0	0	103	121
4	**5**	0	0	0	0	0	0	0	0
4	**6**	0	0	0	0	0	0	0	0
4	**7**	0	0	0	0	0	0	0	0
4	**8**	0	0	0	0	0	0	0	0
Total		**11**	**0**	**0**	**7**	**0**	**0**	**141**	**159**

From	To	04:00	08:00	12:00	16:00	20:00	23:59	Next Day	Total
5	**1**	0	0	0	0	0	0	0	0
5	**2**	0	0	0	0	0	0	0	0
5	**3**	0	0	0	0	0	0	0	0
5	**4**	0	0	0	0	0	0	0	0
5	**5**	0	0	0	0	0	0	133	133
5	**6**	0	0	0	0	0	0	0	0
5	**7**	0	0	0	0	0	0	0	0
5	**8**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**0**	**0**	**0**	**0**	**133**	**133**
6	**1**	0	0	0	0	0	0	0	0
6	**2**	0	0	0	0	0	0	0	0
6	**3**	0	0	0	0	0	0	0	0
6	**4**	0	0	0	0	0	0	0	0
6	**5**	0	0	0	0	0	0	52	52
6	**6**	0	0	0	66	0	81	304	451
6	**7**	0	0	0	0	0	0	0	0
6	**8**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**0**	**66**	**0**	**81**	**356**	**503**
7	**1**	0	0	0	0	0	0	0	0
7	**2**	0	0	0	0	0	0	0	0
7	**3**	0	0	0	0	0	0	0	0
7	**4**	0	0	0	0	0	0	0	0
7	**5**	0	0	0	0	0	0	0	0
7	**6**	0	0	0	0	0	0	0	0
7	**7**	0	0	0	0	0	22	47	69
7	**8**	0	0	0	0	0	0	0	0
Total		**0**	**0**	**0**	**0**	**0**	**22**	**47**	**69**
8	**1**	0	0	0	0	0	0	0	0
8	**2**	0	0	0	0	0	0	0	0
8	**3**	0	0	0	0	0	0	0	0
8	**4**	0	0	0	0	0	0	0	0
8	**5**	0	0	0	0	0	0	0	0
8	**6**	0	0	0	0	0	0	0	0
8	**7**	0	0	0	0	0	0	0	0
8	**8**	8	9	8	8	0	0	394	427
Total		**8**	**9**	**8**	**8**	**0**	**0**	**394**	**427**

After Final Iteration – Block to Train Assignments (1)

O Yard	D Yard	Block	Train Assignment
1	**1**	0	
1	**2**	1	1
1	**3**	2	6
1	**4**	3	2
1	**5**	4	2
1	**6**	5	3
1	**7**	6	4
1	**8**	7	5
2	**1**	1	19
2	**2**	0	
2	**3**	2	7
2	**4**	3	7
2	**5**	4	7
2	**6**	5	7
2	**7**	6	7
2	**8**	7	7
3	**1**	1	8
3	**2**	2	8
3	**3**	0	
3	**4**	3	8
3	**5**	4	8
3	**6**	5	9
3	**7**	6	8
3	**8**	7	9
4	**1**	1	10
4	**2**	2	10
4	**3**	3	10
4	**4**	0	
4	**5**	4	10
4	**6**	5	7
4	**7**	6	10
4	**8**	7	10

After Final Iteration – Block to Train Assignments (2)

O Yard	D Yard	Block	Train Assignment
5	**1**	1	11, 12
5	**2**	2	11, 12
5	**3**	3	11, 12
5	**4**	4	11, 12
5	**5**	0	
5	**6**	5	9
5	**7**	6	11, 12
5	**8**	7	13
6	**1**	1	14, 15
6	**2**	2	16
6	**3**	3	16
6	**4**	4	16
6	**5**	5	17
6	**6**	0	
6	**7**	6	4
6	**8**	7	18
7	**1**	1	19
7	**2**	2	19
7	**3**	3	19
7	**4**	4	19
7	**5**	5	19
7	**6**	6	22, 23
7	**7**	0	
7	**8**	7	5
8	**1**	1	20, 21
8	**2**	2	20, 21
8	**3**	3	20, 21
8	**4**	4	20, 21
8	**5**	5	22, 23
8	**6**	6	22, 23
8	**7**	7	24
8	**8**	0	

Appendix 5 – Software Code

Module 1

```
Option Explicit

' Explicitly create all variables in program

    Public NumYards As Long ' number of yards in the network
    Public MaxSize As Long ' max train size (number of cars)
    Public MinSize As Long ' min train size (number of cars)
    Public NumTrains As Long ' the number of trains which can run in the network

    Public T1() As Date, TC() As Date, T2() As Date  ' collection start times for trains and arrival time for train to next yard
    Public TX1() As Integer, TXC1() As Integer, TX2() As Integer ' conversion of collection times to integer times (out of 1-7)

    Public OYard As Long, CYard As Long, DYard As Long ' Origin and Destination Yards for indivudual Trains

    Public i As Integer, j As Integer, k As Integer, l As Integer, m As Integer, n As Integer, x As Integer, y As Integer ' various counter variables for loops

    Public NumBlocks() As Long ' Maximum number of blocks which can be created at each yard
    Public MinBlockSize() As Long ' Minimum size of pure blocks at each yard (respectively)

    Public Blocks() As Long ' Blocks list
    Public Blocks_Pure_Mixed() As Long ' Blocks list

    Public ODArrayP1() As Long ' Daily OD data table for Priority 1 (in array format)
    Public ODArrayP2() As Long ' Daily OD data table for priority 2(in array format)

    Public SwitchArray() As Long ' Switchting data table (in arrray format)
    Public NumSwitches() As Long ' Number of cars switched per yard
    Public SwitchingPrepTime() As Date ' Switching prep time per train at a given yard
    Public SwitchingTime() As Date ' Switching time per yard
    Public ServiceTime() As Date ' Service time for each train at a given yard

    Public TrainNumber As Long ' the train which is being moved at the specific time
    Public Trains() As Long ' Storage value for number of cars on a train at a given time
    Public TrainSwitchTime() As Date ' Storage value for amount of time it takes to switch cars on train X

    Public TrainTime1() As Date ' the amount of time it takes the specific train to move along a  route
    Public TrainTime2() As Date ' the amount of time it takes the specific train to move along a  route
    Public TrainSpeed() As Long ' the speed associated to the specific train routes


    Public PossibleTrainRoutes1() As String ' the various train routes available for each specific train
    Public PossibleTrainRoutes2() As String ' the various train routes available for each specific train
    Public TrainRoute1() As String ' the various train route for each specific train (based on speed and distance) Leg 1
    Public TrainRoute2() As String ' the various train route for each specific train (based on speed and distance) Leg 2

    Public Fulltrain As Integer ' a 0/1 variable which determines whether the train is full or not
    Public Smalltrain As Integer ' a 0/1 variable which determines whether the train is large enough to run or not

    Public NumRoutes As Integer ' total number of routes in the network
    Public NetworkRoute() As String ' the various train routes available in the network
```

```
   Public RouteDist() As Long ' the various distances wrt the various train routes
'       Public RouteTime() As Date ' the various movement times wrt the various train routes

Sub ReDimAll()
Application.ScreenUpdating = False

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Get Original Value for n (# of yards)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Sheets("Initial Input").Select
   Range("Number_Yards").Select
   NumYards = ActiveCell.Value
   Range("Max_Train_Size").Select
   MaxSize = ActiveCell.Value
   Range("Min_Train_Size").Select
   MinSize = ActiveCell.Value
   Range("Number_Of_Trains").Select
   NumTrains = ActiveCell.Value
   Range("Number_Of_Routes").Select
   NumRoutes = ActiveCell.Value
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Hide Unused Cells
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Sheets("Yard Data").Select
   Rows("1:100").EntireRow.Hidden = False
   Rows(3 + NumYards & ":17").EntireRow.Hidden = True
   Range("B2").Select

      Sheets("Routes").Select
   Rows("1:100").EntireRow.Hidden = False
   Rows(3 + NumRoutes & ":75").EntireRow.Hidden = True
   Range("B2").Select
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Redim all Variable Arrays
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   ReDim ODArrayP1(1 To NumYards, 1 To NumYards, 1 To 7)
   ReDim ODArrayP2(1 To NumYards, 1 To NumYards, 1 To 7)

   ReDim SwitchArray(1 To NumYards, 1 To NumYards, 1 To 7)

   ReDim NumSwitches(1 To NumYards)

   ReDim NumBlocks(1 To NumYards)
   ReDim MinBlockSize(1 To NumYards)

   ReDim SwitchingPrepTime(1 To NumYards)
   ReDim SwitchingTime(1 To NumYards)
   ReDim ServiceTime(1 To NumYards)

   ReDim Trains(1 To NumTrains)
   ReDim TrainSpeed(1 To NumTrains)
   ReDim TrainSwitchTime(1 To NumTrains)

   ReDim T1(1 To NumTrains)
   ReDim TC(1 To NumTrains)
   ReDim T2(1 To NumTrains)
```

```
    ReDim TX1(1 To NumTrains)
    ReDim TXC1(1 To NumTrains)
    ReDim TX2(1 To NumTrains)

    ReDim TrainTime1(1 To NumTrains)
    ReDim TrainTime2(1 To NumTrains)

    ReDim PossibleTrainRoutes1(1 To NumTrains, 1 To NumRoutes)
    ReDim PossibleTrainRoutes2(1 To NumTrains, 1 To NumRoutes)
    ReDim TrainRoute1(1 To NumTrains)
    ReDim TrainRoute2(1 To NumTrains)

    ReDim NetworkRoute(1 To NumRoutes)
    ReDim RouteDist(1 To NumRoutes)

    ReDim Blocks(1 To NumYards, 1 To NumYards)
    ReDim Blocks_Pure_Mixed(1 To NumYards, 1 To NumYards)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Move user to yard Data Sheet
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        Sheets("Yard Data").Select

    Application.ScreenUpdating = True
End Sub

Sub CreateODArray()
Application.ScreenUpdating = False
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Get Values for number of blocks and block sizes for each corresponding yard
        Sheets("Yard Data").Select
        Range("D3").Select
    For i = 1 To NumYards
        NumBlocks(i) = ActiveCell.Value
        ActiveCell.Offset(1, 0).Select
    Next i

        Range("E3").Select
    For i = 1 To NumYards
        MinBlockSize(i) = ActiveCell.Value
        ActiveCell.Offset(1, 0).Select
    Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Get switching time values for each corresponding yard
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        Range("F3").Select
    For i = 1 To NumYards
        SwitchingPrepTime(i) = ActiveCell.Value
        ActiveCell.Offset(1, 0).Select
    Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Get switching time values for each corresponding yard
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        Range("G3").Select
    For i = 1 To NumYards
        SwitchingTime(i) = ActiveCell.Value
        ActiveCell.Offset(1, 0).Select
```

```
   Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Get Service time values for each corresponding yard
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Range("H3").Select
   For i = 1 To NumYards
      ServiceTime(i) = ActiveCell.Value
      ActiveCell.Offset(1, 0).Select
   Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Get Values for Train Route and Train Route Time information
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Sheets("Routes").Select
      Range("Route_Path1").Select
   For i = 1 To NumRoutes
      NetworkRoute(i) = ActiveCell.Value
      ActiveCell.Offset(0, 1).Select
      RouteDist(i) = ActiveCell.Value
      ActiveCell.Offset(1, -1).Select
   Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Get Values for OD Array
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Sheets("O-D Matrices").Select
      Range("J3").Select
   For i = 1 To NumYards
      For j = 1 To NumYards
         For k = 1 To 7
         If ActiveCell.Font.Bold = True Then
            ODArrayP1(i, j, k) = ActiveCell.Value
         Else
            ODArrayP1(i, j, k) = 0
         End If
            ActiveCell.Offset(0, 1).Select
         Next k
            ActiveCell.Offset(1, -7).Select
      Next j
            ActiveCell.Offset(3, 0).Select
   Next i

      Sheets("O-D Matrices").Select
      Range("J3").Select
   For i = 1 To NumYards
      For j = 1 To NumYards
         For k = 1 To 7
         If ActiveCell.Font.Bold = False Then
            ODArrayP2(i, j, k) = ActiveCell.Value
         Else
            ODArrayP2(i, j, k) = 0
         End If
            ActiveCell.Offset(0, 1).Select
         Next k
            ActiveCell.Offset(1, -7).Select
      Next j
            ActiveCell.Offset(3, 0).Select
   Next i
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Create initial values for number of cars in a yard / day and time to switch those cars / day
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        Sheets("O-D Matrices").Select
        Range("Cars_Initial").Select
        ActiveCell.Offset(0, 1).Select
    For i = 1 To NumYards
        For j = 1 To NumYards
            For k = 1 To 7
                ActiveCell = ActiveCell.Value + ODArrayP1(i, j, k) + ODArrayP2(i, j, k)
            Next k
        Next j
            ActiveCell.Offset(0, 1).Select
    Next i

        Sheets("O-D Matrices").Select
    Range("A1").Select

Application.ScreenUpdating = True
End Sub

Sub Blocks_Setup()
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Create setup of blocks
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Sheets("O-D Matrices").Select
        For j = 1 To NumYards
            For k = 1 To NumYards
                If Cells(k + 3, 3).Value <= NumBlocks(j) Then
                        Blocks(j, k) = Cells(3 + (3 + NumYards) * (j - 1) + k, 3).Value
                        Blocks_Pure_Mixed(j, k) = Cells(3 + (3 + NumYards) * (j - 1) + k, 4).Value
                End If
            Next k
        Next j
End Sub

Sub ReCreateODArray()
Application.ScreenUpdating = False
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Get Values for OD Array
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        Sheets("O-D Matrices").Select
        Range("J3").Select
    For i = 1 To NumYards
        For j = 1 To NumYards
            For k = 1 To 7
            If ActiveCell.Font.Bold = True Then
                ODArrayP1(i, j, k) = ActiveCell.Value
            Else
                ODArrayP1(i, j, k) = 0
            End If
                ActiveCell.Offset(0, 1).Select
            Next k
                ActiveCell.Offset(1, -7).Select
        Next j
                ActiveCell.Offset(3, 0).Select
```

```
    Next i

        Sheets("O-D Matrices").Select
        Range("J3").Select
    For i = 1 To NumYards
        For j = 1 To NumYards
            For k = 1 To 7
            If ActiveCell.Font.Bold = False Then
                ODArrayP2(i, j, k) = ActiveCell.Value
            Else
                ODArrayP2(i, j, k) = 0
            End If
                ActiveCell.Offset(0, 1).Select
                'Debug.Print ODArrayP2(i, j, k)
            Next k
                ActiveCell.Offset(1, -7).Select
        Next j
                ActiveCell.Offset(3, 0).Select
    Next i

Application.ScreenUpdating = True
End Sub

Sub Train_Call_Run(TrainNumber)
Application.ScreenUpdating = False
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get OYard
    If Cells(TrainNumber + 2, 26 + 5).Value = "" Then
        MsgBox "You have not chosen an origin yard"
        Exit Sub
    Else
        OYard = Cells(TrainNumber + 2, 26 + 5).Value ' yard number (Origin)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get CYard
    If Cells(TrainNumber + 2, 26 + 6).Value = "" Then
        CYard = 0
    Else
        CYard = Cells(TrainNumber + 2, 26 + 6).Value ' yard number (Connection)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get DYard
    If Cells(TrainNumber + 2, 26 + 7).Value = "" Then
        MsgBox "You have not chosen a destination yard"
        Exit Sub
    Else
        DYard = Cells(TrainNumber + 2, 26 + 7).Value ' yard number (Destination)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    T1(TrainNumber) = Cells(TrainNumber + 2, 26 + 25).Value
    TC(TrainNumber) = Cells(TrainNumber + 2, 26 + 26).Value
    T2(TrainNumber) = Cells(TrainNumber + 2, 26 + 27).Value

    Call TrainTimes(TrainNumber, OYard, CYard, DYard, T1(TrainNumber), TC(TrainNumber), T2(TrainNumber))
    Call TrainMovement(TrainNumber, OYard, CYard, DYard, T1(TrainNumber), TC(TrainNumber),
T2(TrainNumber))
```

```
Application.ScreenUpdating = True
End Sub

Sub TrainTimes(TrainNumber, OYard, CYard, DYard, TSub1, TSubC, TSub2)
Application.ScreenUpdating = False

    T1(TrainNumber) = TSub1
    TC(TrainNumber) = TSubC
    T2(TrainNumber) = TSub2

    Call ReCreateODArray
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Check Minimum Train Size
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Trains(TrainNumber) = 0 ' Reset Trains(TrainNumber)

    If T1(TrainNumber) > #11:59:00 PM# Then TX1(TrainNumber) = 6 ' assumed that only cars from today can be
carried on any given train
    If T1(TrainNumber) <= #11:59:00 PM# Then TX1(TrainNumber) = 6
    If T1(TrainNumber) <= #8:00:00 PM# Then TX1(TrainNumber) = 5
    If T1(TrainNumber) <= #4:00:00 PM# Then TX1(TrainNumber) = 4
    If T1(TrainNumber) <= #12:00:00 PM# Then TX1(TrainNumber) = 3
    If T1(TrainNumber) <= #8:00:00 AM# Then TX1(TrainNumber) = 2
    If T1(TrainNumber) <= #4:00:00 AM# Then TX1(TrainNumber) = 1

    Cells(TrainNumber + 2, 26 + 13).Select
        For i = 1 To NumYards
            If Selection.Font.ColorIndex = xlAutomatic Then
                x = ActiveCell.Value
            Else
                x = 0
            End If
            If x > 0 And x <= NumYards Then
                    For j = 1 To NumYards
                If x = Blocks(OYard, j) Then
                    For k = 1 To TX1(TrainNumber) ' the time period for the cars to be taken from
                        Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP1(OYard, j, k)
                    Next k
                End If
                    Next j
            End If
            ActiveCell.Offset(0, 1).Select
        Next i

    Cells(TrainNumber + 2, 26 + 13).Select
        For i = 1 To NumYards
            If Selection.Font.ColorIndex = xlAutomatic Then
                x = ActiveCell.Value
            Else
                x = 0
            End If
            If x > 0 And x <= NumYards Then
                    For j = 1 To NumYards
                If x = Blocks(OYard, j) Then
                    For k = 1 To TX1(TrainNumber) ' the time period for the cars to be taken from
```

```
                Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP2(OYard, j, k)
              Next k
            End If
              Next j
        End If
        ActiveCell.Offset(0, 1).Select
      Next i

  If CYard = 0 Then

    CYard = 0

  Else

' because the train gets to CYard after some time, therefore additional cars may be picked up

    If TC(TrainNumber) > #11:59:00 PM# Then TXC1(TrainNumber) = 6 ' because cars on any train can only be
picked up from the day of the OD Demand (Assumed)
    If TC(TrainNumber) <= #11:59:00 PM# Then TXC1(TrainNumber) = 6
    If TC(TrainNumber) <= #8:00:00 PM# Then TXC1(TrainNumber) = 5
    If TC(TrainNumber) <= #4:00:00 PM# Then TXC1(TrainNumber) = 4
    If TC(TrainNumber) <= #12:00:00 PM# Then TXC1(TrainNumber) = 3
    If TC(TrainNumber) <= #8:00:00 AM# Then TXC1(TrainNumber) = 2
    If TC(TrainNumber) <= #4:00:00 AM# Then TXC1(TrainNumber) = 1

    Cells(TrainNumber + 2, 26 + 13).Select
      For i = 1 To NumYards
      If Selection.Interior.Color = 65535 Then
        x = ActiveCell.Value
      Else
        x = 0
      End If
        If x > 0 And x <= NumYards Then
            For j = 1 To NumYards
          If x = Blocks(CYard, j) Then
            For k = 1 To TXC1(TrainNumber) ' the time period for the cars to be taken from
              Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP1(CYard, j, k)
            Next k
          End If
            Next j
        End If
        ActiveCell.Offset(0, 1).Select
      Next i

    Cells(TrainNumber + 2, 26 + 13).Select
      For i = 1 To NumYards
      If Selection.Interior.Color = 65535 Then
        x = ActiveCell.Value
      Else
        x = 0
      End If
        If x > 0 And x <= NumYards Then
            For j = 1 To NumYards
          If x = Blocks(CYard, j) Then
            For k = 1 To TXC1(TrainNumber) ' the time period for the cars to be taken from
              Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP2(CYard, j, k)
```

```
                Next k
              End If
                Next j
            End If
            ActiveCell.Offset(0, 1).Select
          Next i

  End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  If Trains(TrainNumber) < MinSize Then
    Smalltrain = 1
    GoTo TrainIsTooSmall
  End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  TrainSwitchTime(TrainNumber) = Trains(TrainNumber) * SwitchingTime(DYard) +
SwitchingPrepTime(DYard)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  Cells(TrainNumber + 2, 26 + 24) = TrainSwitchTime(TrainNumber)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  If T2(TrainNumber) + TrainSwitchTime(TrainNumber) > #11:59:00 PM# Then TX2(TrainNumber) = 7
  If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #11:59:00 PM# Then TX2(TrainNumber) = 6
  If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #8:00:00 PM# Then TX2(TrainNumber) = 5
  If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #4:00:00 PM# Then TX2(TrainNumber) = 4
  If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #12:00:00 PM# Then TX2(TrainNumber) = 3
  If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #8:00:00 AM# Then TX2(TrainNumber) = 2
  If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #4:00:00 AM# Then TX2(TrainNumber) = 1

    Trains(TrainNumber) = 0 ' Reset Trains(TrainNumber)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Error messages for trains which are too small or large
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
TrainIsTooSmall:
  If Smalltrain = 1 Then
    MsgBox ("Train(" & TrainNumber & ") does not have enough cars, and thus it will not" & _
      " run. This train only has " & Trains(TrainNumber) & " cars.")
    Smalltrain = 0
    Trains(TrainNumber) = 0 ' Reset Trains(TrainNumber)
    Exit Sub
  End If

Application.ScreenUpdating = True
End Sub

Sub TrainMovement(TrainNumber, OYard, CYard, DYard, TSub1, TSubC, TSub2)
Application.ScreenUpdating = False

  T1(TrainNumber) = TSub1
  TC(TrainNumber) = TSubC
  T2(TrainNumber) = TSub2
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Movement of cars on Train (i)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  If T1(TrainNumber) > #11:59:00 PM# Then TX1(TrainNumber) = 6 ' because cars on any train can only be
picked up from the day of the OD Demand (Assumed)
  If T1(TrainNumber) <= #11:59:00 PM# Then TX1(TrainNumber) = 6
  If T1(TrainNumber) <= #8:00:00 PM# Then TX1(TrainNumber) = 5
```

```
   If T1(TrainNumber) <= #4:00:00 PM# Then TX1(TrainNumber) = 4
   If T1(TrainNumber) <= #12:00:00 PM# Then TX1(TrainNumber) = 3
   If T1(TrainNumber) <= #8:00:00 AM# Then TX1(TrainNumber) = 2
   If T1(TrainNumber) <= #4:00:00 AM# Then TX1(TrainNumber) = 1

   If T2(TrainNumber) + TrainSwitchTime(TrainNumber) > #11:59:00 PM# Then TX2(TrainNumber) = 7
   If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #11:59:00 PM# Then TX2(TrainNumber) = 6
   If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #8:00:00 PM# Then TX2(TrainNumber) = 5
   If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #4:00:00 PM# Then TX2(TrainNumber) = 4
   If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #12:00:00 PM# Then TX2(TrainNumber) = 3
   If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #8:00:00 AM# Then TX2(TrainNumber) = 2
   If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #4:00:00 AM# Then TX2(TrainNumber) = 1
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Loading of Priority (1) Cars (OYard)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Dim Remainder As Long

      Cells(TrainNumber + 2, 26 + 13).Select
      For i = 1 To NumYards
         If Selection.Font.ColorIndex = xlAutomatic Then
            x = ActiveCell.Value
         Else
            x = 0
         End If
         If x > 0 And x <= NumYards Then
               For j = 1 To NumYards
            If x = Blocks(OYard, j) Then
                  For k = 1 To TX1(TrainNumber) ' the time period for the cars to be taken from
               If Trains(TrainNumber) + ODArrayP1(OYard, j, k) <= MaxSize Then
                  Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP1(OYard, j, k)
                  ODArrayP1(DYard, j, TX2(TrainNumber)) = ODArrayP1(DYard, j, TX2(TrainNumber)) + _
                     ODArrayP1(OYard, j, k) ' destination yard is being added all cars
                  ODArrayP1(OYard, j, k) = 0 ' origin yard is being negated all cars
               ElseIf Trains(TrainNumber) + ODArrayP1(OYard, j, k) > MaxSize Then
                  Remainder = MaxSize - Trains(TrainNumber)
                  Trains(TrainNumber) = Trains(TrainNumber) + Remainder
                  ODArrayP1(DYard, j, TX2(TrainNumber)) = ODArrayP1(DYard, j, TX2(TrainNumber)) + _
                     Remainder
                  ODArrayP1(OYard, j, k) = ODArrayP1(OYard, j, k) - Remainder
                  Remainder = 0

                  Fulltrain = 1
                  GoTo TrainIsFull

               End If
                  Next k
            End If
               Next j
         End If

      ActiveCell.Offset(0, 1).Select
      Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Loading of Priority (2) Cars (OYard)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   Cells(TrainNumber + 2, 26 + 13).Select
```

```
        For i = 1 To NumYards
           If Selection.Font.ColorIndex = xlAutomatic Then
              x = ActiveCell.Value
           Else
              x = 0
           End If
           If x > 0 And x <= NumYards Then
                 For j = 1 To NumYards
              If x = Blocks(OYard, j) Then
                    For k = 1 To TX1(TrainNumber) ' the time period for the cars to be taken from
                 If Trains(TrainNumber) + ODArrayP2(OYard, j, k) <= MaxSize Then
                    Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP2(OYard, j, k)
                    ODArrayP2(DYard, j, TX2(TrainNumber)) = ODArrayP2(DYard, j, TX2(TrainNumber)) + _
                       ODArrayP2(OYard, j, k) ' destination yard is being added all cars
                    ODArrayP2(OYard, j, k) = 0 ' origin yard is being negated all cars
                 ElseIf Trains(TrainNumber) + ODArrayP2(OYard, j, k) > MaxSize Then
                    Remainder = MaxSize - Trains(TrainNumber)
                    Trains(TrainNumber) = Trains(TrainNumber) + Remainder
                    ODArrayP2(DYard, j, TX2(TrainNumber)) = ODArrayP2(DYard, j, TX2(TrainNumber)) + _
                       Remainder
                    ODArrayP2(OYard, j, k) = ODArrayP2(OYard, j, k) - Remainder
                    Remainder = 0

                    Fulltrain = 1
                    GoTo TrainIsFull

                 End If
                    Next k
              End If
                 Next j
           End If

        ActiveCell.Offset(0, 1).Select
        Next i
 '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    If CYard > 0 Then

' because the train gets to CYard after some time, therefore additional cars may be picked up
        If TC(TrainNumber) > #11:59:00 PM# Then TXC1(TrainNumber) = 6 ' because cars on any train can only be
picked up from the day of the OD Demand (Assumed)
        If TC(TrainNumber) <= #11:59:00 PM# Then TXC1(TrainNumber) = 6
        If TC(TrainNumber) <= #8:00:00 PM# Then TXC1(TrainNumber) = 5
        If TC(TrainNumber) <= #4:00:00 PM# Then TXC1(TrainNumber) = 4
        If TC(TrainNumber) <= #12:00:00 PM# Then TXC1(TrainNumber) = 3
        If TC(TrainNumber) <= #8:00:00 AM# Then TXC1(TrainNumber) = 2
        If TC(TrainNumber) <= #4:00:00 AM# Then TXC1(TrainNumber) = 1
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Loading of Priority (1) Cars (CYard)
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Cells(TrainNumber + 2, 26 + 13).Select
        For i = 1 To NumYards
           If Selection.Interior.Color = 65535 Then
              x = ActiveCell.Value
           Else
              x = 0
           End If
```

```
        If x > 0 And x <= NumYards Then
              For j = 1 To NumYards
           If x = Blocks(CYard, j) Then
                 For k = 1 To TXC1(TrainNumber) ' the time period for the cars to be taken from
              If Trains(TrainNumber) + ODArrayP1(CYard, j, k) <= MaxSize Then
                 Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP1(CYard, j, k)
                 ODArrayP1(DYard, j, TX2(TrainNumber)) = ODArrayP1(DYard, j, TX2(TrainNumber)) + _
                    ODArrayP1(CYard, j, k) ' destination yard is being added all cars
                 ODArrayP1(CYard, j, k) = 0 ' origin yard is being negated all cars
              ElseIf Trains(TrainNumber) + ODArrayP1(CYard, j, k) > MaxSize Then
                 Remainder = MaxSize - Trains(TrainNumber)
                 Trains(TrainNumber) = Trains(TrainNumber) + Remainder
                 ODArrayP1(DYard, j, TX2(TrainNumber)) = ODArrayP1(DYard, j, TX2(TrainNumber)) + _
                    Remainder
                 ODArrayP1(CYard, j, k) = ODArrayP1(CYard, j, k) - Remainder
                 Remainder = 0

                 Fulltrain = 1
                 GoTo TrainIsFull
              End If
                 Next k
           End If
              Next j
        End If

     ActiveCell.Offset(0, 1).Select
     Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Loading of Priority (2) Cars (CYard)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  Cells(TrainNumber + 2, 26 + 13).Select
     For i = 1 To NumYards
        If Selection.Interior.Color = 65535 Then
           x = ActiveCell.Value
        Else
           x = 0
        End If
        If x > 0 And x <= NumYards Then
              For j = 1 To NumYards
           If x = Blocks(CYard, j) Then
                 For k = 1 To TXC1(TrainNumber) ' the time period for the cars to be taken from
              If Trains(TrainNumber) + ODArrayP2(CYard, j, k) <= MaxSize Then
                 Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP2(CYard, j, k)
                 ODArrayP2(DYard, j, TX2(TrainNumber)) = ODArrayP2(DYard, j, TX2(TrainNumber)) + _
                    ODArrayP2(CYard, j, k) ' destination yard is being added all cars
                 ODArrayP2(CYard, j, k) = 0 ' origin yard is being negated all cars
              ElseIf Trains(TrainNumber) + ODArrayP2(CYard, j, k) > MaxSize Then
                 Remainder = MaxSize - Trains(TrainNumber)
                 Trains(TrainNumber) = Trains(TrainNumber) + Remainder
                 ODArrayP2(DYard, j, TX2(TrainNumber)) = ODArrayP2(DYard, j, TX2(TrainNumber)) + _
                    Remainder
                 ODArrayP2(CYard, j, k) = ODArrayP2(CYard, j, k) - Remainder
                 Remainder = 0

                 Fulltrain = 1
                 GoTo TrainIsFull
```

```
                    End If
                        Next k
                End If
                    Next j
            End If

        ActiveCell.Offset(0, 1).Select
        Next i

    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Error messages for trains which are too small or large
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
TrainIsTooSmall:
    If Smalltrain = 1 Then
        MsgBox ("Train(" & TrainNumber & ") does not have enough cars, and thus it will not" & _
            " run. This train only has " & Trains(TrainNumber) & " cars.")
        Smalltrain = 0
        Trains(TrainNumber) = 0 ' Reset Trains(TrainNumber)
        Exit Sub
    End If

TrainIsFull:
    If Fulltrain = 1 Then
        MsgBox ("Train(" & TrainNumber & ") is Full, it will only take up to a maximum of " & _
            MaxSize & " cars. The train may only take a full block, it cannot take one in part." & _
                "This train took " & Trains(TrainNumber) & " cars.")
        Fulltrain = 0
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Replace OD Array
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        Sheets("O-D Matrices").Select
        Range("J3").Select
    For i = 1 To NumYards
        For j = 1 To NumYards
            For k = 1 To 7
                ActiveCell.Value = ODArrayP1(i, j, k)
                ActiveCell.Offset(0, 1).Select
            Next k
                ActiveCell.Offset(1, -7).Select
        Next j
                ActiveCell.Offset(3, 0).Select
    Next i

        Range("J3").Select
    For i = 1 To NumYards
        For j = 1 To NumYards
            For k = 1 To 7
                ActiveCell.Offset(0, 1).Select
                If ActiveCell.Value > 0 Then
                    ActiveCell.Font.Bold = True
                Else
                    ActiveCell.Font.Bold = False
                End If
```

```
         Next k
            ActiveCell.Offset(1, -7).Select
      Next j
            ActiveCell.Offset(3, 0).Select
   Next i

      Sheets("O-D Matrices").Select
      Range("J3").Select
   For i = 1 To NumYards
      For j = 1 To NumYards
         For k = 1 To 7
            If ActiveCell.Font.Bold = False Then
               ActiveCell.Value = ODArrayP2(i, j, k)
            End If
            ActiveCell.Offset(0, 1).Select
         Next k
            ActiveCell.Offset(1, -7).Select
      Next j
            ActiveCell.Offset(3, 0).Select
   Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Count Number of Trains Originating at Yard "O" and received at yard "D"
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   Range("OYard_Number_Origin_Trains").Offset(0, OYard) = Range("OYard_Number_Origin_Trains").Offset(0,
OYard).Value + 1
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   If CYard > 0 Then
      Range("CYard_Number_Trains").Offset(0, CYard) = Range("CYard_Number_Trains").Offset(0,
CYard).Value + 1
   End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   Range("DYard_Number_Trains").Offset(0, DYard) = Range("DYard_Number_Trains").Offset(0, DYard).Value
+ 1
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Add to the values for number of cars in a yard / day and time to switch those cars / day
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   Range("Cars_Additional").Offset(0, DYard) = Range("Cars_Additional").Offset(0, DYard).Value +
Trains(TrainNumber)

   Range("Class_Time_Additional").Offset(0, DYard) = Range("Class_Time_Additional").Offset(0, DYard).Value
+ TrainSwitchTime(TrainNumber)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Application.ScreenUpdating = True
End Sub
```

Module 2

```
Sub ButtonAll_Click()
Application.ScreenUpdating = False

' call each train to be run individually, in order of earliest to latest

   Dim Train_Run_List() As Long
   Dim Train_Run_Time() As Date

   ReDim Train_Run_List(1 To NumTrains)
```

```
    ReDim Train_Run_Time(1 To NumTrains)

    For i = 1 To NumTrains
       Train_Run_List(i) = Cells(i + 2, 26 + 3).Value
       Train_Run_Time(i) = Cells(i + 2, 26 + 25).Value
    Next i

    Call Time_Sort_Run(Train_Run_Time, Train_Run_List)

    For m = 1 To NumTrains
       Call Run_Train(Train_Run_List(m))
    Next m

    Call Train_Sort_Run(Train_Run_List, Train_Run_Time)

Application.ScreenUpdating = True
End Sub

Sub Run_Train(S_TrainNumber)

       If S_TrainNumber > NumTrains Then Exit Sub
       Call ReCreateODArray
       Call Blocks_Setup
       Call Train_Call_Run(S_TrainNumber)
       If Trains(S_TrainNumber) < MinSize Then Exit Sub

    Cells(S_TrainNumber + 2, 26 + 21).Select
    ActiveCell = ActiveCell.Value + Trains(S_TrainNumber)

End Sub

Sub Time_Sort_Run(list1() As Date, list2() As Long)
Application.ScreenUpdating = False

' Sorts array

    Dim First As Integer, Last As Long
    Dim i As Long, j As Long
    Dim Temp1
    Dim Temp2

    First = LBound(list1)
    Last = UBound(list1)
    For i = First To Last - 1
       For j = i + 1 To Last
          If list1(i) > list1(j) Then
             Temp1 = list1(j)
             list1(j) = list1(i)
             list1(i) = Temp1

             Temp2 = list2(j)
             list2(j) = list2(i)
             list2(i) = Temp2
          End If
       Next j
    Next i
```

```
Application.ScreenUpdating = True
End Sub

Sub Train_Sort_Run(list1() As Long, list2() As Date)
Application.ScreenUpdating = False

' Sorts array

   Dim First As Integer, Last As Long
   Dim i As Long, j As Long
   Dim Temp1
   Dim Temp2

   First = LBound(list1)
   Last = UBound(list1)
   For i = First To Last - 1
      For j = i + 1 To Last
         If list1(i) > list1(j) Then
            Temp1 = list1(j)
            list1(j) = list1(i)
            list1(i) = Temp1

            Temp2 = list2(j)
            list2(j) = list2(i)
            list2(i) = Temp2
         End If
      Next j
   Next i

Application.ScreenUpdating = True
End Sub
```

Module 3

```
   Public Queue_List_OBuild() As Long
   Public Queue_Time_OBuild() As Date

   Public Queue_List_CBuild() As Long
   Public Queue_Time_CBuild() As Date

   Public Queue_List_DSwitch() As Long
   Public Queue_Time_DSwitch() As Date

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   Public T1Start() As Long
   Public T1Stop() As Long
   Public T1New() As Date
   '''''''''''''''''''''
   Public T2Start() As Long
   Public T2Stop() As Long
   Public T2New() As Date
   '''''''''''''''''''''
   Public T3Start() As Long
   Public T3Stop() As Long
   Public T3New() As Date
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Public OYard_List() As Long
    Public OYard_Train_List() As Long
    Public OYard_Build_Time_List() As Date
    Public OYard_Priority_List() As Long

    Public CYard_List() As Long
    Public CYard_Train_List() As Long
    Public CYard_Build_Time_List() As Date
    Public CYard_Priority_List() As Long

    Public DYard_List() As Long
    Public DYard_Train_List() As Long
    Public DYard_Build_Time_List() As Date
    Public DYard_Priority_List() As Long

Sub Actual_Time_Calculations()
Application.ScreenUpdating = False

    Sheets("Queue Times").Select
        Columns("D:F").Select
        Selection.ClearContents
        Range("D1").Select
        ActiveCell = "OYard Build List"
        Range("E1").Select
        ActiveCell = "CYard Connection Only  List"
        Range("F1").Select
        ActiveCell = "DYard Switching/Classification List"
    Sheets("O-D Matrices").Select

    For i = 1 To NumTrains
        Trains(i) = 75
    Next i

    Call Get_Shortest_Routes
    Call ArrayMovement
    Call Test_Queue
    Call New_OYard_Times
    Call Recalulate_CYard_and_DYard_Arrivals
    Call New_CYard_Times
    Call Recalulate_DYard_Arrivals
    Call New_DYard_Times
    Call Adjust_Times_24

    Sheets("Queue Times").Select
        Columns("D:F").Select
        Selection.ClearContents
        Range("D1").Select
        ActiveCell = "OYard Build List"
        Range("E1").Select
        ActiveCell = "CYard Connection Only  List"
        Range("F1").Select
        ActiveCell = "DYard Switching/Classification List"
    Sheets("O-D Matrices").Select

    Call Test_Queue
```

```
    Call Train_Call_Run_XX
    Call New_OYard_Times
    Call Recalulate_CYard_and_DYard_Arrivals
    Call New_CYard_Times
    Call Recalulate_DYard_Arrivals
    Call New_DYard_Times
    Call Adjust_Times_24

Application.ScreenUpdating = True
End Sub

Sub ArrayMovement()
Application.ScreenUpdating = False

    For TrainNumber = 1 To NumTrains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get build time
    If Cells(TrainNumber + 2, 26 + 4).Value = "" Then
        MsgBox "You have not chosen a build time"
        Exit Sub
    Else
        T1(TrainNumber) = Cells(TrainNumber + 2, 26 + 4).Value ' start time period for collection of cars
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get OYard
    If Cells(TrainNumber + 2, 26 + 5).Value = "" Then
        MsgBox "You have not chosen an origin yard for train number" & TrainNumber & "."
        Exit Sub
    Else
        OYard = Cells(TrainNumber + 2, 26 + 5).Value ' yard number (Origin)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get CYard
    If Cells(TrainNumber + 2, 26 + 6).Value = "" Then
        CYard = 0
    Else
        CYard = Cells(TrainNumber + 2, 26 + 6).Value ' yard number (Connection)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get DYard
    If Cells(TrainNumber + 2, 26 + 7).Value = "" Then
        MsgBox "You have not chosen a destination yard for train number" & TrainNumber & "."
        Exit Sub
    Else
        DYard = Cells(TrainNumber + 2, 26 + 7).Value ' yard number (Destination)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get train speed and determine route(s)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    If Cells(TrainNumber + 2, 26 + 8).Value = "" Then
        MsgBox "You have not provided a train speed"
        Exit Sub
    ElseIf CYard > 0 Then
        TrainSpeed(TrainNumber) = Cells(TrainNumber + 2, 26 + 8).Value

        For i = 1 To NumRoutes
```

```
      If InStr(NetworkRoute(i), OYard) = 1 And InStr(NetworkRoute(i), CYard) = Len(NetworkRoute(i)) Then
        PossibleTrainRoutes1(TrainNumber, i) = 1
      Else
        PossibleTrainRoutes1(TrainNumber, i) = 0
      End If
    Next i

    For i = 1 To NumRoutes
      If InStr(NetworkRoute(i), CYard) = 1 And InStr(NetworkRoute(i), DYard) = Len(NetworkRoute(i)) Then
        PossibleTrainRoutes2(TrainNumber, i) = 1
      Else
        PossibleTrainRoutes2(TrainNumber, i) = 0
      End If
    Next i

  Else

    TrainSpeed(TrainNumber) = Cells(TrainNumber + 2, 26 + 8).Value

    For i = 1 To NumRoutes

      If InStr(NetworkRoute(i), OYard) = 1 And InStr(NetworkRoute(i), DYard) = Len(NetworkRoute(i)) Then
          PossibleTrainRoutes1(TrainNumber, i) = 1
        Else
          PossibleTrainRoutes1(TrainNumber, i) = 0
      End If

    Next i

  End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' determine total train time(s) for each route(s)

  If CYard > 0 Then
    x = 0
  TrainTime1(TrainNumber) = 25 ' intitial value of 23:59:59 hours for train time - set high so that it will be
replaced with a new, smaller time

  For i = 1 To NumRoutes
    x = x + PossibleTrainRoutes1(TrainNumber, i)
  Next i
    If x > 0 Then
      For i = 1 To NumRoutes
        If PossibleTrainRoutes1(TrainNumber, i) = 1 And ((RouteDist(i) / TrainSpeed(TrainNumber)) / 24) <
TrainTime1(TrainNumber) Then
          TrainRoute1(TrainNumber) = NetworkRoute(i)
          TrainTime1(TrainNumber) = ((RouteDist(i) / TrainSpeed(TrainNumber)) / 24)
          Cells(TrainNumber + 2, 26 + 9) = TrainRoute1(TrainNumber)
          Cells(TrainNumber + 2, 26 + 10) = TrainTime1(TrainNumber)
        End If
      Next i
    Else
      MsgBox "There are no direct routes which take this train from Yard " & OYard & " to Yard " & CYard & "."
      Exit Sub
    End If
```

```
      x = 0
   TrainTime2(TrainNumber) = 25 ' intitial value of 23:59:59 hours for train time - set high so that it will be
replaced with a new, smaller time

   For i = 1 To NumRoutes
      x = x + PossibleTrainRoutes2(TrainNumber, i)
   Next i
      If x > 0 Then
         For i = 1 To NumRoutes
            If PossibleTrainRoutes2(TrainNumber, i) = 1 And ((RouteDist(i) / TrainSpeed(TrainNumber)) / 24) <
TrainTime2(TrainNumber) Then
               TrainRoute2(TrainNumber) = NetworkRoute(i)
               TrainTime2(TrainNumber) = ((RouteDist(i) / TrainSpeed(TrainNumber)) / 24)
               Cells(TrainNumber + 2, 26 + 11) = TrainRoute2(TrainNumber)
               Cells(TrainNumber + 2, 26 + 12) = TrainTime2(TrainNumber)
            End If
         Next i
      Else
         MsgBox "There are no direct routes which take this train from Yard " & CYard & " to Yard " & DYard & "."
         Exit Sub
      End If

   Else

   x = 0

   TrainTime1(TrainNumber) = 25 ' intitial value of 23:59:59 hours for train time - set high so that it will be
replaced with a new, smaller time

   For i = 1 To NumRoutes
      x = x + PossibleTrainRoutes1(TrainNumber, i)
   Next i
      If x > 0 Then
         For i = 1 To NumRoutes
            If PossibleTrainRoutes1(TrainNumber, i) = 1 And ((RouteDist(i) / TrainSpeed(TrainNumber)) / 24) <
TrainTime1(TrainNumber) Then
               TrainRoute1(TrainNumber) = NetworkRoute(i)
               TrainTime1(TrainNumber) = ((RouteDist(i) / TrainSpeed(TrainNumber)) / 24)
               Cells(TrainNumber + 2, 26 + 9) = TrainRoute1(TrainNumber)
               Cells(TrainNumber + 2, 26 + 10) = TrainTime1(TrainNumber)
            End If
         Next i
      Else
         MsgBox "There are no direct routes which take this train from Yard " & OYard & " to Yard " & CYard & "."
         Exit Sub
      End If

   End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' determine arrival time of train into CYard and DYard

   If CYard > 0 Then
      T2(TrainNumber) = T1(TrainNumber) + ServiceTime(OYard) + ServiceTime(CYard) / 2 +
TrainTime1(TrainNumber) + TrainTime2(TrainNumber) ' overall travel and service time for each train
   Else
```

```
        T2(TrainNumber) = T1(TrainNumber) + ServiceTime(OYard) + TrainTime1(TrainNumber) ' overall travel and
service time for each train
    End If

    If CYard > 0 Then
        If T1(TrainNumber) + TrainTime1(TrainNumber) + ServiceTime(OYard) > 1 Then
            Cells(TrainNumber + 2, 26 + 22) = T1(TrainNumber) + TrainTime1(TrainNumber) + ServiceTime(OYard) -
1
            Cells(TrainNumber + 2, 26 + 22).Interior.Color = 65535
            Cells(TrainNumber + 2, 26 + 22).Font.Color = -16776961
        Else
            Cells(TrainNumber + 2, 26 + 22) = T1(TrainNumber) + TrainTime1(TrainNumber) + ServiceTime(OYard)
        End If
    End If

    If T2(TrainNumber) > 1 Then
        Cells(TrainNumber + 2, 26 + 23) = T2(TrainNumber) - 1
        Cells(TrainNumber + 2, 26 + 23).Interior.Color = 65535
        Cells(TrainNumber + 2, 26 + 23).Font.Color = -16776961
    Else
        Cells(TrainNumber + 2, 26 + 23) = T2(TrainNumber)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Next TrainNumber

Application.ScreenUpdating = True
End Sub

Sub Test_Queue()
Application.ScreenUpdating = False

    ReDim Queue_List_OBuild(1 To NumYards, 1 To 2880)
    ReDim Queue_Time_OBuild(1 To 2880)
    ReDim Queue_List_CBuild(1 To NumYards, 1 To 2880)
    ReDim Queue_Time_CBuild(1 To 2880)
    ReDim Queue_List_DSwitch(1 To NumYards, 1 To 2880)
    ReDim Queue_Time_DSwitch(1 To 2880)

    Sheets("Queue Times").Select

    For i = 1 To NumYards
        x = (i - 1) * 2880
        For j = 1 To 2880

            Queue_List_OBuild(i, j) = Cells(1 + j + x, 4).Value
            Queue_Time_OBuild(j) = Cells(1 + j, 3).Value

            Queue_List_CBuild(i, j) = Cells(1 + j + x, 5).Value
            Queue_Time_CBuild(j) = Cells(1 + j, 3).Value

            Queue_List_DSwitch(i, j) = Cells(1 + j + x, 6).Value
            Queue_Time_DSwitch(j) = Cells(1 + j, 3).Value

        Next j
    Next i
```

```
        Sheets("O-D Matrices").Select

Application.ScreenUpdating = True
End Sub

Sub New_OYard_Times()
Application.ScreenUpdating = False

    ReDim OYard_List(1 To NumTrains)
    ReDim OYard_Train_List(1 To NumTrains)
    ReDim OYard_Build_Time_List(1 To NumTrains)
    ReDim OYard_Priority_List(1 To NumTrains)

    ReDim CYard_List(1 To NumTrains)
    ReDim CYard_Train_List(1 To NumTrains)
    ReDim CYard_Build_Time_List(1 To NumTrains)
    ReDim CYard_Priority_List(1 To NumTrains)

    ReDim DYard_List(1 To NumTrains)
    ReDim DYard_Train_List(1 To NumTrains)
    ReDim DYard_Build_Time_List(1 To NumTrains)
    ReDim DYard_Priority_List(1 To NumTrains)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
ReDim T1Start(1 To NumTrains)
ReDim T1Stop(1 To NumTrains)

ReDim T1New(1 To NumTrains)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Sheets("O-D Matrices").Select

    For i = 1 To NumTrains
        OYard_Train_List(i) = Cells(2 + i, 26 + 3).Value
        OYard_List(i) = Cells(2 + i, 26 + 5).Value
        OYard_Build_Time_List(i) = Cells(2 + i, 26 + 4).Value
        OYard_Priority_List(i) = Cells(2 + i, 26 + 20).Value
    Next i

Call Time_Sort(OYard_Build_Time_List, OYard_Train_List, OYard_Priority_List, OYard_List)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Priority Trains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    For i = 1 To NumTrains
     If OYard_Priority_List(i) = 1 Then

        T1Start(i) = Round(OYard_Build_Time_List(i) * 2880, 0)
        T1Stop(i) = Round((OYard_Build_Time_List(i) + ServiceTime(OYard_List(i))) * 2880, 0)
            If T1Start(i) = 0 Then
                T1Start(i) = 1
                T1Stop(i) = T1Stop(i) + 1
            End If
' check if T1Stop(i) goes into next day.
        x = 0

        If T1Stop(i) < 2880 Then
            For j = T1Start(i) To T1Stop(i)
                x = Queue_List_OBuild(OYard_List(i), j) + x
```

```
            Next j
        Else
            For j = T1Start(i) To 2880
                x = Queue_List_OBuild(OYard_List(i), j) + x
            Next j

            For j = 1 To T1Stop(i) - 2880
                x = Queue_List_OBuild(OYard_List(i), j) + x
            Next j
        End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        If x = 0 Then
            If T1Stop(i) < 2880 Then
                For j = T1Start(i) To T1Stop(i)
                    Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
                Next j
            Else
                For j = T1Start(i) To 2880
                    Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
                Next j

                For j = 1 To T1Stop(i) - 2880
                   Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
                Next j
            End If

        Else

            Do Until x = 0

                x = 0
                T1Start(i) = T1Start(i) + 1
                T1Stop(i) = T1Stop(i) + 1

                If T1Stop(i) < 2880 Then
                    For j = T1Start(i) To T1Stop(i)
                        x = Queue_List_OBuild(OYard_List(i), j) + x
                    Next j
                Else
                    For j = T1Start(i) To 2880
                        x = Queue_List_OBuild(OYard_List(i), j) + x
                    Next j

                    For j = 1 To T1Stop(i) - 2880
                        x = Queue_List_OBuild(OYard_List(i), j) + x
                    Next j
                End If

            Loop
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
            If T1Stop(i) < 2880 Then
                For j = T1Start(i) To T1Stop(i)
                    Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
                Next j
            Else
                For j = T1Start(i) To 2880
```

```
            Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
          Next j

          For j = 1 To T1Stop(i) - 2880
           Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
          Next j
        End If

        OYard_Build_Time_List(i) = (T1Start(i)) / 2880

      End If
    End If
   Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Non - Priority Trains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   For i = 1 To NumTrains
    If OYard_Priority_List(i) = 0 Then

      T1Start(i) = Round(OYard_Build_Time_List(i) * 2880, 0)
      T1Stop(i) = Round((OYard_Build_Time_List(i) + ServiceTime(OYard_List(i))) * 2880, 0)
        If T1Start(i) = 0 Then
          T1Start(i) = 1
          T1Stop(i) = T1Stop(i) + 1
        End If
' check if T1Stop(i) goes into next day.
      x = 0

      If T1Stop(i) < 2880 Then
        For j = T1Start(i) To T1Stop(i)
          x = Queue_List_OBuild(OYard_List(i), j) + x
        Next j
      Else
        For j = T1Start(i) To 2880
          x = Queue_List_OBuild(OYard_List(i), j) + x
        Next j

        For j = 1 To T1Stop(i) - 2880
          x = Queue_List_OBuild(OYard_List(i), j) + x
        Next j
      End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      If x = 0 Then
        If T1Stop(i) < 2880 Then
          For j = T1Start(i) To T1Stop(i)
            Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
          Next j
        Else
          For j = T1Start(i) To 2880
            Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
          Next j

          For j = 1 To T1Stop(i) - 2880
           Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
          Next j
        End If
```

```
      Else

        Do Until x = 0

          x = 0
          T1Start(i) = T1Start(i) + 1
          T1Stop(i) = T1Stop(i) + 1

          If T1Stop(i) < 2880 Then
            For j = T1Start(i) To T1Stop(i)
              x = Queue_List_OBuild(OYard_List(i), j) + x
            Next j
          Else
            For j = T1Start(i) To 2880
              x = Queue_List_OBuild(OYard_List(i), j) + x
            Next j
            j = 0
            For j = 1 To (T1Stop(i) - 2880)
              If j <= 2880 Then
              x = Queue_List_OBuild(OYard_List(i), j) + x
              End If
            Next j
          End If

        Loop
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        If T1Stop(i) < 2880 Then
          For j = T1Start(i) To T1Stop(i)
            Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
          Next j
        Else
          For j = T1Start(i) To 2880
            Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
          Next j

          For j = 1 To T1Stop(i) - 2880
           Queue_List_OBuild(OYard_List(i), j) = OYard_Train_List(i)
          Next j
        End If

        OYard_Build_Time_List(i) = (T1Start(i)) / 2880

      End If
    End If
   Next
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Sheets("Queue Times").Select

   For i = 1 To NumYards
      x = (i - 1) * 2880
      For j = 1 To 2880

        Cells(1 + j + x, 4) = Queue_List_OBuild(i, j)

      Next j
```

```
    Next i

        Sheets("O-D Matrices").Select
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    For i = 1 To NumTrains
        T1New(i) = (T1Start(i) / 2880)
    Next i
Call Train_Sort(OYard_Train_List, OYard_Priority_List, OYard_List, T1New)
    For i = 1 To NumTrains
        Cells(i + 2, 26 + 25) = T1New(i)
    Next i

Application.ScreenUpdating = True
End Sub

Sub Recalulate_CYard_and_DYard_Arrivals()
Application.ScreenUpdating = False

For TrainNumber = 1 To NumTrains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get build time
    If Cells(TrainNumber + 2, 26 + 25).Value = "" Then
        MsgBox "You have not chosen a build time"
        Exit Sub
    Else
        T1(TrainNumber) = Cells(TrainNumber + 2, 26 + 25).Value ' start time period for collection of cars
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get OYard
    If Cells(TrainNumber + 2, 26 + 5).Value = "" Then
        MsgBox "You have not chosen an origin yard"
        Exit Sub
    Else
        OYard = Cells(TrainNumber + 2, 26 + 5).Value ' yard number (Origin)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get CYard
    If Cells(TrainNumber + 2, 26 + 6).Value = "" Then
        CYard = 0
    Else
        CYard = Cells(TrainNumber + 2, 26 + 6).Value ' yard number (Connection)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get DYard
    If Cells(TrainNumber + 2, 26 + 7).Value = "" Then
        MsgBox "You have not chosen a destination yard"
        Exit Sub
    Else
        DYard = Cells(TrainNumber + 2, 26 + 7).Value ' yard number (Destination)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Recalculate CYard and DYard Arrival Times
    If CYard > 0 Then
        T2(TrainNumber) = T1(TrainNumber) + ServiceTime(OYard) + ServiceTime(CYard) / 2 +
TrainTime1(TrainNumber) + TrainTime2(TrainNumber) ' overall travel and service time for each train
    Else
```

```
        T2(TrainNumber) = T1(TrainNumber) + ServiceTime(OYard) + TrainTime1(TrainNumber) ' overall travel and
service time for each train
    End If

    If CYard > 0 Then
        If T1(TrainNumber) + TrainTime1(TrainNumber) + ServiceTime(OYard) > 1 Then
            Cells(TrainNumber + 2, 26 + 26) = T1(TrainNumber) + TrainTime1(TrainNumber) + ServiceTime(OYard) -
1
            Cells(TrainNumber + 2, 26 + 26).Interior.Color = 65535
            Cells(TrainNumber + 2, 26 + 26).Font.Color = -16776961
        Else
            Cells(TrainNumber + 2, 26 + 26) = T1(TrainNumber) + TrainTime1(TrainNumber) + ServiceTime(OYard)
        End If
    End If

    If T2(TrainNumber) > 1 Then
        Cells(TrainNumber + 2, 26 + 27) = T2(TrainNumber) - 1
        Cells(TrainNumber + 2, 26 + 27).Interior.Color = 65535
        Cells(TrainNumber + 2, 26 + 27).Font.Color = -16776961
    Else
        Cells(TrainNumber + 2, 26 + 27) = T2(TrainNumber)
    End If
Next TrainNumber

Application.ScreenUpdating = True
End Sub

Sub New_CYard_Times()
Application.ScreenUpdating = False
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
ReDim T2Start(1 To NumTrains)
ReDim T2Stop(1 To NumTrains)

ReDim T2New(1 To NumTrains)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Sheets("O-D Matrices").Select

    For i = 1 To NumTrains
        CYard_Train_List(i) = Cells(2 + i, 26 + 3).Value
        CYard_List(i) = Cells(2 + i, 26 + 6).Value
        CYard_Build_Time_List(i) = Cells(2 + i, 26 + 26).Value
        CYard_Priority_List(i) = Cells(2 + i, 26 + 20).Value
    Next i
Call Time_Sort(CYard_Build_Time_List, CYard_Train_List, CYard_Priority_List, CYard_List)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Priority Trains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    For i = 1 To NumTrains
     If CYard_Priority_List(i) = 1 And CYard_List(i) <> 0 Then

        T2Start(i) = Round(CYard_Build_Time_List(i) * 2880, 0)
        T2Stop(i) = Round((CYard_Build_Time_List(i) + ServiceTime(CYard_List(i))) * 2880 / 2, 0) ' assumed that
CYard operations are shorter than OYard
            If T2Start(i) = 0 Then
                T2Start(i) = 1
                T2Stop(i) = T2Stop(i) + 1
```

```
        End If

' check if T2Stop(i) goes into next day.
    x = 0

    If T2Stop(i) < 2880 Then
        For j = T2Start(i) To T2Stop(i)
            x = Queue_List_CBuild(CYard_List(i), j) + x
        Next j
    Else
        For j = T2Start(i) To 2880
            x = Queue_List_CBuild(CYard_List(i), j) + x
        Next j

        For j = 1 To T2Stop(i) - 2880
            x = Queue_List_CBuild(CYard_List(i), j) + x
        Next j
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    If x = 0 Then
        If T2Stop(i) < 2880 Then
            For j = T2Start(i) To T2Stop(i)
                Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
            Next j
        Else
            For j = T2Start(i) To 2880
                Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
            Next j

            For j = 1 To T2Stop(i) - 2880
               Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
            Next j
        End If

    Else

        Do Until x = 0

            x = 0
            T2Start(i) = T2Start(i) + 1
            T2Stop(i) = T2Stop(i) + 1

            If T2Stop(i) < 2880 Then
                For j = T2Start(i) To T2Stop(i)
                    x = Queue_List_CBuild(CYard_List(i), j) + x
                Next j
            Else
                For j = T2Start(i) To 2880
                    x = Queue_List_CBuild(CYard_List(i), j) + x
                Next j

                For j = 1 To T2Stop(i) - 2880
                    x = Queue_List_CBuild(CYard_List(i), j) + x
                Next j
            End If
```

```
            Loop
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
            If T2Stop(i) < 2880 Then
                For j = T2Start(i) To T2Stop(i)
                    Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
                Next j
            Else
                For j = T2Start(i) To 2880
                    Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
                Next j

                For j = 1 To T2Stop(i) - 2880
                   Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
                Next j
            End If

            CYard_Build_Time_List(i) = (T2Start(i)) / 2880

        End If
     End If
    Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Non - Priority Trains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    For i = 1 To NumTrains
     If CYard_Priority_List(i) = 0 And CYard_List(i) <> 0 Then

        T2Start(i) = Round(CYard_Build_Time_List(i) * 2880, 0)
        T2Stop(i) = Round((CYard_Build_Time_List(i) + ServiceTime(CYard_List(i))) * 2880 / 2, 0) ' assumed that
CYard operations are shorter than OYard
            If T2Start(i) = 0 Then
                T2Start(i) = 1
                T2Stop(i) = T2Stop(i) + 1
            End If

' check if T2Stop(i) goes into next day.
        x = 0

        If T2Stop(i) < 2880 Then
            For j = T2Start(i) To T2Stop(i)
                x = Queue_List_CBuild(CYard_List(i), j) + x
            Next j
        Else
            For j = T2Start(i) To 2880
                x = Queue_List_CBuild(CYard_List(i), j) + x
            Next j

            For j = 1 To T2Stop(i) - 2880
                x = Queue_List_CBuild(CYard_List(i), j) + x
            Next j
        End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        If x = 0 Then
            If T2Stop(i) < 2880 Then
                For j = T2Start(i) To T2Stop(i)
                    Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
```

```
          Next j
        Else
          For j = T2Start(i) To 2880
            Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
          Next j

          For j = 1 To T2Stop(i) - 2880
           Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
          Next j
        End If

      Else

        Do Until x = 0

          x = 0
          T2Start(i) = T2Start(i) + 1
          T2Stop(i) = T2Stop(i) + 1

          If T2Stop(i) < 2880 Then
            For j = T2Start(i) To T2Stop(i)
              x = Queue_List_CBuild(CYard_List(i), j) + x
            Next j
          Else
            For j = T2Start(i) To 2880
              x = Queue_List_CBuild(CYard_List(i), j) + x
            Next j

            For j = 1 To T2Stop(i) - 2880
              x = Queue_List_CBuild(CYard_List(i), j) + x
            Next j
          End If

        Loop
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        If T2Stop(i) < 2880 Then
          For j = T2Start(i) To T2Stop(i)
            Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
          Next j
        Else
          For j = T2Start(i) To 2880
            Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
          Next j

          For j = 1 To T2Stop(i) - 2880
           Queue_List_CBuild(CYard_List(i), j) = CYard_Train_List(i)
          Next j
        End If

        CYard_Build_Time_List(i) = (T2Start(i)) / 2880

      End If
    End If
  Next i

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
        Sheets("Queue Times").Select

    For i = 1 To NumYards
        x = (i - 1) * 2880
        For j = 1 To 2880

            Cells(1 + j + x, 5) = Queue_List_CBuild(i, j)

        Next j
    Next i

        Sheets("O-D Matrices").Select
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    For i = 1 To NumTrains
        T2New(i) = (T2Start(i) / 2880)
    Next i

Call Train_Sort(CYard_Train_List, CYard_Priority_List, CYard_List, T2New)

    For i = 1 To NumTrains
        If T2New(i) <> 0 Then Cells(i + 2, 26 + 26) = T2New(i)
    Next i

Application.ScreenUpdating = True
End Sub

Sub Recalulate_DYard_Arrivals()
Application.ScreenUpdating = False

For TrainNumber = 1 To NumTrains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get CYARD build time
    If Cells(TrainNumber + 2, 26 + 26).Value = "" Then
        TC(TrainNumber) = 0
    Else
        TC(TrainNumber) = Cells(TrainNumber + 2, 26 + 26).Value ' start time period for collection of cars at
CYARD ONLY
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get CYard
    If Cells(TrainNumber + 2, 26 + 6).Value = "" Then
        CYard = 0
    Else
        CYard = Cells(TrainNumber + 2, 26 + 6).Value ' yard number (Connection)
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Recalculate CYard and DYard Arrival Times
    If CYard > 0 Then
        T2(TrainNumber) = TC(TrainNumber) + ServiceTime(CYard) / 2 + TrainTime2(TrainNumber) ' overall travel
and service time for each train

        If T2(TrainNumber) > 1 Then
            Cells(TrainNumber + 2, 26 + 27) = T2(TrainNumber) - 1
            Cells(TrainNumber + 2, 26 + 27).Interior.Color = 65535
            Cells(TrainNumber + 2, 26 + 27).Font.Color = -16776961
        Else
```

```
            Cells(TrainNumber + 2, 26 + 27) = T2(TrainNumber)
        End If

    End If

Next TrainNumber

Application.ScreenUpdating = True
End Sub

Sub New_DYard_Times()
Application.ScreenUpdating = True

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
ReDim T3Start(1 To NumTrains)
ReDim T3Stop(1 To NumTrains)

ReDim T3New(1 To NumTrains)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Sheets("O-D Matrices").Select

    For i = 1 To NumTrains
        DYard_Train_List(i) = Cells(2 + i, 26 + 3).Value
        DYard_List(i) = Cells(2 + i, 26 + 7).Value
        DYard_Build_Time_List(i) = Cells(2 + i, 26 + 27).Value
        DYard_Priority_List(i) = Cells(2 + i, 26 + 20).Value
    Next i

Call Time_Sort(DYard_Build_Time_List, DYard_Train_List, DYard_Priority_List, DYard_List)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Priority Trains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    For i = 1 To NumTrains
     If DYard_Priority_List(i) = 1 Then

        T3Start(i) = Round(DYard_Build_Time_List(i) * 2880, 0)
        T3Stop(i) = Round((DYard_Build_Time_List(i) + TrainSwitchTime(DYard_List(i))) * 2880, 0) ' assumed that
DYard operations (Switching are constant per train)
            If T3Start(i) = 0 Then
                T3Start(i) = 1
                T3Stop(i) = T3Stop(i) + 1
            End If

' check if T3Stop(i) goes into next day.
        x = 0

        If T3Stop(i) < 2880 Then
            For j = T3Start(i) To T3Stop(i)
                x = Queue_List_DSwitch(DYard_List(i), j) + x
            Next j
        Else
            For j = T3Start(i) To 2880
                x = Queue_List_DSwitch(DYard_List(i), j) + x
            Next j

            For j = 1 To T3Stop(i) - 2880
```

```
            x = Queue_List_DSwitch(DYard_List(i), j) + x
        Next j
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    If x = 0 Then
        If T3Stop(i) < 2880 Then
            For j = T3Start(i) To T3Stop(i)
                Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j
        Else
            For j = T3Start(i) To 2880
                Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j

            For j = 1 To T3Stop(i) - 2880
              Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j
        End If

    Else

        Do Until x = 0

            x = 0
            T3Start(i) = T3Start(i) + 1
            T3Stop(i) = T3Stop(i) + 1

            If T3Stop(i) < 2880 Then
                For j = T3Start(i) To T3Stop(i)
                    x = Queue_List_DSwitch(DYard_List(i), j) + x
                Next j
            Else
                For j = T3Start(i) To 2880
                    x = Queue_List_DSwitch(DYard_List(i), j) + x
                Next j

                For j = 1 To T3Stop(i) - 2880
                    x = Queue_List_DSwitch(DYard_List(i), j) + x
                Next j
            End If

        Loop
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        If T3Stop(i) < 2880 Then
            For j = T3Start(i) To T3Stop(i)
                Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j
        Else
            For j = T3Start(i) To 2880
                Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j

            For j = 1 To T3Stop(i) - 2880
              Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j
        End If
```

```
          DYard_Build_Time_List(i) = (T3Start(i)) / 2880

        End If
      End If
    Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Non - Priority Trains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    For i = 1 To NumTrains
     If DYard_Priority_List(i) = 0 Then

        T3Start(i) = Round(DYard_Build_Time_List(i) * 2880, 0)
        T3Stop(i) = Round((DYard_Build_Time_List(i) + TrainSwitchTime(DYard_List(i))) * 2880, 0) ' assumed that
DYard operations (Switching are constant per train)
          If T3Start(i) = 0 Then
            T3Start(i) = 1
            T3Stop(i) = T3Stop(i) + 1
          End If

' check if T3Stop(i) goes into next day.
        x = 0

        If T3Stop(i) < 2880 Then
          For j = T3Start(i) To T3Stop(i)
            x = Queue_List_DSwitch(DYard_List(i), j) + x
          Next j
        Else
          For j = T3Start(i) To 2880
            x = Queue_List_DSwitch(DYard_List(i), j) + x
          Next j

          For j = 1 To T3Stop(i) - 2880
            x = Queue_List_DSwitch(DYard_List(i), j) + x
          Next j
        End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        If x = 0 Then
          If T3Stop(i) < 2880 Then
            For j = T3Start(i) To T3Stop(i)
              Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j
          Else
            For j = T3Start(i) To 2880
              Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j

            For j = 1 To T3Stop(i) - 2880
             Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j
          End If

        Else

          Do Until x = 0
```

```
            x = 0
            T3Start(i) = T3Start(i) + 1
            T3Stop(i) = T3Stop(i) + 1

            If T3Stop(i) < 2880 Then
                For j = T3Start(i) To T3Stop(i)
                    x = Queue_List_DSwitch(DYard_List(i), j) + x
                Next j
            Else
                For j = T3Start(i) To 2880
                    x = Queue_List_DSwitch(DYard_List(i), j) + x
                Next j

                For j = 1 To T3Stop(i) - 2880
                    x = Queue_List_DSwitch(DYard_List(i), j) + x
                Next j
            End If

        Loop
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        If T3Stop(i) < 2880 Then
            For j = T3Start(i) To T3Stop(i)
                Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j
        Else
            For j = T3Start(i) To 2880
                Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j

            For j = 1 To T3Stop(i) - 2880
              Queue_List_DSwitch(DYard_List(i), j) = DYard_Train_List(i)
            Next j
        End If

        DYard_Build_Time_List(i) = (T3Start(i)) / 2880

    End If
  End If
 Next i
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Sheets("Queue Times").Select

 For i = 1 To NumYards
    x = (i - 1) * 2880
    For j = 1 To 2880

        Cells(1 + j + x, 6) = Queue_List_DSwitch(i, j)

    Next j
 Next i

    Sheets("O-D Matrices").Select
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
 For i = 1 To NumTrains
    T3New(i) = (T3Start(i) / 2880)
 Next i
```

```
Call Train_Sort(DYard_Train_List, DYard_Priority_List, DYard_List, T3New)

   For i = 1 To NumTrains
      Cells(i + 2, 26 + 27) = T3New(i)
   Next i

Application.ScreenUpdating = True
End Sub

Sub Time_Sort(list1() As Date, list2() As Long, list3() As Long, list4() As Long)
Application.ScreenUpdating = False

' Sorts array

   Dim First As Integer, Last As Long
   Dim i As Long, j As Long
   Dim Temp1
   Dim Temp2
   Dim Temp3
   Dim Temp4

   First = LBound(list1)
   Last = UBound(list1)
   For i = First To Last - 1
      For j = i + 1 To Last
         If list1(i) > list1(j) Then
            Temp1 = list1(j)
            list1(j) = list1(i)
            list1(i) = Temp1

            Temp2 = list2(j)
            list2(j) = list2(i)
            list2(i) = Temp2

            Temp3 = list3(j)
            list3(j) = list3(i)
            list3(i) = Temp3

            Temp4 = list4(j)
            list4(j) = list4(i)
            list4(i) = Temp4
         End If
      Next j
   Next i

Application.ScreenUpdating = True
End Sub

Sub Train_Sort(list1() As Long, list2() As Long, list3() As Long, list4() As Date)
Application.ScreenUpdating = False

' Sorts array

   Dim First As Integer, Last As Long
   Dim i As Long, j As Long
```

```
    Dim Temp1
    Dim Temp2
    Dim Temp3
    Dim Temp4

    First = LBound(list1)
    Last = UBound(list1)
    For i = First To Last - 1
       For j = i + 1 To Last
          If list1(i) > list1(j) Then
             Temp1 = list1(j)
             list1(j) = list1(i)
             list1(i) = Temp1

             Temp2 = list2(j)
             list2(j) = list2(i)
             list2(i) = Temp2

             Temp3 = list3(j)
             list3(j) = list3(i)
             list3(i) = Temp3

             Temp4 = list4(j)
             list4(j) = list4(i)
             list4(i) = Temp4
          End If
       Next j
    Next i

Application.ScreenUpdating = True
End Sub


Sub Adjust_Times_24()
For TrainNumber = 1 To NumTrains
    Cells(TrainNumber + 2, 26 + 22).Select
    If Selection.Interior.Color = 65535 Then
       ActiveCell = ActiveCell.Value + 1
    End If
Next TrainNumber

For TrainNumber = 1 To NumTrains
    Cells(TrainNumber + 2, 26 + 23).Select
    If Selection.Interior.Color = 65535 Then
       ActiveCell = ActiveCell.Value + 1
    End If

For TrainNumber = 1 To NumTrains
    Cells(TrainNumber + 2, 26 + 26).Select
    If Selection.Interior.Color = 65535 Then
       ActiveCell = ActiveCell.Value + 1
    End If
Next TrainNumber

For TrainNumber = 1 To NumTrains
    Cells(TrainNumber + 2, 26 + 27).Select
```

```
    If Selection.Interior.Color = 65535 Then
        ActiveCell = ActiveCell.Value + 1
End If
Next TrainNumber
End Sub

Sub Get_Shortest_Routes()

    Dim OYard_Dist_List() As Long ' oyards
    Dim DYard_Dist_List() As Long ' dyard
    Dim Length_Dist_List() As Long ' dyard

    Sheets("Routes").Select

    ReDim Shortest_Yard_Dist(1 To NumYards, 1 To NumYards)
    ReDim OYard_Dist_List(1 To NumRoutes)
    ReDim DYard_Dist_List(1 To NumRoutes)
    ReDim Length_Dist_List(1 To NumRoutes)

    For i = 1 To NumRoutes
        OYard_Dist_List(i) = Cells(2 + i, 3).Value
        DYard_Dist_List(i) = Cells(2 + i, 4).Value
        Length_Dist_List(i) = Cells(2 + i, 6).Value
    Next i

    For i = 1 To NumRoutes
        If Shortest_Yard_Dist(OYard_Dist_List(i), DYard_Dist_List(i)) = 0 Then
            Shortest_Yard_Dist(OYard_Dist_List(i), DYard_Dist_List(i)) = Length_Dist_List(i)
        End If

        If Shortest_Yard_Dist(OYard_Dist_List(i), DYard_Dist_List(i)) <> 0 Then
            If Shortest_Yard_Dist(OYard_Dist_List(i), DYard_Dist_List(i)) > Length_Dist_List(i) Then
                Shortest_Yard_Dist(OYard_Dist_List(i), DYard_Dist_List(i)) = Length_Dist_List(i)
            End If
        End If
    Next i

    Range("List_of_Shortest_Paths").Select
    For i = 1 To NumYards
        For j = i To NumYards
            ActiveCell.Offset(i, j) = Shortest_Yard_Dist(i, j)
            ActiveCell.Offset(j, i) = Shortest_Yard_Dist(j, i)
        Next j
    Next i

End Sub
```

Module 4

```
Public Shortest_Yard_Dist() As Long ' shortest distances between yards - based on routes provided by user

Sub Assign_Blocks()
Application.ScreenUpdating = False
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' GET SHORTEST ROUTES
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
   Call Get_Shortest_Routes
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' GET BLOCKS SETUP
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   Sheets("O-D Matrices").Select

   For j = 1 To NumYards
      For k = 1 To NumYards
         If Cells(k + 3, 3).Value <= NumBlocks(j) Then
               Blocks(j, k) = Cells(3 + (3 + NumYards) * (j - 1) + k, 3).Value
               Blocks_Pure_Mixed(j, k) = Cells(3 + (3 + NumYards) * (j - 1) + k, 4).Value
         End If
      Next k
   Next j
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' ASSIGN DIRECT BLOCKS (Pure)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

   For TrainNumber = 1 To NumTrains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Get OYard, CYard, and DYard
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      If Cells(TrainNumber + 2, 26 + 5).Value = "" Then
         MsgBox "You have not chosen an origin yard for train number" & TrainNumber & "."
         Exit Sub
      Else
         OYard = Cells(TrainNumber + 2, 26 + 5).Value ' yard number (Origin)
      End If


      If Cells(TrainNumber + 2, 26 + 6).Value = "" Then
         CYard = 0
      Else
         CYard = Cells(TrainNumber + 2, 26 + 6).Value ' yard number (Connection)
      End If


      If Cells(TrainNumber + 2, 26 + 7).Value = "" Then
         MsgBox "You have not chosen a destination yard for train number" & TrainNumber & "."
         Exit Sub
      Else
         DYard = Cells(TrainNumber + 2, 26 + 7).Value ' yard number (Destination)
      End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Pure Blocks Direct Assignment Only
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Cells(TrainNumber + 2, 26 + 13).Select

         Do While ActiveCell <> ""
            ActiveCell.Offset(0, 1).Select
         Loop

         If ActiveCell = "" Then
            If Blocks_Pure_Mixed(OYard, DYard) = 1 Then
               ActiveCell.Value = Blocks(OYard, DYard)
```

```
                If Cells(3 + (3 + NumYards) * (OYard - 1) + DYard, 5) = "" Then
                    Cells(3 + (3 + NumYards) * (OYard - 1) + DYard, 5) = TrainNumber
                Else
                    Cells(3 + (3 + NumYards) * (OYard - 1) + DYard, 5) = Cells(3 + (3 + NumYards) * (OYard - 1) +
DYard, 5).Value & ", " & TrainNumber
                End If

            End If
        End If

        Cells(TrainNumber + 2, 26 + 13).Select

        Do While ActiveCell <> ""
            ActiveCell.Offset(0, 1).Select
        Loop

        If ActiveCell = "" Then
            If CYard > 0 Then
                If Blocks_Pure_Mixed(CYard, DYard) = 1 Then
                    ActiveCell.Value = Blocks(CYard, DYard)
                    Selection.Font.Color = -16776961
                    Selection.Interior.Color = 65535

                    If CYard > 0 Then
                        If Cells(3 + (3 + NumYards) * (CYard - 1) + DYard, 5) = "" Then
                            Cells(3 + (3 + NumYards) * (CYard - 1) + DYard, 5) = TrainNumber
                        Else
                            Cells(3 + (3 + NumYards) * (CYard - 1) + DYard, 5) = Cells(3 + (3 + NumYards) * (CYard - 1)
+ DYard, 5).Value & ", " & TrainNumber
                        End If
                    End If

                End If
            End If
        End If

    Next TrainNumber
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' ASSIGN DIRECT BLOCKS (Impure)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    For TrainNumber = 1 To NumTrains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Get OYard, CYard, and DYard
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        OYard = Cells(TrainNumber + 2, 26 + 5).Value ' yard number (Origin)

        If Cells(TrainNumber + 2, 26 + 6).Value = "" Then
            CYard = 0
        Else
            CYard = Cells(TrainNumber + 2, 26 + 6).Value ' yard number (Connection)
        End If

        DYard = Cells(TrainNumber + 2, 26 + 7).Value ' yard number (Destination)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Impure Blocks Direct Assignment Only
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
        Cells(TrainNumber + 2, 26 + 13).Select

          Do While ActiveCell <> ""
            ActiveCell.Offset(0, 1).Select
          Loop

          If ActiveCell = "" Then
            If Blocks_Pure_Mixed(OYard, DYard) = 0 Then
              ActiveCell.Value = Blocks(OYard, DYard)

              If Cells(3 + (3 + NumYards) * (OYard - 1) + DYard, 5) = "" Then
                Cells(3 + (3 + NumYards) * (OYard - 1) + DYard, 5) = TrainNumber
              Else
                Cells(3 + (3 + NumYards) * (OYard - 1) + DYard, 5) = Cells(3 + (3 + NumYards) * (OYard - 1) +
DYard, 5).Value & ", " & TrainNumber
              End If

            End If
          End If

        Cells(TrainNumber + 2, 26 + 13).Select

          Do While ActiveCell <> ""
            ActiveCell.Offset(0, 1).Select
          Loop

          If ActiveCell = "" Then
            If CYard > 0 Then
              If Blocks_Pure_Mixed(CYard, DYard) = 0 Then
                ActiveCell.Value = Blocks(CYard, DYard)
                Selection.Font.Color = -16776961
                Selection.Interior.Color = 65535

                If CYard > 0 Then
                  If Cells(3 + (3 + NumYards) * (CYard - 1) + DYard, 5) = "" Then
                    Cells(3 + (3 + NumYards) * (CYard - 1) + DYard, 5) = TrainNumber
                  Else
                    Cells(3 + (3 + NumYards) * (CYard - 1) + DYard, 5) = Cells(3 + (3 + NumYards) * (CYard - 1)
+ DYard, 5).Value & ", " & TrainNumber
                  End If
                End If

              End If
            End If
          End If

    Next TrainNumber
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'Unassigned - BLOCKS ASSIGNMENT (Pure/Impure)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Dim Train_Assignment_List() As Long
    Dim OYard_Assignment_List() As Long
    Dim CYard_Assignment_List() As Long
    Dim DYard_Assignment_List() As Long
    Dim x As Long, y As Long, z As Long
```

```
   Dim Blocks_Assigned() As String

   ReDim Blocks_Assigned(1 To NumYards, 1 To NumYards)

   ReDim Train_Assignment_List(1 To NumTrains)
   ReDim OYard_Assignment_List(1 To NumTrains)
   ReDim CYard_Assignment_List(1 To NumTrains)
   ReDim DYard_Assignment_List(1 To NumTrains)

   For TrainNumber = 1 To NumTrains

      Train_Assignment_List(TrainNumber) = Cells(TrainNumber + 2, 26 + 3).Value ' yard number (Origin)

      OYard_Assignment_List(TrainNumber) = Cells(TrainNumber + 2, 26 + 5).Value ' yard number (Origin)

      If Cells(TrainNumber + 2, 26 + 6).Value = "" Then
         CYard_Assignment_List(TrainNumber) = 0
      Else
         CYard_Assignment_List(TrainNumber) = Cells(TrainNumber + 2, 26 + 6).Value ' yard number
(Connection)
      End If

      DYard_Assignment_List(TrainNumber) = Cells(TrainNumber + 2, 26 + 7).Value ' yard number (Destination)

   Next TrainNumber
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   For j = 1 To NumYards
      For k = 1 To NumYards
         If Cells(k + 3, 3).Value <= NumBlocks(j) Then
               Blocks_Assigned(j, k) = Cells(3 + (3 + NumYards) * (j - 1) + k, 5).Value
         End If
      Next k
   Next j
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
For j = 1 To NumYards
      For k = 1 To NumYards
         If j <> k Then
            If Blocks_Assigned(j, k) = "" Then

               x = 0

               For TrainNumber = 1 To NumTrains
                  If x = 0 And OYard_Assignment_List(TrainNumber) = j Then

                     ' Shortest_Yard_Dist(x,x)

                     If Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) < Shortest_Yard_Dist(j, k) Then
                        y = Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) - Shortest_Yard_Dist(j, k)
                     End If
                        x = TrainNumber ' assign first available trin no matter where it is going
                  ElseIf x <> 0 And OYard_Assignment_List(TrainNumber) = j Then

                     If Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) < Shortest_Yard_Dist(j, k) Then
                        z = Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) - Shortest_Yard_Dist(j, k)
                     End If
```

```
            '''' assign new train only when it is more sensible to do so (eg the train is going
            '''' towards but not passing the dyard and is the closest to the dyard)
            If y < z Then
                x = TrainNumber
                y = z
            End If
        End If
    Next TrainNumber
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    For TrainNumber = 1 To NumTrains
        If x = 0 And OYard_Assignment_List(TrainNumber) = j Then

            ' Shortest_Yard_Dist(x,x)

            If Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) > Shortest_Yard_Dist(j, k) Then
                y = Shortest_Yard_Dist(j, k) - Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber))
            End If
                x = TrainNumber ' assign first available trin no matter where it is going
        ElseIf x <> 0 And OYard_Assignment_List(TrainNumber) = j Then

            If Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) > Shortest_Yard_Dist(j, k) Then
                z = Shortest_Yard_Dist(j, k) - Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber))
            End If

            '''' assign new train only when it is more sensible to do so (eg the train is going
            '''' towards but not passing the dyard and is the closest to the dyard)
            If y < z Then
                x = TrainNumber
                y = z
            End If
        End If
    Next TrainNumber
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' cyard trains only if no oyard trains are available
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    For TrainNumber = 1 To NumTrains
        If x = 0 And CYard_Assignment_List(TrainNumber) = j Then

            ' Shortest_Yard_Dist(x,x)

            If Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) < Shortest_Yard_Dist(j, k) Then
                y = Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) - Shortest_Yard_Dist(j, k)
            End If
                x = TrainNumber ' assign first available trin no matter where it is going
        ElseIf x <> 0 And CYard_Assignment_List(TrainNumber) = j Then

            If Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) < Shortest_Yard_Dist(j, k) Then
                z = Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) - Shortest_Yard_Dist(j, k)
            End If

            '''' assign new train only when it is more sensible to do so (eg the train is going
            '''' towards but not passing the dyard and is the closest to the dyard)
            If y < z Then
                x = TrainNumber
                y = z
            End If
```

```
                    End If
                Next TrainNumber
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
                For TrainNumber = 1 To NumTrains
                    If x = 0 And CYard_Assignment_List(TrainNumber) = j Then

                        ' Shortest_Yard_Dist(x,x)

                        If Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) > Shortest_Yard_Dist(j, k) Then
                            y = Shortest_Yard_Dist(j, k) - Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber))
                        End If
                            x = TrainNumber ' assign first available trin no matter where it is going
                    ElseIf x <> 0 And CYard_Assignment_List(TrainNumber) = j Then

                        If Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber)) > Shortest_Yard_Dist(j, k) Then
                            z = Shortest_Yard_Dist(j, k) - Shortest_Yard_Dist(j, DYard_Assignment_List(TrainNumber))
                        End If

                        '''' assign new train only when it is more sensible to do so (eg the train is going
                        '''' towards but not passing the dyard and is the closest to the dyard)
                        If y < z Then
                            x = TrainNumber
                            y = z
                        End If
                    End If
                Next TrainNumber
            ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
            ' print data onto spreadsheet
            ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
            Cells(x + 2, 26 + 13).Select

                Do While ActiveCell > 0
                    ActiveCell.Offset(0, 1).Select
                Loop

                If ActiveCell = "" Then
                If x <> 0 Then
                    If OYard_Assignment_List(x) = j Then
                        ActiveCell.Value = Blocks(j, k)

                        If Cells(3 + (3 + NumYards) * (j - 1) + k, 5) = "" Then
                            Cells(3 + (3 + NumYards) * (j - 1) + k, 5) = x
                        Else
                            Cells(3 + (3 + NumYards) * (j - 1) + k, 5) = Cells(3 + (3 + NumYards) * (j - 1) + k, 5).Value &
", " & x
                        End If
                    End If
                End If
                End If

                If ActiveCell = "" Then
                If x <> 0 Then
                    If CYard_Assignment_List(x) = j Then
                        ActiveCell.Value = Blocks(j, k)
                        Selection.Font.Color = -16776961
                        Selection.Interior.Color = 65535
```

```
                    If CYard > 0 Then
                        If Cells(3 + (3 + NumYards) * (j - 1) + k, 5) = "" Then
                            Cells(3 + (3 + NumYards) * (j - 1) + k, 5) = x
                        Else
                            Cells(3 + (3 + NumYards) * (j - 1) + k, 5) = Cells(3 + (3 + NumYards) * (j - 1) + k, 5).Value
& ", " & x
                        End If
                    End If
                End If
            End If
            End If
        End If
    End If
  Next k
Next j
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Assure that Each Impure and Indirect Block is Assigned to the Same Train(s) - ELSE ALERT USER
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  ReDim Blocks_Assigned(1 To NumYards, 1 To NumYards)

  For j = 1 To NumYards
    For k = 1 To NumYards
      If Cells(k + 3, 3).Value <= NumBlocks(j) Then
          Blocks(j, k) = Cells(3 + (3 + NumYards) * (j - 1) + k, 3).Value
          Blocks_Pure_Mixed(j, k) = Cells(3 + (3 + NumYards) * (j - 1) + k, 4).Value
          Blocks_Assigned(j, k) = Cells(3 + (3 + NumYards) * (j - 1) + k, 5).Value
        Else
          Blocks(j, k) = 0
          Blocks_Pure_Mixed(j, k) = 0
          Blocks_Assigned(j, k) = 0
      End If
    Next k
  Next j

  For i = 1 To NumYards
    For j = 1 To NumYards
      For k = 1 To NumYards
        If j <> k Then
          If Blocks_Pure_Mixed(i, j) = 0 And Blocks_Pure_Mixed(i, k) = 0 Then
            If Blocks(i, j) = Blocks(i, k) Then
              If Blocks_Assigned(i, j) <> Blocks_Assigned(i, k) Then

                If Blocks_Assigned(i, j) = "" And Blocks_Assigned(i, k) <> "" Then
                  Cells(3 + (3 + NumYards) * (i - 1) + j, 5).Value = Cells(3 + (3 + NumYards) * (i - 1) + k,
5).Value
                End If

                If Blocks_Assigned(i, j) <> "" And Blocks_Assigned(i, k) = "" Then
                  Cells(3 + (3 + NumYards) * (i - 1) + k, 5).Value = Cells(3 + (3 + NumYards) * (i - 1) + j,
5).Value
                End If

                If Blocks_Assigned(i, j) <> "" And Blocks_Assigned(i, k) <> "" And Blocks_Assigned(i, j) <>
Blocks_Assigned(i, k) Then
                  Cells(3 + (3 + NumYards) * (i - 1) + j, 5).Select
```

```
                    Selection.Interior.ThemeColor = xlThemeColorLight2
                    Selection.Font.ThemeColor = xlThemeColorDark1
                    Selection.Font.Bold = True
                    Selection.Font.Size = 16

                  Cells(3 + (3 + NumYards) * (i - 1) + k, 5).Select
                    Selection.Interior.ThemeColor = xlThemeColorLight2
                    Selection.Font.ThemeColor = xlThemeColorDark1
                    Selection.Font.Bold = True
                    Selection.Font.Size = 16

                End If
              End If
            End If
          End If
        End If
      Next k
    Next j
  Next i

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Assure that only only OD is to each train - ELSE ALERT USER
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
  Call TEST_SAME_TRAIN
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Application.ScreenUpdating = True
End Sub

Sub TEST_SAME_TRAIN()
  Dim saad_1() As Long
  Dim saad_2() As Long
  Dim saad_3() As Long

  ReDim saad_1(1 To NumTrains)
  ReDim saad_2(1 To NumTrains)
  ReDim saad_3(1 To NumTrains)

  For i = 1 To NumTrains

    saad_1(i) = Cells(i + 2, 26 + 5)
    saad_2(i) = Cells(i + 2, 26 + 6)
    saad_3(i) = Cells(i + 2, 26 + 7)

  Next i

  For i = 1 To NumTrains
    For j = 1 To NumTrains

      If i <> j Then
        If saad_1(i) = saad_1(j) And saad_2(i) = saad_2(j) And saad_3(i) = saad_3(j) Then
          Cells(i + 2, 26 + 5).Font.Bold = True
          Cells(i + 2, 26 + 6).Font.Bold = True
          Cells(i + 2, 26 + 7).Font.Bold = True

          Cells(i + 2, 26 + 5).Interior.Color = 65535
          Cells(i + 2, 26 + 6).Interior.Color = 65535
```

```
            Cells(i + 2, 26 + 7).Interior.Color = 65535
          End If
        End If

      Next j
    Next i

End Sub
```

Module 5

```
Sub Train_Call_Run_XX()
Application.ScreenUpdating = False
   For TrainNumber = 1 To NumTrains
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get OYard
   If Cells(TrainNumber + 2, 26 + 5).Value = "" Then
      MsgBox "You have not chosen an origin yard"
      Exit Sub
   Else
      OYard = Cells(TrainNumber + 2, 26 + 5).Value ' yard number (Origin)
   End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get CYard
   If Cells(TrainNumber + 2, 26 + 6).Value = "" Then
      CYard = 0
   Else
      CYard = Cells(TrainNumber + 2, 26 + 6).Value ' yard number (Connection)
   End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' get DYard
   If Cells(TrainNumber + 2, 26 + 7).Value = "" Then
      MsgBox "You have not chosen a destination yard"
      Exit Sub
   Else
      DYard = Cells(TrainNumber + 2, 26 + 7).Value ' yard number (Destination)
   End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   T1(TrainNumber) = Cells(TrainNumber + 2, 26 + 25).Value
   TC(TrainNumber) = Cells(TrainNumber + 2, 26 + 26).Value
   T2(TrainNumber) = Cells(TrainNumber + 2, 26 + 27).Value

   Call TrainTimes_XX(TrainNumber, OYard, CYard, DYard, T1(TrainNumber), TC(TrainNumber),
T2(TrainNumber))

   Next TrainNumber

Application.ScreenUpdating = True
End Sub
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Sub TrainTimes_XX(TrainNumber, OYard, CYard, DYard, TSub1, TSubC, TSub2)
Application.ScreenUpdating = False

   T1(TrainNumber) = TSub1
   TC(TrainNumber) = TSubC
   T2(TrainNumber) = TSub2
```

```
   Call ReCreateODArray
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Check Minimum Train Size
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   Trains(TrainNumber) = 0 ' Reset Trains(TrainNumber)

   If T1(TrainNumber) > #11:59:00 PM# Then TX1(TrainNumber) = 6 ' assumed that only cars from today can be
carried on any given train
   If T1(TrainNumber) <= #11:59:00 PM# Then TX1(TrainNumber) = 6
   If T1(TrainNumber) <= #8:00:00 PM# Then TX1(TrainNumber) = 5
   If T1(TrainNumber) <= #4:00:00 PM# Then TX1(TrainNumber) = 4
   If T1(TrainNumber) <= #12:00:00 PM# Then TX1(TrainNumber) = 3
   If T1(TrainNumber) <= #8:00:00 AM# Then TX1(TrainNumber) = 2
   If T1(TrainNumber) <= #4:00:00 AM# Then TX1(TrainNumber) = 1

   Cells(TrainNumber + 2, 26 + 13).Select
      For i = 1 To NumYards
         If Selection.Font.ColorIndex = xlAutomatic Then
            x = ActiveCell.Value
         Else
            x = 0
         End If
         If x > 0 And x <= NumYards Then
               For j = 1 To NumYards
            If x = Blocks(OYard, j) Then
               For k = 1 To TX1(TrainNumber) ' the time period for the cars to be taken from
                  Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP1(OYard, j, k)
               Next k
            End If
               Next j
         End If
         ActiveCell.Offset(0, 1).Select
      Next i

   Cells(TrainNumber + 2, 26 + 13).Select
      For i = 1 To NumYards
         If Selection.Font.ColorIndex = xlAutomatic Then
            x = ActiveCell.Value
         Else
            x = 0
         End If
         If x > 0 And x <= NumYards Then
               For j = 1 To NumYards
            If x = Blocks(OYard, j) Then
               For k = 1 To TX1(TrainNumber) ' the time period for the cars to be taken from
                  Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP2(OYard, j, k)
               Next k
            End If
               Next j
         End If
         ActiveCell.Offset(0, 1).Select
      Next i

   If CYard = 0 Then
```

```
        CYard = 0

    Else

' because the train gets to CYard after some time, therefore additional cars may be picked up

        If TC(TrainNumber) > #11:59:00 PM# Then TXC1(TrainNumber) = 6 ' because cars on any train can only be
picked up from the day of the OD Demand (Assumed)
        If TC(TrainNumber) <= #11:59:00 PM# Then TXC1(TrainNumber) = 6
        If TC(TrainNumber) <= #8:00:00 PM# Then TXC1(TrainNumber) = 5
        If TC(TrainNumber) <= #4:00:00 PM# Then TXC1(TrainNumber) = 4
        If TC(TrainNumber) <= #12:00:00 PM# Then TXC1(TrainNumber) = 3
        If TC(TrainNumber) <= #8:00:00 AM# Then TXC1(TrainNumber) = 2
        If TC(TrainNumber) <= #4:00:00 AM# Then TXC1(TrainNumber) = 1

        Cells(TrainNumber + 2, 26 + 13).Select
            For i = 1 To NumYards
            If Selection.Interior.Color = 65535 Then
                x = ActiveCell.Value
            Else
                x = 0
            End If
                If x > 0 And x <= NumYards Then
                        For j = 1 To NumYards
                    If x = Blocks(CYard, j) Then
                        For k = 1 To TXC1(TrainNumber) ' the time period for the cars to be taken from
                            Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP1(CYard, j, k)
                        Next k
                    End If
                        Next j
                End If
                ActiveCell.Offset(0, 1).Select
            Next i

        Cells(TrainNumber + 2, 26 + 13).Select
            For i = 1 To NumYards
            If Selection.Interior.Color = 65535 Then
                x = ActiveCell.Value
            Else
                x = 0
            End If
                If x > 0 And x <= NumYards Then
                        For j = 1 To NumYards
                    If x = Blocks(CYard, j) Then
                        For k = 1 To TXC1(TrainNumber) ' the time period for the cars to be taken from
                            Trains(TrainNumber) = Trains(TrainNumber) + ODArrayP2(CYard, j, k)
                        Next k
                    End If
                        Next j
                End If
                ActiveCell.Offset(0, 1).Select
            Next i

    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    If Trains(TrainNumber) < MinSize Then
```

```
        Smalltrain = 1
        GoTo TrainIsTooSmall
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    TrainSwitchTime(TrainNumber) = Trains(TrainNumber) * SwitchingTime(DYard) +
SwitchingPrepTime(DYard)

    Cells(TrainNumber + 2, 26 + 24) = TrainSwitchTime(TrainNumber)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    If T2(TrainNumber) + TrainSwitchTime(TrainNumber) > #11:59:00 PM# Then TX2(TrainNumber) = 7
    If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #11:59:00 PM# Then TX2(TrainNumber) = 6
    If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #8:00:00 PM# Then TX2(TrainNumber) = 5
    If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #4:00:00 PM# Then TX2(TrainNumber) = 4
    If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #12:00:00 PM# Then TX2(TrainNumber) = 3
    If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #8:00:00 AM# Then TX2(TrainNumber) = 2
    If T2(TrainNumber) + TrainSwitchTime(TrainNumber) <= #4:00:00 AM# Then TX2(TrainNumber) = 1

        Trains(TrainNumber) = 0 ' Reset Trains(TrainNumber)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   Error messages for trains which are too small or large
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
TrainIsTooSmall:
    If Smalltrain = 1 Then
        MsgBox ("Train(" & TrainNumber & ") does not have enough cars, and thus it will not" & _
            " run. This train only has " & Trains(TrainNumber) & " cars.")
        Smalltrain = 0
        Trains(TrainNumber) = 0 ' Reset Trains(TrainNumber)
        Exit Sub
    End If

Application.ScreenUpdating = True
End Sub
```