PHYSICAL INTERNET, CONVENTIONAL, AND HYBRID LOGISTIC SYSTEMS: AN OPTIMIZATION BASED COMPARISON

by

Mehran Fazili

Submitted in partial fulfilment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
April 2014

به نام آن که جان را فکرت آموخت
چراغ دل به نور جان برافروخت

ز فضلش هر دو عالم گشت روشن
ز فیضش خاک آدم گشت گلشن

گلشن راز، شیخ محمود شبستری

This thesis is dedicated to

my father and mother, Morteza Fazili, and Simin Mirkhani

for their constant support and unconditional love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The purpose of this thesis is to compare the performance of the Conventional (CO), Physical Internet (PI), and Hybrid (HY) logistics systems in a road network in order to understand and quantify the advantages and disadvantages of PI.

The comparison presented in this work is carried out through Monte-Carlo simulation in which loads are generated randomly and a sequential three phase optimization framework to optimize the CO, PI, and HY logistical networks.

By applying the methodology to the CO, PI, and HY logistic systems for the example road network, differences are observed in the total number of the instances containers requiring loading and unloading and total driving time to carry all the loads to their final destination.

Finally, the total cost of each logistic policy is calculated and the most economical logistical policy is identified for different scenarios.

LIST OF ABBREVIATIONS USED

ILP	Integer Linear Programming
TTD	Total Travel Distance
ADR	Average Duration of Routes
TDC	Total Delivery Cost
CO	Conventional Logistic Policy or System
PI	Physical Internet Logistic Policy or System
HY	Hybrid Logistic Policy or System
LTNL	Low Traffic Normal Loads Network
MTNL	Moderate Traffic Normal Loads Network
HTNL	High Traffic Normal Loads Network
HTLL	High Traffic Large Loads Network
HTSL	High Traffic Small Loads Network
RD	Random Traffic Network
k	Thousand
m	Million

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Dr. Uday Venkatadri for giving me vision, continuous encouragement, and expert guidance throughout this research. This thesis would not have been possible without his invaluable comments and advice in all stages of the work.

I would like to extend my sincerest thanks and appreciation to Dr. M.Mahdi Tajbakhsh, for co-supervising and funding this project. His support and feedback was extremely valuable to enrich this work.

A special thanks goes to Dr. Pemberton Cyrus, for his guidance, valuable comments, and great collaboration during this research.

Thanks are due to Dr. Ahsan Habib for serving on my defense committee, and to Dr. Alireza Ghasemi for moderating my defense session.

Special appreciation is due to my dear brother, Yashar Fazili for providing expert recommendations on selecting the right tools I needed for computation. Special thanks also goes to my dear sister Mehrnaz Fazili for her continuous kindness and support.

I do not know how to thank Nazanin Hooshmandi, my soul mate and companion who always believed in me and encouraged me. She made this chapter of my life a very fine and unforgettable one.

I want to also thank my dear friend Sina Raeisi, without whom my graduate studies at Dalhousie University would not have been even half as fun.

Last but not least, I would like to thank all the staff and graduate students of department of Industrial Engineering at Dalhousie University and wish them all the best.

Chapter 1: Introduction

Physical Internet (PI) is the way physical objects are moved, stored, realized, supplied and used, aiming towards greater efficiency and sustainability [1]. PI has the potential of introducing ground breaking improvements notable to fields of material handling, logistics, and facility design. The drive for PI is from the claim that "*the way physical objects are moved handled, stored, realized, supplied, and used throughout the world is not sustainable economically, environmentally, and socially*" [2].

The term "*Physical Internet*" was first used on the front page of The Economist magazine, in June 2006 [3]. PI refers to transportation of physical goods using data transmission analogy to Internet. The vision for PI is on thirteen characteristics that build the PI logistic model [1]. Most of these characteristics are inspired from data transmission protocols that are shaping Internet.

In PI, goods are encapsulated in globally standard, smart, green, and modular containers. The PI-containers are different from the conventional containers currently used in conventional logistic system. The concept of encapsulation of goods into PI-containers is similar to transmission of data by means of data packets in data networks.

Current logistic systems have number of unsustainability symptoms, such as: shipping large amounts of air and packaging, large amount of empty travel or dead-heads, slow response to incidental products demand, poor utilization of logistic resources, and etc [1]. It is believed that PI can alleviate some of these unsustainable logistics practices.

1.1 Thesis Objectives

The purpose of this thesis is to compare the performance of Conventional (CO), PI, and a Hybrid (HY) logistic systems within a road network using an optimization framework. As we will see in the literature review in the next chapter, there is a need to build an optimization framework to support research in PI and help understand its advantages and limitations.

1.2 Thesis Organization

The remainder of this thesis is organized as follows: A literature review relevant to the theme of the research is presented in Chapter 2. This includes an introduction to PI and an overview of models in vehicle routing that are pertinent to this research. In Chapter 3, the structure of the logistics network and routing systems for the three systems are being compared (PI, CO, and HY). In Chapter 4, a three-phased optimization framework is presented to enable the performance comparison. The load generation Monte-Carlo simulation is discussed in Chapter 5. The numerical results of the thesis are presented in Chapter 6. In Chapter 7 the main contributions of this thesis are summarized, and areas for future research are suggested.

Chapter 2: Literature Review

The literature review in this thesis first focuses on literature related to the Physical Internet. Since field of Physical Internet is fairly recent, the number of published papers in the domain are relatively low. The second part of this literature review focuses on literature of the Vehicle Routing Problem (VRP).

2.1 Physical Internet (PI)

The PI project (or initiative) has an official and dedicated website [4] accessible at "*http://www.physicalinternetinitiative.org*". This website contains a complete list of publications in this field. Active projects, events, and news of Physical Internet are also reported in this website. Figure 1, shows a brief overview of PI on a timeline.

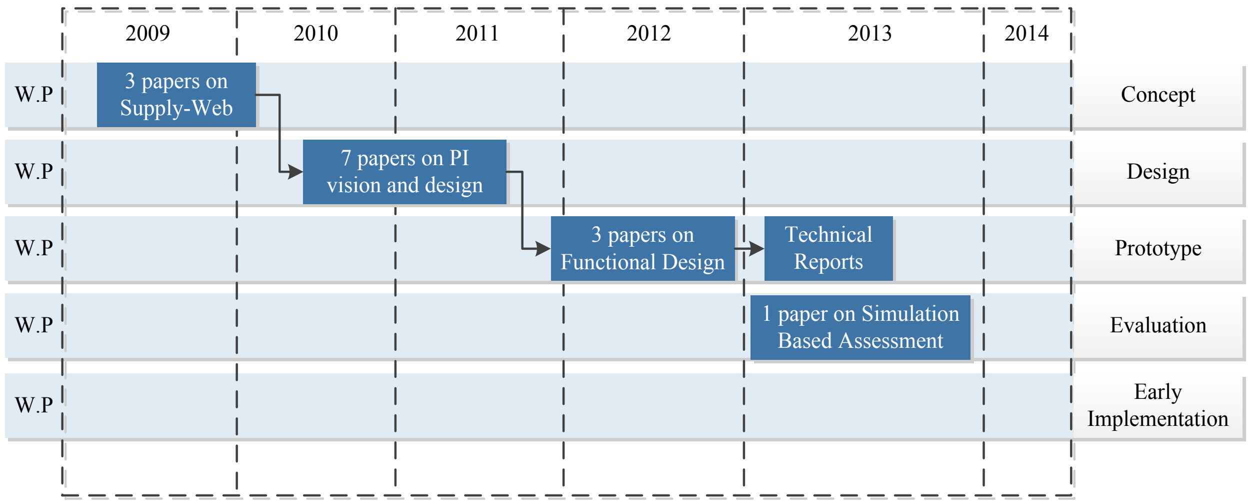

Figure 1 - PI on Time line

The Supply Web is described in [5], as a network of interconnected supply chain networks. In the traditional system, supply chain networks of various organizations do not share much information or infrastructure with each other. In PI, much of the logistic infrastructure such as PI hubs or PI transit centers are open for use to every agent who is willing to sign up for the service.

The concept of an integrated supply web, which brings together disparate supply chain networks, is one of the most important building blocks of the PI.

An overview of the physical aspects of PI is provided in [6]. The physical elements of the PI are listed as: *containers*, *movers*, and *nodes*. The containers used in PI are named PI-containers and a number of standard sizes are proposed for these containers. Movers are grouped into PI-transporters, PI-conveyors, and PI-handlers. For the nodes, PI-hubs and PI-transit centers are introduced. This forms the backbone of PI. PI logistics system can be described as follows: when a shipment is made from A to B, the PI container used to make the shipment may be routed through one or more PI-hubs or PI-transit centers where they are handled by PI-transporters, PI-conveyors, and PI-handlers. The PI-containers themselves are modular and may fit within or alongside other PI-containers. Since the PI logistics system is shared, the expected advantage of such a system comes from sharing and pooling resources.

In [7], a supply chain visualization software called the Supply Web Mapper is introduced. This software enables parties and organizations using the supply web to get live information on status of the Supply Web in several multi-dimensional synthetic forms and diagrams. Also, a generic objective of the Supply Web Mapper is to obtain a representation of the existing relationship between several selected actors, resources and products within the Supply Web. This prototype software consists of three main interfaces "*Conceptual Map Viewer*", "*Geographical Map Viewer*", and "*Data Mining viewer*" to achieve this goal.

In [8], an agent based simulation platform to simulate the complex scenarios that could occur on supply web is introduced. The motive behind creation of such simulation environment was to develop an inexpensive, yet effective method of assessing dynamics of supply chain behaviour of an organization with shared resources.

A global sustainability grand challenge issue is presented in [1] with statistics showing the weak performance of current global logistics systems. This paper also lists a number of global unsustainability symptoms and the PI vision is introduced through 13 points. Encapsulating merchandise in standard PI-container sizes, and moving from point to point transportation to distributed multi segment transportation are among the 13 points related to this thesis.

The first study on the impact of network topology on the performance of PI is shown in [9]. In this research, which does not involve an optimization framework, the performance of the conventional logistic vs. PI enabled networks are described using parameters for transportation throughput requirements, flow travel, and total cost.

The design and development phases of a simulation tool for a future comprehensive study of the PI are explained in [10]. In this research, the mechanism for capturing and quantifying the impact of PI in terms of economic, environmental, and social efficiency is shown.

Finally, proposals for functional design of the PI facilities for road based transit centers, road-rail hubs, and road based cross-docking hub are presented in [11], [12], and [13]. Each of these studies show the facility components, introduce a set of key performance indicators, and graphical simulation models built to evaluate operations. At the end of these studies, performance results under various conditions are reported.

It has been hypothesized that an interconnected logistics network is better than a fragmented logistics network. A simulation platform is developed in [14] to evaluate the performance of an interconnected logistic network. The goal of the platform is to evaluate the efficiency of an interconnected logistic network in terms of delivery times, carbon emission levels, and travel times. To carry out the demonstration, data obtained from corporations for fast moving items in the consumer goods sector in France is used. The performance measures from the simulation for the PI logistics system are compared against standard industry KPI's. The conclusion in this paper is that PI does not compromise operational efficiency of the logistics system while significantly reducing the carbon foot print and logistics costs.

In summary, the PI literature consists of two parts: the development of the conceptual PI framework and the development of infrastructural concepts for the realization for PI. The predominant methodology is to build simulation environments. For example, the simulation environments in [8], [11], [12], and [13] were designed to show the expected performance of various logistic activities within a PI facility. The simulation platform shown in [14] has a slightly broader mandate and looks at the performance of a broader interconnected logistic network (of which PI is a good example).

The PI literature is in a very nascent stage. There are several opportunities within PI for cross-discipline research which includes research in several areas of OR such as supply chain management, logistics and operations planning, and transportation. Other potential areas of research include mechanical design, social sciences, information technology, etc.

From the literature, it appears that the PI logistics system has both advantages and disadvantages. The advantage appears to be that through consolidation and deconsolidation at PI hubs, transportation is more efficient. This is because transport occurs through modular containers and PI hubs offer better opportunities to mix and match loads for easier routing. Moreover, the PI hubs themselves can be automated for efficiency in handling. From a social point of view, drivers are generally not required to travel long distances from their home base. On the other hand, the disadvantage of the PI logistics system seems to be that a greater level of material handling is required inside the PI hubs, which adds to both cost and lead time.

It is believed that more research is needed to understand the above trade-offs. There is no optimization framework in the literature that can help decision makers quantify the advantages and disadvantages of PI over the traditional logistics system. This gap provides the motivation for this thesis.

2.2 Vehicle Routing Problems

Since the objective of this thesis is to compare PI, CO, and HY, it is important to understand how logistics networks are optimized. The *Vehicle Routing Problem* (VRP) and its variants is the basic building block to optimize transportation in a logistics network.

A VRP is defined as a set of routes or sequences that start from one or several locations, and visit a number of geographically scattered points to minimize total travel distance or time [15]. There are several classifications of the VRP and comprehensive literature reviews on the subject. For example [16] provides a classification of VRP according to the number of pick-up and delivery locations.

An overall division in literature is between the *exact* algorithmic approaches, and *approximate* or *meta-heuristic* approaches to solve variations of the VRP. The literature on both is comprehensive. Each division again has a number of subdivisions.

The exact algorithms are classified in three broad categories of: (i) direct tree search methods, (ii) dynamic programming, and (iii) integer linear programming [17]. The *meta-heuristic* approaches are classified according to the heuristic algorithms used to solve the VRP. Among the most popular approximation approaches, the solution methods built using *Tabu Search* and *Simulated Annealing* have received extensive attention from several researchers.

As linear integer programming is the methodology applied in this thesis, the literature in integer programming is the focus of this literature review.

VRP is a specific form of a more general problem set called *Pick-up and Delivery Problems* (PDP). For PDP two problem classes can be distinguished. These classes are: (i) *Vehicle Routing Problems with Backhaul* (VRPB), and (ii) *Pick-up and Delivery Vehicle Routing Problem* (PDVRP) [18]. The first class deals with transportation of goods from number of depots or central locations to a set of linehaul customers, and from backhaul customers to the depots. The second class refers to scenarios where goods are picked up at customer's location and directly delivered to other customers. Figure 2, illustrates an overall presentation of these two problem classes and their subclasses.

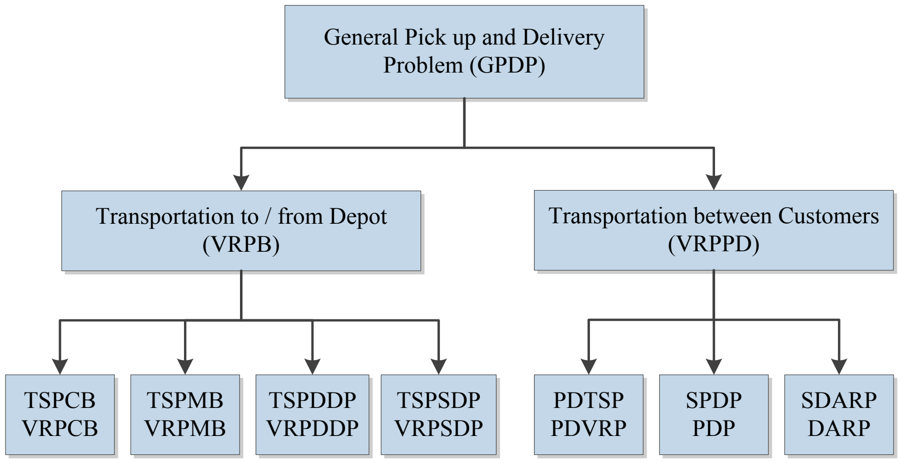
General Pick up and Delivery Problem (GPDP)
Transportation to / from Depot (VRPB)
Transportation between Customers (VRPPD)
TSPCB VRPCB
TSPMB VRPMB
TSPDDP VRPDDP
TSPSDP VRPSDP
PDTSP PDVRP
SPDP PDP
SDARP DARP

Figure 2 - Pick-up and Delivery Problems [18]

VRPB can be divided into four subclasses. Each of the four subclasses in the VRPB are shown with a prefix – VRP or TSP. When the notation starts with "*TSP*", the problem only involves one vehicle (TSP refers to the travelling salesman problem). When a subclass has the "*VRP*" prefix, there are several vehicles. In *Vehicle Routing Problem with Clustered Backhaul* (VRPCB) and *Vehicle Routing Problem with Mixed Backhaul* (VRPMB), customers are either pick-up customers or delivery customers, but cannot be both. In VRPCB problems, the requirement is to visit all the delivery customers or delivery cluster before the first customer in the pick-up group is served. However, in the mixed backhaul case (VRPMB), drop-off and pickup customers could be mixed.

In the last two subclasses, *Vehicle Routing Problem with Divisible Delivery and Pickup* (VRPDDP), and *Vehicle Routing Problem with Simultaneous Delivery and Pickup* (VRPSDP), customers may need both delivery and pick up. In the VRPDDP each customer can be visited twice, one for delivery, and one for pick up. However, in the VRPSDP, a customer can only be visited once.

A graphical illustration of these subclasses is presented in Figure 3.

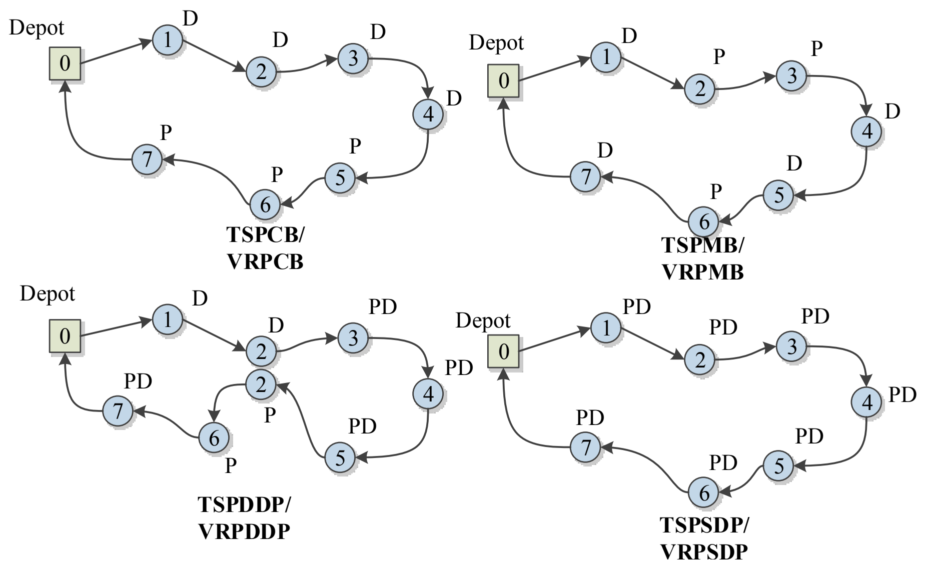

Figure 3 - VRPB Sub-Classes

2.2.1 Formulation of VRPB Problems

The mathematical formulation of the VRPB problems are introduced and explained in this section. First, using a consistent notation, a general form of single vehicle and multi vehicle formulation will be introduced and explained. Second, the modification required to show any of the four subclasses of the problem will be introduced. The notations and formulations used in this section are adopted from [18].

Sets:

$P = \{1, \dots, n\}$	Set of Pick up Customers
$D = \{n+1, \dots n+\tilde{n}\}$	Set of Delivery Customers
K	Set of vehicles

Parameters:

n	Number of pick up nodes.
$\tilde{n}$	Number of delivery nodes.
q_i	Demand or supply quantity at node *i*. Supply quantity is shown by positive value and demand quantity with negative value. Demand and supply at the start or the end deport is set to 0
e_i	Earliest time to begin service at node *i*
l_i	Latest time to begin service at node *i*
d_i	Service duration at node *i*
c_{ij}^k	Cost of traveling from node *i* to node *j* by vehicle *k*
r_{ij}^k	Travel time from node *i* to node *j* by vehicle *k*
C^k	Capacity of vehicle *k*
T^k	Maximum route duration of vehicle / route *k*

Variables:

x_{ij}^k	Binary, 1 if arc (i, j) is used by vehicle *k*, 0 otherwise
Q_i^k	Load of vehicle *k* when leaving node *i*
B_i^k	Beginning of service of vehicle *k* at node *i*

Note that in a single vehicle formulation of the problems, the subscript *k* can be omitted.

The single vehicle formulation of the pick-up and delivery problems is based on Traveling Salesman (TSP) problem and is formulated as follows:

$$Minimize \sum_{(i,j)\in A} c_{ij} \cdot x_{ij} \qquad \text{(L1)}$$

Subject to:

$$\sum_{i:(i,j)\in A} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\} \qquad \text{(L2)}$$

$$\sum_{j:(i,j)\in A} x_{ij} = 1 \qquad \forall i \in V \setminus \{n + \tilde{n} + 1\} \qquad \text{(L3)}$$

$$x_{ij} \in 0,1 \qquad \forall (i,j) \in A \qquad \text{(L4)}$$

$$\sum_{(i,j)\in A(S,\tilde{S})} x_{ij} \geq 1 \qquad \forall S \subseteq V \setminus \{n + \tilde{n} + 1\},\ S \neq 0 \qquad \text{(L5)}$$

$$x_{ij} = 1 \Rightarrow B_j \geq B_i + d_i + t_{ij} \qquad \forall (i,j) \in A \qquad \text{(L6)}$$

$$s_i - s_j + 1 \leq M \cdot (1 - x_{ij}) \qquad \forall\, n \subseteq V \setminus \{n + \tilde{n} + 1\} \qquad \text{(L7)}$$

In the single vehicle formulation of the VRPB, the objective function L1 minimizes the total routing cost. L2 and L3 enforce that each node is visited exactly once. Constraints L5, L6, and L7 are the three optional methods used for sub tour elimination. Therefore presence of only one of them in the formulation is sufficient. In constraint L5 at least one route has to leave every non-empty subset S ⊆ V \ $\{n + \tilde{n} + 1\}$. Constraint L6 is used as a time window constraint to eliminate the sub tours. Another option for sub tour elimination is constraint L7 (Miller, Tucker, and Zemlin (MTZ) constraint [19]).

The following is the general formulation of the multi-vehicle VRP adopted from [20]:

$$Minimize \sum_{k\in K} \sum_{(i,j)\in A} c_{ij}^k \cdot x_{ij}^k \qquad \text{(L8)}$$

Subject to:

$$\sum_{k\in K} \sum_{i:(i,j)\in A} x_{ij}^k = 1 \qquad \forall i \in P \cup D \qquad \text{(L9)}$$

$$\sum_{j:(0,j)\in A} x_{0j}^k = 1 \qquad \forall k \in K \qquad \text{(L10)}$$

$$\sum_{i:(i,n+\tilde{n}+1)\in A} x_{i,n+\tilde{n}+1}^k = 1 \qquad \forall k \in K \qquad \text{(L11)}$$

$$\sum_{i:(i,j)\in A} x_{ij}^k - \sum_{j:(i,j)\in A} x_{ji}^k = 0 \qquad \forall S \subseteq V \setminus \{n+\tilde{n}+1\}, S \neq 0 \qquad \text{(L12)}$$

$$x_{ij} = 1 \Rightarrow B_j^k \geq B_i^k + d_i + t_{ij}^k \qquad \forall (i,j) \in A,\ k \in K \qquad \text{(L13)}$$

$$s_i - s_j + 1 \leq M \cdot (1 - x_{ij}) \qquad \forall n \subseteq V \setminus \{n+\tilde{n}+1\} \qquad \text{(L14)}$$

$$x_{ij} = 1 \Rightarrow Q_j^k = Q_i^k + q_j \qquad \forall (i,j) \in A,\ k \in K \qquad \text{(L15)}$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{C^k, C^k + q_i\} \qquad \forall i \in V,\ k \in K \qquad \text{(L16)}$$

$$x_{ij} \in 0,1 \qquad \forall (i,j) \in A \qquad \text{(L17)}$$

The objective function L8 minimizes the routing cost over all the vehicles. Constraint L9 enforces that each node is visited exactly once. Constraints L10, and L11 ensure that all vehicles begin their routes from the depot and return to the depot at the end of their routes. Constraint L12 is for vehicle flow conservation. Constraint L18 is the time window constraint, used for sub tour elimination. As the single vehicle case, the MTZ constraint in L14 can be used instead of L13 for sub tour elimination. Constraints L15 and L16 ensures that vehicle capacity is not exceeded at any time in the route.

2.2.1.1 TSPCB and VRPCB Extensions

One can formulate this TSPCB as a double-cluster problem: one for a pickup and one for a delivery customer. An additional condition is needed to enforce that no pickup customer is visited until all delivery customers are visited. This can be achieved by the following constraint:

$$\sum_{i}\sum_{j} x_{ij} = 1 \qquad \forall i \in P, \forall j \in D \qquad \text{(L18)}$$

Constraint L18 guides the model to use only one arc connecting pick-up and delivery customers. For this to work, the cost of an arc from j to i is assigned to a high value so that no arc from a delivery customer to a pickup customer is chosen.

A similar approach can be used in the case of VRPCB. Since in the VRPCB there are multiple number of vehicles to be routed, constraint L18 is modified to constraint L19 as shown below:

$$\sum_{i}\sum_{j} x_{ij}^{k} = 1 \qquad \forall i \in P, \forall j \in D\ \forall k \in K \qquad \text{(L19)}$$

A number of papers deal with these problem subclasses. For example, [21] and [22] use exact algorithms, while [23] uses a Tabu Search based heuristic to solve VRPCB problems.

2.2.1.2 TSPMB and VRPMB Extensions

For TSPMB the order of delivery and pick up customers is only restricted by the vehicle capacity. So to formulate the TSPMB problem, the following constraints are added to the general formulation.

$$\mathrm{Q}_0 = -\sum_{i \in D} q_i \qquad \text{(L20)}$$

$$x_{ij} = 1 \implies Q_j \geq Q_i + q_j \qquad \forall (i, j) \in P, \forall j \in D \qquad \text{(L21)}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{C, C + q_i\} \qquad \forall i \in V \qquad \text{(L22)}$$

Constraint L20 ensures that vehicle leaves the depot with load equal to the total amount of deliveries. Constraint L21, and L22 ensure that vehicle capacity is not exceeded at any node through the route.

A similar concept is used in the VRPMP case. In this form of problem, one has to ensure that every vehicle leaves the depot with no more than total amount of deliveries to nodes enroute. This is achieved by L23.

$$Q_0^k = -\sum_{j \in D} q_i \sum_{i \in V} x_{ij}^k \qquad \forall k \in K \qquad \text{(L23)}$$

There are papers in the literature that deal with the VRPMB. For the exact case, an example is [24]. A heuristic approach to deal with the VRPMB is shown in [25].

2.2.1.3 TSPDDP and VRPDDP Extensions

The TSPDDP, and VRPDDP can use the same formulations introduced for TSPMB and VRPMB. The only difference is that each customer is visited twice, once for delivery and once for pick up. To enforce this in the model, each customer node can be entered twice in the model. This subclass of the VRP problem is also known as the "*Lasso*" or "*Double Path*" VRP, in literature.

2.2.1.4 TSPSDP and VRPSDP Extensions

In this subclass, each customer can be only visited once. Each customer requires both delivery and pick at the same time. Therefore, at each node i, the difference between pick-up and delivery amount can be shown as: $q_i = (q_i^+ - q_i^-)$. q_i^+ denotes the pickup amount and q_i^- is the delivery amount at node *i*. If the amount of delivery to node *i* is larger than the amount to be picked up, q_i becomes a negative number.

In addition, L24 and L25, are used for TSPSDP and VRPSDP cases respectively, to ensure that the load that vehicle(s) carry when leaving a depot is equal to the total amount to be delivered to all customers in the route.

For the TSPSDP case:

$$\mathrm{Q}_0 = \sum_{i \in D} q_i^- \qquad \text{(L24)}$$

For the VRPSDP case:

$$Q_0^k = \sum_{j \in p} q_j^- \sum_{i \in V} x_{ij}^k \qquad \forall k \in K \qquad \text{(L25)}$$

In addition to the extension introduced for the subclasses above, time window constraints and maximum route length constraints may be added to the general form of the TSP or VRP model to achieve custom requirements. For example, one can use the maximum route length constraint to enforce a cap on the allowed driving time of a driver. VRPSDP is also known as the

"*Hamiltonian*" VRP in literature. An exact algorithm for VRPSDP is introduced in [26]. For Tabu Search heuristic approaches, the reader may refer to [27] and [28].

2.2.2 Transportation between Customers

In VRPDP problems, a set of vehicles have to satisfy a set of transportation request. Each transportation request specifies a load size and has a distinctive origin, where it has to be picked up, and destination, where it has to be delivered [29]. VRPDP has three categories shown in Figure 2. Similar to the classification presented in the previous section, subclasses of VRPDP can be shown with extension to a general pick-up and delivery model. As these classification are not distinctively different from VRPB problems, their mathematical formulations s are not shown here. A reader may refer to general PDP formulation available in the literature. It should be noted that Dial a Ride Problem (DARP) is different from Pickup and Delivery Vehicle Routing Problem (PDVRP) and Pickup and Delivery Problem (PDP), as in DARP the loads are passengers and therefore consideration is given to level of convenience in travel. Taxi, or any charter service provider companies use variations of DARP models. An exact algorithm of DARP is presented in [30]. For a comprehensive review of PDP models a reader is encouraged to refer to [29].

To summarize, the VRP is a very mature research area. There are several optimization algorithms and heuristic approaches to solve this broad area of problems. Since the objective of this thesis is to compare the PI, CO, and HY logistics systems, there are plenty of tools and techniques to model these systems within an optimization framework. The ideas discussed in this review will be used extensively in the remainder of this thesis.

Chapter 3: Network Structure and Logistic Systems

In this section, the road network used for this thesis will be introduced since it is important to study the road network topology and its physical characteristics. In order to understand the PI, CO, and HY logistics systems, we describe a simplified road network in Eastern Canada in section 3.1. This road network will also be used in our computational case studies. The three logistic systems are discussed in detail in sections 3.2-3.4. For a better understanding of these logistic polices, routes created with each of these logistic polices for two example container flows will be illustrated.

3.1 Eastern Canada Road Network

This thesis focuses on transportation within a simplified Eastern Canadian road network. As shown in Figure 4, 11 cities in Eastern Canada are the nodes in the network. Being a large urban area, and for better simulation of traffic flows, Montreal is represented by two nodes: Montreal East and Montreal West. Figure 5 shows only the logical connection of the nodes in the network and is not to scale. The network shown in Figure 5 is a tier 4 *Intra-Country Inter-State Network* according to the PI network classification in [1].

The Canada West node represents the rest of the Canadian road network to West. Similarly, Ontario North, and Ontario South nodes represent the northern and southern Ontario road networks as one single node. The traffic going to and from Canada West, Ontario North, and Ontario South will pass through these three nodes. In the same fashion, the traffic to and from United States will pass from three nodes called US Border Gate 1, 2, and 3.

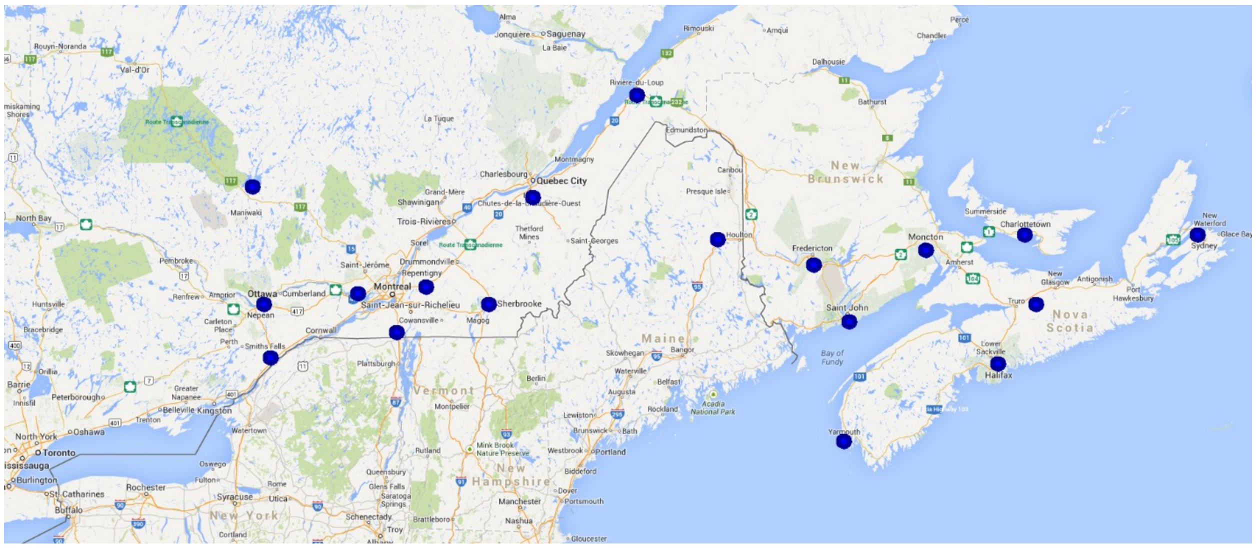

Figure 4 - Candidate Cities in Road Network

(Dots placed after retrieving map from maps.Google.com – retrieved on September 20, 2013)

The network shown below, has a tree structure, with the exception to the connection between Fredericton, Moncton, and Saint John. As will be shown, logistics decision making can get very complicated even on tree networks, especially when there are a large number of loads to be transported. Without the proper tools and routing optimization algorithms, drawing a clear conclusion on the true performance of the logistics systems will not be possible.

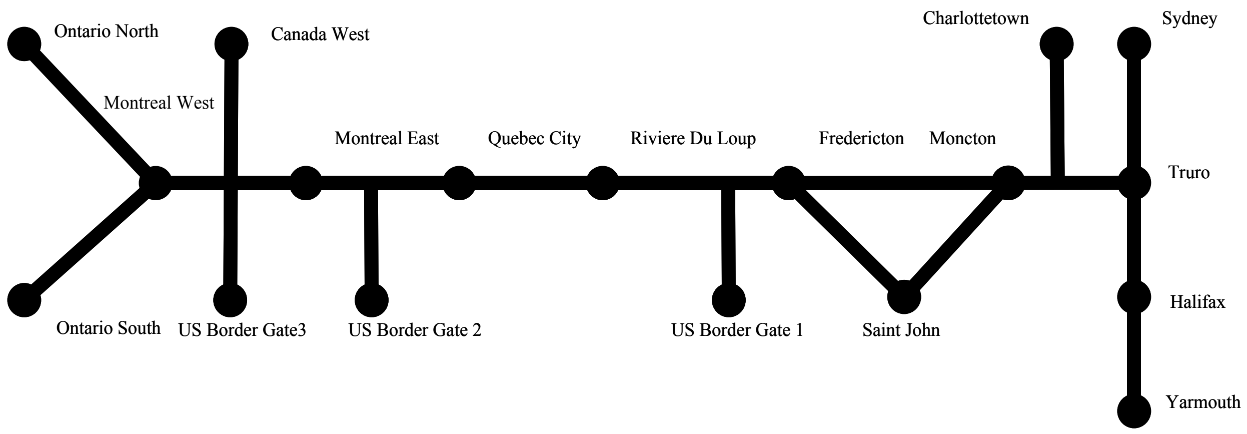

Figure 5 - Logical Connection of the Nodes

Since there are 7 network tiers in the PI logistic system [1], logical connections between the nodes in the network is sufficient to define the tier 4 Intra-country inter-state network in Figure 5. However, to study other network tiers, such as a tier 3 *Intra-City Inter Facility Network,* factors such as urban congestion, routes capacity, and routes availability should be considered in the network definition. Such issues are beyond the scope of this thesis.

3.2 Conventional (Door-to-Door) Logistics Policy

Loads at each source node with identical final destinations are packed into a container in the Door-to-Door (DTD) logistics system. This container is then transported to the final destination without any en-route loading and unloading. In other words, transshipment is not allowed in Conventional logistics (CO). As an example, a container carrying loads from Yarmouth to US Border Gate 2 will be loaded in Yarmouth and only unloaded when it arrives at its final destination. Similarly a container from Sydney to US Border Gate 2 will have a non-stop trip to this node. These two container flows are illustrated in Figure 6, with orange and green arrows respectively.

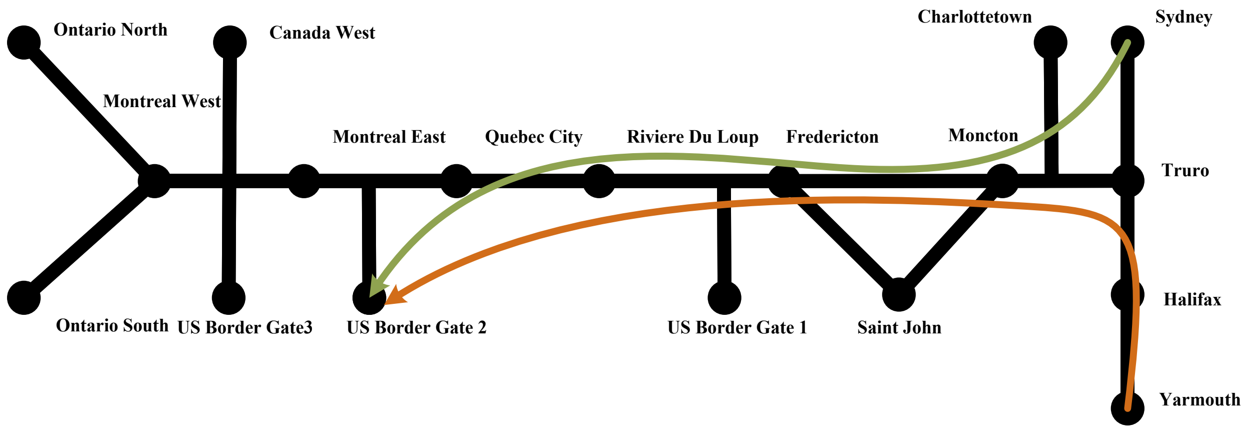

Figure 6 - Example Flows in CO Logistic Policy

Appendix A: Conventional Arcs Dictionary shows, in dictionary form, the list of nodes from any source node to any destination node. Elements of this dictionary will be explained in detail in Chapter 5: Data Generation and .

3.3 PI Logistic Policy Flows

The PI Logistic system one the same road network uses five road based PI transit centers, as described in [11]. These are located in Truro, Fredericton, Quebec City, Montreal East, and Montreal West. Selection of these nodes to host the PI Transit Centers is because of population or due to their strategic location. For example, although Truro is a small town, it is located on a Trans-Canada highway split.

Each of these PI transit centers represents a cluster of nodes as shown in Figure 7. In this logistics system, at the source node, loads are packed into a container. A container then is transported to a PI node with a road based PI transit center. The loads are therefore transported from an origin to one or more PI transit centers (or PI hubs) before reaching their final destination through the PI hub of the destination cluster.

In the PI logistics system, loads with different sources but the same final destination could be packed into the same container at any of the en-route PI nodes. As an example, a container going from Yarmouth to US Border Gate 2 will be packed in Yarmouth, and unpacked in Truro. Since Truro is the PI hub of cluster 1, all the other containers arriving to Truro from the other nodes in cluster 1 (such as Sydney) will be also unpacked in Truro. Loads with the final destination beyond the neighbour PI hub (ex. Fredericton) will be packed into other containers and transported to Fredericton. The same process will repeat in Fredericton until loads arrive to Quebec City where they are packed into containers with the final destination of US Border Gate 2.

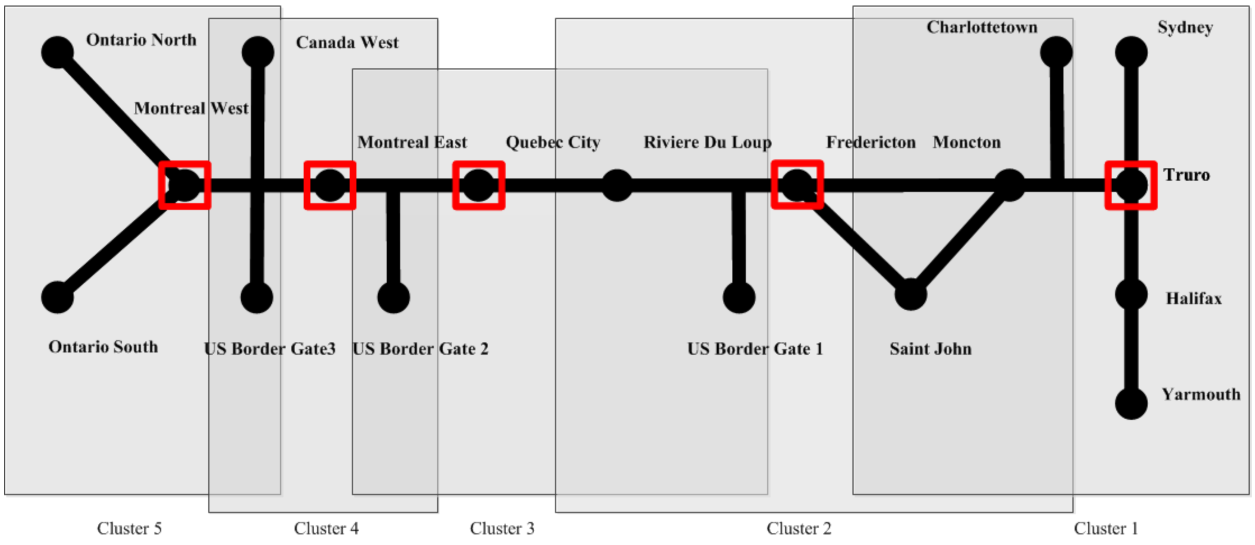

Figure 7 - PI Road Network

Figure 8 is a representation of how loads from Sydney and Yarmouth could be packed in the same container in Truro and travel together for the rest of their journey using PI logistic systems.

Appendix B: PI Arcs Dictionary shows a dictionary form of the list of nodes allowed for transshipment from any source node to any destination node when PI Logistics is used. Elements of this dictionary will be explained in detail in Chapter 5: Data Generation and Optimization.

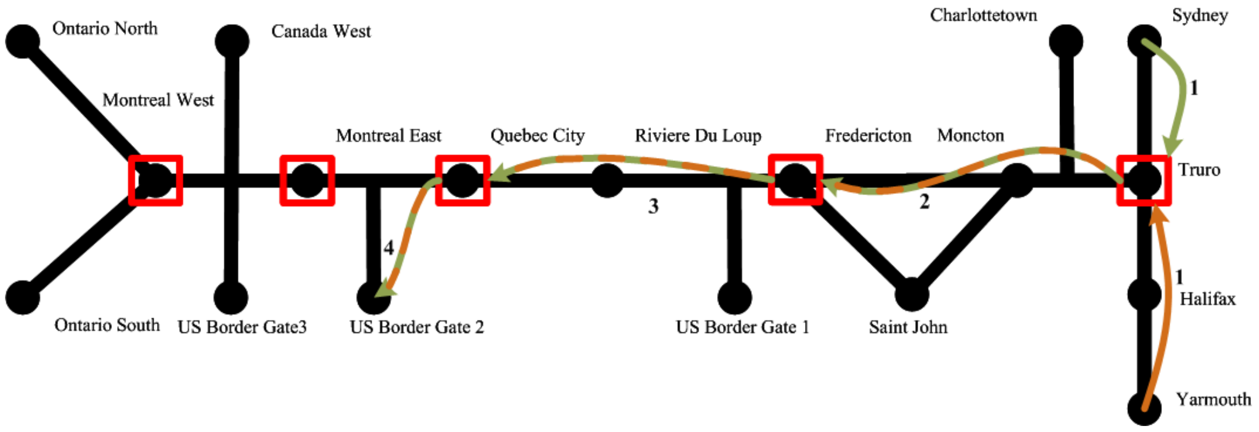

Figure 8 - Example Flows in PI Logistic Policy

Figure 9 shows a proposed layout for PI Hub. Trucks enter the facility from the gates at the bottom left and park in the designated load switching area in the center of the facility. When loaded, trucks leave the facility from the gates at the bottom right side of the facility.

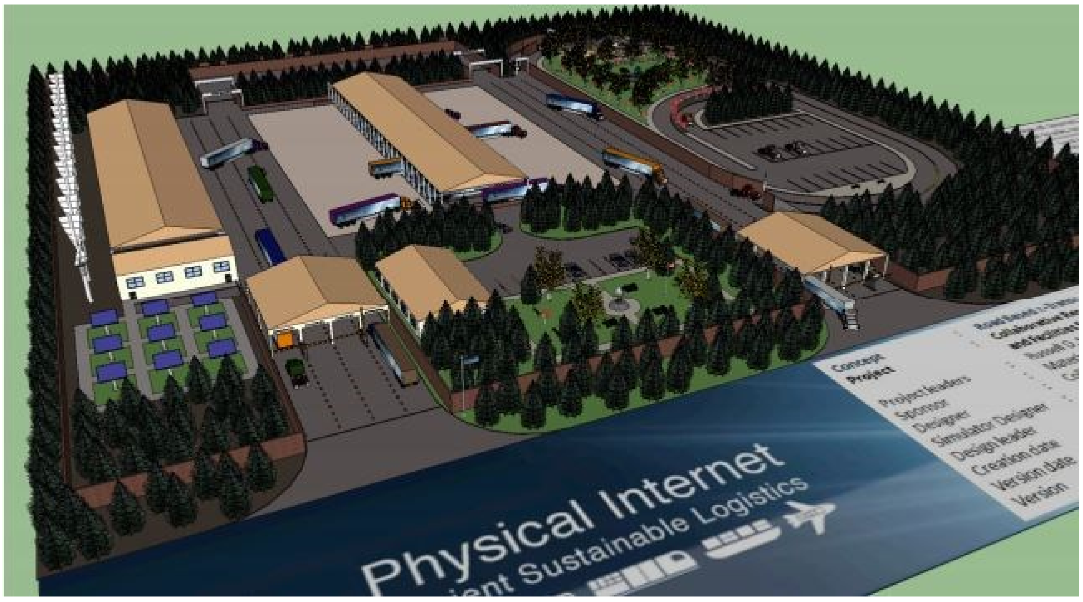

Figure 9 - Isometric View of the Proposed Layout for PI Hubs [31]

3.4 Hybrid Logistic Policy

The Hybrid Logistics policy is a combination of CO and PI. Packing and unpacking is done according to the PI policy inside the PI transit centers of source and destination clusters. In contrast, the containers transiting between PI hubs are not packed and unpacked at every intermediate PI hub in the network.

Therefore, the loads are packed into a container at a source node. The container is then transported to the source cluster's PI Hub. At the PI Hub, the request is packed with all the other requests going to the same final cluster destination. The container is then transported to the destination cluster's PI Hub, where the loads are unloaded and loaded with all other loads going to the same final destination node.

As an example, a load going from Yarmouth to US Border Gate 2 will be loaded into a container going from Yarmouth to Truro. In Truro, all the loads arriving from the other nodes in cluster 1 (ex.Sydney) are unloaded. Loads with the final destination node in Cluster 3 are then packed into container going to Quebec City. In Quebec City loads with final destination of US Border Gate 2 are packed into a container going to this node.

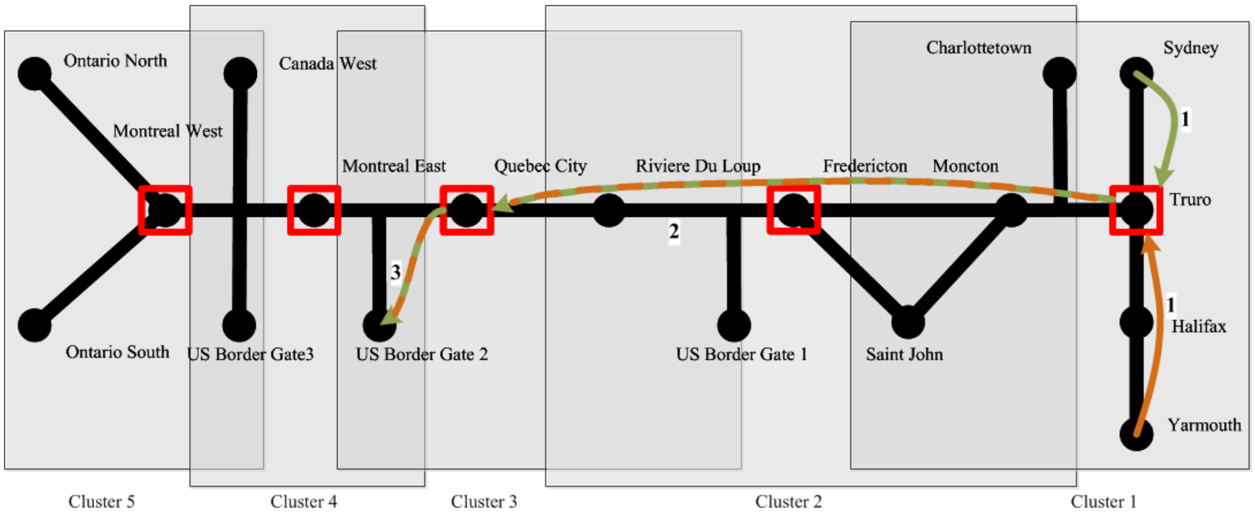

Figure 10 is a representation of how loads from Sydney and Yarmouth could be packed in the same container in Truro and travel together for the rest of their journey using the HY logistic policy.

Appendix C: Hybrid Arcs Dictionary, shows a dictionary form of a list of nodes allowed for transshipment from any source node to any destination node when HY Logistics is used. Elements of this dictionary will be explained in detail in Chapter 5: Data Generation and Optimization.

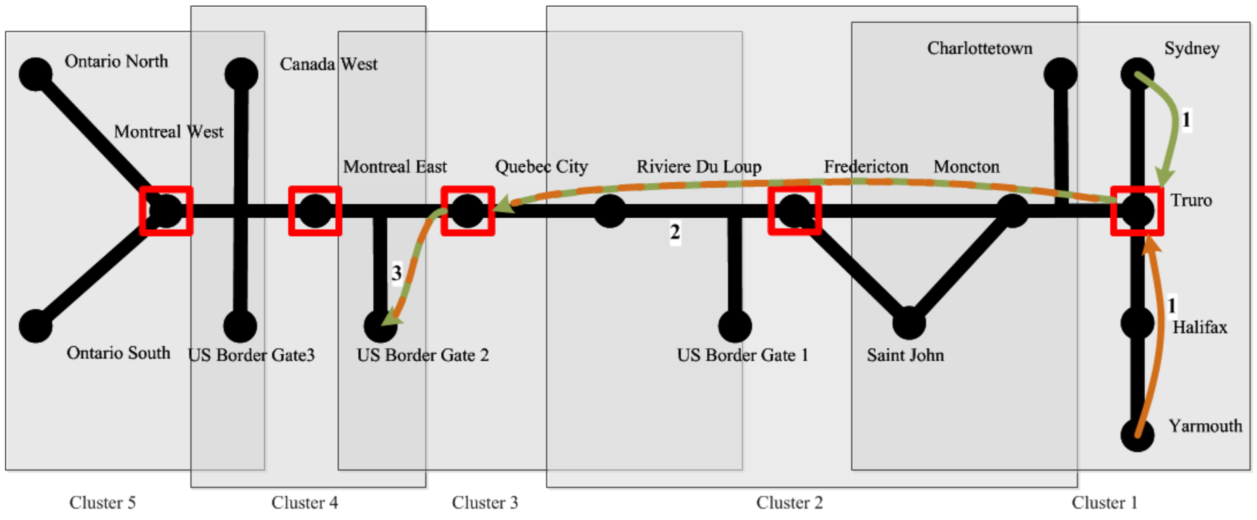

Figure 10 - Example Flows in HY Logistic Policy

3.5 Request Sizes and Physical Characteristics of Containers

The modularity of containers is a very important element of the PI vision [1], [6]. In [6] the standard modules used to encapsulate goods are called π containers. In this thesis, the term PI-container is switched with PI-box. Therefore, PI-boxes are the actual boxes that encapsulate merchandise. The term PI-containers will refer to the containers that encapsulate PI- boxes.

Conventional containers have only one door at the back of the container to load and unload. This implies that merchandise is unloaded in Last in First out (LIFO) fashion. The size of the containers varies from 20 feet to 53 feet depending on the mode of transportation and geographical region of use [32]. Figure 11 shows a 53-foot container widely used in North America.

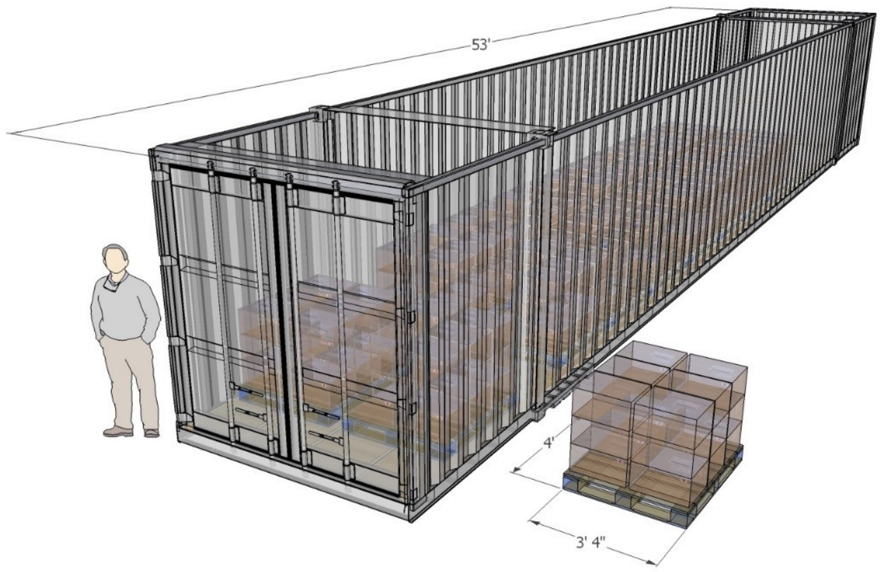

Figure 11 - Conventional Pallets Loaded in Container [33]

The load (or request) sizes introduced in this thesis are the sizes used for PI-boxes. PI-boxes are placed in a conceptual PI-containers. The conceptual PI-containers introduced in this thesis can be loaded and unloaded from the side. Loading and unloading a PI-container from side reduces the time spent to pack and unpack containers at PI Hubs. It should be noted that the mechanical design and proof of concept of PI- containers is not within the scope of this thesis. The standard size of a PI-container is assumed to be 40 feet.

Figure 12 and Figure 13 show PI-boxes and PI-containers. Figure 12 shows how a fully packed PI-container looks like with four 0.125 (5 feet) and two 0.25 (10 feet) PI-boxes.

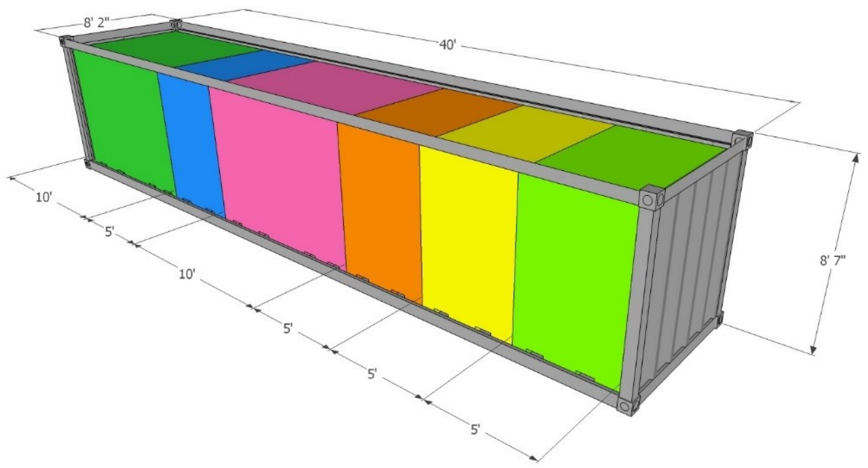

Figure 12 - Packed PI-Container [33]

For the sake of consistency in this thesis, all logistic systems use PI-containers for transportation of the loads. Using PI-containers to investigate performance of logistic systems ensures that factors such as difference in the shape of the loads, container packing and unpacking methods, and container space utilization, have no effect.

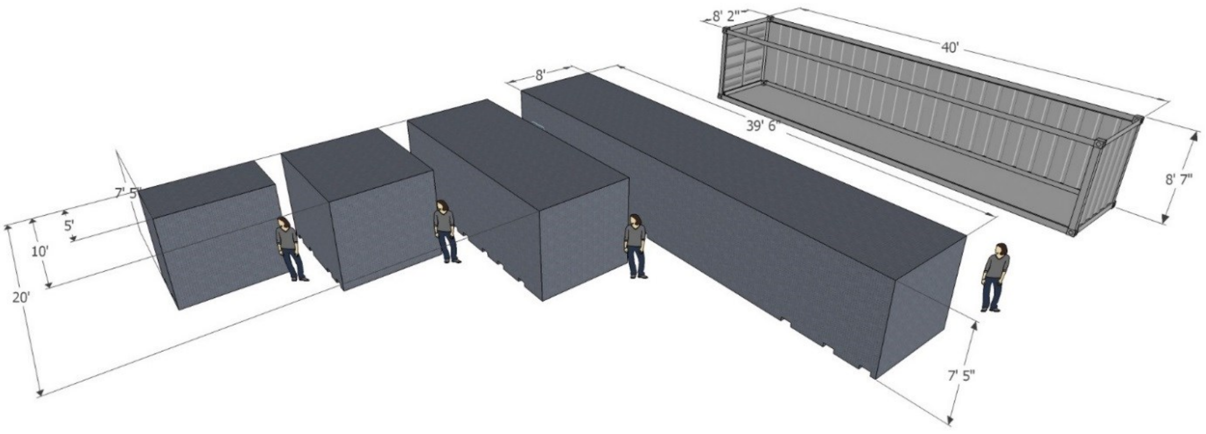

Figure 13 - PI-Boxes and a PI-Container

Chapter 4: Three-Phase Optimization Framework

In this chapter, we develop an optimization framework to compare and contrast the PI, CO, and HY logistics networks. Once a set of container transportation requests (with load sizes and origin-destination information) is received, a typical logistics operator needs to consolidate these requests and build routes with existing assets (trucks) to realize the shipments. We assume that the logistics operator of any of the three network types will use optimization to come up with an operations plan.

4.1 Optimization Methodology

The comparison presented in this work is carried out through Monte-Carlo simulation in which loads are generated randomly and a sequential three phase optimization framework optimizes CO, PI, and HY logistical networks.

The comparison begins with generating a list of loads using Monte-Carlo simulation to be shipped from each source to each destination. The operations plan for container movements is developed by solving optimization models for packing and consolidation, routing, and truck assignment. While the packing/consolidation strategy varies depending on the logistics system (PI, CO, HY), this part of the logistics operations plan deals with creating container shipments. A set of routes are then developed to transport containers through the network to satisfy the requests. Finally, trucks (drivers) are assigned to routes in order execute the operations plan and make sure that trucks (drivers) return to their home base. Figure 14, is a graphical illustration of the work done at each step in the optimization framework.

By applying the three phase optimization framework methodology to CO, PI, and HY logistic systems, differences are observed in the total number of the instances containers requiring loading and unloading and total driving time to carry all the loads to their final destination. The complete list of KPI used to make the comparisons is as follows:

1. Number of container packing and unpacking instances
2. Total hours of container routing
3. Number of trucks in service
4. Average hours worked per truck

5. Percentage drivers back home at the end of the day
6. Total system cost as function of operation, social, material handling, and a fix cost

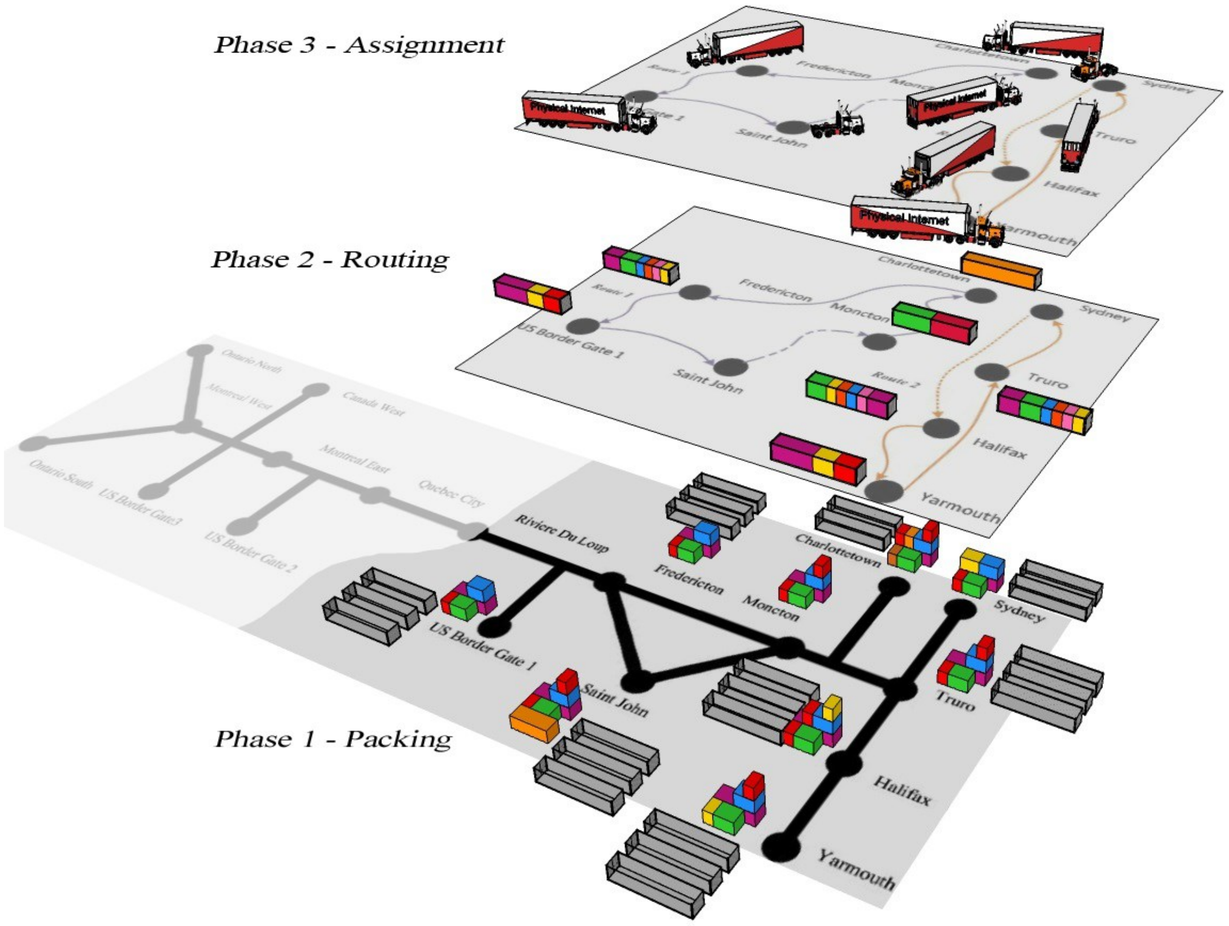

Figure 14 - Three Phase Optimization Framework

These three steps (or optimization phases) are applied across the various logistics systems studied in this thesis using the optimization framework presented below to compare PI, CO, and HY for the set of KPI's discussed.

4.2 Model Assumptions

The following assumptions are made in this thesis:

- A set of discrete load sizes are generated. All loads need to be transported by the end of the planning period.
- The number containers at each node is unlimited.

- Trucks (drivers) have home bases. All trucks (drivers) need to be return to their home bases by the end of the planning period.
- For the PI logistics policy, consolidation and deconsolidation occurs at all nodes (source, destination, and PI-transit centers). For the CO logistics policy, consolidation and deconsolidation occurs only at source and destination nodes.
- The three optimization models described below are all single-period models. This means there is no load movement requests carried over from one period to the next. In other words, the entire systems resets to its initial state at the beginning of each period.

4.3 Elements of the Optimization Framework

The three optimization are models written using the GLPK (GNU Linear Programming Kit) package. The GLPK models are converted to *.MPS* or *.LP* format and executed by Gurobi Optimizer© Version 5.6. These optimization models work in the following fashion:

1) *Packing* Optimization (PO) - Phase 1

   PO reads the generated requests and packs them into containers. The purpose of this optimization step is to find the minimum number of the containers required to transport all the requests from their initial location or source to their destination node. This depends on the number and size of requests and also the type of logistical network (CO, PI, or HY).

2) *Routing* Optimization (RO) - Phase 2

   RO reads the result of PO and finds a series of node sequences to determine the order in which a truck can transport containers so that all the containers reach their final destination. The sequence of nodes visited by a truck carrying different containers is called a route. Routes are created such that first and last node visited by a truck coincide. In other words, each route created by the RO is a loop.

3) *Scheduling* Optimization (SO) - Phase 3

   SO model reads the list of the routes created by RO and associates each route as a unique job that should be assigned to a truck. The scheduling models works similar to an assignment problem to find the minimized cost of performing all the routes with the given set trucks in a given delivery time span. This optimization model is executed three times with different delivery time spans to find the minimum number of the trucks required at each node location.

An auxiliary module for loop elimination uses a property specific to tree networks to reduce the volume of the data transferred from PO to RO. Reducing the volume of the data entering RO reduces the solution time of RO model significantly without affecting the value of optimal solution.

Figure 15 shows the data flows in the optimization framework and how the optimization models are connected to each other. Load movement requests (from node to node in the network) are generated by the data generation module. They are then passed on to the PO model in the first phase of the optimization. The output of the PO model is passed on to the RO model the result of which is then passed on to the SO. Once SO is run, the results are stored in a database and the comparison KPIs evaluated.

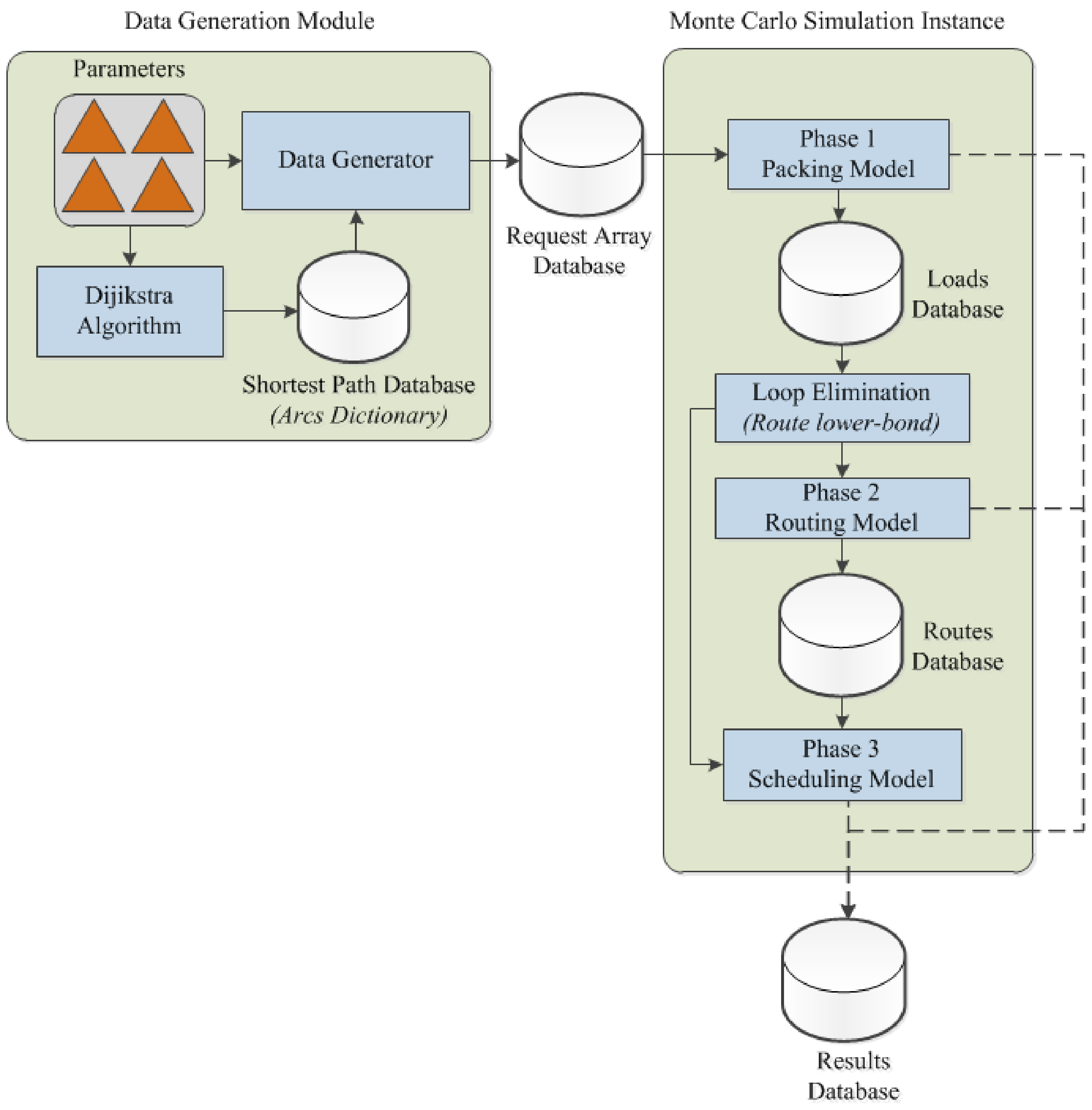

Figure 15 - Data Module and the Optimization Framework

4.4 Phase 1: Packing Model (PO)

The PO model is a one dimensional Bin-Packing Problem (BPP). In the BPP a list of items with fractional volumes are given. The objective is to find the minimum number of the bins to fit these items such that total volume of the items assigned to a bin is less or equal to one [34]. In the PO model, the generated requests are the items (or PI-boxes) and the 40 foot PI-containers are the bins.

In the PO model, two sets *Request* and *Container* are defined. Data in the *Request* set is generated in the Data Generation Module. In contrast, number of containers in the *Container* set is fixed at an arbitrary large integer number *k.* This number defines the maximum number of the containers that could be used to solve an instance of the BPP. Setting too small a value for k could result in infeasibility, and too large a value will increase solution time. An appropriate value for k can be set by investigating the number of elements in *Request* set. Obviously, this model is always feasible when the value of k is any large enough.

In the PO model, request sizes are stored in REQ_SIZE_r. This parameter is defined over the set of requests. Parameter *Capacity* is defined as a constant value set to 1. Binary assignment variable as_{rc} captures the assignment of request *r* to container *c*. No partial assignment of request *r* to container *c* is not allowed in the model. Finally con_n_C variable counts the minimum number of conatiners required to transport all the requests.

The following are the sets, parameters, and variables of this model:

Sets:

$Req = \{1 \dots q\}$	Set of Requests. *q: integer*
$Cont = \{1 \dots k\}$	Set of Containers parked at each node. *k: integer*

Parameters:

REQ_SIZE_r	Fractional size of request *r* from node *i* to node *j.* Set of fractional sizes to choose from. Available Sizes are: {0.125, 0.25, 0.5, 0.75, 1, 0}
$CAPACITY$	Capacity of each container. Default = 1.

Variables:

as_{rc}	Binary, 1 if request *r* is assigned to container *c*, 0 otherwise
con_n_C	Binary, 1 if container *c* is used

Mixed Integer Linear Model:

$$Minimize\ Z_1 = \sum_{c=1}^{k} con\_n_c \tag{1}$$

Subject to:

$$\sum_{c=1}^{k} as_{rc} = 1 \qquad r \in Req \qquad (2)$$

$$\sum_{r=1}^{q} as_{rc} \cdot REQ\_SIZE_r \leq con\_n_c \cdot CAPACITY \qquad c \in Cont \qquad (3)$$

Objective function minimizes the total number of the containers required. Constraint 2 assigns every request *r* to a container. Constraint 3 ensures the total volume of the requests assigned to container *c* is less than or equal to the capacity of the container.

This model minimizes the number of container for each source and destination pair. Therefore, PO model should be called and executed for every possible source destination pair determined by the CO, PI, and HY logistic system. Appendix F: GLPK Code of Packing MIP shows the GLPK code for the Packing MIP.

Depending on the number of loads generated, instances of this model can have between 4 to approximately 300K integer and binary variables. The time required to reach 0.005% MIP gap depends on the number of variables and varies from a fraction of a second for a 4 variable instance to approximately 20 seconds for a 300K instance on an Intel Xeon processor with 8 logical cores, 16 GB (ECC) memory, and a processing speed of 3.4 Ghz.

4.4.1 Packing MIP - Calling Algorithm

The PO model is called for each source and destination pair to minimize the number of containers required for transportation across the entire network. One may ask what the valid source and destination pairs are in each of the logistic systems. The valid source-destination pairs *Active Arcs* are the arcs that directly connect a source node to a destination node. Each active arc is a dictionary key in the logistic dictionaries.

For example, in the CO logistic system Halifax-Quebec City is an active arc. In PI and HY logistic systems, there is no direct flow between Halifax and Quebec City, therefore this arc is not present in these two logistic systems but is replaced by three active arcs in PI: Halifax-Truro, Truro-Fredericton, and Fredericton-Quebec City.

The number of active arcs (or number of dictionary rows) in CO, PI, and HY are 308, 80, and 68 respectively. There are 18 nodes in the network used in this thesis. Since the active arcs in CO are from any node to any other node, there are 18*17, i.e., 308 active arcs. The number of arcs in PI and HY are fewer, and are a function of the number of arcs and transit nodes. The PO model should be called and executed these many times as there are active arcs.

The PO calling algorithm is written in Python 2.7. The PO model is also written in Python using standard code in a Python library called "*Gurobipy*". This library enables user to call Gurobi optimization within the Python environment and feeds the MIP model to the optimizer directly [35]. The following, is the pseudo code of the calling algorithm used in this thesis. Appendix G: Packing Model Calling Algorithm, contains the actual Python code of this part.

```
Read List of Loads from the Request Array Database
Read List of Active Arcs from the Logistics Policy dictionaries

  For i in Node:
    For j in Node:
      If ((i,j) pair exist in the logistics policy dictionary):
        Loads_array → Request Array Database
        --------------------------------------------------
        Create PO Model sets, parameters, variables
        Create constraint set 1 (assignment constraint)
        Create constraint set 2 (capacity constraint)
        Define the objective function
        Call model.optimize()
       Read the solution variable values from the optimizer
       Write solution into results database
```

4.5 Phase 2: Routing Model (RO)

The heart of the optimization framework is the routing optimization model (RO). The Routing model reads the number of containers on each active arc and finds sets of sequences called *Routes* of minimum total length. The logical constraints in RO are on *Demand*, *Flow Conservation*, and *Subtour elimination*. There are also optional constraints that could be added to RO such as *Vehicle Capacity*, and *Routing Policy* constraints. *Routing policy* constraints could include constraints on the total routing time or the number of routes generated.

4.5.1 Routing MIP

The RO MIP uses the reduced flows on an active arc as the number of the times that arc could be used in routing. In the RO model, two sets of *Routes* and *Nodes* are defined. The nodes set contains the list of the 18 nodes defined in the network.

Routes set contains m elements defined as the maximum number of the routes that could be created to solve an instance of the RO problem. Setting too small a value for m could result in infeasibility. An appropriate value for m can be set by investigating the number of reduced flows to be routed. This model is always feasible for any large enough value of m.

If an arc is used to transport a full container, the time required to travel over this arc is shown by $FTTIME$. However, as the speed of commute increases while deadheading, deadheading over the same arc requires $ETTIM$ time. M is an auxiliary parameter used in the model for subtour elimination. The value of M can be also used as a maximum number of the stops in a route if variable s_{ik} is set to be integer. Variable r_k is used to capture the total duration of a route, and s_{ik} is used as an auxiliary variable in sub tour elimination. All the variables, parameters, and sets used in RO model are shown in the table below:

Sets:

Nodes = {1…n}	Set of nodes in the network
Route = {1…m}	Set of empty routes

Parameters:

RED_CONT_{ij}	Reduced Flow or number of containers to be sent from node *i* to node *j*
$FTTIME_{ij}$	Loaded (or full) travel time from node *i* to node *j*
$ETTIME_{ij}$	Deadhead (or empty) travel time from node *i* to node *j*
M	Auxiliary parameter set at an arbitrary large value

Variables:

fa_{ijk}	Binary, 1 if route *k* includes the loaded arc from node *i* to node *j*, 0 otherwise
ea_{ijk}	Binary, 1 if route *k* includes the deadhead arc from node *i* to node *j*, 0 otherwise
r_k	Total duration (or length) of route *k*
s_{ik}	Auxiliary positive integer variable used for sub tour elimination

Mixed Integer Linear RO Model:

$$Minimize \;\; Z_2 = \sum_{k=1}^{m} r_k \tag{7}$$

Subject to:

$$\sum_{k=1}^{m} fa_{ijk} = RED\_CONT_{ij} \qquad \forall i, j \in Node \tag{8}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}(FTTIME \cdot fa_{ijk} + \; ETTIME \cdot ea_{ijk}) = \; r_k \qquad \forall k \in Route \tag{9}$$

$$\sum_{i=1}^{n}(fa_{ipk} + ea_{ipk}) = \sum_{j=1}^{n}(fa_{pjk} + ea_{pjk}) \qquad \forall p \in Node, \; \forall k \in Route \tag{10}$$

$$\sum_{i=1}^{n}\left(fa_{ipk} + ea_{ipk}\right) + \sum_{j=1}^{n}\left(fa_{pjk} + ea_{pjk}\right) \; \leq 2 \qquad \forall p \in Node, \; \forall k \in Route \tag{11}$$

$$\sum_{k=1}^{m} ea_{ijk} \; \leq 1 \qquad \forall i, j \in Node, \; \forall k \in Route \tag{12}$$

$$s_{ik} - s_{jk} \; + 1 \leq M \cdot (1 - \; fa_{ijk}) \qquad \forall i, j \in Node, \; \forall k \in Route \tag{13}$$

$$s_{ik} \; \geq 1, s_{ik} \leq M \qquad \forall i \in Node, \; \forall k \in Route \tag{14}$$

The objective function of the RO model in (7) minimizes the total duration of the routes created. Constraint 8 is the demand constraint. As explained, number of the times an arc can be used for transportation of containers should be equal to the reduced flow. Constraint 9 is used to capture the total duration of a route *k*. In constraint 10, sum of all the full and deadhead flows in to a node should be equal to the sum of full and deadhead flows out of the same node. Constraint 11 ensures that a flow is either full (with a container) or deadhead but not both at the same time. Constraints 10 and 11 together are used to force flow conservation in the model. Constraints 12, 13, 14 are used for sub tour elimination. 13 and 14 are the MTZ constraints described in [19] and discussed in the VRP literature review. These constraints assign an auxiliary value to node *j* if node j is immediately visited after node i in route k. Using this method, nodes are visited in just one single and fully integrated sequence. Constraints 13, and 14 impose sub tour elimination on the container flows (fa_{ipk}) but not on the deadhead flows (ea_{ipk}). Constraint 12 sets the number of deadhead flows per route to maximum of one. A single flow cannot create a sub tour by itself, so sub tour elimination constraints are not required for this type of flow. With use of flow conservation constraints in 10 and 11, a single deadhead flow is assigned such that it connects the very last node visited by route k to the very first node in route k. Therefore every route created in RO model is a loop starting at an arbitrary node and ending at the same node. Having loops as the routes is very beneficial and reduces the complexity of the resource scheduling in the next optimization phase. Appendix I: GLPK Code of Routing MIP, shows the GLPK code of RO model.

In addition to the standard set of constraints just described, auxiliary constraints may also be added to RO to restrict route durations, lengths, number of routes, maximum allowed deadhead length, etc. The following parameters and a new binary variable may be introduced through optional constraints 15 to 19 in the RO model to accomplish this.

Binary variable act_r_k is used to indicate whether a route is active or not. An active route is a route that has a duration greater than zero.

$TTTIME$	Maximum allowed duration (or length) for each route
$MITIME$	Minimum allowed duration (or length) for each route
AVG_R_LEN	Average duration (or length) of each route
MAX_DE	Max allowed deadhead length as percentage of r_k
NUM_ACT_R	Max number of the routes allowed to be created to serve all the requests
act_r_k	Binary, 1 if $r_k > 0$, 0 otherwise

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left(ea_{ijk} \cdot ETTIME\right) / MAX\_DE \le r_k \qquad \forall k \in Route \tag{15}$$

$$\sum_{k=1}^{m}(r_k) \;/\; AVG\_R\_LEN \;\le\; \sum_{k=1}^{m} act\_r_k \tag{16}$$

$$\sum_{k=1}^{m} act\_r_k \;\le NUM\_ACT\_R \tag{17}$$

$$r_k \geq act\_r_k \cdot MITIME \qquad \forall k \in Route \qquad (18)$$

$$r_k \leq act\_r_k \cdot TTTIME \qquad \forall k \in Route \qquad (19)$$

Constraint 15 sets an upper limit on the maximum duration of deadhead allowed in a route as a percentage of total route length. Constraint 16 sets an average route length for all the routes. This constraint enforces RO model to only build such routes that together produce an average route length of not greater than the fixed Average Route Length parameter. Constraint 17 sets a cap on the number of routes that could be activated by the RO model, enforcing the model to perform all routing activities with a pre specified number of routes. Constraints 18 and 19 set the minimum and maximum duration on the length of the generated routes.

Unfortunately, RO is unable to find a set of minimum length routes (with 0.005% MIP gap on an Intel® Xeon® E3 @ 3.4 GHz) in a reasonable amount of time. Since the vehicle routing problem on tree is a *NP-hard* problem [36] and it is not surprising to observe very long solution times for the instances of the problems in this thesis. To reduce the solution time a routing property on trees is used. This property is explained in the section 4.5.2.

4.5.2 Loop Elimination on Tree Shaped Networks

It is well known that a reasonably sized VRP instance is very difficult to solve [37]. A common approach to solving the VRP is column generation [38]. In this approach, a set of feasible routes which satisfy the routing constraints are generated in a separate model. The main model then selects and combines these routes in a feasible and optimal manner till an optimal solution is found.

Column generation is the one approach to deal with the complexity of VRP models. However, the strategy used in the thesis to solve the RO model relies on a property that tree networks could benefit from, and therefore is specific to the unique network structure of the routing network considered in this thesis. This strategy involves elimination of loops in VRP for trees and is explained in [39].

As mentioned, PO model finds the minimum number of the containers on each active arc. This number determines the number of instances a loaded travel from source to destination on that active arc is possible. These container flows are shown with solid lines in Figure 15. Empty travel is also permitted on the active arcs. There might be situations where there is no container to be carried from a source to a destination, but a truck has to still use that active arc to be able to complete a route, or go back to a home location. The empty flows h_{ij} (deadheads hereafter) are shown with dashed lines in Figure 15. If node *i* can be connected to node *j* by an active arc, four distinctive flows are generated. Full or empty travel out of node *i* towards node *j* and full and empty travel out of node *j* towards node *i*.

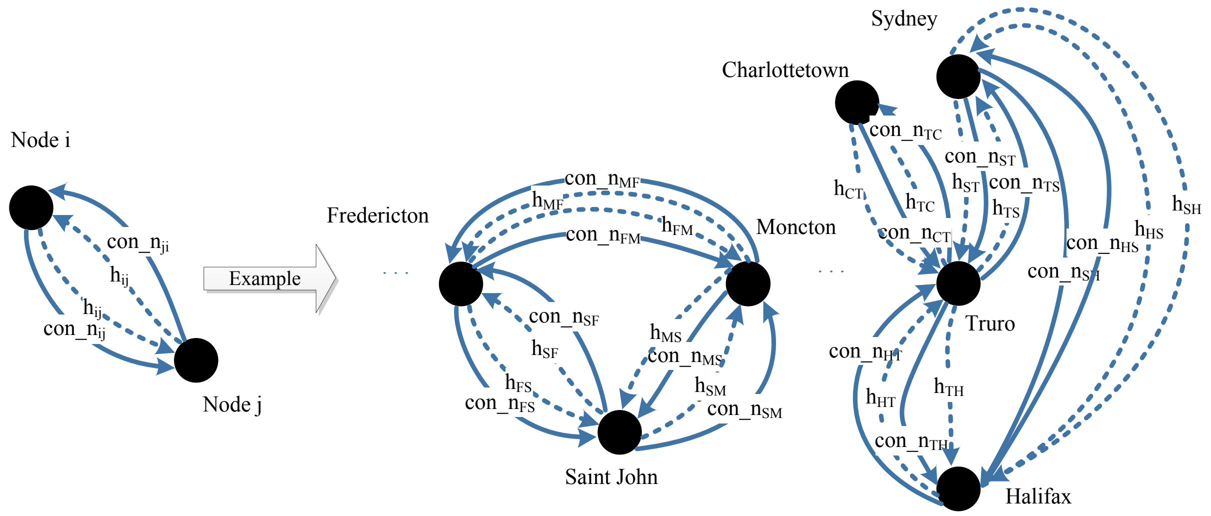

Figure 15 - Illustration of the Flows in the Network

To visualize this strategy, the deadhead flows are eliminated and container flows in and out of nodes *i* and *j* are shown with only one solid line in Figure 16.

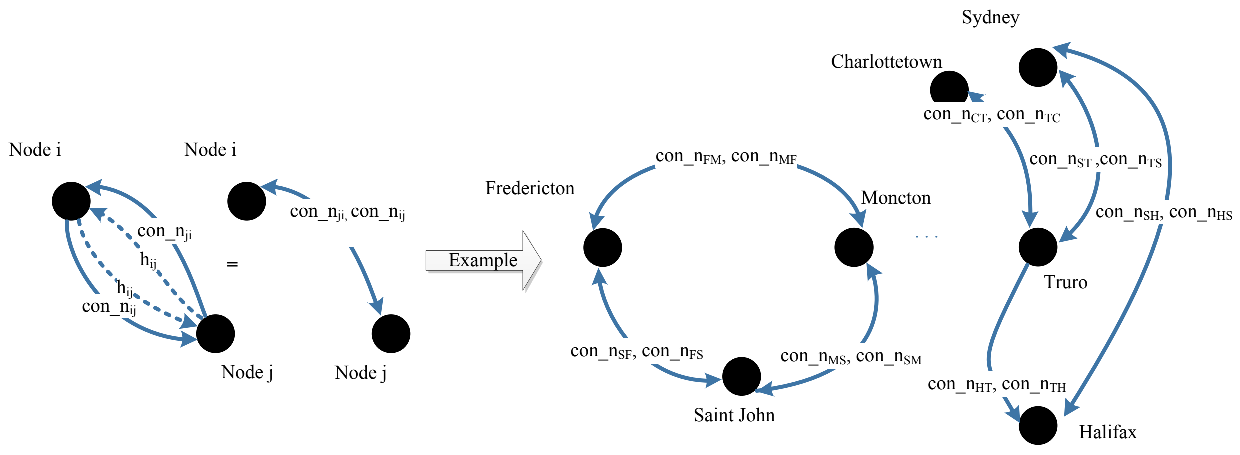
Node i
con_n$_{ji}$
h$_{ij}$
h$_{ij}$
con_n$_{ij}$
Node j
=
Node i
con_n$_{ji}$, con_n$_{ij}$
Node j
Example
Fredericton
con_n$_{FM}$, con_n$_{MF}$
Moncton
con_n$_{SF}$, con_n$_{FS}$
con_n$_{MS}$, con_n$_{SM}$
Saint John
Sydney
Charlottetown
con_n$_{CT}$, con_n$_{TC}$
con_n$_{ST}$,con_n$_{TS}$
con_n$_{SH}$, con_n$_{HS}$
Truro
con_n$_{HT}$, con_n$_{TH}$
Halifax

Figure 16 - Illustration of the Equivalent Flows in the Network

Since the network has a tree structure except for the part of the network creating a loop among Moncton, Fredericton, and Saint John. These nodes can be connected to each other with more than one distinctive routes. However, since the shortest path between any two nodes is a unique path, there is only one logical route to go to node i from node j and vice versa and our network therefore can benefit from the loop elimination property for trees.

Note that con_n_{ij} is the number of containers to be transported from source i to destination j and con_n_{ji} is the number of containers to be transported from source j to destination i. Therefore the smaller of these two values are the number of complete loops created between node i and node j.

$$Complete\ Loop_{ij} = min(con\_n_{ij}\ , con\_n_{ji}) \tag{5}$$

Since there is only one feasible arc to go from i to j, and the length of this arc is equal to the lower bound on the travel distance from i to j, complete loops must be a subset of the optimal route set and therefore may be eliminated for optimization purposes (however, the cost of these loops are recorded to calculate total logistics costs).

The absolute difference between con_n_{ij} and con_n_{ji} is called the *reduced flow* (rf_{ij}) between node i and node j and its direction and magnitude are calculated below:

$$\overrightarrow{rf}_{ij} = max(\overrightarrow{con\_n}_{ij}, \overrightarrow{con\_n}_{ji}), \qquad rf_{ij} = |con\_n_{ij} - con\_n_{ji}| \tag{6}$$

The direction of the reduced flow is the same as the direction of the larger of $\overrightarrow{con\_n}_{ij}$, and $\overrightarrow{con\_n}_{ji}$.

In loop elimination, only the reduced flows are entered in RO. The value of the total duration of the eliminated loops is then added to the optimal solution of RO to report the result of the routing optimization phase.

Appendix H: Calculation of Flow Lower Bond shows the Python program written to find the loops created after running the PO model. This program calculates the total duration of each loop, calculates the cumulative duration of all routes, and in the end, eliminates these loops from the data set sent to RO.

Depending on the number of routes and number of containers to be routed, various instances of this model (without auxiliary constraints 15-19) can have between 80K to approximately 400K integer and binary variables combined. The time required to reach a 0.005% MIP gap depends on number of variables. As explained, if loop elimination is performed, the number of variables entered into the routing model significantly decreases. If the loop elimination step is performed, the solution time of a problem instance with 400K variables is reduced to approximately 50 minutes in the worst case. On average, the solution time is approximately 25 minutes on an Intel® Xeon® E3 @ 3.4 GHz.

With auxiliary constraints, the solutions performance is sometimes faster because they act as valid inequalities and constrain the search space of the MIP. The reader is referred to [37] to see examples of integer programs that are solved more easily when additional constraints are used.

In grid shaped networks there are more than one route connecting nodes to each other. Flows between any two nodes could be assigned to different routes connecting the two nodes arbitrarily or according to other considerations such as route capacity, route congestions, and etc. Loop elimination in grid type networks therefore is a multi-step process. In the first step, (among all the route options available) flows between a source-destination pair should be assigned to a specific route. In the second step, the net flow on each route should be calculated. Finally, in third step loop elimination can be performed when the net flows of any source destination pair on each route is known.

4.6 Phase 3: Assignment and Cost Model

Assignment model is the third and the last phase in the optimization framework. It uses the identified loops and the generated routes in the RO model as input, and assigns these routes to a set of available vehicles. The assignments of the routes (jobs hereafter) to the vehicles are such that the total cost of logistic system is minimized. The total cost in this model consists of three components: operational, social, and fixed costs.

4.6.1 Operational Cost

The operational cost is the most significant component of the total cost functions. This cost is incurred when a tractor truck (truck hereafter) is driven from one node to another node. A research report [40] presented to Transport Canada is used to extract the operational cost of truck driving activities. The following is a summary of the factors considered in calculation of the operation cost:

1) *Labour Cost of Drivers*: Drivers cost include hourly wage for all the time a driver is on duty (ex. driving, loading, unloading time) with an additional percentage added to include any expected medical insurance or pension payments.
2) *Average Fuel Cost*: Fuel cost is an average fuel consumption rate of a truck per kilometer or per hour multiplied by the fuel price. Some logistics companies have long term fuel purchase contract with fuel suppliers in Canada. Although the exact terms of these contracts are confidential, an expected bulk fuel price is used in calculation of the fuel cost of a truck.
3) *Repair and Tire Cost*: repair costs are incurred to maintain a vehicle operable over its life span. Repair cost includes, vehicle maintenance costs, cleaning, and miscellaneous costs. Tires are the significant contributors to operational cost of trucks and need very frequent change and maintenance. Estimated tire wear rate and average price of tires across Canadian provinces are used to calculate the tire costs of vehicle.
4) *Registration and Licence Fee*: This is a fixed annual cost for registration a commercial truck in Canada. Provincial and territorial charges slightly vary in Canada. A complete hand book of territorial charges, regulations, and the taxes can be found in [41].
5) *Cost of Capital*: This includes all the costs associated with the amortization of a vehicle such as interest rate and opportunity cost of the money. An average truck market price and an average residual value of trucks at the end of their life is used in this calculation.
6) *Vehicle Insurance Cost*: Insurance cost is historically calculated as roughly 3% of the revenue generated by a truck with full annual utilization.
7) *Average Toll Charges*: The extent to which toll charges effect the truck's operational cost depends on usage frequency of such toll facilities for transportation. In Canada there are 17 toll facilities in total: MacDonald, and MacKay bridges, and Highway 104 in NS, Confederation Bridge in PEI, Saint John Harbour Bridge in NB, Coquihalla Highway in BC, Highway 407, and 10 Canada US border bridges in Ontario. The total revenue from these toll facilities is approximately $206 million dollars [40]. This number divided by the total number of the trucks used toll facilities can be used as an estimate of the average toll charges in operational cost of a truck.

Congestion is calculated to increase the truck operational costs on average by 15% [40]. An average of \$3.15 per kilometer is calculated as the operational cost of a truck. This value increases to \$3.58 cents per kilometer for operation in congested areas. Since the network road network in this thesis excludes congested urban areas, the \$3.58 cents per kilometer was not used in this thesis. Therefore, for an average truck speed of 65 kilometers per hour (used in this thesis), the hourly operational cost of truck is approximately \$200.

4.6.2 Social Cost

According to Transport Canada's regulation no driver shall exceed 14 hours of on duty (or driving) time per day. After each 14 hours of driving, a driver should take 8 hours of rest time [42]. Therefore a driver has to park in a designated road side park and sleeps areas or stops at a bed and breakfast overnight if not going back home in 14 hours. The social cost is defined as the cost incurred per night if a truck driver has to take rest while on duty. An average cost of park and sleep or economy bed and breakfast for truck drivers is approximately \$60 per night. The effect of seasonality or weather condition is not considered in this cost. Moreover, truck drivers struggle with number of common occupational health and safety issues such as sleep disorders, spine and lumbar problems due to continuous sitting, etc [43]. Due to the complexity and lack of universal data for cost of such treatments, these costs are hard to measure and were not included in this thesis.

Jobs with driving duration longer than 14 hours will therefore result in a social cost on top of the operation cost. Also these jobs will require a longer completion time because a driver needs rest while enroute. A job completion time is called "*enroute time*" in this thesis. For example, if a job has driving time of 30 hours its social cost and total enroute time can be calculated as follows:

$$Total\ Social\ Cost\ = \left\lfloor \frac{Driving\ time}{Daily\ Driving\ Limit} \right\rfloor \cdot (Social\ Cost)\ \rightarrow\ \left\lfloor \frac{30}{14} \right\rfloor \cdot \$60 = \$120$$

$$Enroute\ time = Driving\ time + \left\lfloor \frac{Driving\ time}{Daily\ Driving\ Limit} \right\rfloor \cdot (Rest\ Time) \rightarrow\ 30 + \left\lfloor \frac{30}{14} \right\rfloor \cdot 8 = 46\ hrs$$

Jobs created in Model 2 have a maximum duration of 72 hours. Jobs with required driving time between 14 to 72 hours are called "*Long Jobs*" and jobs with required driving time of less than

14 hours are called "*Short Jobs*". Appendix J: Calculation of the Jobs Enroute Time in Python includes the Python code used to calculate the enroute time of each job.

4.6.3 Fixed Costs

As explained in the social cost section, jobs can be divided into two major categories: day jobs and multiple day jobs. Trucks that are assigned to serve multiple day jobs require a sleeper compartment. Trucks with and without sleeper compartments are called "*Long Haul*" and "*Short Haul*" trucks respectively. Figure 17 uses the objects available in 3D Warehouse [33] of Google SketchUp™ to display a sample short haul and a sample long haul truck used for one day and multiple day job assignments respectively.

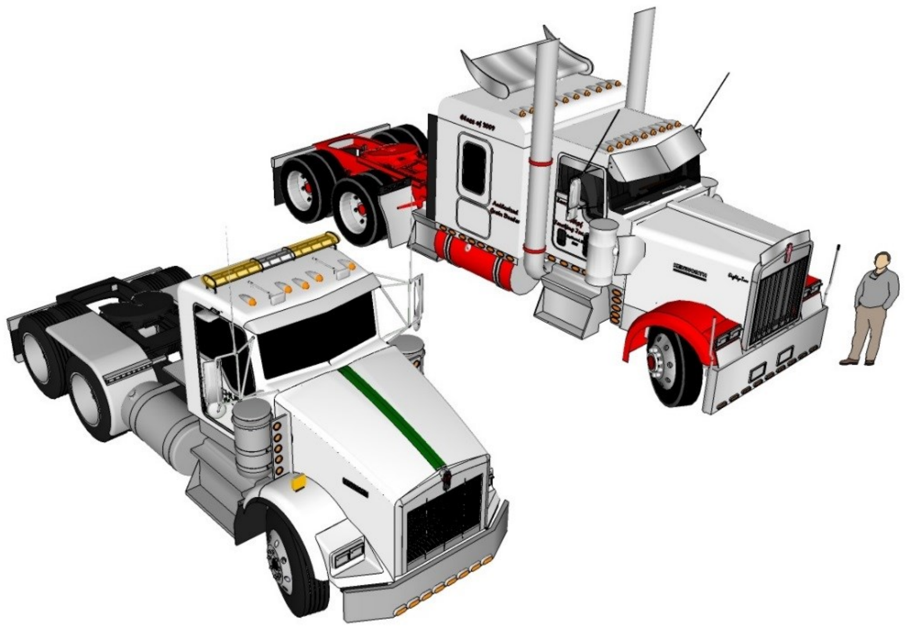

Figure 17 - Short and Long Haul Trucks [33]

The fixed cost of entering a short haul and a long haul truck into service is approximately $10, and $100 respectively. This cost includes administration, dispatching agent's time, paper work and other costs that may be incurred if a driver needs to carry any special equipment (such as the rental cost for auxiliary power generation units, safety gear for special loads, etc.) while enroute. These fixed costs were inferred from an interview with two logistic coordinators of at a logistics company in Halifax, Nova Scotia.

There are three sets (nodes, jobs, and trucks) in the Assignment model. The route information is entered into the jobs sets. Jobs are identified by their starting node location, driving time, and enroute duration. These values are stored in *JL*, *JDD* and *JD* parameters. As mentioned, each job is a loop, therefore only the starting node of a job is sufficient to identify a job.

Each job can be served with only one truck. Similar to the jobs, each truck is identified by its home node. Also each truck is either a short haul or long haul truck. Model 3 assumes that both types of trucks are available at each node; therefore the problem is never infeasible due the lack of required trucks. If truck *i* at node n_1 is assigned to serve job *j* at node n_2, the deadhead duration incurred because of this assignment is the round trip duration between node n_1 and n_2. Parameter DDUR in Model 3 calculates the deadhead durations.

4.6.4 Assignment and Cost MIP

Model 3 calculates the minimum number of truck required to serve all the jobs in a 72 hour time window. Finding the minimum number of trucks to serve the jobs is equal to minimizing the total assignment cost. The model objective function minimizes the operation, social, and fix costs of assignment.

Sets:

Node	Set of nodes in the network
Job = {1…*m*}	Set of jobs
Truck = {1…*n*}	Set of trucks. Trucks are either short haul or long haul.

Parameters:

TL_i	Location node of truck *i*
JL_j	Location node of job *j*
$DIST_{n1,n2}$	Distance in time from node n_1 to node n_2
$DDUR_{ij}$	Round trip distance in time from location node of truck *i* to location node of job *j*
JDD_j	Driving time required to serve job *j*.
JD_j	Total time required to serve job *j*.
$DELVSPAN_i$	Maximum driving time allowed per day (set to 14 hours per day for short haul and 72 hours total for long haul trucks).
$DAYDRIVE_i$	Maximum on duty time allowed per day (set to 14 hours).

$FIXC$	Fixed cost of entering a new short haul truck into service
$HOURC$	Average hourly operational cost of a short haul truck (set to \$220 /hour)
$SOCIALC$	Social cost per day per truck driver when away from home (set to \$50 /day)

Variables:

x_{ij}	Binary, 1 if truck *i* is assigned to serve job *j*
fix_i	Binary, 1 if truck *i* enters service, 0 otherwise
$enroutet_i$	Total working time of truck *i*
opr_cost_i	Total operational cost incurred by truck *i.*
soc_cost_i	Total social cost incurred by truck *i*
fix_cost_i	Fixed cost of entering truck *i* into service

Objective Function:

$$Minimize\ \ Z_3 = \sum_{i=1}^{n}(opr\_cost_i + soc\_cost_i + fix\_cost_i) \tag{20}$$

Subject to:

$$\sum_{j=1}^{m}(DDUR_{ij} + JD_j\ ) \cdot x_{ij} = enroutet_i \qquad \forall i \in Truck \tag{21}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in Job \tag{22}$$

$$enroutet_i \leq DELVSPAN_i \cdot fix_i \qquad \forall i \in Truck \tag{23}$$

$$\sum_{j=1}^{m}(DDUR_{ij} + JDD_j\ ) \cdot HOURC_i \cdot x_{ij} = opr\_cost_i \qquad \forall i \in Truck \tag{24}$$

$$\sum_{\forall j \in Job \mid JDD_j \geq DAYDRIVE}^{n} \frac{JDD_j \cdot x_{ij}}{DAYDRIVE} \cdot SOCIALC = \ soc\_cost_i \qquad \forall i \in Truck \tag{25}$$

$$FIXC \cdot fix_i = fix\_cost_i \qquad \forall i \in Truck \tag{26}$$

In the model, constraint 21 calculates the total duration of truck *i* (job duration and deadhead travel). Constraint 22 ensures that each job is done by one and only one truck. Constraint 23 ensures that the enroute time of each truck remains less than the allowed delivery span. Constraint 24 calculates the operational cost incurred by the operation of truck *i*. Constraint 25 calculates the social cost of performing a job considering only jobs that require a duty duration of longer than 14 hours. Constraint 26 calculates the fixed cost of assignment.

Instances of the model solved in this thesis have between 400K and 3M variables. The solution times to generate 0.005 % MIP gap varied from 10 minutes to 6 hours on an 8 core Intel® Xeon® E3 @ 3.4 Ghz. Appendix K: GLPK Code of Assignment MIP.

Chapter 5: Data Generation and Optimization

This chapter describes how data is generated for the PI, CO, and HY logistics systems for comparison purposes.

Data is generated randomly using Monte-Carlo simulation and stored in the Request Array Database. The basic data consists of a move request from a source node to a destination node with a load size specified in terms of PI-container capacities.

This is done for each of the three logistic polices. Therefore to complete an experiment instance, the three phase optimization framework is performed on each of the CO, PI, and HY loads. Figure 18 is a conceptual representation of the application of the three phase optimization framework applied to the simulation instances. The results generated from CO, PI and HY logistic systems are shown in red, green, and blue respectively. These results are investigated and discussed in detail in Chapter 6: Results, of this report.

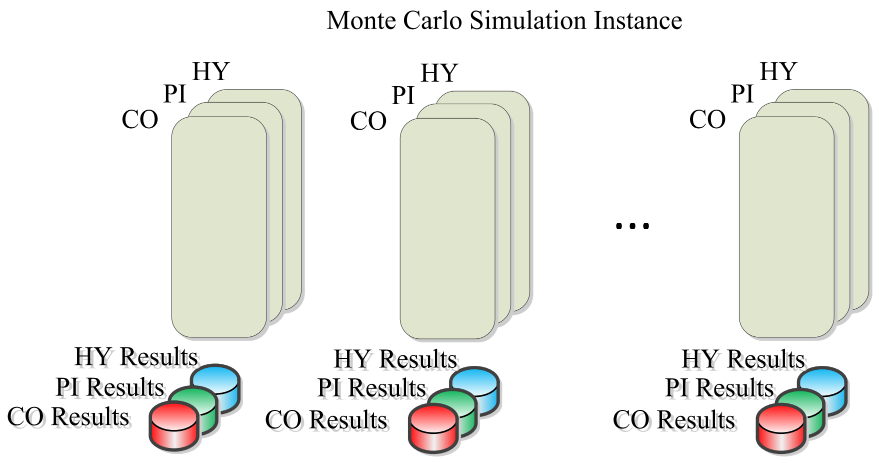

Figure 18 - Monte Carlo Simulation Instances

The data used in this thesis is generated using the parameters below:

- The average speed of trucks set to 80 km per hour
- A set of five different load sizes with the following probability distribution

$$P(0.125) = \frac{8}{15}, P(0.25) = \frac{4}{15}, P(0.5) = \frac{2}{15}, P(1) = \frac{1}{15}, P(0) = \frac{1}{15}$$

- The nodes in the Eastern Canada road network as used with actual distances in kilometers.
- The number of loads to be generated (explained in section 5.2)

The data generation module uses Monte-Carlo simulation generate an array of loads. Each line in this array is called a "*request*". Each request therefore is identified by its row number, source node, destination node, and load size.

5.1 Data Generation Module

Data generation module has four parts, all of which are written in the Python programming language (Figure 19). The set of parameters is recorded in a python file called "*param.py*". The python code in this file is given in Appendix D: Parameters File. This file is kept as a reference and is read by the data generator every time a new loads array is required.

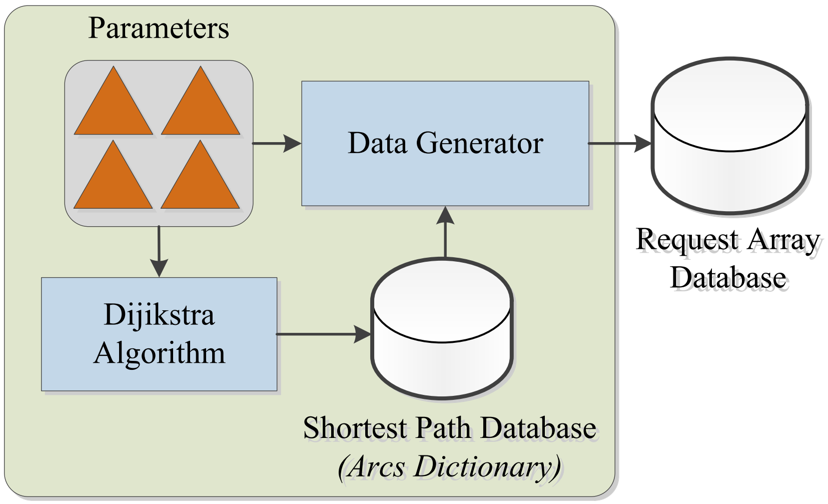

Figure 19 - Data Generation Module

Python code implementing the Dijkstra algorithm [44] is used to find the shortest paths between each node in the network. This algorithm finds the shortest route (or path with the lowest cost) between a node and all the other nodes in the network. The solution time complexity of this algorithm can be shown to be $O|V^2|$ where V is the number of vertices or nodes in the network [44]. Dijkstra's algorithm has to be executed V times to find the complete set of shortest paths among all the nodes in the network. This implies that finding the shortest paths using the Dijkstra algorithm has a complexity of $V \cdot O|V^2|$. There are other shortest path algorithms such as Floyd Warshall [45] with a solution time complexity of $O|V^3|$. Floyd-Warshall is required to run only once to get the full set of shortest paths among all the nodes in the network. While we implemented the Dijkstra algorithm, it should be noted that this is done only once. The network in this study does not have any dynamic or changing features such as variation in availability of source-destination arcs or varying route capacity. Therefore, it is sufficient to find the shortest paths among the nodes in the network once and store the results in a database.

The solution time required for the Dijkstra Algorithm to find the shortest paths in the road network was between 30 seconds in CO, to approximately 2 minutes in PI. This solution time varied because each logistic system permitted different sets of source-destination arcs to be used.

After execution of the Dijikstra algorithm, a Shortest Path Dictionary was generated for each logistic system. Each line in Shortest Path Dictionary is called a recordset. A recordset itself consists of a *key pair* and an *array element*. The key pair consists of a unique combination of a source and a destination node. The array element presents the arcs in the shortest path of the key pair. Elements of these dictionaries presented in Appendix A: Conventional Arcs Dictionary, Appendix B: PI Arcs Dictionary, and Appendix C: Hybrid Arcs Dictionary.

Since each of the CO, PI, and HY systems use different routing policies, the array elements for each identical key pair varies in each Shortest Path Dictionary. For example, the key pair of (*Yarmouth, Ontario South*) will have the following array elements in CO, PI, and HY dictionaries respectively:

('Yar','Ots'): [['Yar','Ots']]

('Yar','Ots'): [['Yar','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']]

('Yar','Ots'): [['Yar','Tru'],['Tru','Mtw'],['Mtw','Ots']]

The top row from the CO dictionary shows no nodes visited enroute. The second row (from the PI dictionary) shows multiple arcs in the array element due to transshipment nodes corresponding to the PI Hubs. The HY dictionary has only two transhipment nodes, one at the source cluster and the other at the destination cluster.

The actual data generation occurs by executing the code in "Generator.py". This code consists of a number of data connection handles to read data from inputs and a number of nested loops to generate the loads array. Appendix E: Data Generator File, shows the Python code of the data grantor file. The pseudo code for the data generation mechanism is as follows:

```
Read Parameters file
Read Policy Dictionary → CO, PI, HY
For r in Req:
  For k in Node:
    For j in Node:
      Size = randomly choose a size from the list of available load sizes
      For i in range (Dictionary_keys()):
        If (Dictionary_keys[i][0] = Node[k] and Dictionary_keys[i][0] = Node [j]) :
        For l in range (Dictionary_array_element):
          For counter in range(Policy_node_counter[counter][0] = Dictionary_array_element[i][l][0] and
          Policy_node_counter[counter][1] = Dictionary_array_element[i][l][1]):
            Policy_node_counter[counter][2] =  Policy_node_counter[counter][2] +1
            Policy_Loads_array = (Dictionary_array_element[i][l][0],   Dictionary_array_element[i][l][1],
          size)
Import Policy_Loads_array into Request Array Database
```

5.2 Designed of Experiments and Loading Scenarios

In this section, the experimental design to investigate the performance of the logistic systems is explained. There are four types of experiments as explained in the following four sub-sections.

5.2.1 Low, Medium, and High Traffic Levels Experiments

In these experiments, two load requests are assigned for every (source, destination) pair. This scenario is to simulate a network with consistently low traffic across all nodes. For example, the total number of the requests in this scenario is:

$$18\,Sources \cdot 17\,Destinations \cdot 2\,\frac{Requests}{Source - Destination\ Pair} = \ 612\ Requests$$

This number is then set to five to simulate consistently moderate traffic across all the nodes in the network.

Finally, ten requests for every source-destination pair is generated to simulate a network with high traffic. Table 1, Table 2, and Table 3 show number of the requests and number of the implied requests for each logistic policy in each scenario. The implied requests are generated according to the array elements of the dictionaries. For example a generated request from Moncton to Montreal West would imply one request in CO policy, four requests in PI, (Moncton to Fredericton to Quebec City to Montreal East, to Montreal West) and two requests in HY (Moncton to Fredericton to Montreal West. Six instances of each scenario was generated using Monte Carlo simulation. Therefore, 18 set of results were generated through this experiment.

Table 1- Number of Requests in CO Logistic System

	Low Traffic Experiment	Moderate Traffic Experiment	High Traffic Experiment
Generated Requests	*612*	*1530*	*3060*
Total Implied Requests	*612*	*1530*	*3060*

Table 2 - Number of Requests to Simulate PI Logistic System

	Low Traffic Experiment	Moderate Traffic Experiment	High Traffic Experiment
Generated Requests	*612*	*1530*	*3060*
Total Implied Requests	*1700*	*4250*	*8500*

Table 3 - Number of Requests to Simulate HY Logistic System

	Low Traffic Experiment	Moderate Traffic Experiment	High Traffic Experiment
Generated Requests	*612*	*1530*	*3060*
Total Implied Requests	*1250*	*3125*	*6250*

5.2.2 High Traffic and Small Loads Experiment

This experiment is similar to the consistent traffic level experiment with ten loads for each source-destination pair. The only change is in the probability distribution of the load sizes. This experiment is designed to investigate the performance of the logistic systems when sizes of the requests in the network tend to be small. The following is the probability distribution of the loads in this experiment:

$$P(0.125) = \frac{8}{15}\,, P(0.25) = \frac{4}{15}\,, P(0.5) = \frac{3}{15}\,, P(0) = \frac{1}{15}$$

Similar to the consistent traffic level experiment, six instances of this experiment type were generated using Monte Carlo simulation.

5.2.3 High Traffic and Large Loads Experiment

This experiment is designed to investigate the performance of the logistic systems when the sizes of the requests in the network tend to be large. The following is the probability distribution of the loads in this experiment:

$$P(0.125) = \frac{1}{15}\,, P(0.25) = \frac{2}{15}\,, P(0.5) = \frac{4}{15}\,, P(1) = \frac{8}{15}$$

Six instances of this experiment type were generated using Monte Carlo simulation.

5.2.4 Random Traffic Experiment

In this experiment the probability distribution of the load sizes are returned to the original as follows:

$$P(0.125) = \frac{8}{15}\,, P(0.25) = \frac{4}{15}\,, P(0.5) = \frac{2}{15}\,, P(1) = \frac{1}{15}\,, P(0) = \frac{1}{15}$$

The difference in this experiment with previous five experiments is that the number of loads generated for each source-destination pair is set to a normally distributed non-negative value with mean of 10 and standard deviation of 3. This experiment is designed to investigate the performance of the logistic systems when number of loads in the various parts of the network is not consistent and therefore network could have a high traffic in some areas and low traffic in other areas. Once again, six instances of this experiment type were generated using Monte Carlo simulation.

Chapter 6: Results

This section provides the results gathered from each optimization model for each scenario. Graphs of results and the result of a one way ANNOVA test [46] are provided for each experiment. A reader may refer to Appendix L: One-Way ANOVA Test to review the statistical method used in the one way ANNOVA test.

The only factor affecting the difference in the results generated from solution of all optimization models is the routing polices (CO, PI, and HY) used. Routing systems are applied on identical data ensuring that no other source of variation effects the quality of the results. As there is only one factor affecting the results gathered, one can conclude that we have a *one factor* statistical problem [46]. The one way ANNOVA test is sufficient to build the confidence intervals around the mean values of results generated in a one factor problem.

Minitab™ 16 is used to perform the ANNOVA test and graph the confidence intervals (CI) around the mean values of the results.

6.1 Model 1 Results – Counting Packing/ Unpacking Instances

Model 1 was used to calculate the minimum number of containers trips from one node to any other node. As each container needs one loading and one unloading per trip, the total number of containers in the system also refers to the total number of container packing and unpacking (i.e. loading and unloading) instances. Investigating the number of packing and unpacking is important as it captures the resources required in material handling. The logistic system which results in lower number of packing and unpacking is superior to other systems for this criteria. The following six sections present the results generated for Criteria 1.

6.1.1 Low Traffic Experiment

In the low traffic experiment, CO resulted in the smallest number of packing and unpacking instances. PI logistic has the largest number of packing and unpacking instances. As expected, the number of packing and unpacking instances in HY is in between. Figure 20 shows the number of packing and unpacking in CO, HY, and PI logistics in this experiment.

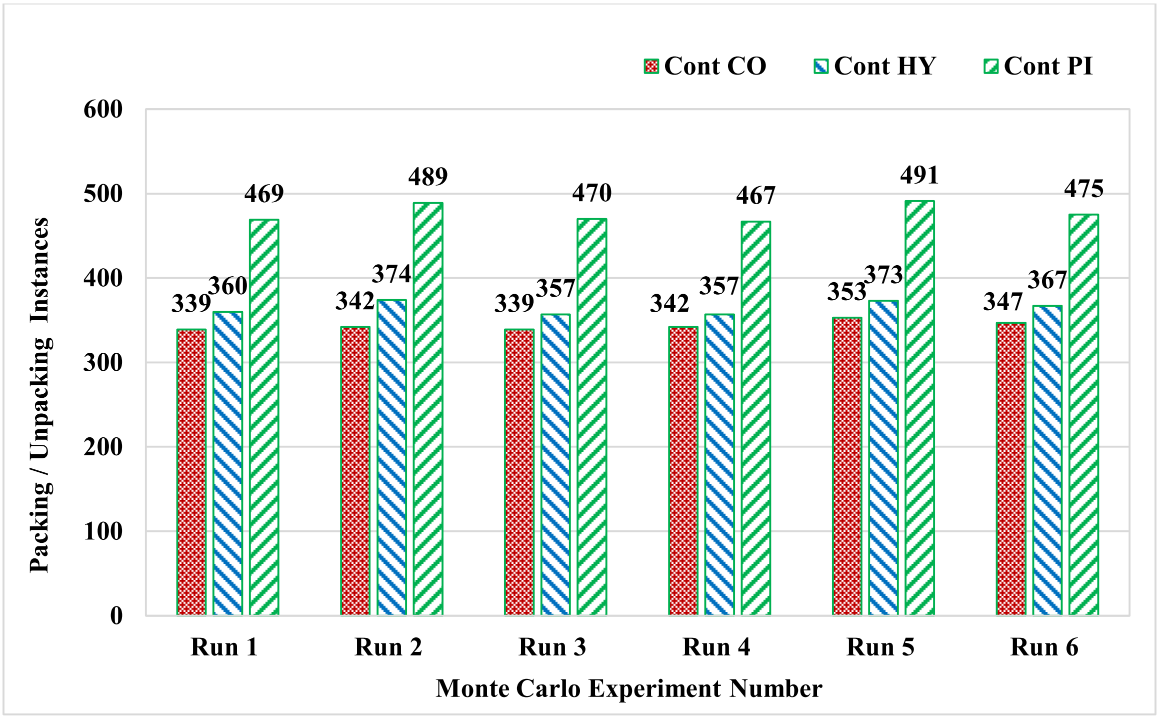

Figure 20 - Packing / Unpacking Instance in Low Traffic Network

Statistical Analysis

The ANNOVA result shows that the generated 95% CIs do not overlap. This means that statistically, the three logistic systems perform differently in terms of number of packing and unpacking required. CO performers better than HY, and HY performs better than PI in the low traffic experiment.

```
One-way ANOVA: Cont CO, Cont HY, Cont PI

Source  DF       SS       MS       F      P
Factor   2  61511.4  30755.7  458.81  0.000
Error   15   1005.5     67.0
Total   17  62516.9

S = 8.187   R-Sq = 98.39%   R-Sq(adj) = 98.18%
                           Individual 95% CIs For Mean Based on
                           Pooled StDev
Level     N    Mean  StDev  ------+---------+---------+---------+---
Cont CO   6  343.67   5.43  (-*-)
Cont HY   6  364.67   7.76       (-*-)
Cont PI   6  476.83  10.55                               (-*-)
                           ------+---------+---------+---------+---
                                360       400       440       480
Pooled StDev = 8.19
```

6.1.2 Moderate Traffic Experiment

In the moderate traffic experiment, CO performed best followed by HY and PI. Figure 21 shows the results for this experiment.

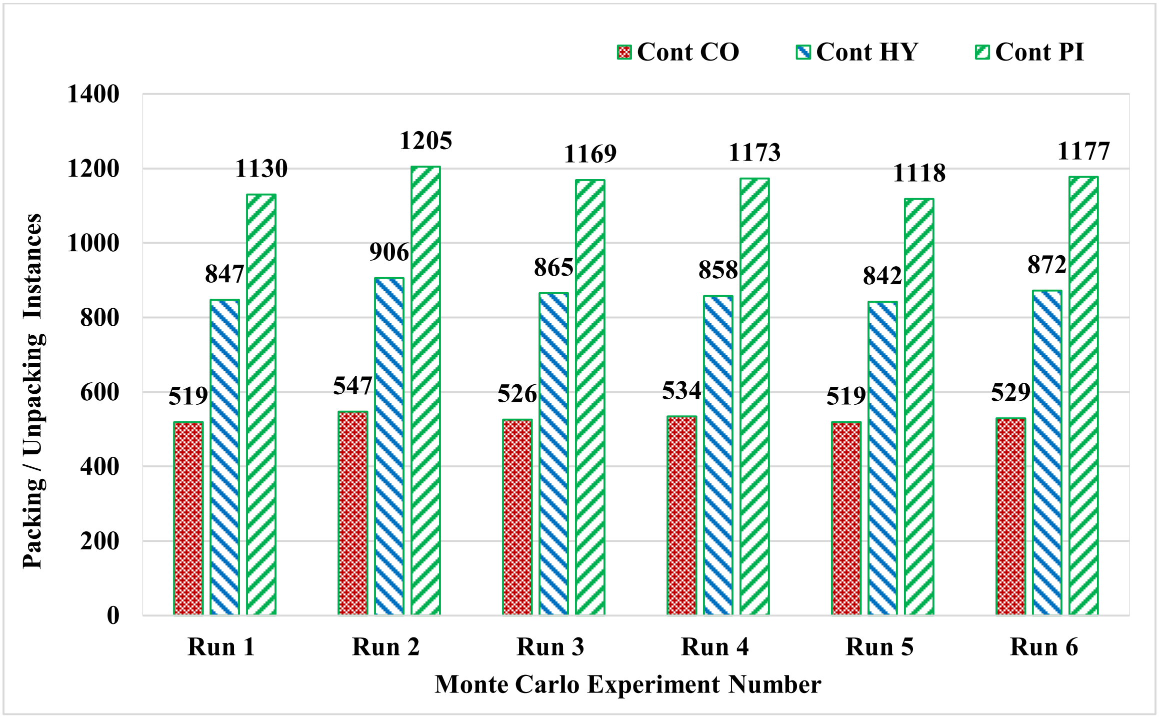

Figure 21 - Packing / Unpacking Instance in Moderate Traffic Network

Statistical Analysis

The ANNOVA result shows that the generated 95% CIs do not overlap implying that the three logistic systems perform differently in terms of the number of packing and unpacking instances required.

```
One-way ANOVA: Cont CO, Cont HY, Cont PI

Source  DF       SS      MS        F      P
Factor   2  1203588  601794  1075.40  0.000
Error   15     8394     560
Total   17  1211982

S = 23.66   R-Sq = 99.31%   R-Sq(adj) = 99.22%
                           Individual 95% CIs For Mean Based on
                           Pooled StDev
Level     N    Mean  StDev  -----+---------+---------+---------+----
Cont CO   6   529.0   10.6  (*)
Cont HY   6   865.0   22.9                  (*)
Cont PI   6  1162.0   32.3                                (*)
                            -----+---------+---------+---------+----
```

```
Pooled StDev = 23.7
```

6.1.3 High Traffic Experiment

In the high traffic experiment, CO performed best followed by HY and PI. Figure 22, shows the number of packing and unpacking in CO, HY, and PI logistics in this experiment.

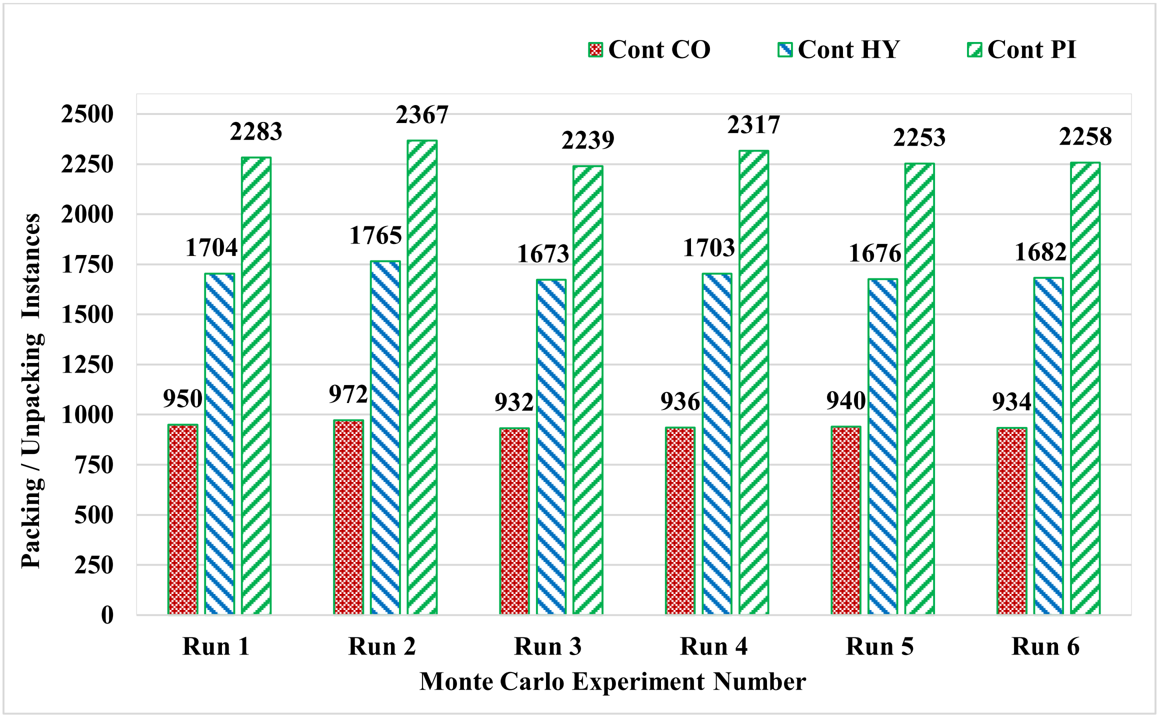

Figure 22 - Packing / Unpacking Instance in High Traffic Network

Statistical Analysis

The ANNOVA result shows that the generated 95% CIs do not overlap once again and the results are statistically significant.

```
One-way ANOVA: Cont CO, Cont HY, Cont PI

Source   DF       SS       MS        F      P
Factor    2  5433418  2716709  2186.86  0.000
Error    15    18634     1242
Total    17  5452052

S = 35.25   R-Sq = 99.66%   R-Sq(adj) = 99.61%

                                Individual 95% CIs For Mean Based on
                                Pooled StDev
Level     N    Mean  StDev  -------+---------+---------+---------+--
Cont CO   6   944.0   15.1  (*
Cont HY   6  1700.5   34.3                      (*
Cont PI   6  2286.2   48.2                                   (*)
```

```
                              -------+---------+---------+---------+--
                                  1200      1600      2000      2400
```

```
Pooled StDev = 35.2
```

6.1.4 High Traffic Small Loads Experiment

In the High Traffic Small Loads Experiment, ten loads of random sizes from each node to every other node are packed into containers. The load size probability distribution and the total number of loads in the experiment was shown in section 5.2.2.

CO resulted in the smallest number of packing and unpacking instances. PI logistic has the largest number of packing and unpacking. Number of packing and unpacking in HY logistics was between the results for CO and PI. Figure 23, shows the number of packing and unpacking in CO, HY, and PI logistics in this experiment.

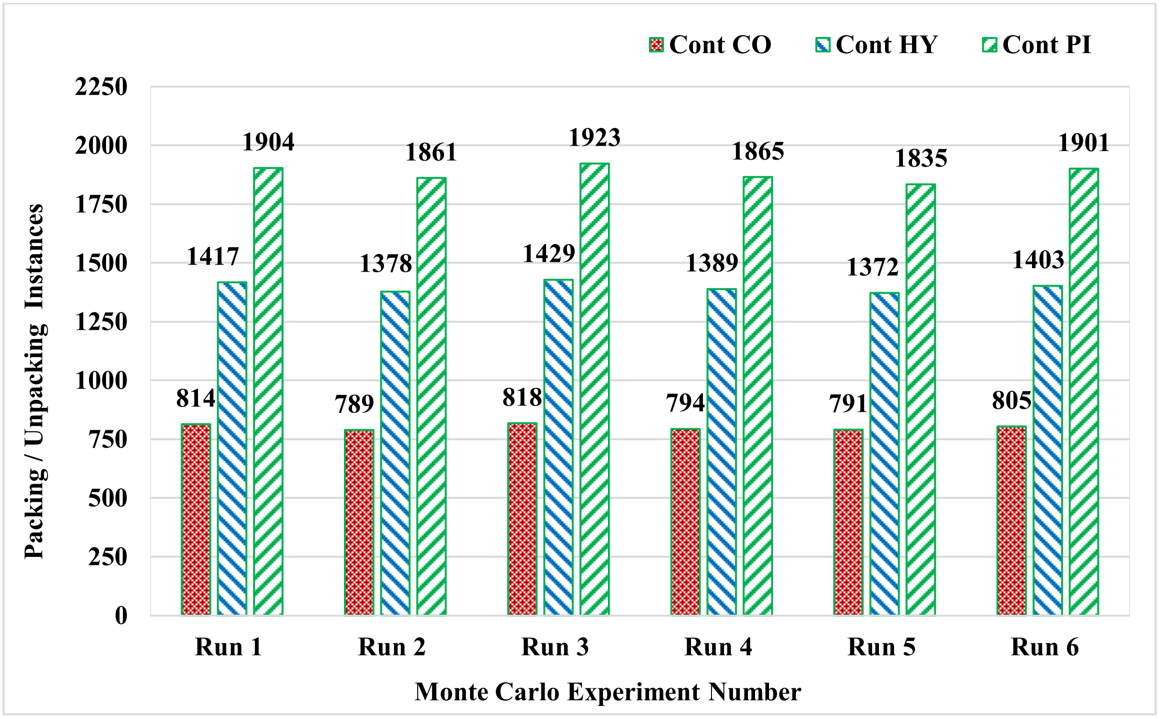

Figure 23 - Packing / Unpacking Instance in High Traffic Small Loads Network

Statistical Analysis

The ANNOVA result shows that the generated 95% CI do not overlap. This means that statistically speaking, the three logistic systems perform differently in terms of number of packing and unpacking required. CO performs better than HY, and HY performs better than PI in

High Traffic Small Loads Experiment. The following is the result of ANNOVA test generated in Minitab.

```
One-way ANOVA: Cont CO, Cont HY, Cont PI

Source   DF       SS       MS        F      P
Factor    2  3509734  1754867  3015.12  0.000
Error    15     8730      582
Total    17  3518464

S = 24.13   R-Sq = 99.75%   R-Sq(adj) = 99.72%

                               Individual 95% CIs For Mean Based on
                               Pooled StDev
Level     N    Mean  StDev  ----+---------+---------+---------+-----
Cont CO   6   801.8   12.4  (*
Cont HY   6  1398.0   22.4                      (*
Cont PI   6  1881.5   33.1                                     (*
                            ----+---------+---------+---------+-----
                              900      1200      1500      1800
```

6.1.5 High Traffic Large Loads Experiment

The results for the High Traffic Large Load Experiment follow the same pattern as for the High Traffic Small Loads Experiment. Figure 24, shows the number of packing and unpacking in CO, HY, and PI logistics in this experiment.

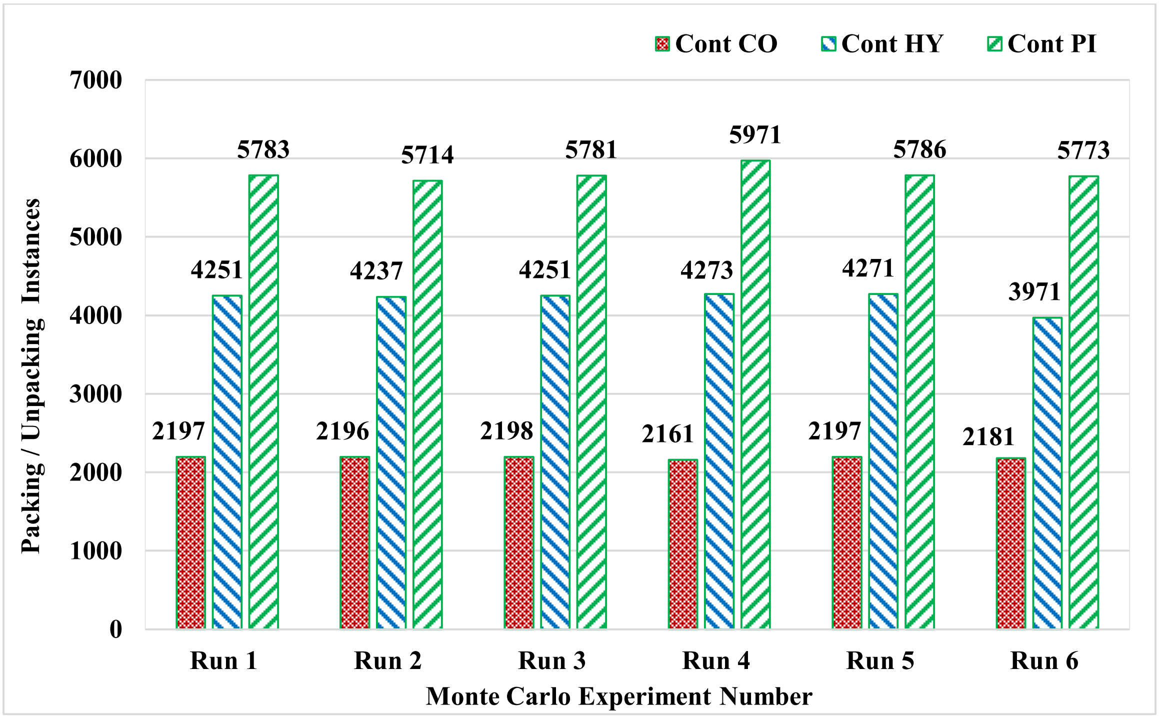

Figure 24 - Packing / Unpacking Instance in High Traffic Large Loads Network

Statistical Analysis

The ANNOVA result shows that the generated 95% CI do not overlap. This means that statistically, the three logistic systems perform differently in terms of number of packing and unpacking required. The following is the result of ANNOVA test generated in Minitab.

```
One-way ANOVA: Cont CO, Cont HY, Cont PI

Source  DF        SS        MS        F      P
Factor   2  39344776  19672388  2727.21  0.000
Error   15    108201      7213
Total   17  39452977

S = 84.93   R-Sq = 99.73%   R-Sq(adj) = 99.69%

                           Individual 95% CIs For Mean Based on
                           Pooled StDev
Level     N    Mean  StDev  ---------+---------+---------+---------+
Cont CO   6  2188.3   14.9  (*)
Cont HY   6  4209.0  117.4                    (*)
Cont PI   6  5801.3   87.4                                    (*)
                            ---------+---------+---------+---------+
                                  3000      4000      5000      6000
Pooled StDev = 24.1
```

6.1.6 Random Traffic Experiment

In the Random Traffic Experiment, the number of loads from any node to other nodes is normally distributed with mean of ten and standard deviation of four.

In the Random Traffic Experiment, CO logistic resulted in smallest number of packing and unpacking instances. PI logistic has the highest number of packing and unpacking. Number of packing and unpacking in HY logistics was between the results in CO and PI. Figure 25, shows the number of packing and unpacking in CO, HY, and PI logistics in this experiment.

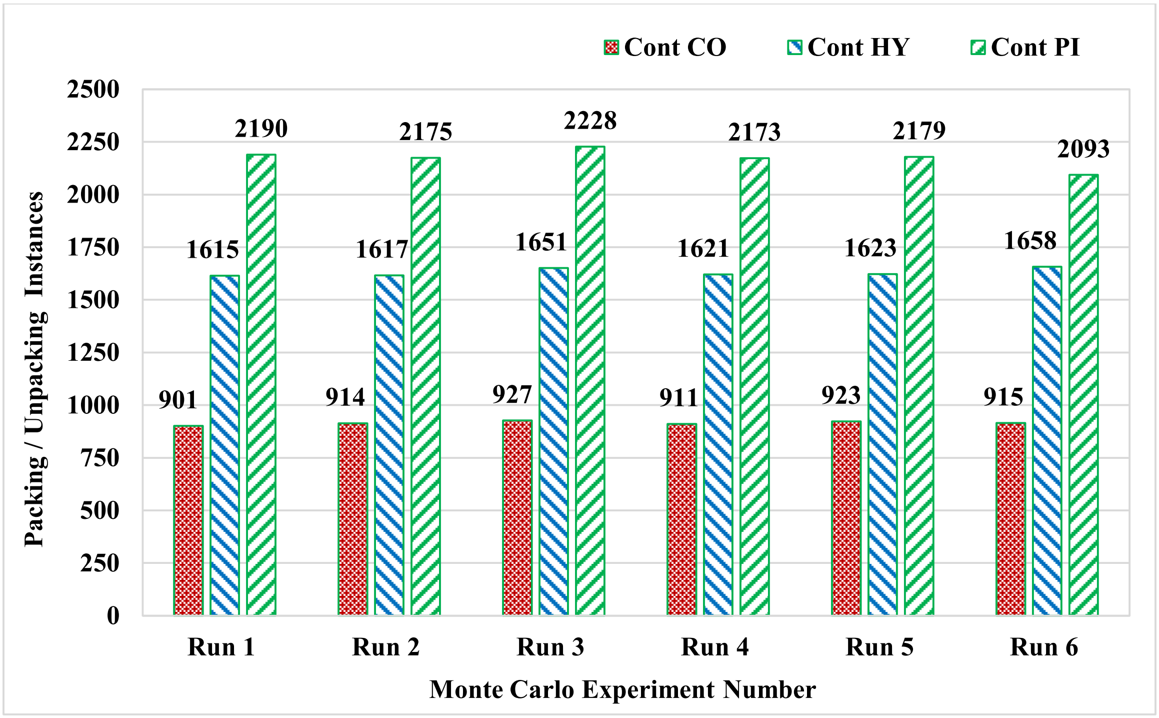

Figure 25 - Packing / Unpacking Instance in Random Traffic Network

Statistical Analysis

The ANNOVA result shows that the generated 95% CI do not overlap. This means that statistically, the three logistic systems perform differently in terms of number of packing and unpacking required. The following is the result of ANNOVA test generated in Minitab.

```
One-way ANOVA: Cont CO, Cont HY, Cont PI

Source   DF       SS       MS        F      P
Factor    2  4776536  2388268  3005.46  0.000
Error    15    11920      795
Total    17  4788456

S = 28.19   R-Sq = 99.75%   R-Sq(adj) = 99.72%

                               Individual 95% CIs For Mean Based on
                               Pooled StDev
Level     N    Mean  StDev  -----+---------+---------+---------+----
Cont CO   6   915.2    9.2  (*)
Cont HY   6  1630.8   18.7                      (*
Cont PI   6  2173.0   44.2                                    (*)
                            -----+---------+---------+---------+----
                              1050      1400      1750      2100

Pooled StDev = 28.2
```

6.2 Model 2 Results - Route Driving Time Criteria

Model 2 was used to calculate the minimum driving time of routes required to transport all the containers from their sources to their destinations. Minimizing the total duration of routes is important as it has direct effect on transportation cost. The amount of CO_2 emission due to transportation is also correlated with the driving time of the routes.

The logistic system with the lowest route duration is superior to the other systems for this criteria. The following six sections present the results generated from the Model 2.

6.2.1 Low Traffic Network

The routing result of the Low Traffic Experiment shows a clear gap between PI and HY compared to CO. The PI and HY results are very close to each other. These results indicate that PI and HY are more efficient in terms of driving time required. However, the gap between PI and HY is very small. Figure 26 shows the total routing time in hours for each of the logistic systems.

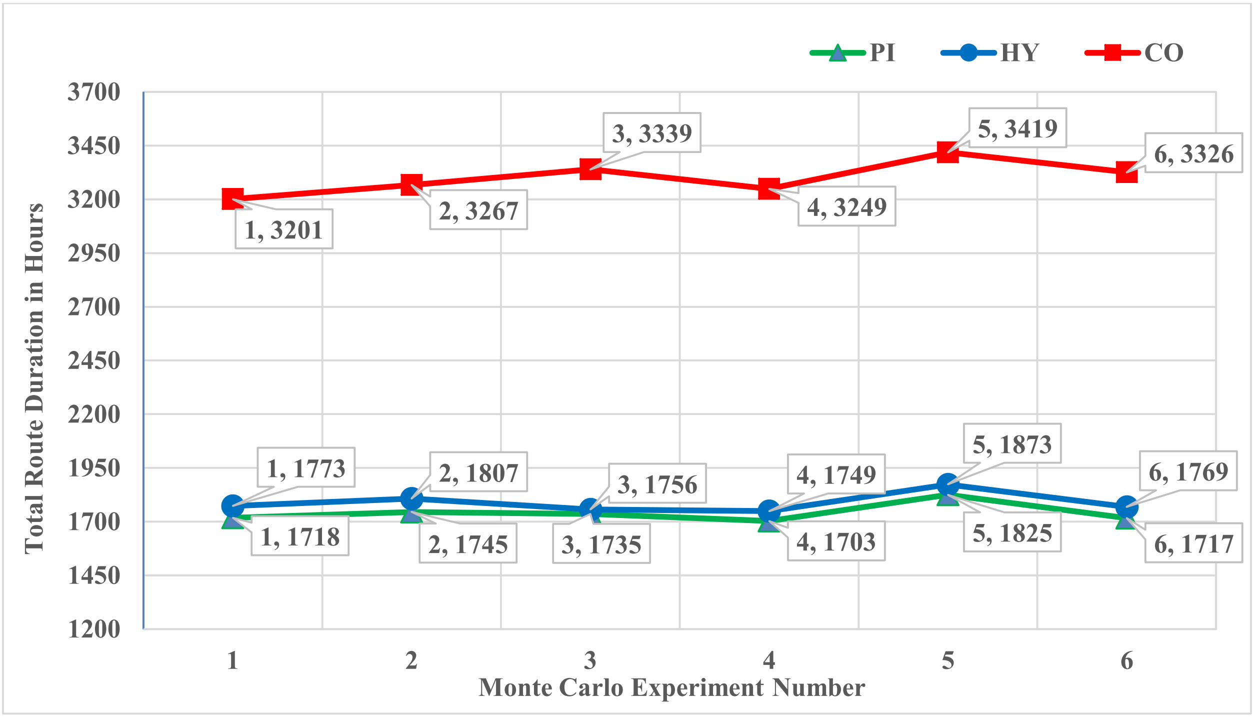

Figure 26 - Optimal Routing Time in Low Traffic Network

Statistical Analysis

Result of the ANNOVA test indicates that PI and HY are not statistically different for this criteria, as the 95% Confidence Interval (CI) of systems overlap, although the mean value of PI is slightly less than the mean value of HY. On the other hand, the CI of both PI and HY are significantly different from CO. The following is the result of ANNOVA test generated in Minitab.

```
One-way ANOVA: Hours PI, Hours HY, Hours CO

Source  DF       SS       MS        F      P
Factor   2  9442174  4721087  1406.28  0.000
Error   15    50357     3357
Total   17  9492531

S = 57.94   R-Sq = 99.47%   R-Sq(adj) = 99.40%

                            Individual 95% CIs For Mean Based on
                            Pooled StDev
Level      N    Mean  StDev  ------+---------+---------+---------+---
Hours PI   6  1740.6   43.9  (*)
Hours HY   6  1788.0   46.3   (*)
Hours CO   6  3300.2   77.4                                (*)
                            ------+---------+---------+---------+---
                               2000      2500      3000      3500
Pooled StDev = 57.9
```

6.2.2 Moderate Traffic Experiment

The routing result of the Moderate Traffic Experiment follows a similar pattern. Figure 27 shows the total routing time in hours for each of the logistic systems.

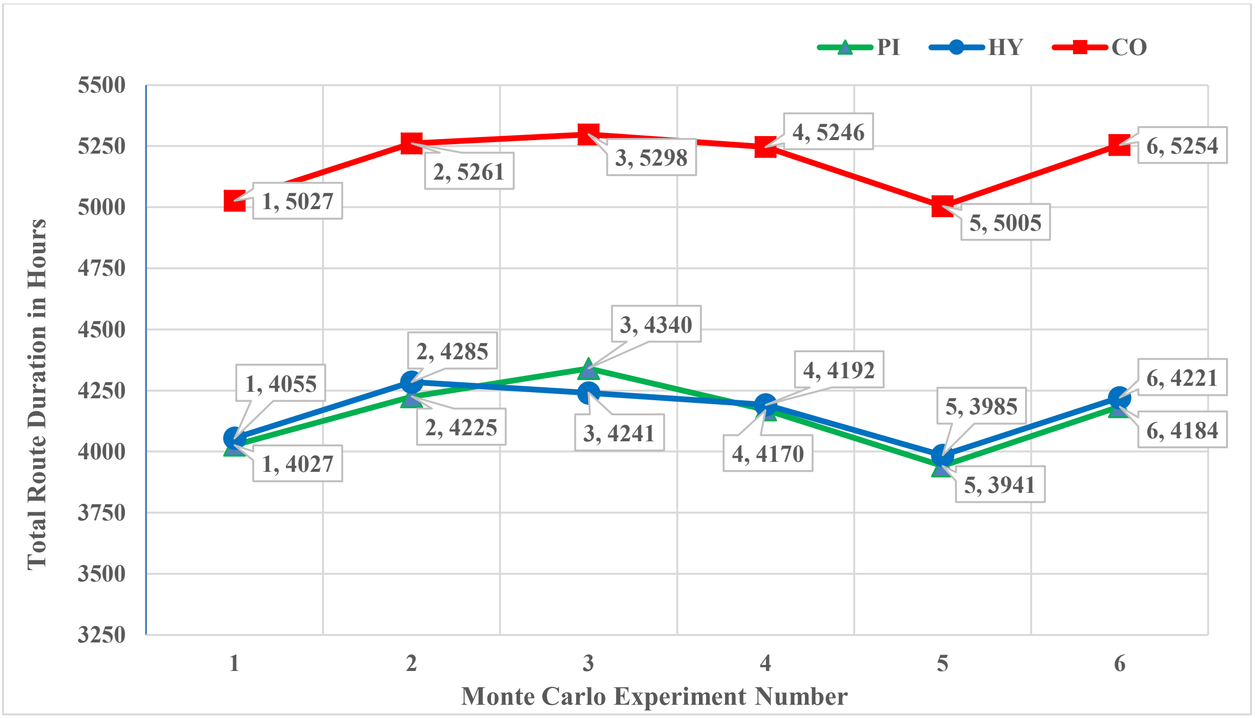

Figure 27 - Optimal Routing Time in Moderate Traffic Network

Statistical Analysis

Result of the ANNOVA test indicates that PI and HY are not statistically different as the 95% CI of systems overlap. PI and HY are statistically different from CO. The following is the result of ANNOVA test generated in Minitab.

```
One-way ANOVA: Hours PI, Hours HY, Hours CO

Source  DF       SS       MS       F      P
Factor   2  4213616  2106808  123.81  0.000
Error   15   255253    17017
Total   17  4468869

S = 130.4   R-Sq = 94.29%   R-Sq(adj) = 93.53%

                            Individual 95% CIs For Mean Based on
                            Pooled StDev
Level      N    Mean  StDev  -----+---------+---------+---------+----
Hours PI   6  4147.9  142.9  (---*--)
Hours HY   6  4163.2  117.2   (--*--)
Hours CO   6  5181.8  130.0                               (--*--)
                            -----+---------+---------+---------+----
                              4200      4550      4900      5250

Pooled StDev = 130.4
```

6.2.3 High Traffic Experiment

The routing result of the High Traffic Experiment follows a similar pattern. Figure 28 shows the total routing time in hours for each of the logistic systems.

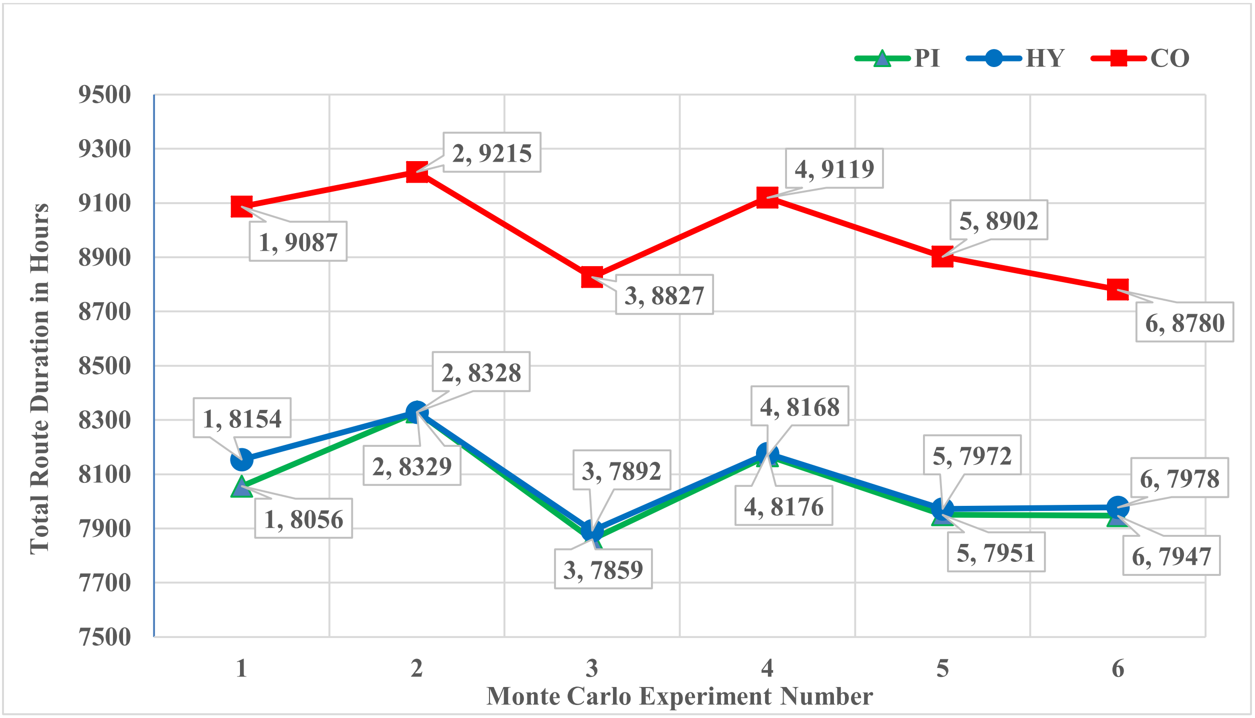

Figure 28 - Optimal Routing Time in High Traffic Network

Statistical Analysis

Result of the ANNOVA test indicates that PI and HY are not statistically different as the 95% CI of systems overlap. PI and HY are statistically different from CO. The following is the result of ANNOVA test generated in Minitab.

```
One-way ANOVA: Hours PI, Hours HY, Hours CO

Source  DF       SS       MS      F      P
Factor   2  3395088  1697544  58.23  0.000
Error   15   437319    29155
Total   17  3832407

S = 170.7   R-Sq = 88.59%   R-Sq(adj) = 87.07%

                            Individual 95% CIs For Mean Based on
                            Pooled StDev
Level     N    Mean  StDev  ----+---------+---------+---------+-----
Hours PI  6  8051.7  172.4  (---*---)
Hours HY  6  8083.3  163.5   (---*---)
Hours CO  6  8988.3  176.1                           (---*---)
                            ----+---------+---------+---------+-----
```

```
                             8050      8400      8750      9100
Pooled StDev = 170.7
```

6.2.4 High Traffic Small Loads Experiment

The routing result of the High Traffic Small Loads Experiment follows a similar pattern. Figure 29 shows the total routing time in hours for each of the logistic systems.

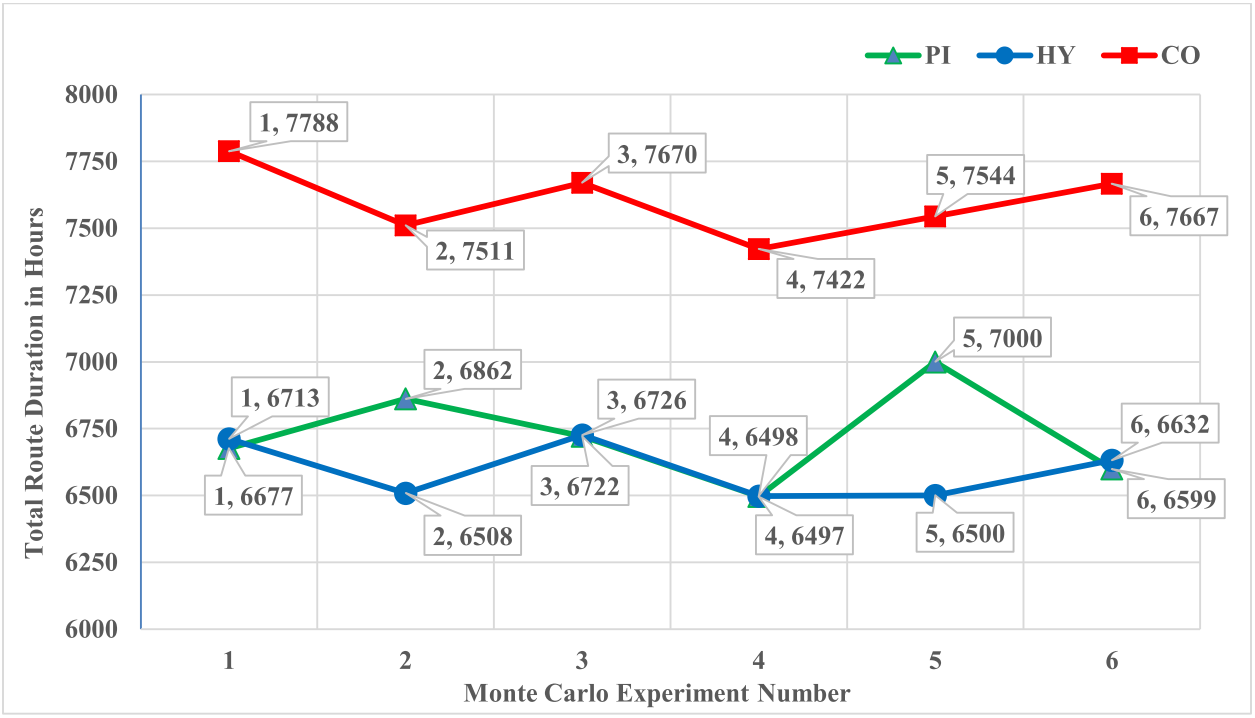

Figure 29 - Optimal Routing Time in High Traffic Small Loads Network

Statistical Analysis

Result of the ANNOVA test indicates that PI and HY are not statistically different as the 95% CI of systems overlap. PI and HY are statistically different from CO. The following is the result of ANNOVA test generated in Minitab.

```
One-way ANOVA: Hours PI, Hours HY, Hours CO

Source  DF       SS      MS      F      P
Factor   2  1474647  737324  67.88  0.000
Error   15   162943   10863
Total   17  1637591

S = 104.2   R-Sq = 90.05%   R-Sq(adj) = 88.72%

                            Individual 95% CIs For Mean Based on Pooled StDev
Level      N    Mean  StDev    -+---------+---------+---------+--------
Hours PI   6  5308.3  101.0    (--*---)
Hours HY   6  5355.8   97.9       (--*---)
Hours CO   6  5937.8  113.2                               (---*--)
                                 -+---------+---------+---------+--------
                               5250      5500      5750      6000

Pooled StDev = 104.2
```

6.2.5 High Traffic Large Loads Experiment

The routing result of the High Traffic Large Loads Experiment follows a similar pattern. Figure 30 shows the total routing time in hours for each of the logistic systems.

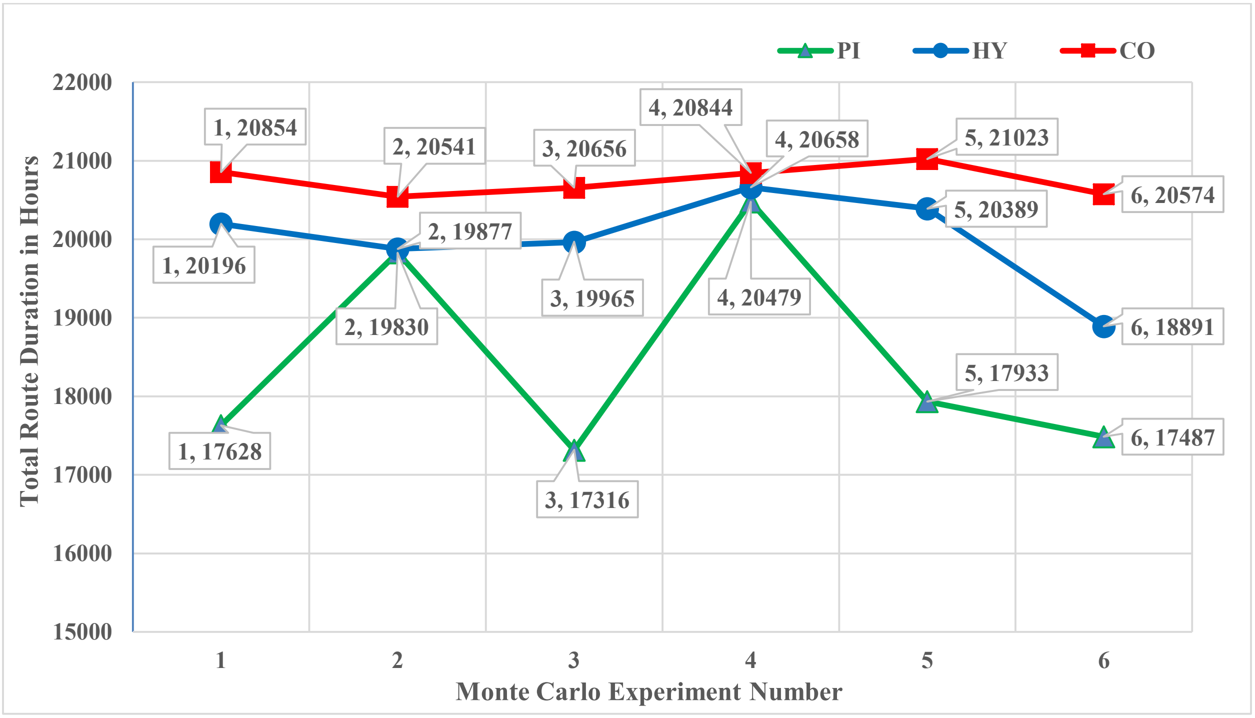

Figure 30 - Optimal Routing Time in High Traffic Large Loads Network

Statistical Analysis

Result of the ANNOVA test indicates that PI and HY are not statistically different as the 95% CI of systems overlap. PI and HY are statistically different from CO. The following is the result of ANNOVA test generated in Minitab.

```
One-way ANOVA: Hours PI, Hours HY, Hours CO

Source  DF        SS       MS      F      P
Factor   2  16552020  8276010  11.06  0.001
Error   15  11225698   748380
Total   17  27777718

S = 865.1   R-Sq = 59.59%   R-Sq(adj) = 54.20%
```

```
                                   Individual 95% CIs For Mean Based on
                                   Pooled StDev
Level       N    Mean    StDev    ---+---------+---------+---------+------
Hours PI    6   18445    1355     (------*-------)
Hours HY    6   19996     612                    (-------*------)
Hours CO    6   20749     189                              (------*-------)
                                  ---+---------+---------+---------+------
                                 18000     19000     20000     21000

Pooled StDev = 865
```

6.2.6 Random Traffic Experiment

The routing result of the Random Traffic Experiment shows approximately equal gaps in the means for PI, HY, and CO. Figure 31 shows the total routing time in hours for each of the logistic systems.

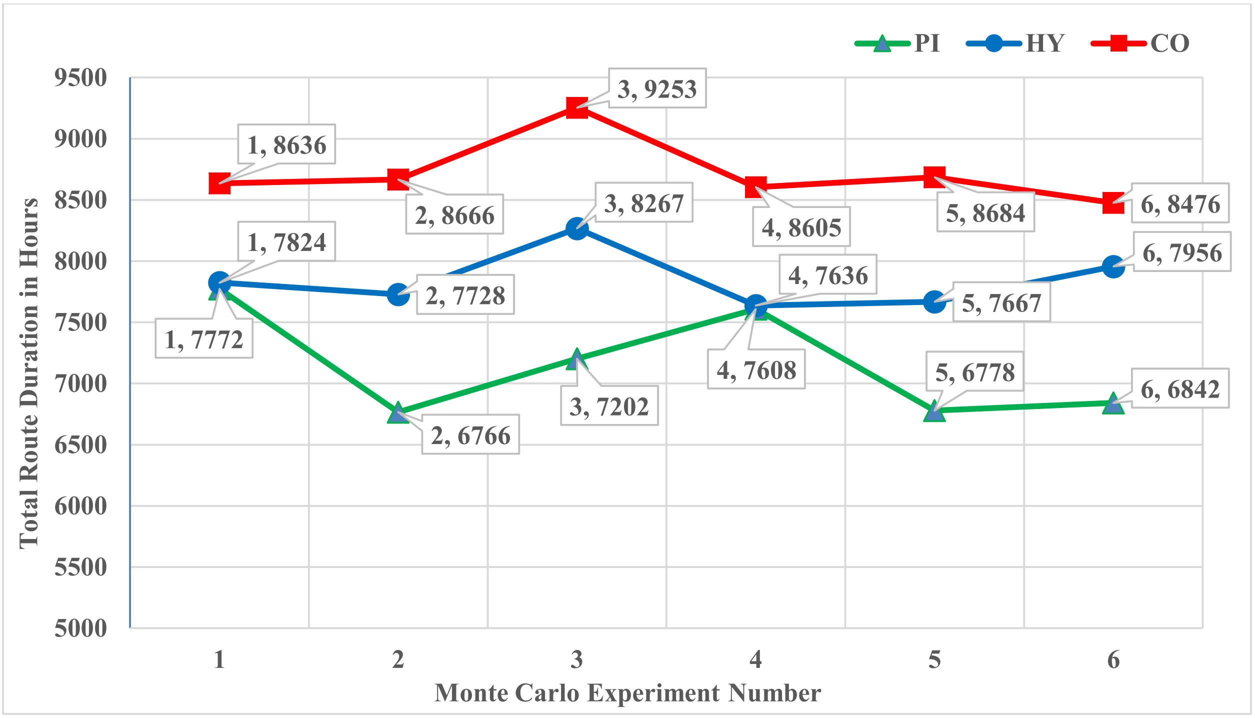

Figure 31 - Optimal Routing Time in Random Traffic Network

Statistical Analysis

The result of the ANNOVA test indicates that all the logistic systems are statistically different as none of the 95% CI of the systems overlap each other. The mean value for PI logistic is less than the mean value of the HY. Therefore, PI is the most efficient logistic policy in a random traffic network. The second most efficient policy is HY with a lower mean value compared to CO.

```
One-way ANOVA: Hours PI, Hours HY, Hours CO

Source  DF       SS       MS      F      P
Factor   2  7324606  3662303  33.76  0.000
Error   15  1627129   108475
Total   17  8951735

S = 329.4   R-Sq = 81.82%   R-Sq(adj) = 79.40%

                            Individual 95% CIs For Mean Based on
                            Pooled StDev
Level     N    Mean  StDev  -----+---------+---------+---------+----
Hours PI  6  7161.2  442.6  (---*----)
Hours HY  6  7846.3  236.6             (----*----)
Hours CO  6  8719.9  271.2                            (---*----)
                            -----+---------+---------+---------+----
                              7200      7800      8400      9000

Pooled StDev = 329.4
```

6.3 Model 2: Driving Time Trends

The six experiments introduced in the previous sections creates two general trends for analysis.

6.3.1 Traffic Level Trend

In the Traffic Level Trend Analysis, the number of loads generated for each source, destination pair is increased, but the probability distribution of the loads selected for simulation stays unchanged. In other word, traffic trend only investigated the effect of increasing traffic level on the overall routing performance of the individual logistic systems.

Side by side comparison of the results generated from Model 2 indicates that the highest gap between the average performance of the logistic systems is in a low traffic network. This gap decreases as the traffic level increases. When traffic level increases, regardless of the routing system used, the likelihood of full or close to full container movements greatly increases. Therefore, in a scenario in which, every single container sent from a source to a destination is completely full, the routing performance gap among these logistic systems is minimized. Table 4, provides a summary of the logistic systems routing times under different traffic levels.

Table 4 - Driving Time of Logistic Systems with Different Traffic Level

	Total Driving Time Low Traffic			**Total Driving Time Moderate Traffic**			**Total Driving Time High Traffic**		
Exp	PI	HY	CO	PI	HY	CO	PI	HY	CO
1	1718	1773	3201	4027	4055	5027	8056	8154	9087
2	1745	1807	3267	4225	4285	5261	8329	8328	9215
3	1735	1756	3339	4340	4241	5298	7859	7892	8827
4	1703	1749	3249	4170	4192	5246	8168	8176	9119
5	1825	1873	3419	3941	3985	5005	7951	7972	8902
6	1717	1769	3326	4184	4221	5254	7947	7978	8780
Avg	**1740.6**	**1788.0**	**3300.2**	**4147.9**	**4163.2**	**5181.8**	**8051.7**	**8083.3**	**8988.3**
Diff	**Base**	**+2.72%**	**+89.60%**	**Base**	**+0.37%**	**+24.93%**	**Base**	**+0.39%**	**+11.63%**

6.3.2 Model 2: Load Size Trend

In the Load Size Trend Analysis, the number of loads generated for each source, destination pair remains unchanged, but the probability distribution of the loads selected for simulation changes according to the discrete distributions described in section 5.2.

The highest gap in the routing performance of the logistic systems occurs when large loads are transported over the network. The size of this gap decreases as size of the loads decrease. When large loads are transported, utilizing different logistic systems results in large differences in routing time. It is believed that when loads are large, the packing efficiency of CO, for example, compared to PI may not be very good. Therefore, a large gap is observed. However, in the case of smaller average loads, the packing efficiency of CO improves and therefore, the performance gap between CO and PI decreases.

Table 5 provides a summary of the logistic systems routing times with different load size distributions.

Table 5 - Driving Time of Logistic Systems with Different Load Size Distributions

	Total Driving Time High Traffic Large Loads			Total Driving Time High Traffic Normal Loads			Total Driving Time High Traffic Small Loads		
Exp	PI	HY	CO	PI	HY	CO	PI	HY	CO
1	14268	16370	17428	6540	6653	7575	6677	6713	7788
2	15966	16074	17278	6766	6769	7561	6862	6508	7511
3	13949	16128	17349	6366	6437	7328	6722	6726	7670
4	17924	17215	17284	6630	6645	7611	6497	6498	7422
5	14592	16579	17805	6453	6525	7366	7322	6500	7544
6	20315	22565	24676	6464	6527	7099	6599	6632	7667
Avg	**16169.2**	**17488.5**	**18636.8**	**6536.5**	**6592.6**	**7423.3**	**6779.7**	**6596.2**	**7600.4**
Diff	**Base**	**+8.16%**	**+15.26%**	**Base**	**+0.86%**	**+13.57%**	**Base**	**-2.71%**	**+12.11%**

The random traffic experiment was performed to test the routing performance of each logistic scenario with varying traffic levels in the network. PI resulted in the lowest driving time once again, followed by HY, and CO. Table 5 provides a summary of driving times in the three logistic systems for the Random Traffic Network.

Table 6 - Driving Time of Logistic Systems with Random Traffic

	Random Traffic Experiment		
Exp	PI	HY	CO
1	7772	7824	8636
2	6766	7728	8666
3	7202	8267	9253
4	7608	7636	8605
5	6778	7667	8684
6	6842	7956	8476
Avg	**7161**	**7846**	**8720**
Diff	**Base**	**+9.57%**	**+21.77%**

Figure 32 shows the driving time of all logistic systems across all the experiments performed in this thesis.

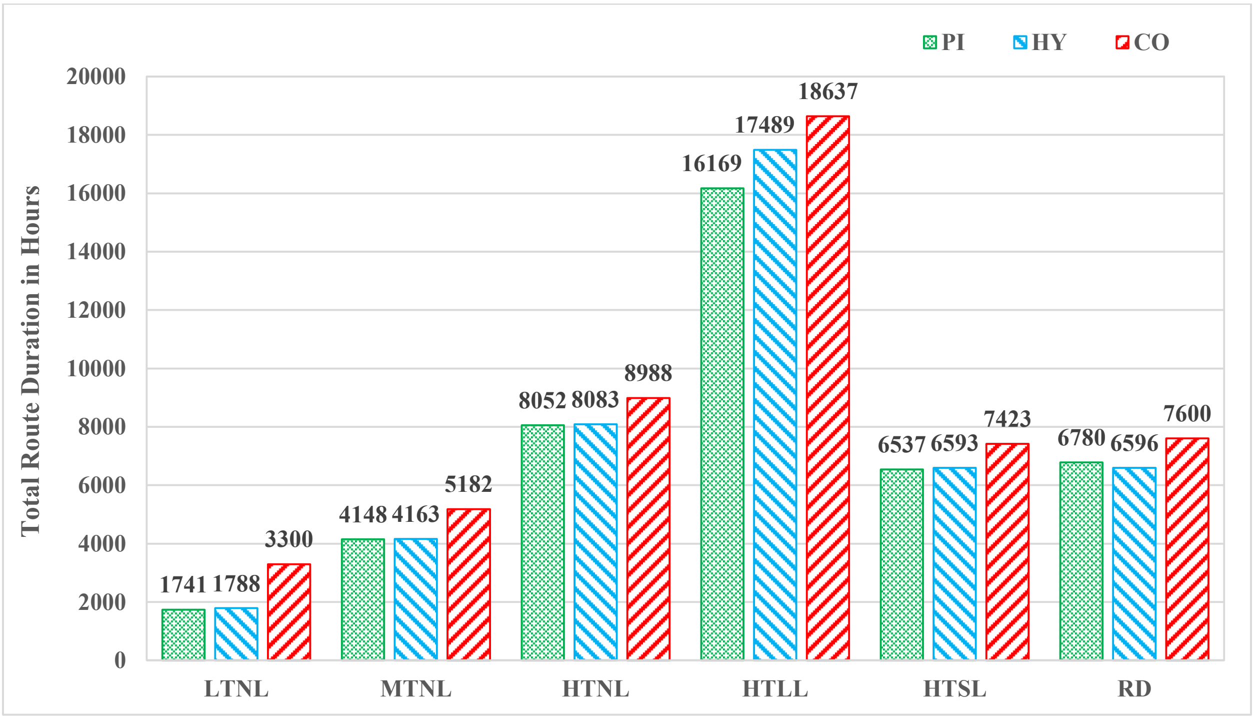

Figure 32 - Overall Comparison of Routing Time of Logistic Systems

The average difference in driving time across all six experiments is shown in Table 7. As can be seen, CO has a gap of approximately 27% in driving hours compared to PI. It can be seen that the performance of HY is much closer to PI.

Table 7 - Percentage Difference in Driving Time of HY and CO with Respect to PI

PI	HY	CO
Base	**+2.77%**	**+26.98%**

6.4 Model 3 Results

The numerical values for the rest of the KPIs in this thesis are calculated after phase three of the optimization framework. The results are presented in the next two sections. The KPI values are shown and discussed first, and then a method to calculate the total cost of the logistic systems is proposed.

6.4.1 Values of the Key Performance Indicators

The number of trucks in service, average hours worked per truck, percentage of drivers back home at the end of the day, and the total hours of operation are the rest of the KPIs calculated for the three systems.

PI needs the largest number of trucks for service. Most of the jobs created in PI policy are short jobs and can be completed in one day. Therefore, a large number of short haul trucks enter into service. The average work time per truck is also low in PI. Percentage of drivers that can return home at the end of the day is also highest in PI (it should be noted that not all the drivers can go back home at the end of the day as a number of routes have a duration of longer than 14 hours). Table 8, and Table 9 list the numerical values of the KPIs. The effect of increase in traffic level on the KPIs is shown in Table 8 and the effect of changing the load size distribution on the KPIs is shown in Table 9.

Table 8 - Logistic Systems KPI values with Difference Traffic Levels

	Low Traffic			**Moderate Traffic**			**High Traffic**		
Policy	PI	HY	CO	PI	HY	CO	PI	HY	CO
Trucks in Service	102	64	64	183	136	92	528	371	227
Long Haul	6	21	52	19	41	75	11	86	140
Short Haul	96	43	12	164	95	17	517	285	87
Total Enroute Hours	1,434.6	1,797.2	3,510.7	3,205.0	3,830.2	5,051.5	6,661.8	7,948.8	10,206.2
Total Driving Hours	1,402.6	1,469.2	2,743.2	3,191.1	3,256.6	4,212.1	6,525.8	6,636.8	7,574.2
Avg Hours on Duty	14.06	28.08	54.85	17.51	28.16	54.91	12.62	21.43	44.96
Avg Hours Driving	13.75	22.96	42.86	17.44	23.95	45.78	12.36	17.89	33.37
% Drivers Back Home	94.12%	67.19%	18.75%	89.62%	69.85%	18.48%	97.92%	76.82%	38.33%

Table 9 - Logistic Systems KPI values with Different Load Size Distributions

	High Traffic, Small Loads			Random Traffic			High Traffic, Large Loads		
Policy	PI	HY	CO	PI	HY	CO	PI	HY	CO
Trucks in Service	413	224	168	428	322	210	529	477	429
Long Haul	20	66	116	18	86	138	153	242	233
Short Haul	393	158	52	410	236	72	376	235	196
Total Enroute Hours	5,579.1	6,375.7	8,487.7	5,682.7	7,715.5	9,877.8	14,837	18,916	22,596
Total Driving Hours	5,443.1	5,319.7	6,319.7	5,506.7	6,339.5	7,269.8	14,637	15,556	16,868
Avg Hours on Duty	13.51	28.46	50.52	13.28	23.96	47.04	28.05	39.66	52.67
Avg Hours Driving	13.18	23.75	37.62	12.87	19.69	34.62	27.67	32.61	39.32
% Drivers Back Home	95.16%	70.54%	30.95%	95.79%	73.29%	34.29%	71.08%	49.27%	45.69%

The number of drivers who can go back home at the end of a work day remains consistently high in PI, regardless of the traffic level.

In CO, when the traffic level is low, Model 3 may find it efficient to assign some of the short jobs to long haul trucks because if long haul truck is needed anyway for a job, it can be used to serve other short haul jobs. Therefore, for lower traffic levels, the number of the drivers who can go back home in CO is low. However as the network gets busier and long haul trucks get enough work, the shorter jobs are done mostly with short haul trucks and therefore, the number of drivers who go back home at the end of a work day increases.

Change in load size values does not seem to create a visible change in the pattern of KPI values. As shown in Table 9, the PI again remains the best logistics policy in terms of a truck driver's social life. This is followed by HY and CO.

6.5 Costs

The cost of the logistic systems are calculated by considering the results generated from Model 1 and Model 3. Model 2 has no direct effect on the costs:

$$Total\ Cost = (Material\ Handling, from\ Model\ 1) + (Operational\ Cost + Social\ Cost + Fixed\ Cost, all\ from\ Model\ 3)$$

While all cost components from Model 3 have been already discussed, the Material Handling cost is incurred within the PI transit centers. It is assumed that the measurement unit for Material Handling cost is in dollars per container. Although there are a number of functional designs and detailed simulations of PI transit centers in [12], [13], [31], it is difficult to infer the realistic material handling cost incurred inside these type of facilities.

Therefore, a cost ratio analysis is performed to calculate the total logistic policy cost. The cost ratio is calculated as follows:

$$Cost\ Ratio\ = \frac{Material\ Handling\ Cost}{Operational\ \ Cost}$$

Since the main difference between PI, and CO is the trade-off between the level of material handling and driving activities, this can be quantified using the cost ratio.

Current Material Handling Costs

Many logistic companies charge a flat rate fee to pack and unpack containers regardless of the weight and size of the load inside the container. However, material handling cost differs according to the handling process used. For example, on pallet movement of load from a 40 foot container to another container of the same size, called a "*Pallet to Pallet*" movement, costs approximately $150 for a 40-foot container. The "*Pallet to Floor*" process removes skids from the incoming load and puts the loads in the floor of the outgoing container. Finally, "*Floor to Floor*" process is movement of loads from the floor of incoming containers to the floor of the outgoing containers. The *Pallet to Floor* process costs $250, and the *Floor to Floor* movement process costs $350.

In this thesis, \$150 is used as the packing/ unpacking cost of a PI container. This is a conservative estimate or the worst case scenario cost of the actual material handling cost that could be incurred in future PI transit centers for the following two reasons:

1) Conventional containers are loaded from the rear, meaning only one or two forklifts can pack a container at once due to the limited working space. Hence the loading and unloading process of conventional containers capture the warehouse resources for long periods of time. Also load placed in a container are accessible in the first in last out order. Therefore accessing the load placed into the container first requires removal of the entire load. Whereas PI containers can be loaded from the sides making loading and unloading process very quick. Also, removal of a specific PI load from a container does not require movement of other loads.
2) Future innovations in material handling processes can potentially reduce the cost of container loading/ unloading in transit centers. Automated material handling equipment such as PI-conveyer, PI-sorters, and PI-composers [6] will be used in PI-transit centers, which could result in lower cost material handling processes.

Sections 6.5.1 to 6.5.5 present the results of the total cost analysis. The dashed line on the cost ratio value of 0.75 in the charts shows the point where the material handling cost is \$150.

6.5.1 Low Traffic Experiment

The summary of the results required for calculation of the total cost in the Low Traffic Experiment is shown in Table 10.

Table 10 - Social, Operational, and Fix Costs in Low Traffic Experiment

Policy	# Packing / Unpacking	Social Cost	Operational Cost	Fix Cost
PI	475	\$3.23k	\$280.5k	\$1.6k
HY	367	\$3.3k	\$293.8k	\$2.5k
CO	347	\$9.2k	\$548.6k	\$5.3k

13 instances of the total cost for each of the logistic systems were calculated using the cost ratio introduced in the previous section. Table 11 lists the total cost of each logistics system for each cost ratio.

Table 11 - Total Cost as a function of Cost Ratio in Low Traffic Experiment

Policy	Total Cost 1	Total Cost 2	Total Cost 3	Total Cost 4	Total Cost 5	Total Cost 6	Total Cost 7	Total Cost 8	Total Cost 9	Total Cost 10	Total Cost 11	Total Cost 12	Total Cost 13
PI	$328.3k	$375.8k	$470.8k	$565.8k	$660.8k	$755.8k	$850.8k	$945.8k	$1m	$1.1m	$1.2m	$2.2m	$4.1m
HY	$333.9k	$370.6k	$444k	$517.4k	$590.8k	$664.2k	$737.6k	$811k	$884.4k	$957.8k	$1m	$1.8m	$3.2m
CO	$592.6k	$627.3k	$696.7k	$766.1k	$835.5k	$904.9k	$974.3k	$1m	$1.1m	$1.2m	$1.3m	$1.9m	$3.3m
Ratio	**0.5**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**20**	**40**
Packing Cost	100	200	400	600	800	1000	1200	1400	1600	1800	2000	4000	8000
Driving Cost	200	200	200	200	200	200	200	200	200	200	200	200	200

Figure 33, illustrates the grow rate of the total cost of logistic systems as the cost ratio increases in the Low Traffic Experiment. From Table 11, and Figure 33, one can conclude that PI is the most cost efficient logistics system if the material handling cost remains less than $200 per container. HY has the lowest total cost between cost ratios of approximately 1 to 40. CO becomes the most cost efficient logistics system for cost ratio of approximately 40 and higher.

PI and HY have a lower total cost compared to CO on a wide cost ratio range. This implies that in a low traffic networks, the efficiency of PI transit centers can vary widely without affecting the best performing logistic policy. At the cost ratio of 0.75, both PI and HY logistics systems are superior to CO.

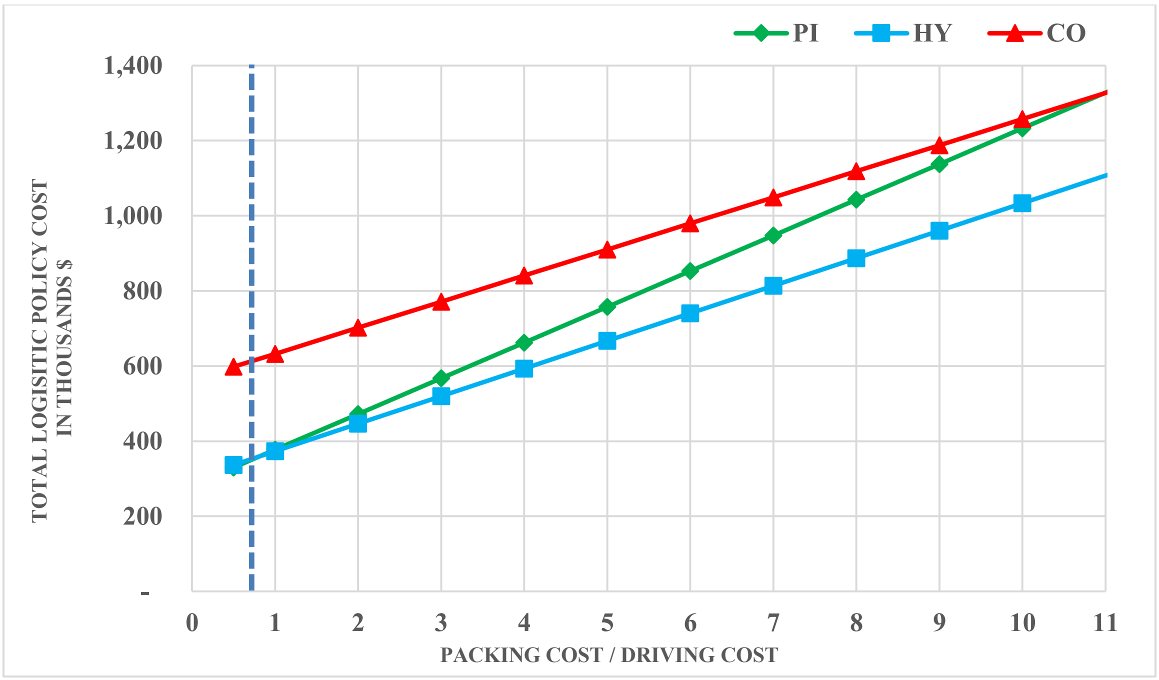

Figure 33 - Comparison of Total Cost of Logistic Systems in Low Traffic Experiment

6.5.2 Moderate Traffic Experiment

The summary of the results required for calculation of the total cost in the Moderate Traffic Experiment is shown in Table 12.

Table 12 - Social, Operational, and Fix Costs in Moderate Traffic Experiment

Policy	# Packing / Unpacking	Social Cost	Operational Cost	Fix Cost
PI	1118	559	$638.2k	$3.5k
HY	842	$7.1k	$651.3k	$5.1k
CO	519	$14.2k	$842.4k	$7.7k

Table 13 lists the total cost of each logistics system for each cost ratio.

Table 13 - Total Cost as a function of Cost Ratio in Moderate Traffic Experiment

Policy	Total Cost 1	Total Cost 2	Total Cost 3	Total Cost 4	Total Cost 5	Total Cost 6	Total Cost 7	Total Cost 8	Total Cost 9	Total Cost 10	Total Cost 11	Total Cost 12	Total Cost 13
PI	$754.1k	$865.9k	$977.7k	$1.1m	$1.2m	$1.3m	$1.4m	$1.5m	$1.6m	$1.8m	$1.9m	$2m	$2.1m
HY	$747.6k	$831.8k	$916k	$1m	$1.1m	$1.2m	$1.3m	$1.3m	$1.4m	$1.5m	$1.6m	$1.7m	$1.8m
CO	$916.2k	$968.1k	$1m	$1.1m	$1.1m	$1.2m	$1.2m	$1.3m	$1.3m	$1.4m	$1.4m	$1.5m	$1.5m
Ratio	**0.5**	**1**	**1.5**	**2**	**2.5**	**3**	**3.5**	**4**	**4.5**	**5**	**5.5**	**6**	**6.5**
Packing Cost	100	200	300	400	500	600	700	800	900	1000	1100	1200	1300
Driving Cost	200	200	200	200	200	200	200	200	200	200	200	200	200

Figure 34 illustrates the grow rate of the total cost of logistic systems as the cost ratio increases for the Moderate Traffic Experiment. From Table 13 and Figure 34, one can conclude that PI is the most cost efficient logistic policy if the material handling cost remains less than $110 per container. In other words, the cost ratio should stay below 0.55 for PI to be the least cost logistic policy. HY has the lowest total cost between cost ratios of approximately 0.55 up to 3.1. CO becomes the most cost efficient logistic policy for cost ratio of approximately 3.1 and higher.

In other words, there is a shift towards the left of all break even points in the lines representing total costs.

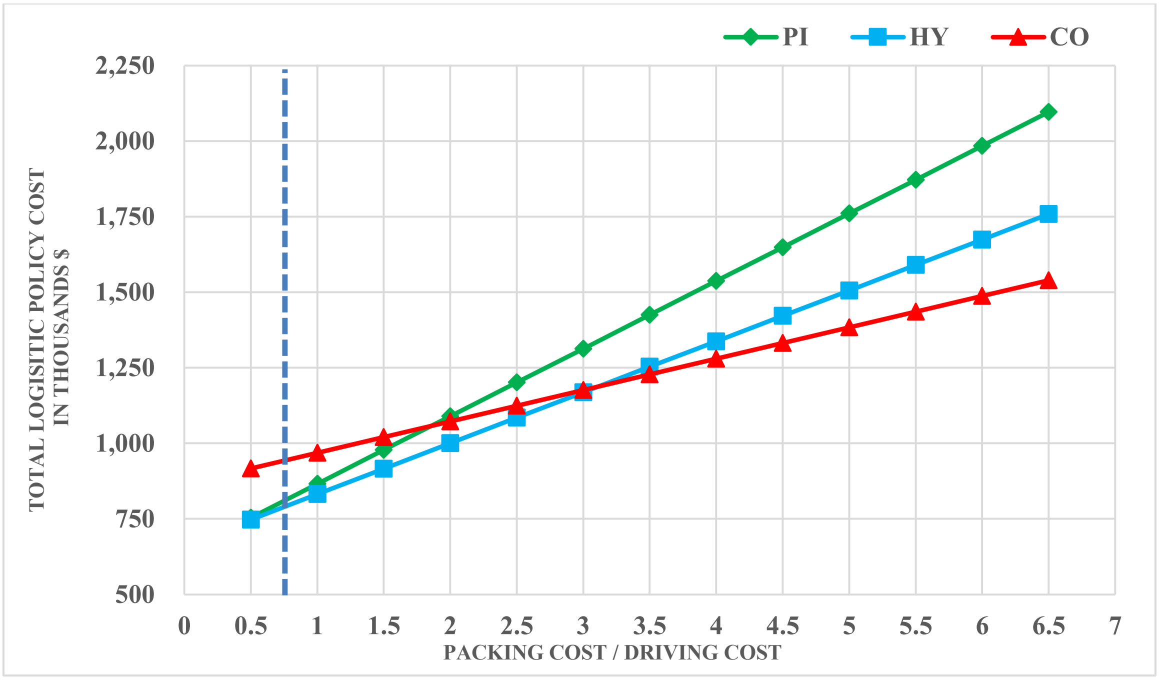

Figure 34 - Comparison of Total Cost of Logistic Systems in Low Traffic Experiment

6.5.3 High Traffic Experiment

The summary of the results required for calculation of the total cost in the High Traffic Experiment is shown in Table 14.

Table 14 - Social, Operational, and Fix Costs in High Traffic Experiment

Policy	# Packing / Unpacking	Social Cost	Operational Cost	Fix Cost
PI	2283	$1.3k	$1.3m	$6.3k
HY	1704	$14k	$1.3m	$11.5k
CO	950	$25.8k	$1.5m	$14.9k

Table 15 lists the total cost of each logistic system for each cost ratio.

Table 15 - Total Cost as a function of Cost Ratio in High Traffic Experiment

Policy	Total Cost 1	Total Cost 2	Total Cost 3	Total Cost 4	Total Cost 5	Total Cost 6	Total Cost 7	Total Cost 8	Total Cost 9	Total Cost 10	Total Cost 11	Total Cost 12	Total Cost 13
PI	$1.4m	$1.4m	$1.4m	$1.5m	$1.5m	$1.6m	$1.6m	$1.7m	$1.7m	$1.8m	$1.8m	$1.9m	$1.9m
HY	$1.4m	$1.4m	$1.5m	$1.5m	$1.5m	$1.6m	$1.6m	$1.6m	$1.7m	$1.7m	$1.7m	$1.8m	$1.8m
CO	$1.6m	$1.6m	$1.6m	$1.6m	$1.7m	$1.7m	$1.7m	$1.7m	$1.7m	$1.7m	$1.8m	$1.8m	$1.8m
Ratio	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**	**1.1**	**1.2**	**1.3**
Packing Cost	20	40	60	80	100	120	140	160	180	200	220	240	260
Driving Cost	200	200	200	200	200	200	200	200	200	200	200	200	200

Figure 35 illustrates the grow rate of the total cost of logistic systems as the cost ratio increases in the High Traffic Experiment. From Table 15 and Figure 35, one can conclude that PI is the most cost efficient logistics system if the material handling cost remains less than $80 per container. In other words, the cost ratio should stay below 0.4 for PI to be the least cost logistics system. HY has the lowest total cost between cost ratios of approximately 0.4 and 1.3. CO becomes the most cost efficient logistics system for cost ratios of approximately 1.3 and higher.

In other words, there is once again a shift towards the left of all break even points in the lines representing total costs.

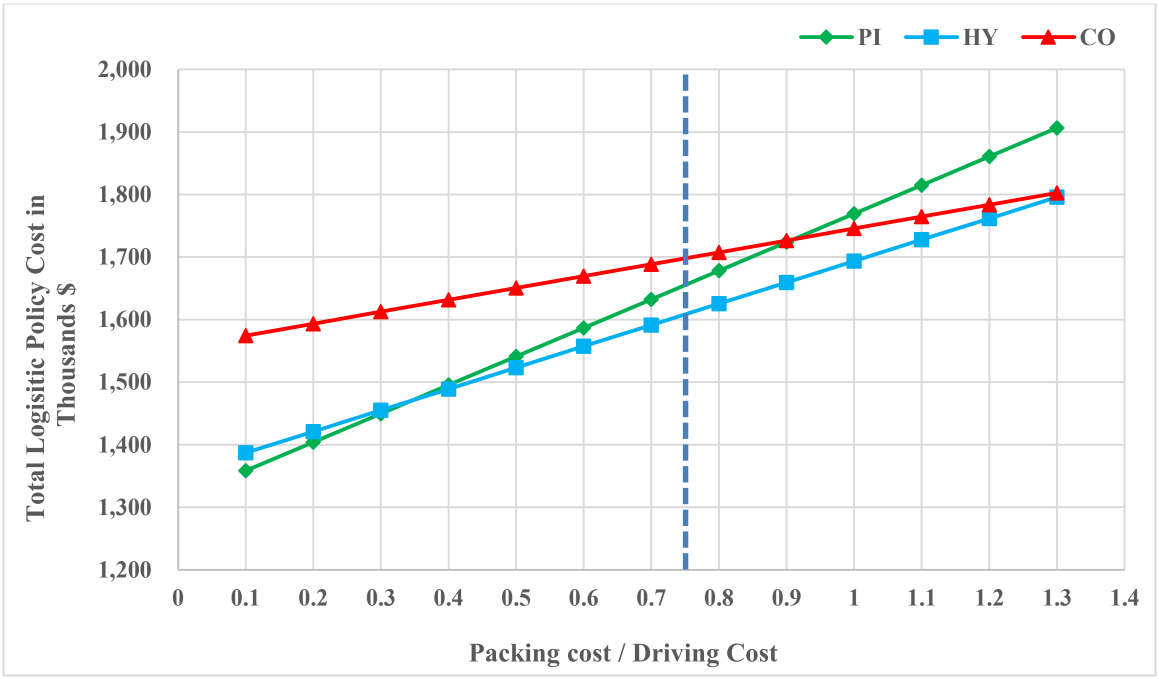

Figure 35 - Comparison of Total Cost of Logistic Systems in High Traffic Experiment

6.5.4 High Traffic Small Loads Experiment

The summary of the results required for calculation of the total cost in the High Traffic Small Loads Experiment is shown in Table 16.

Table 16 - Social, Operational, and Fix Costs in High Traffic Small Loads Experiment

Policy	# Packing / Unpacking	Social Cost	Operational Cost	Fix Cost
PI	1904	$1.3k	$1.1m	$5.9k
HY	1417	$11.2k	$1.1m	$8.2k
CO	814	$21.3k	$1.3m	$12.1k

Table 17 lists the total cost of each logistics systems for each cost ratio.

Table 17 - Total Cost as a function of Cost Ratio in High Traffic Small Loads Experiment

Policy	Total Cost 1	Total Cost 2	Total Cost 3	Total Cost 4	Total Cost 5	Total Cost 6	Total Cost 7	Total Cost 8	Total Cost 9	Total Cost 10	Total Cost 11	Total Cost 12	Total Cost 13
PI	$1.1m	$1.2m	$1.3m	$1.4m	$1.4m	$1.5m	$1.6m	$1.7m	$1.7m	$1.8m	$1.9m	$2m	$2m
HY	$1.1m	$1.2m	$1.2m	$1.3m	$1.3m	$1.4m	$1.5m	$1.5m	$1.6m	$1.6m	$1.7m	$1.7m	$1.8m
CO	$1.3m	$1.3m	$1.4m	$1.4m	$1.4m	$1.5m	$1.5m	$1.5m	$1.6m	$1.6m	$1.6m	$1.7m	$1.7m
Ratio	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**	**1.1**	**1.3**	**1.5**	**1.7**	**1.9**	**2.1**	**2.3**	**2.5**
Packing Cost	20	60	100	140	180	220	260	300	340	380	420	460	500
Driving Cost	200	200	200	200	200	200	200	200	200	200	200	200	200

Figure 36 illustrates the growth rate of the total cost of logistic systems as the cost ratio increases in the High Traffic Small Loads Experiment. From Table 17 and Figure 36 one can conclude that PI has the lowest total cost below cost ratios of approximately 0.12. HY is the most cost efficient logistics system if the cost ratio is between 0.12 and 1.7. CO becomes the most cost efficient logistic policy for cost ratio of approximately 1.7 and higher.

In a High Traffic Small Loads Network, the HY logistics system is the cheapest for a relatively wide range.

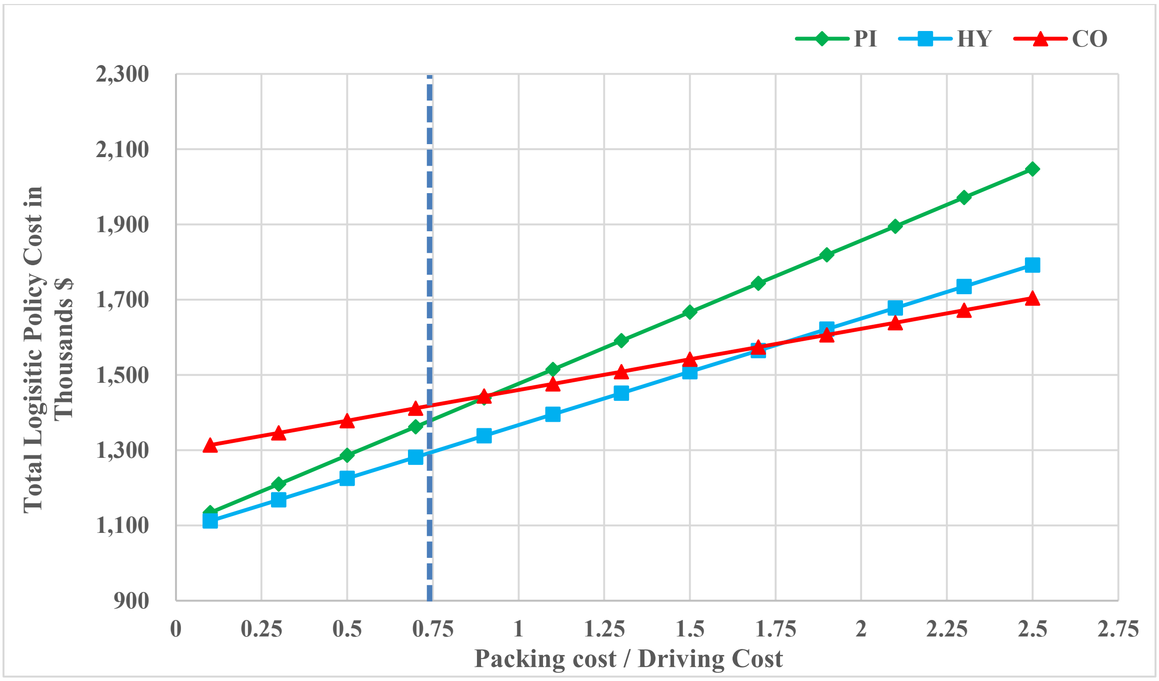

Figure 36 - Comparison of Total Cost of Logistic Systems in High Traffic Small Loads Experiment

6.5.5 High Traffic Large Loads Experiment

The summary of the results required for calculation of the total cost in the High Traffic Large Loads Experiment is shown in Table 18.

Table 18 - Social, Operational, and Fix Costs in High Traffic Large Loads Experiment

Policy	# Packing / Unpacking	Social Cost	Operational Cost	Fix Cost
PI	5773	$1.9k	$2.9m	$19.1k
HY	3971	$35.8k	$3.1m	$26.6k
CO	2181	$56.5k	$3.4m	$25.3k

Table 19 lists the total cost of each logistics system for each cost ratio.

Table 19 - Total Cost as a function of Cost Ratio in High Traffic Large Loads Experiment

Policy	Total Cost 1	Total Cost 2	Total Cost 3	Total Cost 4	Total Cost 5	Total Cost 6	Total Cost 7	Total Cost 8	Total Cost 9	Total Cost 10	Total Cost 11	Total Cost 12	Total Cost 13
PI	$3.1m	$3.2m	$3.3m	$3.4m	$3.5m	$3.6m	$3.8m	$3.9m	$4m	$4.1m	$4.2m	$4.3m	$4.4m
HY	$3.3m	$3.3m	$3.4m	$3.5m	$3.6m	$3.7m	$3.7m	$3.8m	$3.9m	$4m	$4m	$4.1m	$4.2m
CO	$3.5m	$3.5m	$3.6m	$3.6m	$3.7m	$3.7m	$3.8m	$3.8m	$3.8m	$3.9m	$3.9m	$4m	$4m
Ratio	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**	**1.1**	**1.2**	**1.3**
Packing Cost	20	40	60	80	100	120	140	160	180	200	220	240	260
Driving Cost	200	200	200	200	200	200	200	200	200	200	200	200	200

Figure 37 illustrates the growth rate of the total cost of the logistic systems as the cost ratio increases. From Table 19 and Figure 37, one can conclude that PI is the most cost efficient logistic policy if the material handling cost remains less than $120 per container. In other words, the cost ratio should stay below 0.6 for PI to be the least cost logistics system. HY has the lowest total cost between cost ratios of approximately 0.6 and 0.8. CO becomes the most cost efficient logistics system for cost ratios of approximately 0.8 and higher.

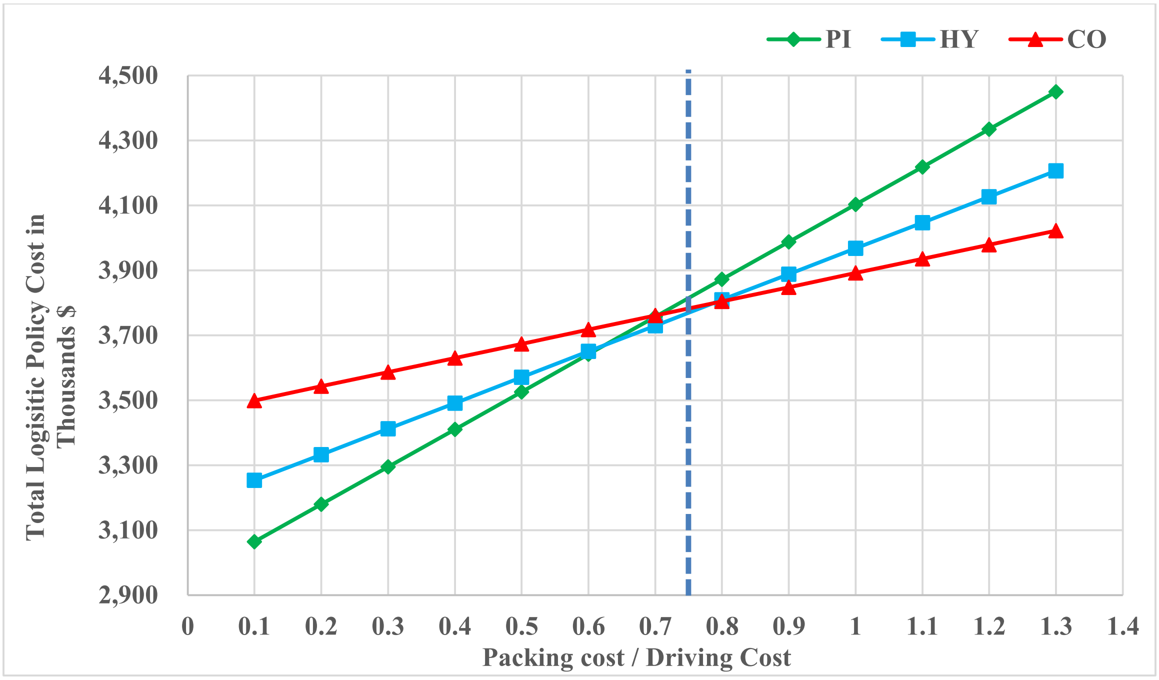

Figure 37 - Comparison of Total Cost of Logistic Systems in High Traffic Large Loads Experiment

6.5.6 Random Traffic Experiment

The summary of the results required for calculation of the total cost in the Random Traffic Experiment is shown in Table 20.

Table 20 - Social, Operational, and Fix Costs in Random Traffic Experiment

Policy	# Packing / Unpacking	Social Cost	Operational Cost	Fix Cost
PI	2175	$1.7k	$1.1m	$5.9k
HY	1617	$14.2k	$1.3m	$11k
CO	914	$24.9k	$1.5m	$14.5k

Table 21 lists the total cost of each logistics system for each cost ratio.

Table 21 - Total Cost as a function of Cost Ratio in Random Traffic Experiment

Policy	Total Cost 1	Total Cost 2	Total Cost 3	Total Cost 4	Total Cost 5	Total Cost 6	Total Cost 7	Total Cost 8	Total Cost 9	Total Cost 10	Total Cost 11	Total Cost 12	Total Cost 13
PI	$1.1m	$1.2m	$1.3m	$1.4m	$1.5m	$1.5m	$1.6m	$1.7m	$1.8m	$1.8m	$1.9m	$2m	$2.1m
HY	$1.3m	$1.4m	$1.4m	$1.5m	$1.5m	$1.6m	$1.7m	$1.7m	$1.8m	$1.8m	$1.9m	$1.9m	$2m
CO	$1.5m	$1.5m	$1.6m	$1.6m	$1.6m	$1.7m	$1.7m	$1.7m	$1.8m	$1.8m	$1.8m	$1.9m	$1.9m
Ratio	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**	**1.1**	**1.3**	**1.5**	**1.7**	**1.9**	**2.1**	**2.3**	**2.5**
Packing Cost	20	60	100	140	180	220	260	300	340	380	420	460	500
Driving Cost	200	200	200	200	200	200	200	200	200	200	200	200	200

Figure 38 illustrates the growth rate of the total cost of logistic systems as the cost ratio increases. From Table 21 and Figure 38, one can conclude that PI is the most cost efficient logistics policy if the material handling cost remains less than \$300 per container. In other words, the cost ratio should stay below 1.5 for PI to be the least cost logistics systems. The least cost logistics system changes very quickly from PI to CO, leaving a very small optimal range for HY. Very quickly after a cost ratio of 1.5, CO becomes the most cost efficient logistic policy.

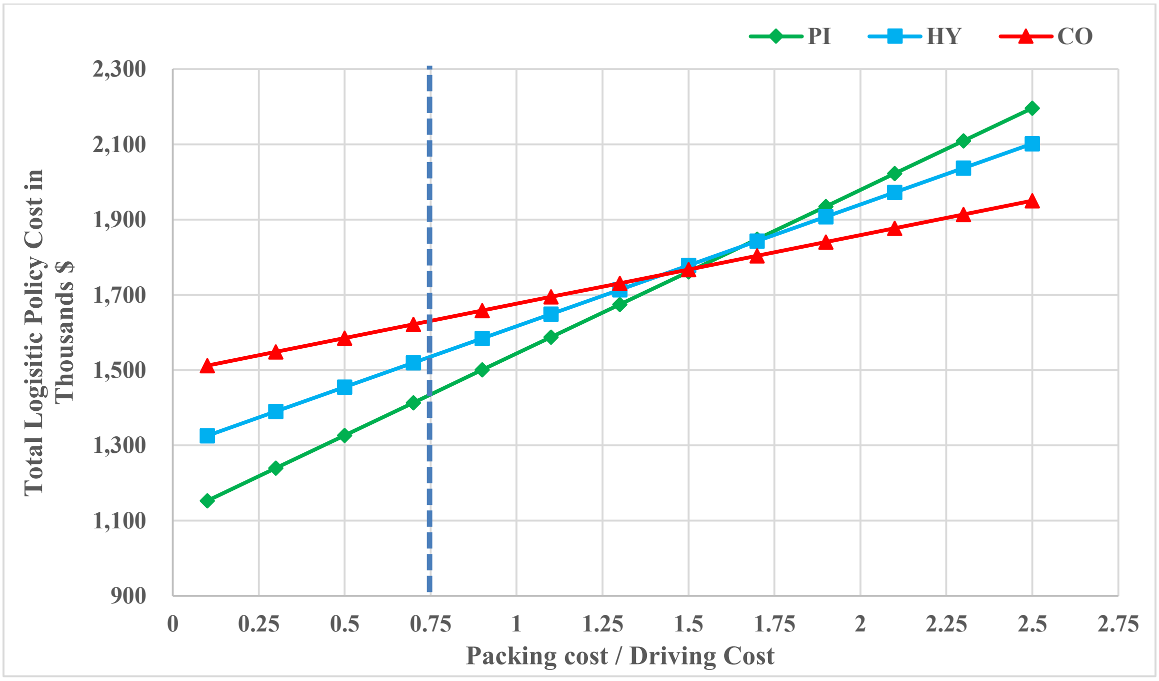

Figure 38 - Comparison of Total Cost of Logistic Systems in Random Traffic Experiment

Chapter 7: Conclusions

The three phase optimization framework has shown to be an effective method of investigating the overall performance of the three logistic systems in this thesis. By comparing the CO, PI, and HY logistic systems using the three phase optimization methodology, a major trade-off was observed between the total number of instances containers requiring loading and unloading and total required driving time to carry loads from source to final destination.

Compared to CO and HY, PI benefits from lower total driving time required to transfer loads from sources to destinations across the introduced road network. However, number of the instances containers require loading and unloading is higher in PI and HY compared to CO. In other words, PI and HY decreases the work done by drivers and increases the work done in the PI transit centers.

The cost performance of each of these logistic systems varies under various traffic and load selection scenarios. In networks with low ratio of packing to driving costs, PI seems to be the superior logistics system followed by HY and CO. As this ratio increases, HY and CO become increasingly attractive. For intermediate values of this ratio, HY is the most attractive while for very high values of this ratio, CO is the most attractive.

From an environmental point of view, PI shows substantial decrease in the logistic systems' carbon foot print from driving and reduces truck traffic on the roads. Another major benefit of PI is that it reduces the social costs associated with truck driving. It should be noted that success of PI logistic system highly depends on the efficiency of PI transit center. Without efficient PI transit centers, PI logistics policy will not reduce the total cost of logistic activities.

7.1 Direction for Future Studies

This thesis is one of the early analytical studies on the subject of PI. Further research is required to fully understand the capabilities and shortcoming of PI logistic systems. Some areas for potentially fruitful research are:

Extension to Grid Networks

This research mainly focussed on the Eastern Canada road network, which is practically speaking, a tree. This framework can be used to investigate these logistic systems on other network topologies such as grid networks.

Extension to City Level Delivery

The effect of urban area congestion and road network capacity were not addressed in this research. One of the reasons why PI may succeed in denser topographies (such as Europe) is that packing/unpacking may become required for delivery inside urban cores due to environmental regulations. If the cost of packing/unpacking is already paid for, the PI or HY logistics systems might become much more attractive. Therefore, it worth investigating this issue in realistic urban or dense population settings.

Effect of Loads Sizes

As explained in [9], the global standard PI container and load sizes are still not determined. In this thesis, it is shown than the load sizes is one of the main factors effecting the cost performance of PI. However, the combination of load sizes that could result in the optimal performance of PI is unknown and requires research.

Improvement of Packing Model

Model 1 of this thesis uses a one dimensional bin packing MIP to calculate the minimum number of the containers needed to be transported from sources to destinations. Load sizes chosen in this thesis made a one dimensional bin packing MIP sufficient to calculate the optimal number of bins in Model 1. However, if smaller load sizes are introduced, two or three dimensional bin packing will be required to calculate the minimum number of containers required. The effect of this should be considered

Delivery Times and Queuing

In this research, we do not look at delivery lead times or queues at PI centers which increase the total transit time. Investigating the effect of each logistic policy on delivery times and queuing behaviour in the PI transit centers can be very beneficial. In a simulation environment, the average effect of delays, and queues on the total cost of a PI logistic system can be examined and compared to other logistic systems.

Relaxation of the Reset Assumption and Effect of Uncertainty

This thesis only dealt with static loads. Load information is assumed to be available at the time of optimization. A one period planning horizon assumed with a state reset at the end of the period. However, in real-life, containers and trucks move constantly as demand evolves.

Dynamic optimization approaches can be developed and tested to investigate the cost performance of these logistic systems in real-time. There is also the effect of uncertainty that could be taken into account.

Multi-agent simulation models can be a means to evaluate the performance of real-time PI systems to understand the complex interplay involved in the presence of multiple decision makers systems with uncertainty in the logistics system. As discussed in section 2, and in [8], a multi-agent simulation of PI was shown to be effective methods to simulate supply webs. These types of simulation studies will potentially provide a more accurate evaluation of the logistic systems' behaviour when dealing with networks with complex demand, capacity, and routing constraints.

REFERENCES

[1] B. Montreuil, "Toward a Physical Internet: Meeting the Global Logisitcs Sustainability Grand Challenge," CIRRELT - Interuniversity Rsearch Center on Enterprise Networks, Logistics and Transportation, Quebec, 2011.

[2] B. Montreuil, "Physical Internet Manifesto," 2 April 2010. [Online]. Available: www.physicalinternetinitiative.org. [Accessed 12 March 2013].

[3] "Physical Internet: A survey of Logisitcs," *The Economist Newspaper,* 17 June 2006.

[4] "Physical Internet," CIRRLET, March 2010. [Online]. Available: http://www.physicalinternetinitiative.org/. [Accessed 20 January 2014].

[5] D. HAKIMI, M. Benoit and O. LABARTHE, "Supply Web: Concept and Technology," in *Seventh Annual International Symposium* , Toronto, 2009.

[6] B. Montreuil, R. D. Meller and E. Ballot, "Innovation, Toward a Physical Internet: The Impact on Logistics Facilities and Material Handling Systems Design and," 2010.

[7] B. Montreuil, O. Labarthe, D. Hakimi, A. Larcher and M. Audet, "Supply Web Mapper," in *International Conference on Industrial Engineering and Systems Management* , Montreal , 2009.

[8] D. Hakimi, B. Montreuil and O. Labarthe, "Supply Web Agent-Based Simulation Platform," in *3rd International Conference on Information Systems, Logistics and Supply Chain. Creating value through green supply chains*, Casablanca, 2010.

[9] E. Ballot, B. Montreuil and F. Fontane, "Topology of the Logistic Networks and the Potential of Physical Internet," CIRRELT, Quebec City, 2010.

[10] D. HAKIMI, B. MONTREUIL, R. SARRAJ, E. BALLOT and S. PAN, "SIMULATING A PHYSICAL INTERNET ENABLED MOBILITY WEB: THE CASE OF MASS

DISTRIBUTION IN FRANCE," in *9th International Conference of Modeling, Optimization and Simulation* , Bordeaux, 2012.

[11] R. D. Meller, B. Montreuil, C. Thivierge and Z. Montreuil, "Functional Design og Physical Internet Facilities: A Road Based Transit Center," in *Progress in Material Handling Research* , Charlotte, MHIA, 2012.

[12] E. Ballot, M. Benoit and C. Thivierge, "FUNCTIONAL DESIGN OF PHYSICAL INTERNET FACILITIES: A ROAD-RAIL HUB," in *Progress in Material Handling Research 2012*, Charlotte, MHIA, 2012.

[13] B. Montreuil, R. D. Meller, C. Thivierge and Z. Montreuil, "Functional Design of Physical Internet Facilities: A Road-Based Crossdocking hub," in *Progress in Material Handling Research 2012'*, Charlotte, MHIA, 2012.

[14] R. Sarraj, E. Ballot, S. Pan, D. Hakimi and B. Montreuil, "Interconnected logistic networks and protocols: simulation-based efficiency assessment," *International Journal of Production Research,* vol. 52, no. 11, pp. 3185-3208, 2013.

[15] G. Laporte, "Vehicle Routing Problem: An overview of exact and approximate algorithms," *European Journal of Operational Research,* pp. 345-358, 1992.

[16] L. Bodin, B. Golden, A. Assad and M. Ball, "Routing and scheduling of vehicles and crews. The state of the art," *Computers and Operations Research,* pp. 69-211, 1983.

[17] G. Laporte and Y. Nobert, "Exact algorithms for the vehicle routing problem," in *Surveys in Combinatorial Optimization*, Amsterdam, North-Holland, 1987, pp. 147-184.

[18] S. N. Parragh, K. F. Doerner and R. F. Hartl, "A survey on pickup and delivery problems. Part I: Transportation between customers and depots," Wein, Springer, 2008, pp. 21-51.

[19] C. E. Miller, A. W. Tucker and R. A. Zemlin, "Integer Programming Formulation of Traveling," *JACM,* vol. 7, no. 4, pp. 326-329, 4 October 1960.

[20] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research,* vol. 59, no. 3, p. 345–358, 1992.

[21] M. Goetschalckx and C. J.-B. Jacobs-Blecha, "The vehicle routing problem with backhauls," *European Journal of Operational Research,* vol. 42, no. 1, p. 39–51, 1989.

[22] P. Toth and D. Vigo, "An Exact Algorithm for the Vehicle Routing Problem with Backhauls," *Transportation Science,* vol. 31, no. 4, pp. 372 - 385, 1997.

[23] D. Duhamel, J.-Y. Potvin and J.-M. Rousseau, "A Tabu Search Heuristic for the Vehicle Routing Problem with Backhauls and Time Windows," *Transportation Science,* vol. 31, no. 1, pp. 49 - 59, 1997.

[24] G. Mosheiov, "The Travelling Salesman Problem with pick-up and delivery," *European Journal of Operational Research,* vol. 79, no. 2, p. 299–310, 1994.

[25] S. Salhi and G. Nagy, "A Cluster Insertion Heuristic for Single and Multiple Depot Vehicle Routing Problems with Backhauling," *The Journal of the Operational Research Society,* vol. 50, no. 2, pp. 1034-1042, 1999.

[26] H. Min, "The multiple vehicle routing problem with simultaneous delivery and pick-up points," *Transportation Research Part A: General,* vol. 23, no. 5, p. 377–386, 1989.

[27] F. Alfredo, T. Montané and G. R. Diéguez, "A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service," *Computers & Operations Research,* vol. 33, no. 3, p. 595–619, 2006.

[28] N. Bianchessi and G. Righini, "Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery," *Computers & Operations Research,* vol. 34, no. 2, p. 578–594, 2007.

[29] M. V. P. Savelsbergh and M. Sol, "The General Pickup and Delivery Problem," *Transportation Science,* vol. 29, no. 1, pp. 17 - 29 , 1995.

[30] J.-F. Cordeau and G. Laporte, "The dial-a-ride problem: models and algorithms," *Annals of Operations Research,* vol. 153, no. 1, pp. 29 - 46, 2007.

[31] R. D. Meller, B. Montreuil, C. Thivierge and Z. Montreuil, "FUNCTIONAL DESIGN OF PHYSICAL INTERNET FACILITIES: A ROAD-BASED TRANSIT CENTER," in *Progress in Material Handling Research 2012*, Charlotte, MHIA, 2012.

[32] World Shipping Council, "CONTAINERS," 8 February 2014. [Online]. Available: http://www.worldshipping.org/about-the-industry/containers.

[33] "3D Warehouse," Trimble Navigation Limited, [Online]. Available: https://3dwarehouse.sketchup.com/. [Accessed 12 January 2014].

[34] W. d. l. V. Frenandez and G. S. Lueker, "Bin Packing can be solved within 1 + e in Linear Time," *Combinatorica,* pp. 349-355, 22 May 1981.

[35] Gurobi Optimization Inc., "The Gurobi Python Interface for Python Users," 5 September 2012. [Online]. Available: http://www.gurobi.com/documentation/5.5/quick-start-guide/node39. [Accessed 10 June 2013].

[36] M. Labbe, G. Laporte and H. Mercure, "Capacitated vehicle routing on trees," *Operations Research,* pp. 39: 616 - 622, 1991.

[37] H. P. Williams, "Building Integer Programming Models II," in *Model Building in Mathmatical Programming*, London, Wiley, Sons, Ltd, 2013, pp. 199 - 211.

[38] J. Desrosiers, F. Soumis and M. Desrochers, "Networks," in *Routing with time windows by column generation*, vol. 14, London, UK: Wiley Periodicals, 1984, pp. 545 - 565.

[39] B. Chandran and S. Raghavan, "Modeling and Solving the Capacitated Vehicle Routing Problem on Trees," in *Vehicle Routing Problem* , Springer Science, 2008, pp. 243 - 244.

[40] RAY BARTON & ASSOCIATES, "Estimation of Costs of Heavy Vehicle Use per Vehicle-Kilometer in Canada," Transport Canada, Ottawa, 2006.

[41] J. J. Keller and Associates, "Hours of Service Canada: A Driver's Guide - DVD Training," 2006.

[42] "Commercial Vehicle Drivers Hours of Service Regulations (SOR/2005-313)," 17 February 2014. [Online]. Available: http://laws-lois.justice.gc.ca/eng/regulations/SOR-2005-313/page-3.html#h-9. [Accessed 3 March 2014].

[43] G. M. Saltzman and M. H. Belzer, "Truck Driver Occupational Safety and Health," NIOSH - Publications Dissemination, Cincinnati, OH, 2007.

[44] E. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik,* pp. 269-271, 1959.

[45] R. W. Floyd, "Algorithm 97: shortest path.," *Communications of the ACM,* vol. 5, no. 6, p. 345, 1962.

[46] R. E. Walpole, R. H. Myers, S. L. Myers and K. Ye, Probability & Statistics for Engineers & Scientists, 8th, Ed., San Antonio, Texas: Prentice Hall, 2006, p. 510.

[47] J. Francois Cordeau and G. Laporte, "Static pickup and delivery problems: a classification scheme and survey," *TOP,* vol. 15, no. 1, pp. 1 - 31, 2007.

[48] E. Angelelli and M. Renata, "The Vehicle Routing Problem with Time Window and Simultaneous Pick-up and Delivery," in *Quantitative Approaches to Distribution Logistics and Supply Chain Management*, Berlin Heidelberg, Springer, 2002, pp. 249 - 267.

APPENDICES

Appendix A: Conventional Arcs Dictionary

```
route_co = {
   ('Yar','Hal'): [['Yar','Hal']],
   ('Yar','Tru'): [['Yar','Tru']],
   ('Yar','Syd'): [['Yar','Syd']],
   ('Yar','Mon'): [['Yar','Mon']],
   ('Yar','Cht'): [['Yar','Cht']],
   ('Yar','Stj'): [['Yar','Stj']],
   ('Yar','Frd'): [['Yar','Frd']],
   ('Yar','Gt1'): [['Yar','Gt1']],
   ('Yar','Riv'): [['Yar','Riv']],
   ('Yar','Qbc'): [['Yar','Qbc']],
   ('Yar','Gt2'): [['Yar','Gt2']],
   ('Yar','Mte'): [['Yar','Mte']],
   ('Yar','Gt3'): [['Yar','Gt3']],
   ('Yar','Caw'): [['Yar','Caw']],
   ('Yar','Mtw'): [['Yar','Mtw']],
   ('Yar','Otn'): [['Yar','Otn']],
   ('Yar','Ots'): [['Yar','Ots']],

   ('Hal','Yar'): [['Hal','Yar']],
   ('Hal','Tru'): [['Hal','Tru']],
   ('Hal','Syd'): [['Hal','Syd']],
   ('Hal','Mon'): [['Hal','Mon']],
   ('Hal','Cht'): [['Hal','Cht']],
   ('Hal','Stj'): [['Hal','Stj']],
   ('Hal','Frd'): [['Hal','Frd']],
   ('Hal','Gt1'): [['Hal','Gt1']],
   ('Hal','Riv'): [['Hal','Riv']],
   ('Hal','Qbc'): [['Hal','Qbc']],
   ('Hal','Gt2'): [['Hal','Gt2']],
   ('Hal','Mte'): [['Hal','Mte']],
   ('Hal','Gt3'): [['Hal','Gt3']],
   ('Hal','Caw'): [['Hal','Caw']],
   ('Hal','Mtw'): [['Hal','Mtw']],
   ('Hal','Otn'): [['Hal','Otn']],
   ('Hal','Ots'): [['Hal','Ots']],

   ('Syd','Yar'): [['Syd','Yar']],
   ('Syd','Tru'): [['Syd','Tru']],
   ('Syd','Hal'): [['Syd','Hal']],
   ('Syd','Mon'): [['Syd','Mon']],
   ('Syd','Cht'): [['Syd','Cht']],
   ('Syd','Stj'): [['Syd','Stj']],
   ('Syd','Frd'): [['Syd','Frd']],
   ('Syd','Gt1'): [['Syd','Gt1']],
   ('Syd','Riv'): [['Syd','Riv']],
   ('Syd','Qbc'): [['Syd','Qbc']],
   ('Syd','Gt2'): [['Syd','Gt2']],
   ('Syd','Mte'): [['Syd','Mte']],
   ('Syd','Gt3'): [['Syd','Gt3']],
   ('Syd','Caw'): [['Syd','Caw']],
   ('Syd','Mtw'): [['Syd','Mtw']],
   ('Syd','Otn'): [['Syd','Otn']],
   ('Syd','Ots'): [['Syd','Ots']],
   ('Tru','Yar'): [['Tru','Yar']],
   ('Tru','Syd'): [['Tru','Syd']],
   ('Tru','Hal'): [['Tru','Hal']],
   ('Tru','Mon'): [['Tru','Mon']],
   ('Tru','Cht'): [['Tru','Cht']],
   ('Tru','Stj'): [['Tru','Stj']],
   ('Tru','Frd'): [['Tru','Frd']],
   ('Tru','Gt1'): [['Tru','Gt1']],
   ('Tru','Riv'): [['Tru','Riv']],
   ('Tru','Qbc'): [['Tru','Qbc']],
   ('Tru','Gt2'): [['Tru','Gt2']],
   ('Tru','Mte'): [['Tru','Mte']],
   ('Tru','Gt3'): [['Tru','Gt3']],
   ('Tru','Caw'): [['Tru','Caw']],
   ('Tru','Mtw'): [['Tru','Mtw']],
   ('Tru','Otn'): [['Tru','Otn']],
   ('Tru','Ots'): [['Tru','Ots']],

   ('Mon','Yar'): [['Mon','Yar']],
   ('Mon','Syd'): [['Mon','Syd']],
   ('Mon','Hal'): [['Mon','Hal']],
   ('Mon','Tru'): [['Mon','Tru']],
   ('Mon','Cht'): [['Mon','Cht']],
   ('Mon','Stj'): [['Mon','Stj']],
   ('Mon','Frd'): [['Mon','Frd']],
   ('Mon','Gt1'): [['Mon','Gt1']],
   ('Mon','Riv'): [['Mon','Riv']],
   ('Mon','Qbc'): [['Mon','Qbc']],
   ('Mon','Gt2'): [['Mon','Gt2']],
   ('Mon','Mte'): [['Mon','Mte']],
   ('Mon','Gt3'): [['Mon','Gt3']],
   ('Mon','Caw'): [['Mon','Caw']],
   ('Mon','Mtw'): [['Mon','Mtw']],
   ('Mon','Otn'): [['Mon','Otn']],
   ('Mon','Ots'): [['Mon','Ots']],

   ('Cht','Yar'): [['Cht','Yar']],
   ('Cht','Syd'): [['Cht','Syd']],
   ('Cht','Hal'): [['Cht','Hal']],
   ('Cht','Tru'): [['Cht','Tru']],
   ('Cht','Mon'): [['Cht','Mon']],
   ('Cht','Stj'): [['Cht','Stj']],
   ('Cht','Frd'): [['Cht','Frd']],
   ('Cht','Gt1'): [['Cht','Gt1']],
   ('Cht','Riv'): [['Cht','Riv']],
   ('Cht','Qbc'): [['Cht','Qbc']],
   ('Cht','Gt2'): [['Cht','Gt2']],
   ('Cht','Mte'): [['Cht','Mte']],
   ('Cht','Gt3'): [['Cht','Gt3']],
   ('Cht','Caw'): [['Cht','Caw']],
   ('Cht','Mtw'): [['Cht','Mtw']],
   ('Cht','Otn'): [['Cht','Otn']],
   ('Cht','Ots'): [['Cht','Ots']],

   ('Stj','Yar'): [['Stj','Yar']],
   ('Stj','Syd'): [['Stj','Syd']],
   ('Stj','Hal'): [['Stj','Hal']],
   ('Stj','Tru'): [['Stj','Tru']],
   ('Stj','Cht'): [['Stj','Cht']],
   ('Stj','Mon'): [['Stj','Mon']],
   ('Stj','Frd'): [['Stj','Frd']],
   ('Stj','Gt1'): [['Stj','Gt1']],
   ('Stj','Riv'): [['Stj','Riv']],
   ('Stj','Qbc'): [['Stj','Qbc']],
   ('Stj','Gt2'): [['Stj','Gt2']],
   ('Stj','Mte'): [['Stj','Mte']],
   ('Stj','Gt3'): [['Stj','Gt3']],
   ('Stj','Caw'): [['Stj','Caw']],
   ('Stj','Mtw'): [['Stj','Mtw']],
   ('Stj','Otn'): [['Stj','Otn']],
   ('Stj','Ots'): [['Stj','Ots']],

   ('Frd','Yar'): [['Frd','Yar']],
   ('Frd','Syd'): [['Frd','Syd']],
   ('Frd','Hal'): [['Frd','Hal']],
   ('Frd','Tru'): [['Frd','Tru']],
   ('Frd','Cht'): [['Frd','Cht']],
   ('Frd','Mon'): [['Frd','Mon']],
   ('Frd','Stj'): [['Frd','Stj']],
   ('Frd','Gt1'): [['Frd','Gt1']],
   ('Frd','Riv'): [['Frd','Riv']],
   ('Frd','Qbc'): [['Frd','Qbc']],
   ('Frd','Gt2'): [['Frd','Gt2']],
   ('Frd','Mte'): [['Frd','Mte']],
   ('Frd','Gt3'): [['Frd','Gt3']],
   ('Frd','Caw'): [['Frd','Caw']],
   ('Frd','Mtw'): [['Frd','Mtw']],
   ('Frd','Otn'): [['Frd','Otn']],
   ('Frd','Ots'): [['Frd','Ots']],

   ('Gt1','Yar'): [['Gt1','Yar']],
   ('Gt1','Syd'): [['Gt1','Syd']],
   ('Gt1','Hal'): [['Gt1','Hal']],
   ('Gt1','Tru'): [['Gt1','Tru']],
   ('Gt1','Cht'): [['Gt1','Cht']],
   ('Gt1','Mon'): [['Gt1','Mon']],
   ('Gt1','Stj'): [['Gt1','Stj']],
   ('Gt1','Frd'): [['Gt1','Frd']],
   ('Gt1','Riv'): [['Gt1','Riv']],
   ('Gt1','Qbc'): [['Gt1','Qbc']],
   ('Gt1','Mte'): [['Gt1','Mte']],
   ('Gt1','Gt2'): [['Gt1','Gt2']],
   ('Gt1','Gt3'): [['Gt1','Gt3']],
   ('Gt1','Caw'): [['Gt1','Caw']],
   ('Gt1','Mtw'): [['Gt1','Mtw']],
   ('Gt1','Otn'): [['Gt1','Otn']],
   ('Gt1','Ots'): [['Gt1','Ots']],

   ('Riv','Yar'): [['Riv','Yar']],
   ('Riv','Syd'): [['Riv','Syd']],
   ('Riv','Hal'): [['Riv','Hal']],
   ('Riv','Tru'): [['Riv','Tru']],
   ('Riv','Cht'): [['Riv','Cht']],
   ('Riv','Mon'): [['Riv','Mon']],
   ('Riv','Stj'): [['Riv','Stj']],
   ('Riv','Frd'): [['Riv','Frd']],
   ('Riv','Gt1'): [['Riv','Gt1']],
   ('Riv','Qbc'): [['Riv','Qbc']],
   ('Riv','Mte'): [['Riv','Mte']],
   ('Riv','Gt2'): [['Riv','Gt2']],
   ('Riv','Gt3'): [['Riv','Gt3']],
   ('Riv','Caw'): [['Riv','Caw']],
   ('Riv','Mtw'): [['Riv','Mtw']],
   ('Riv','Otn'): [['Riv','Otn']],
   ('Riv','Ots'): [['Riv','Ots']],

   ('Qbc','Yar'): [['Qbc','Yar']],
   ('Qbc','Syd'): [['Qbc','Syd']],
   ('Qbc','Hal'): [['Qbc','Hal']],
   ('Qbc','Tru'): [['Qbc','Tru']],
   ('Qbc','Cht'): [['Qbc','Cht']],
   ('Qbc','Mon'): [['Qbc','Mon']],
   ('Qbc','Stj'): [['Qbc','Stj']],
   ('Qbc','Frd'): [['Qbc','Frd']],
   ('Qbc','Gt1'): [['Qbc','Gt1']],
   ('Qbc','Riv'): [['Qbc','Riv']],
   ('Qbc','Mte'): [['Qbc','Mte']],
   ('Qbc','Gt2'): [['Qbc','Gt2']],
   ('Qbc','Gt3'): [['Qbc','Gt3']],
   ('Qbc','Caw'): [['Qbc','Caw']],
   ('Qbc','Mtw'): [['Qbc','Mtw']],
   ('Qbc','Otn'): [['Qbc','Otn']],
   ('Qbc','Ots'): [['Qbc','Ots']],

   ('Gt2','Yar'): [['Gt2','Yar']],
   ('Gt2','Syd'): [['Gt2','Syd']],
   ('Gt2','Hal'): [['Gt2','Hal']],
   ('Gt2','Tru'): [['Gt2','Tru']],
   ('Gt2','Cht'): [['Gt2','Cht']],
   ('Gt2','Mon'): [['Gt2','Mon']],
   ('Gt2','Stj'): [['Gt2','Stj']],
   ('Gt2','Frd'): [['Gt2','Frd']],
   ('Gt2','Gt1'): [['Gt2','Gt1']],
   ('Gt2','Riv'): [['Gt2','Riv']],
   ('Gt2','Qbc'): [['Gt2','Qbc']],
   ('Gt2','Mte'): [['Gt2','Mte']],
   ('Gt2','Gt3'): [['Gt2','Gt3']],
   ('Gt2','Caw'): [['Gt2','Caw']],
   ('Gt2','Mtw'): [['Gt2','Mtw']],
   ('Gt2','Otn'): [['Gt2','Otn']],
   ('Gt2','Ots'): [['Gt2','Ots']],

   ('Mte','Yar'): [['Mte','Yar']],
   ('Mte','Syd'): [['Mte','Syd']],
   ('Mte','Hal'): [['Mte','Hal']],
   ('Mte','Tru'): [['Mte','Tru']],
   ('Mte','Cht'): [['Mte','Cht']],
   ('Mte','Mon'): [['Mte','Mon']],
   ('Mte','Stj'): [['Mte','Stj']],
   ('Mte','Frd'): [['Mte','Frd']],
   ('Mte','Gt1'): [['Mte','Gt1']],
   ('Mte','Riv'): [['Mte','Riv']],
   ('Mte','Qbc'): [['Mte','Qbc']],
   ('Mte','Gt2'): [['Mte','Gt2']],
   ('Mte','Gt3'): [['Mte','Gt3']],
   ('Mte','Caw'): [['Mte','Caw']],
   ('Mte','Mtw'): [['Mte','Mtw']],
   ('Mte','Otn'): [['Mte','Otn']],
   ('Mte','Ots'): [['Mte','Ots']],

   ('Gt3','Yar'): [['Gt3','Yar']],
   ('Gt3','Syd'): [['Gt3','Syd']],
   ('Gt3','Hal'): [['Gt3','Hal']],
   ('Gt3','Tru'): [['Gt3','Tru']],
   ('Gt3','Cht'): [['Gt3','Cht']],
   ('Gt3','Mon'): [['Gt3','Mon']],
   ('Gt3','Stj'): [['Gt3','Stj']],
   ('Gt3','Frd'): [['Gt3','Frd']],
   ('Gt3','Gt1'): [['Gt3','Gt1']],
   ('Gt3','Riv'): [['Gt3','Riv']],
   ('Gt3','Qbc'): [['Gt3','Qbc']],
   ('Gt3','Gt2'): [['Gt3','Gt2']],
   ('Gt3','Mte'): [['Gt3','Mte']],
   ('Gt3','Caw'): [['Gt3','Caw']],
   ('Gt3','Mtw'): [['Gt3','Mtw']],
   ('Gt3','Otn'): [['Gt3','Otn']],
   ('Gt3','Ots'): [['Gt3','Ots']],

   ('Caw','Yar'): [['Caw','Yar']],
   ('Caw','Syd'): [['Caw','Syd']],
   ('Caw','Hal'): [['Caw','Hal']],
   ('Caw','Tru'): [['Caw','Tru']],
   ('Caw','Cht'): [['Caw','Cht']],
   ('Caw','Mon'): [['Caw','Mon']],
   ('Caw','Stj'): [['Caw','Stj']],
   ('Caw','Frd'): [['Caw','Frd']],
   ('Caw','Gt1'): [['Caw','Gt1']],
   ('Caw','Riv'): [['Caw','Riv']],
   ('Caw','Qbc'): [['Caw','Qbc']],
   ('Caw','Gt2'): [['Caw','Gt2']],
   ('Caw','Mte'): [['Caw','Mte']],
   ('Caw','Gt3'): [['Caw','Gt3']],
   ('Caw','Mtw'): [['Caw','Mtw']],
   ('Caw','Otn'): [['Caw','Otn']],
   ('Caw','Ots'): [['Caw','Ots']],

   ('Mtw','Yar'): [['Mtw','Yar']],
   ('Mtw','Syd'): [['Mtw','Syd']],
   ('Mtw','Hal'): [['Mtw','Hal']],
   ('Mtw','Tru'): [['Mtw','Tru']],
   ('Mtw','Cht'): [['Mtw','Cht']],
   ('Mtw','Mon'): [['Mtw','Mon']],
   ('Mtw','Stj'): [['Mtw','Stj']],
   ('Mtw','Frd'): [['Mtw','Frd']],
   ('Mtw','Gt1'): [['Mtw','Gt1']],
   ('Mtw','Riv'): [['Mtw','Riv']],
   ('Mtw','Qbc'): [['Mtw','Qbc']],
   ('Mtw','Gt2'): [['Mtw','Gt2']],
   ('Mtw','Mte'): [['Mtw','Mte']],
   ('Mtw','Gt3'): [['Mtw','Gt3']],
   ('Mtw','Caw'): [['Mtw','Caw']],
   ('Mtw','Otn'): [['Mtw','Otn']],
   ('Mtw','Ots'): [['Mtw','Ots']],

   ('Otn','Yar'): [['Otn','Yar']],
   ('Otn','Syd'): [['Otn','Syd']],
   ('Otn','Hal'): [['Otn','Hal']],
   ('Otn','Tru'): [['Otn','Tru']],
   ('Otn','Cht'): [['Otn','Cht']],
   ('Otn','Mon'): [['Otn','Mon']],
   ('Otn','Stj'): [['Otn','Stj']],
   ('Otn','Frd'): [['Otn','Frd']],
   ('Otn','Gt1'): [['Otn','Gt1']],
   ('Otn','Riv'): [['Otn','Riv']],
   ('Otn','Qbc'): [['Otn','Qbc']],
   ('Otn','Gt2'): [['Otn','Gt2']],
   ('Otn','Mte'): [['Otn','Mte']],
   ('Otn','Gt3'): [['Otn','Gt3']],
   ('Otn','Caw'): [['Otn','Caw']],
   ('Otn','Mtw'): [['Otn','Mtw']],
   ('Otn','Ots'): [['Otn','Ots']],

   ('Ots','Yar'): [['Ots','Yar']],
   ('Ots','Syd'): [['Ots','Syd']],
   ('Ots','Hal'): [['Ots','Hal']],
   ('Ots','Tru'): [['Ots','Tru']],
   ('Ots','Cht'): [['Ots','Cht']],
   ('Ots','Mon'): [['Ots','Mon']],
   ('Ots','Stj'): [['Ots','Stj']],
   ('Ots','Frd'): [['Ots','Frd']],
   ('Ots','Gt1'): [['Ots','Gt1']],
   ('Ots','Riv'): [['Ots','Riv']],
   ('Ots','Qbc'): [['Ots','Qbc']],
   ('Ots','Gt2'): [['Ots','Gt2']],
   ('Ots','Mte'): [['Ots','Mte']],
   ('Ots','Gt3'): [['Ots','Gt3']],
   ('Ots','Caw'): [['Ots','Caw']],
   ('Ots','Mtw'): [['Ots','Mtw']],
   ('Ots','Otn'): [['Ots','Otn']]
   }
```

Appendix B: PI Arcs Dictionary

```
route_pi = {
  ('Yar','Hal'): [['Yar','Hal']],
  ('Yar','Tru'): [['Yar','Tru']],
  ('Yar','Syd'): [['Yar','Syd']],
  ('Yar','Mon'): [['Yar','Tru'],['Tru','Mon']],
  ('Yar','Cht'): [['Yar','Tru'],['Tru','Cht']],
  ('Yar','Stj'): [['Yar','Tru'],['Tru','Stj']],
  ('Yar','Frd'): [['Yar','Tru'],['Tru','Frd']],
  ('Yar','Gt1'): [['Yar','Tru'],['Tru','Frd'],['Frd','Gt1']],
  ('Yar','Riv'): [['Yar','Tru'],['Tru','Frd'],['Frd','Riv']],
  ('Yar','Qbc'): [['Yar','Tru'],['Tru','Frd'],['Frd','Qbc']],
  ('Yar','Gt2'): [['Yar','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Gt2']],
  ('Yar','Mte'): [['Yar','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte']],
  ('Yar','Gt3'): [['Yar','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
  ('Yar','Caw'): [['Yar','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Caw']],
  ('Yar','Mtw'): [['Yar','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw']],
  ('Yar','Otn'): [['Yar','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
  ('Yar','Ots'): [['Yar','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],

  ('Hal','Yar'): [['Hal','Yar']],
  ('Hal','Tru'): [['Hal','Tru']],
  ('Hal','Syd'): [['Hal','Syd']],
  ('Hal','Mon'): [['Hal','Tru'],['Tru','Mon']],
  ('Hal','Cht'): [['Hal','Tru'],['Tru','Cht']],
  ('Hal','Stj'): [['Hal','Tru'],['Tru','Stj']],
  ('Hal','Frd'): [['Hal','Tru'],['Tru','Frd']],
  ('Hal','Gt1'): [['Hal','Tru'],['Tru','Frd'],['Frd','Gt1']],
  ('Hal','Riv'): [['Hal','Tru'],['Tru','Frd'],['Frd','Riv']],
  ('Hal','Qbc'): [['Hal','Tru'],['Tru','Frd'],['Frd','Qbc']],
  ('Hal','Gt2'): [['Hal','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Gt2']],
  ('Hal','Mte'): [['Hal','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte']],
  ('Hal','Gt3'): [['Hal','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
  ('Hal','Caw'): [['Hal','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Caw']],
  ('Hal','Mtw'): [['Hal','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw']],
  ('Hal','Otn'): [['Hal','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
  ('Hal','Ots'): [['Hal','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],

  ('Syd','Yar'): [['Syd','Yar']],
  ('Syd','Tru'): [['Syd','Tru']],
  ('Syd','Hal'): [['Syd','Hal']],
  ('Syd','Mon'): [['Syd','Tru'],['Tru','Mon']],
  ('Syd','Cht'): [['Syd','Tru'],['Tru','Cht']],
  ('Syd','Stj'): [['Syd','Tru'],['Tru','Stj']],
  ('Syd','Frd'): [['Syd','Tru'],['Tru','Frd']],
  ('Syd','Gt1'): [['Syd','Tru'],['Tru','Frd'],['Frd','Gt1']],
  ('Syd','Riv'): [['Syd','Tru'],['Tru','Frd'],['Frd','Riv']],
  ('Syd','Qbc'): [['Syd','Tru'],['Tru','Frd'],['Frd','Qbc']],
  ('Syd','Gt2'): [['Syd','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Gt2']],
  ('Syd','Mte'): [['Syd','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte']],
  ('Syd','Gt3'): [['Syd','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
  ('Syd','Caw'): [['Syd','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Caw']],
  ('Syd','Mtw'): [['Syd','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw']],
  ('Syd','Otn'): [['Syd','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
  ('Syd','Ots'): [['Syd','Tru'],['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],

  ('Tru','Yar'): [['Tru','Yar']],
  ('Tru','Syd'): [['Tru','Syd']],
  ('Tru','Hal'): [['Tru','Hal']],
('Tru','Mon'): [['Tru','Mon']],
('Tru','Cht'): [['Tru','Cht']],
('Tru','Stj'): [['Tru','Stj']],
('Tru','Frd'): [['Tru','Frd']],
('Tru','Gt1'): [['Tru','Frd'],['Frd','Gt1']],
('Tru','Riv'): [['Tru','Frd'],['Frd','Riv']],
('Tru','Qbc'): [['Tru','Frd'],['Frd','Qbc']],
('Tru','Gt2'): [['Tru','Frd'],['Frd','Qbc'],['Qbc','Gt2']],
('Tru','Mte'): [['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte']],
('Tru','Gt3'): [['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
('Tru','Caw'): [['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Caw']],
('Tru','Mtw'): [['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw']],
('Tru','Otn'): [['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
('Tru','Ots'): [['Tru','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],

('Mon','Yar'): [['Mon','Tru'],['Tru','Yar']],
('Mon','Syd'): [['Mon','Tru'],['Tru','Syd']],
('Mon','Hal'): [['Mon','Tru'],['Tru','Hal']],
('Mon','Tru'): [['Mon','Tru']],
('Mon','Cht'): [['Mon','Cht']],
('Mon','Stj'): [['Mon','Stj']],
('Mon','Frd'): [['Mon','Frd']],
('Mon','Gt1'): [['Mon','Frd'],['Frd','Gt1']],
('Mon','Riv'): [['Mon','Frd'],['Frd','Riv']],
('Mon','Qbc'): [['Mon','Frd'],['Frd','Qbc']],
('Mon','Gt2'): [['Mon','Frd'],['Frd','Qbc'],['Qbc','Gt2']],
('Mon','Mte'): [['Mon','Frd'],['Frd','Qbc'],['Qbc','Mte']],
('Mon','Gt3'): [['Mon','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
('Mon','Caw'): [['Mon','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Caw']],
('Mon','Mtw'): [['Mon','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw']],
('Mon','Otn'): [['Mon','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
('Mon','Ots'): [['Mon','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],

('Cht','Yar'): [['Cht','Tru'],['Tru','Yar']],
('Cht','Syd'): [['Cht','Tru'],['Tru','Syd']],
('Cht','Hal'): [['Cht','Tru'],['Tru','Hal']],
('Cht','Tru'): [['Cht','Tru']],
('Cht','Mon'): [['Cht','Mon']],
('Cht','Stj'): [['Cht','Stj']],
('Cht','Frd'): [['Cht','Frd']],
('Cht','Gt1'): [['Cht','Frd'],['Frd','Gt1']],
('Cht','Riv'): [['Cht','Frd'],['Frd','Riv']],
('Cht','Qbc'): [['Cht','Frd'],['Frd','Qbc']],
('Cht','Gt2'): [['Cht','Frd'],['Frd','Qbc'],['Qbc','Gt2']],
('Cht','Mte'): [['Cht','Frd'],['Frd','Qbc'],['Qbc','Mte']],
('Cht','Gt3'): [['Cht','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
('Cht','Caw'): [['Cht','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Caw']],
('Cht','Mtw'): [['Cht','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw']],
('Cht','Otn'): [['Cht','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
('Cht','Ots'): [['Cht','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],

('Stj','Yar'): [['Stj','Tru'],['Tru','Yar']],
('Stj','Syd'): [['Stj','Tru'],['Tru','Syd']],
('Stj','Hal'): [['Stj','Tru'],['Tru','Hal']],
('Stj','Tru'): [['Stj','Tru']],
('Stj','Cht'): [['Stj','Cht']],
('Stj','Mon'): [['Stj','Mon']],
('Stj','Frd'): [['Stj','Frd']],
('Stj','Gt1'): [['Stj','Frd'],['Frd','Gt1']],
('Stj','Riv'): [['Stj','Frd'],['Frd','Riv']],
('Stj','Qbc'): [['Stj','Frd'],['Frd','Qbc']],
('Stj','Gt2'): [['Stj','Frd'],['Frd','Qbc'],['Qbc','Gt2']],
('Stj','Mte'): [['Stj','Frd'],['Frd','Qbc'],['Qbc','Mte']],
('Stj','Gt3'): [['Stj','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
('Stj','Caw'): [['Stj','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Caw']],
('Stj','Mtw'): [['Stj','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw']],
('Stj','Otn'): [['Stj','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
('Stj','Ots'): [['Stj','Frd'],['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],

('Frd','Yar'): [['Frd','Tru'],['Tru','Yar']],
('Frd','Syd'): [['Frd','Tru'],['Tru','Syd']],
('Frd','Hal'): [['Frd','Tru'],['Tru','Hal']],
('Frd','Tru'): [['Frd','Tru']],
('Frd','Cht'): [['Frd','Cht']],
('Frd','Mon'): [['Frd','Mon']],
('Frd','Stj'): [['Frd','Stj']],
('Frd','Gt1'): [['Frd','Gt1']],
('Frd','Riv'): [['Frd','Riv']],
('Frd','Qbc'): [['Frd','Qbc']],
('Frd','Gt2'): [['Frd','Qbc'],['Qbc','Gt2']],
('Frd','Mte'): [['Frd','Qbc'],['Qbc','Mte']],
('Frd','Gt3'): [['Frd','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
('Frd','Caw'): [['Frd','Qbc'],['Qbc','Mte'],['Mte','Caw']],
('Frd','Mtw'): [['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw']],
('Frd','Otn'): [['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
('Frd','Ots'): [['Frd','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],

('Gt1','Yar'): [['Gt1','Frd'],['Frd','Tru'],['Tru','Yar']],
('Gt1','Syd'): [['Gt1','Frd'],['Frd','Tru'],['Tru','Syd']],
('Gt1','Hal'): [['Gt1','Frd'],['Frd','Tru'],['Tru','Hal']],
('Gt1','Tru'): [['Gt1','Frd'],['Frd','Tru']],
('Gt1','Cht'): [['Gt1','Frd'],['Frd','Cht']],
('Gt1','Mon'): [['Gt1','Frd'],['Frd','Mon']],
('Gt1','Stj'): [['Gt1','Frd'],['Frd','Stj']],
('Gt1','Frd'): [['Gt1','Frd']],
('Gt1','Riv'): [['Gt1','Riv']],
('Gt1','Qbc'): [['Gt1','Qbc']],
('Gt1','Mte'): [['Gt1','Qbc'],['Qbc','Mte']],
('Gt1','Gt2'): [['Gt1','Qbc'],['Qbc','Gt2']],
('Gt1','Gt3'): [['Gt1','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
('Gt1','Caw'): [['Gt1','Qbc'],['Qbc','Mte'],['Mte','Caw']],
('Gt1','Mtw'): [['Gt1','Qbc'],['Qbc','Mte'],['Mte','Mtw']],
('Gt1','Otn'): [['Gt1','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
('Gt1','Ots'): [['Gt1','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],

('Riv','Yar'): [['Riv','Frd'],['Frd','Tru'],['Tru','Yar']],
('Riv','Syd'): [['Riv','Frd'],['Frd','Tru'],['Tru','Syd']],
('Riv','Hal'): [['Riv','Frd'],['Frd','Tru'],['Tru','Hal']],
('Riv','Tru'): [['Riv','Frd'],['Frd','Tru']],
('Riv','Cht'): [['Riv','Frd'],['Frd','Cht']],
('Riv','Mon'): [['Riv','Frd'],['Frd','Mon']],
('Riv','Stj'): [['Riv','Frd'],['Frd','Stj']],
('Riv','Frd'): [['Riv','Frd']],
('Riv','Gt1'): [['Riv','Gt1']],
('Riv','Qbc'): [['Riv','Qbc']],
('Riv','Mte'): [['Riv','Qbc'],['Qbc','Mte']],
('Riv','Gt2'): [['Riv','Qbc'],['Qbc','Gt2']],
('Riv','Gt3'): [['Riv','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
('Riv','Caw'): [['Riv','Qbc'],['Qbc','Mte'],['Mte','Caw']],
('Riv','Mtw'): [['Riv','Qbc'],['Qbc','Mte'],['Mte','Mtw']],
('Riv','Otn'): [['Riv','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
('Riv','Ots'): [['Riv','Qbc'],['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],
('Qbc','Yar'): [['Qbc','Frd'],['Frd','Tru'],['Tru','Yar']],
('Qbc','Syd'): [['Qbc','Frd'],['Frd','Tru'],['Tru','Syd']],
('Qbc','Hal'): [['Qbc','Frd'],['Frd','Tru'],['Tru','Hal']],
('Qbc','Tru'): [['Qbc','Frd'],['Frd','Tru']],
('Qbc','Cht'): [['Qbc','Frd'],['Frd','Cht']],
('Qbc','Mon'): [['Qbc','Frd'],['Frd','Mon']],
('Qbc','Stj'): [['Qbc','Frd'],['Frd','Stj']],
('Qbc','Frd'): [['Qbc','Frd']],
('Qbc','Gt1'): [['Qbc','Gt1']],
('Qbc','Riv'): [['Qbc','Riv']],
('Qbc','Mte'): [['Qbc','Mte']],
('Qbc','Gt2'): [['Qbc','Gt2']],
('Qbc','Gt3'): [['Qbc','Mte'],['Mte','Gt3']],
('Qbc','Caw'): [['Qbc','Mte'],['Mte','Caw']],
('Qbc','Mtw'): [['Qbc','Mte'],['Mte','Mtw']],
('Qbc','Otn'): [['Qbc','Mte'],['Mte','Mtw'],['Mtw','Otn']],
('Qbc','Ots'): [['Qbc','Mte'],['Mte','Mtw'],['Mtw','Ots']],

('Gt2','Yar'): [['Gt2','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Yar']],
('Gt2','Syd'): [['Gt2','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Syd']],
('Gt2','Hal'): [['Gt2','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Hal']],
('Gt2','Tru'): [['Gt2','Qbc'],['Qbc','Frd'],['Frd','Tru']],
('Gt2','Cht'): [['Gt2','Qbc'],['Qbc','Frd'],['Frd','Cht']],
('Gt2','Mon'): [['Gt2','Qbc'],['Qbc','Frd'],['Frd','Mon']],
('Gt2','Stj'): [['Gt2','Qbc'],['Qbc','Frd'],['Frd','Stj']],
('Gt2','Frd'): [['Gt2','Qbc'],['Qbc','Frd']],
('Gt2','Gt1'): [['Gt2','Qbc'],['Qbc','Gt1']],
('Gt2','Riv'): [['Gt2','Qbc'],['Qbc','Riv']],
('Gt2','Qbc'): [['Gt2','Qbc']],
('Gt2','Mte'): [['Gt2','Mte']],
('Gt2','Gt3'): [['Gt2','Mte'],['Mte','Gt3']],
('Gt2','Caw'): [['Gt2','Mte'],['Mte','Caw']],
('Gt2','Mtw'): [['Gt2','Mte'],['Mte','Mtw']],
('Gt2','Otn'): [['Gt2','Mte'],['Mte','Mtw'],['Mtw','Otn']],
('Gt2','Ots'): [['Gt2','Mte'],['Mte','Mtw'],['Mtw','Ots']],

('Mte','Yar'): [['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Yar']],
('Mte','Syd'): [['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Syd']],
('Mte','Hal'): [['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Hal']],
('Mte','Tru'): [['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru']],
('Mte','Cht'): [['Mte','Qbc'],['Qbc','Frd'],['Frd','Cht']],
('Mte','Mon'): [['Mte','Qbc'],['Qbc','Frd'],['Frd','Mon']],
('Mte','Stj'): [['Mte','Qbc'],['Qbc','Frd'],['Frd','Stj']],
('Mte','Frd'): [['Mte','Qbc'],['Qbc','Frd']],
('Mte','Gt1'): [['Mte','Qbc'],['Qbc','Gt1']],
('Mte','Riv'): [['Mte','Qbc'],['Qbc','Riv']],
('Mte','Qbc'): [['Mte','Qbc']],
('Mte','Gt2'): [['Mte','Gt2']],
('Mte','Gt3'): [['Mte','Gt3']],
('Mte','Caw'): [['Mte','Caw']],
('Mte','Mtw'): [['Mte','Mtw']],
('Mte','Otn'): [['Mte','Mtw'],['Mtw','Otn']],
('Mte','Ots'): [['Mte','Mtw'],['Mtw','Ots']],

('Gt3','Yar'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Yar']],
('Gt3','Syd'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Syd']],
('Gt3','Hal'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Hal']],
('Gt3','Tru'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru']],
('Gt3','Cht'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Cht']],
('Gt3','Mon'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Mon']],
('Gt3','Stj'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Stj']],
('Gt3','Frd'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Frd']],
('Gt3','Gt1'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Gt1']],
('Gt3','Riv'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Riv']],
('Gt3','Qbc'): [['Gt3','Mte'],['Mte','Qbc']],
('Gt3','Gt2'): [['Gt3','Mte'],['Mte','Gt2']],
('Gt3','Mte'): [['Gt3','Mte']],
('Gt3','Caw'): [['Gt3','Caw']],
('Gt3','Mtw'): [['Gt3','Mtw']],
('Gt3','Otn'): [['Gt3','Mtw'],['Mtw','Otn']],
('Gt3','Ots'): [['Gt3','Mtw'],['Mtw','Ots']],

('Caw','Yar'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Yar']],
('Caw','Syd'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Syd']],
('Caw','Hal'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Hal']],
('Caw','Tru'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru']],
('Caw','Cht'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Cht']],
('Caw','Mon'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Mon']],
('Caw','Stj'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Stj']],
('Caw','Frd'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Frd']],
('Caw','Gt1'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Gt1']],
('Caw','Riv'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Riv']],
('Caw','Qbc'): [['Caw','Mte'],['Mte','Qbc']],
('Caw','Gt2'): [['Caw','Mte'],['Mte','Gt2']],
('Caw','Mte'): [['Caw','Mte']],
('Caw','Gt3'): [['Caw','Gt3']],
('Caw','Mtw'): [['Caw','Mtw']],
('Caw','Otn'): [['Caw','Mtw'],['Mtw','Otn']],
('Caw','Ots'): [['Caw','Mtw'],['Mtw','Ots']],

('Mtw','Yar'): [['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Yar']],
('Mtw','Syd'): [['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Syd']],
('Mtw','Hal'): [['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Hal']],
('Mtw','Tru'): [['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru']],
('Mtw','Cht'): [['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Cht']],
('Mtw','Mon'): [['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Mon']],
('Mtw','Stj'): [['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Stj']],
('Mtw','Frd'): [['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd']],
('Mtw','Gt1'): [['Mtw','Mte'],['Mte','Qbc'],['Qbc','Gt1']],
('Mtw','Riv'): [['Mtw','Mte'],['Mte','Qbc'],['Qbc','Riv']],
('Mtw','Qbc'): [['Mtw','Mte'],['Mte','Qbc']],
('Mtw','Gt2'): [['Mtw','Mte'],['Mte','Gt2']],
('Mtw','Mte'): [['Mtw','Mte']],
('Mtw','Gt3'): [['Mtw','Gt3']],
('Mtw','Caw'): [['Mtw','Caw']],
('Mtw','Otn'): [['Mtw','Otn']],
('Mtw','Ots'): [['Mtw','Ots']],

('Otn','Yar'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Yar']],
('Otn','Syd'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Syd']],
('Otn','Hal'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Hal']],
('Otn','Tru'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru']],
('Otn','Cht'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Cht']],
('Otn','Mon'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Mon']],
('Otn','Stj'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Stj']],
('Otn','Frd'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd']],
('Otn','Gt1'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Gt1']],
('Otn','Riv'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Riv']],
('Otn','Qbc'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Qbc']],
('Otn','Gt2'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Gt2']],
('Otn','Mte'): [['Otn','Mtw'],['Mtw','Mte']],
('Otn','Gt3'): [['Otn','Mtw'],['Mtw','Gt3']],
('Otn','Caw'): [['Otn','Mtw'],['Mtw','Caw']],
('Otn','Mtw'): [['Otn','Mtw']],
('Otn','Ots'): [['Otn','Ots']],

('Ots','Yar'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Yar']],
('Ots','Syd'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Syd']],
('Ots','Hal'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru'],['Tru','Hal']],
('Ots','Tru'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Tru']],
('Ots','Cht'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Cht']],
('Ots','Mon'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Mon']],
('Ots','Stj'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd'],['Frd','Stj']],
('Ots','Frd'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Frd']],
('Ots','Gt1'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Gt1']],
('Ots','Riv'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc'],['Qbc','Riv']],
('Ots','Qbc'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Qbc']],
('Ots','Gt2'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Gt2']],
('Ots','Mte'): [['Ots','Mtw'],['Mtw','Mte']],
('Ots','Gt3'): [['Ots','Mtw'],['Mtw','Gt3']],
('Ots','Caw'): [['Ots','Mtw'],['Mtw','Caw']],
('Ots','Mtw'): [['Ots','Mtw']],
('Ots','Otn'): [['Ots','Otn']]
}
```

Appendix C: Hybrid Arcs Dictionary

```
route_hy = {
  ('Yar','Hal'): [['Yar','Hal']],
  ('Yar','Tru'): [['Yar','Tru']],
  ('Yar','Syd'): [['Yar','Syd']],
  ('Yar','Mon'): [['Yar','Tru'],['Tru','Mon']],
  ('Yar','Cht'): [['Yar','Tru'],['Tru','Cht']],
  ('Yar','Stj'): [['Yar','Tru'],['Tru','Stj']],
  ('Yar','Frd'): [['Yar','Tru'],['Tru','Frd']],
  ('Yar','Gt1'): [['Yar','Tru'],['Tru','Frd'],['Frd','Gt1']],
  ('Yar','Riv'): [['Yar','Tru'],['Tru','Frd'],['Frd','Riv']],
  ('Yar','Qbc'): [['Yar','Tru'],['Tru','Qbc']],
  ('Yar','Gt2'): [['Yar','Tru'],['Tru','Qbc'],['Qbc','Gt2']],
  ('Yar','Mte'): [['Yar','Tru'],['Tru','Mte']],
  ('Yar','Gt3'): [['Yar','Tru'],['Tru','Mte'],['Mte','Gt3']],
  ('Yar','Caw'): [['Yar','Tru'],['Tru','Mte'],['Mte','Caw']],
  ('Yar','Mtw'): [['Yar','Tru'],['Tru','Mtw']],
  ('Yar','Otn'): [['Yar','Tru'],['Tru','Mtw'],['Mtw','Otn']],
  ('Yar','Ots'): [['Yar','Tru'],['Tru','Mtw'],['Mtw','Ots']],

  ('Hal','Yar'): [['Hal','Yar']],
  ('Hal','Tru'): [['Hal','Tru']],
  ('Hal','Syd'): [['Hal','Syd']],
  ('Hal','Mon'): [['Hal','Tru'],['Tru','Mon']],
  ('Hal','Cht'): [['Hal','Tru'],['Tru','Cht']],
  ('Hal','Stj'): [['Hal','Tru'],['Tru','Stj']],
  ('Hal','Frd'): [['Hal','Tru'],['Tru','Frd']],
  ('Hal','Gt1'): [['Hal','Tru'],['Tru','Frd'],['Frd','Gt1']],
  ('Hal','Riv'): [['Hal','Tru'],['Tru','Frd'],['Frd','Riv']],
  ('Hal','Qbc'): [['Hal','Tru'],['Tru','Qbc']],
  ('Hal','Gt2'): [['Hal','Tru'],['Tru','Qbc'],['Qbc','Gt2']],
  ('Hal','Mte'): [['Hal','Tru'],['Tru','Mte']],
  ('Hal','Gt3'): [['Hal','Tru'],['Tru','Mte'],['Mte','Gt3']],
  ('Hal','Caw'): [['Hal','Tru'],['Tru','Mte'],['Mte','Caw']],
  ('Hal','Mtw'): [['Hal','Tru'],['Tru','Mtw']],
  ('Hal','Otn'): [['Hal','Tru'],['Tru','Mtw'],['Mtw','Otn']],
  ('Hal','Ots'): [['Hal','Tru'],['Tru','Mtw'],['Mtw','Ots']],

  ('Syd','Hal'): [['Syd','Hal']],
  ('Syd','Tru'): [['Syd','Tru']],
  ('Syd','Yar'): [['Syd','Yar']],
  ('Syd','Mon'): [['Syd','Tru'],['Tru','Mon']],
  ('Syd','Cht'): [['Syd','Tru'],['Tru','Cht']],
  ('Syd','Stj'): [['Syd','Tru'],['Tru','Stj']],
  ('Syd','Frd'): [['Syd','Tru'],['Tru','Frd']],
  ('Syd','Gt1'): [['Syd','Tru'],['Tru','Frd'],['Frd','Gt1']],
  ('Syd','Riv'): [['Syd','Tru'],['Tru','Frd'],['Frd','Riv']],
  ('Syd','Qbc'): [['Syd','Tru'],['Tru','Frd'],['Frd','Qbc']],
  ('Syd','Gt2'): [['Syd','Tru'],['Tru','Qbc'],['Qbc','Gt2']],
  ('Syd','Mte'): [['Syd','Tru'],['Tru','Mte']],
  ('Syd','Gt3'): [['Syd','Tru'],['Tru','Mte'],['Mte','Gt3']],
  ('Syd','Caw'): [['Syd','Tru'],['Tru','Mte'],['Mte','Caw']],
  ('Syd','Mtw'): [['Syd','Tru'],['Tru','Mtw']],
  ('Syd','Otn'): [['Syd','Tru'],['Tru','Mtw'],['Mtw','Otn']],
  ('Syd','Ots'): [['Syd','Tru'],['Tru','Mtw'],['Mtw','Ots']],

  ('Tru','Hal'): [['Tru','Hal']],
('Tru','Tru'): [['Tru','Frd']],
('Tru','Syd'): [['Tru','Syd']],
('Tru','Mon'): [['Tru','Mon']],
('Tru','Cht'): [['Tru','Cht']],
('Tru','Stj'): [['Tru','Stj']],
('Tru','Frd'): [['Tru','Frd']],
('Tru','Gt1'): [['Tru','Frd'],['Frd','Gt1']],
('Tru','Riv'): [['Tru','Frd'],['Frd','Riv']],
('Tru','Qbc'): [['Tru','Qbc']],
('Tru','Gt2'): [['Tru','Qbc'],['Qbc','Gt2']],
('Tru','Mte'): [['Tru','Mte']],
('Tru','Gt3'): [['Tru','Mte'],['Mte','Gt3']],
('Tru','Caw'): [['Tru','Mte'],['Mte','Caw']],
('Tru','Mtw'): [['Tru','Mtw']],
('Tru','Otn'): [['Tru','Mtw'],['Mtw','Otn']],
('Tru','Ots'): [['Tru','Mtw'],['Mtw','Ots']],

('Mon','Yar'): [['Mon','Tru'],['Tru','Yar']],
('Mon','Syd'): [['Mon','Tru'],['Tru','Syd']],
('Mon','Hal'): [['Mon','Tru'],['Tru','Hal']],
('Mon','Tru'): [['Mon','Tru']],
('Mon','Cht'): [['Mon','Cht']],
('Mon','Stj'): [['Mon','Stj']],
('Mon','Frd'): [['Mon','Frd']],
('Mon','Gt1'): [['Mon','Frd'],['Frd','Gt1']],
('Mon','Riv'): [['Mon','Frd'],['Frd','Riv']],
('Mon','Qbc'): [['Mon','Frd'],['Frd','Qbc']],
('Mon','Gt2'): [['Mon','Frd'],['Frd','Qbc'],['Qbc','Gt2']],
('Mon','Mte'): [['Mon','Frd'],['Frd','Mte']],
('Mon','Gt3'): [['Mon','Frd'],['Frd','Mte'],['Mte','Gt3']],
('Mon','Caw'): [['Mon','Frd'],['Frd','Mte'],['Mte','Caw']],
('Mon','Mtw'): [['Mon','Frd'],['Frd','Mtw']],
('Mon','Otn'): [['Mon','Frd'],['Frd','Mtw'],['Mtw','Otn']],
('Mon','Ots'): [['Mon','Frd'],['Frd','Mtw'],['Mtw','Ots']],

('Cht','Yar'): [['Cht','Tru'],['Tru','Yar']],
('Cht','Syd'): [['Cht','Tru'],['Tru','Syd']],
('Cht','Hal'): [['Cht','Tru'],['Tru','Hal']],
('Cht','Tru'): [['Cht','Tru']],
('Cht','Mon'): [['Cht','Mon']],
('Cht','Stj'): [['Cht','Stj']],
('Cht','Frd'): [['Cht','Frd']],
('Cht','Gt1'): [['Cht','Frd'],['Frd','Gt1']],
('Cht','Riv'): [['Cht','Frd'],['Frd','Riv']],
('Cht','Qbc'): [['Cht','Frd'],['Frd','Qbc']],
('Cht','Gt2'): [['Cht','Frd'],['Frd','Qbc'],['Qbc','Gt2']],
('Cht','Mte'): [['Cht','Frd'],['Frd','Mte']],
('Cht','Gt3'): [['Cht','Frd'],['Frd','Mte'],['Mte','Gt3']],
('Cht','Caw'): [['Cht','Frd'],['Frd','Mte'],['Mte','Caw']],
('Cht','Mtw'): [['Cht','Frd'],['Frd','Mtw']],
('Cht','Otn'): [['Cht','Frd'],['Frd','Mtw'],['Mtw','Otn']],
('Cht','Ots'): [['Cht','Frd'],['Frd','Mtw'],['Mtw','Ots']],

('Stj','Yar'): [['Stj','Tru'],['Tru','Yar']],
('Stj','Syd'): [['Stj','Tru'],['Tru','Syd']],
('Stj','Hal'): [['Stj','Tru'],['Tru','Hal']],
('Stj','Tru'): [['Stj','Tru']],
('Stj','Cht'): [['Stj','Cht']],
('Stj','Mon'): [['Stj','Mon']],
('Stj','Frd'): [['Stj','Frd']],
('Stj','Gt1'): [['Stj','Frd'],['Frd','Gt1']],
('Stj','Riv'): [['Stj','Frd'],['Frd','Riv']],
('Stj','Qbc'): [['Stj','Frd'],['Frd','Qbc']],
('Stj','Gt2'): [['Stj','Frd'],['Frd','Qbc'],['Qbc','Gt2']],
('Stj','Mte'): [['Stj','Frd'],['Frd','Mte']],
('Stj','Gt3'): [['Stj','Frd'],['Frd','Mte'],['Mte','Gt3']],
('Stj','Caw'): [['Stj','Frd'],['Frd','Mte'],['Mte','Caw']],
('Stj','Mtw'): [['Stj','Frd'],['Frd','Mtw']],
('Stj','Otn'): [['Stj','Frd'],['Frd','Mtw'],['Mtw','Otn']],
('Stj','Ots'): [['Stj','Frd'],['Frd','Mtw'],['Mtw','Ots']],

('Frd','Yar'): [['Frd','Tru'],['Tru','Yar']],
('Frd','Syd'): [['Frd','Tru'],['Tru','Syd']],
('Frd','Hal'): [['Frd','Tru'],['Tru','Hal']],
('Frd','Tru'): [['Frd','Tru']],
('Frd','Cht'): [['Frd','Cht']],
('Frd','Mon'): [['Frd','Mon']],
('Frd','Stj'): [['Frd','Stj']],
('Frd','Gt1'): [['Frd','Gt1']],
('Frd','Riv'): [['Frd','Riv']],
('Frd','Qbc'): [['Frd','Qbc']],
('Frd','Gt2'): [['Frd','Qbc'],['Qbc','Gt2']],
('Frd','Mte'): [['Frd','Mte']],
('Frd','Gt3'): [['Frd','Mte'],['Mte','Gt3']],
('Frd','Caw'): [['Frd','Mte'],['Mte','Caw']],
('Frd','Mtw'): [['Frd','Mtw']],
('Frd','Otn'): [['Frd','Mtw'],['Mtw','Otn']],
('Frd','Ots'): [['Frd','Mtw'],['Mtw','Ots']],

('Gt1','Yar'): [['Gt1','Frd'],['Frd','Tru'],['Tru','Yar']],
('Gt1','Syd'): [['Gt1','Frd'],['Frd','Tru'],['Tru','Syd']],
('Gt1','Hal'): [['Gt1','Frd'],['Frd','Tru'],['Tru','Hal']],
('Gt1','Tru'): [['Gt1','Frd'],['Frd','Tru']],
('Gt1','Cht'): [['Gt1','Frd'],['Frd','Cht']],
('Gt1','Mon'): [['Gt1','Frd'],['Frd','Mon']],
('Gt1','Stj'): [['Gt1','Frd'],['Frd','Stj']],
('Gt1','Frd'): [['Gt1','Frd']],
('Gt1','Riv'): [['Gt1','Riv']],
('Gt1','Qbc'): [['Gt1','Qbc']],
('Gt1','Mte'): [['Gt1','Qbc'],['Qbc','Mte']],
('Gt1','Gt2'): [['Gt1','Qbc'],['Qbc','Gt2']],
('Gt1','Gt3'): [['Gt1','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
('Gt1','Caw'): [['Gt1','Qbc'],['Qbc','Mte'],['Mte','Caw']],
('Gt1','Mtw'): [['Gt1','Qbc'],['Qbc','Mtw']],
('Gt1','Otn'): [['Gt1','Qbc'],['Qbc','Mtw'],['Mtw','Otn']],
('Gt1','Ots'): [['Gt1','Qbc'],['Qbc','Mtw'],['Mtw','Ots']],

('Riv','Yar'): [['Riv','Frd'],['Frd','Tru'],['Tru','Yar']],
('Riv','Syd'): [['Riv','Frd'],['Frd','Tru'],['Tru','Syd']],
('Riv','Hal'): [['Riv','Frd'],['Frd','Tru'],['Tru','Hal']],
('Riv','Tru'): [['Riv','Frd'],['Frd','Tru']],
('Riv','Cht'): [['Riv','Frd'],['Frd','Cht']],
('Riv','Mon'): [['Riv','Frd'],['Frd','Mon']],
('Riv','Stj'): [['Riv','Frd'],['Frd','Stj']],
('Riv','Frd'): [['Riv','Frd']],
('Riv','Gt1'): [['Riv','Gt1']],
('Riv','Qbc'): [['Riv','Qbc']],
('Riv','Mte'): [['Riv','Qbc'],['Qbc','Mte']],
('Riv','Gt2'): [['Riv','Qbc'],['Qbc','Gt2']],
('Riv','Gt3'): [['Riv','Qbc'],['Qbc','Mte'],['Mte','Gt3']],
('Riv','Caw'): [['Riv','Qbc'],['Qbc','Mte'],['Mte','Caw']],
('Riv','Mtw'): [['Riv','Qbc'],['Qbc','Mtw']],
('Riv','Otn'): [['Riv','Qbc'],['Qbc','Mtw'],['Mtw','Otn']],
('Riv','Ots'): [['Riv','Qbc'],['Qbc','Mtw'],['Mtw','Ots']],

('Qbc','Yar'): [['Qbc','Tru'],['Tru','Yar']],
('Qbc','Syd'): [['Qbc','Tru'],['Tru','Syd']],
('Qbc','Hal'): [['Qbc','Tru'],['Tru','Hal']],
('Qbc','Tru'): [['Qbc','Tru']],
('Qbc','Cht'): [['Qbc','Frd'],['Frd','Cht']],
('Qbc','Mon'): [['Qbc','Frd'],['Frd','Mon']],
('Qbc','Stj'): [['Qbc','Frd'],['Frd','Stj']],
('Qbc','Frd'): [['Qbc','Frd']],
('Qbc','Gt1'): [['Qbc','Gt1']],
('Qbc','Riv'): [['Qbc','Riv']],
('Qbc','Mte'): [['Qbc','Mte']],
('Qbc','Gt2'): [['Qbc','Gt2']],
('Qbc','Gt3'): [['Qbc','Mte'],['Mte','Gt3']],
('Qbc','Caw'): [['Qbc','Mte'],['Mte','Caw']],
('Qbc','Mtw'): [['Qbc','Mtw']],
('Qbc','Otn'): [['Qbc','Mtw'],['Mtw','Otn']],
('Qbc','Ots'): [['Qbc','Mtw'],['Mtw','Ots']],

('Gt2','Yar'): [['Gt2','Qbc'],['Qbc','Tru'],['Tru','Yar']],
('Gt2','Syd'): [['Gt2','Qbc'],['Qbc','Tru'],['Tru','Syd']],
('Gt2','Hal'): [['Gt2','Qbc'],['Qbc','Tru'],['Tru','Hal']],
('Gt2','Tru'): [['Gt2','Qbc'],['Qbc','Tru']],
('Gt2','Cht'): [['Gt2','Qbc'],['Qbc','Frd'],['Frd','Cht']],
('Gt2','Mon'): [['Gt2','Qbc'],['Qbc','Frd'],['Frd','Mon']],
('Gt2','Stj'): [['Gt2','Qbc'],['Qbc','Frd'],['Frd','Stj']],
('Gt2','Frd'): [['Gt2','Qbc'],['Qbc','Frd']],
('Gt2','Gt1'): [['Gt2','Qbc'],['Qbc','Gt1']],
('Gt2','Riv'): [['Gt2','Qbc'],['Qbc','Riv']],
('Gt2','Qbc'): [['Gt2','Qbc']],
('Gt2','Mte'): [['Gt2','Mte']],
('Gt2','Gt3'): [['Gt2','Mte'],['Mte','Gt3']],
('Gt2','Caw'): [['Gt2','Mte'],['Mte','Caw']],
('Gt2','Mtw'): [['Gt2','Mte'],['Mte','Mtw']],
('Gt2','Otn'): [['Gt2','Mte'],['Mte','Mtw'],['Mtw','Otn']],
('Gt2','Ots'): [['Gt2','Mte'],['Mte','Mtw'],['Mtw','Ots']],


('Mte','Yar'): [['Mte','Tru'],['Tru','Yar']],
('Mte','Syd'): [['Mte','Tru'],['Tru','Syd']],
('Mte','Hal'): [['Mte','Tru'],['Tru','Hal']],
('Mte','Tru'): [['Mte','Tru']],
('Mte','Cht'): [['Mte','Frd'],['Frd','Cht']],
('Mte','Mon'): [['Mte','Frd'],['Frd','Mon']],
('Mte','Stj'): [['Mte','Frd'],['Frd','Stj']],
('Mte','Frd'): [['Mte','Frd']],
('Mte','Gt1'): [['Mte','Qbc'],['Qbc','Gt1']],
('Mte','Riv'): [['Mte','Qbc'],['Qbc','Riv']],
('Mte','Qbc'): [['Mte','Qbc']],
('Mte','Gt2'): [['Mte','Gt2']],
('Mte','Gt3'): [['Mte','Gt3']],
('Mte','Caw'): [['Mte','Caw']],
('Mte','Mtw'): [['Mte','Mtw']],
('Mte','Otn'): [['Mte','Mtw'],['Mtw','Otn']],
('Mte','Ots'): [['Mte','Mtw'],['Mtw','Ots']],

('Gt3','Yar'): [['Gt3','Mte'],['Mte','Tru'],['Tru','Yar']],
('Gt3','Syd'): [['Gt3','Mte'],['Mte','Tru'],['Tru','Syd']],
('Gt3','Hal'): [['Gt3','Mte'],['Mte','Tru'],['Tru','Hal']],
('Gt3','Tru'): [['Gt3','Mte'],['Mte','Tru']],
('Gt3','Cht'): [['Gt3','Mte'],['Mte','Frd'],['Frd','Cht']],
('Gt3','Mon'): [['Gt3','Mte'],['Mte','Frd'],['Frd','Mon']],
('Gt3','Stj'): [['Gt3','Mte'],['Mte','Frd'],['Frd','Stj']],
('Gt3','Frd'): [['Gt3','Mte'],['Mte','Frd']],
('Gt3','Gt1'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Gt1']],
('Gt3','Riv'): [['Gt3','Mte'],['Mte','Qbc'],['Qbc','Riv']],
('Gt3','Qbc'): [['Gt3','Mte'],['Mte','Qbc']],
('Gt3','Gt2'): [['Gt3','Mte'],['Mte','Gt2']],
('Gt3','Mte'): [['Gt3','Mte']],
('Gt3','Caw'): [['Gt3','Caw']],
('Gt3','Mtw'): [['Gt3','Mtw']],
('Gt3','Otn'): [['Gt3','Mtw'],['Mtw','Otn']],
('Gt3','Ots'): [['Gt3','Mtw'],['Mtw','Ots']],

('Caw','Yar'): [['Caw','Mte'],['Mte','Tru'],['Tru','Yar']],
('Caw','Syd'): [['Caw','Mte'],['Mte','Tru'],['Tru','Syd']],
('Caw','Hal'): [['Caw','Mte'],['Mte','Tru'],['Tru','Hal']],
('Caw','Tru'): [['Caw','Mte'],['Mte','Tru']],
('Caw','Cht'): [['Caw','Mte'],['Mte','Frd'],['Frd','Cht']],
('Caw','Mon'): [['Caw','Mte'],['Mte','Frd'],['Frd','Mon']],
('Caw','Stj'): [['Caw','Mte'],['Mte','Frd'],['Frd','Stj']],
('Caw','Frd'): [['Caw','Mte'],['Mte','Frd']],
('Caw','Gt1'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Gt1']],
('Caw','Riv'): [['Caw','Mte'],['Mte','Qbc'],['Qbc','Riv']],
('Caw','Qbc'): [['Caw','Mte'],['Mte','Qbc']],
('Caw','Gt2'): [['Caw','Mte'],['Mte','Gt2']],
('Caw','Mte'): [['Caw','Mte']],
('Caw','Gt3'): [['Caw','Gt3']],
('Caw','Mtw'): [['Caw','Mtw']],
('Caw','Otn'): [['Caw','Mtw'],['Mtw','Otn']],
('Caw','Ots'): [['Caw','Mtw'],['Mtw','Ots']],

('Mtw','Yar'): [['Mtw','Tru'],['Tru','Yar']],
('Mtw','Syd'): [['Mtw','Tru'],['Tru','Syd']],
('Mtw','Hal'): [['Mtw','Tru'],['Tru','Hal']],
('Mtw','Tru'): [['Mtw','Tru']],
('Mtw','Cht'): [['Mtw','Frd'],['Frd','Cht']],
('Mtw','Mon'): [['Mtw','Frd'],['Frd','Mon']],
('Mtw','Stj'): [['Mtw','Frd'],['Frd','Stj']],
('Mtw','Frd'): [['Mtw','Frd']],
('Mtw','Gt1'): [['Mtw','Qbc'],['Qbc','Gt1']],
('Mtw','Riv'): [['Mtw','Qbc'],['Qbc','Riv']],
('Mtw','Qbc'): [['Mtw','Qbc']],
('Mtw','Gt2'): [['Mtw','Mte'],['Mte','Gt2']],
('Mtw','Mte'): [['Mtw','Mte']],
('Mtw','Gt3'): [['Mtw','Gt3']],
('Mtw','Caw'): [['Mtw','Caw']],
('Mtw','Otn'): [['Mtw','Otn']],
('Mtw','Ots'): [['Mtw','Ots']],

('Otn','Yar'): [['Otn','Mtw'],['Mtw','Tru'],['Tru','Yar']],
('Otn','Syd'): [['Otn','Mtw'],['Mtw','Tru'],['Tru','Syd']],
('Otn','Hal'): [['Otn','Mtw'],['Mtw','Tru'],['Tru','Hal']],
('Otn','Tru'): [['Otn','Mtw'],['Mtw','Tru']],
('Otn','Cht'): [['Otn','Mtw'],['Mtw','Frd'],['Frd','Cht']],
('Otn','Mon'): [['Otn','Mtw'],['Mtw','Frd'],['Frd','Mon']],
('Otn','Stj'): [['Otn','Mtw'],['Mtw','Frd'],['Frd','Stj']],
('Otn','Frd'): [['Otn','Mtw'],['Mtw','Frd']],
('Otn','Gt1'): [['Otn','Mtw'],['Mtw','Qbc'],['Qbc','Gt1']],
('Otn','Riv'): [['Otn','Mtw'],['Mtw','Qbc'],['Qbc','Riv']],
('Otn','Qbc'): [['Otn','Mtw'],['Mtw','Qbc']],
('Otn','Gt2'): [['Otn','Mtw'],['Mtw','Mte'],['Mte','Gt2']],
('Otn','Mte'): [['Otn','Mtw'],['Mtw','Mte']],
('Otn','Gt3'): [['Otn','Mtw'],['Mtw','Gt3']],
('Otn','Caw'): [['Otn','Mtw'],['Mtw','Caw']],
('Otn','Mtw'): [['Otn','Mtw']],
('Otn','Ots'): [['Otn','Ots']],

('Ots','Yar'): [['Ots','Mtw'],['Mtw','Tru'],['Tru','Yar']],
('Ots','Syd'): [['Ots','Mtw'],['Mtw','Tru'],['Tru','Syd']],
('Ots','Hal'): [['Ots','Mtw'],['Mtw','Tru'],['Tru','Hal']],
('Ots','Tru'): [['Ots','Mtw'],['Mtw','Tru']],
('Ots','Cht'): [['Ots','Mtw'],['Mtw','Frd'],['Frd','Cht']],
('Ots','Mon'): [['Ots','Mtw'],['Mtw','Frd'],['Frd','Mon']],
('Ots','Stj'): [['Ots','Mtw'],['Mtw','Frd'],['Frd','Stj']],
('Ots','Frd'): [['Ots','Mtw'],['Mtw','Frd']],
('Ots','Gt1'): [['Ots','Mtw'],['Mtw','Qbc'],['Qbc','Gt1']],
('Ots','Riv'): [['Ots','Mtw'],['Mtw','Qbc'],['Qbc','Riv']],
('Ots','Qbc'): [['Ots','Mtw'],['Mtw','Qbc']],
('Ots','Gt2'): [['Ots','Mtw'],['Mtw','Mte'],['Mte','Gt2']],
('Ots','Mte'): [['Ots','Mtw'],['Mtw','Mte']],
('Ots','Gt3'): [['Ots','Mtw'],['Mtw','Gt3']],
('Ots','Caw'): [['Ots','Mtw'],['Mtw','Caw']],
('Ots','Mtw'): [['Ots','Mtw']],
('Ots','Otn'): [['Ots','Otn']],
}
```

Appendix D: Parameters File

```
avg_speed = 80

debug = 'No'

# ------------------------------------------------------------------------------
# Load sizes based in the simulation ((Fraction of 40 foot container)

load_size = [0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.25,0.25,0.25,0.25,0.5,0.5,1]
#Regular loads distribution

#load_size = [0,0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.125,0.25,0.25,0.25,0.25,0.5,0.5,0.5]
#for small loads


#load_size = [1,1,1,1,1,1,1,1,0.5,0.5,0.5,0.5,0.25,0.25,0.125,0.125]
#for large loads

# ------------------------------------------------------------------------------


if debug == 'No':
   # Nodes in the entire network
   Node = ['Hal','Tru','Syd','Yar','Cht','Mon','Frd','Stj','Gt1','Riv','Qbc','Gt2','Mte','Gt3','Caw','Mtw','Otn','Ots']

if debug == 'Yes':
   # Nodes in the network
   Node = ['Hal','Tru','Syd','Yar','Cht','Mon','Frd','Stj','Gt1','Riv','Qbc']


# ------------------------------------------------------------------------------
# Number of requests from each node to each destination

Req = 10
```

Appendix E: Data Generator File

```
from CO_TA import *
from HY_TA import *
from PI_TA import *
from Param import *
from random import *

import csv

# -------------------------------------------------------------------------------
# The follwoing key and values are used to call the route_pi dictionary

# General form of the python dictionary dict = { 'key' : value }
# Form of dictionary used in the program  route_xx = { ('key m',key n'): [[Node m, Node q],...,[Node r, Node n] }

key_pi = route_pi.keys()
key_hy = route_hy.keys()
key_co = route_co.keys()
# Puts all the dictionary keys in the key list. There is no need to iterate over the dictionary and collect all the keys

value_pi = route_pi.values()
value_hy = route_hy.values()
value_co = route_co.values()
# Puts all the dictionary values int the value list.There is no need iterate over the dictionary and collect all the
values

# -------------------------------------------------------------------------------
# Thses list are created to count the number of loads between each source and destination in the pi, hy , and co
network
# First list is temp list used at each iteration

temp_count_pi = []
count_pi = []

temp_count_hy = []
count_hy = []

temp_count_co = []
count_co = []

# -------------------------------------------------------------------------------
# Following two for loops create the empty counter list
# General form of the counter list
# count_xx[i][0] --> Source
# count_xx[i][1] --> Destination
# count_xx[i][2] --> Counter

for p in range(len(Node)):
   for q in range(len(Node)):
      temp_count_pi = []
      temp_count_pi.append(Node[p])
      temp_count_pi.append(Node[q])
      temp_count_pi.append(0)
      count_pi.append(temp_count_pi)

for p in range(len(Node)):
   for q in range(len(Node)):
      temp_count_hy = []
      temp_count_hy.append(Node[p])
      temp_count_hy.append(Node[q])
```

```
        temp_count_hy.append(0)
        count_hy.append(temp_count_hy)

for p in range(len(Node)):
    for q in range(len(Node)):
        temp_count_co = []
        temp_count_co.append(Node[p])
        temp_count_co.append(Node[q])
        temp_count_co.append(0)
        count_co.append(temp_count_co)

# -------------------------------------------------------------------------------
# First list is temp list used at each iteration. 2nd list is complete list of all the  loads.

temp_pi = []
list_pi = []


temp_hy = []
list_hy = []

temp_co = []
list_co = []

# -------------------------------------------------------------------------------

new_counter = 1
for r in range(Req):
#The entire process will repeat accroding to the number of requets

    for k in range(len(Node)):

        # General for loops to iterate over the set of nodes in the network
        for j in range(len(Node)):

          new_counter +=1

          # size of loads randomly chosen from the load size list in the topo.py
          size = choice(load_size)

          for i in range(len(route_pi)):

              new_counter = 1

              # -------------------------------------------------------------------------------
              # Generation of list of loads for PI senario
              if key_pi[i][0] == Node[k] and key_pi[i][1]== Node[j]:



                mehran = 0
                for l in range(len(value_pi[i])):

                  # -------------------------------------------------------------------------------
                  # Generate counter values
                  # when there is a load from spesific source to specific destination,
                  # the value of the counter for that source and destination is incremented

                  for pi in range(len(count_pi)):
                      if count_pi[pi][0] == value_pi[i][l][0] and  count_pi[pi][1] == value_pi[i][l][1]:
                          count_pi[pi][2] = count_pi[pi][2]+1
    # -------------------------------------------------------------------------------
    # Insertion of values in to the PI load list
        mehran =3
        if (mehran%2==1):
          temp_pi = []
          temp_pi.append(value_pi[i][l][0])
          temp_pi.append(value_pi[i][l][1])
          temp_pi.append(count_pi[pi][2])
          temp_pi.append(size)
          list_pi.append(temp_pi)

          temp_pi = []

# -------------------------------------------------------------------------------
# Generation of list of loads for Hybrid senario
if key_hy[i][0] == Node[k] and key_hy[i][1]== Node[j]:

  mehran = 0
  for l in range(len(value_hy[i])):

    # -------------------------------------------------------------------------------
    # Generate counter values
    # when there is a load from spesific source to specific destination,
    # the value of the counter for that source and destination is incremented

    for hy in range(len(count_hy)):
      if count_hy[hy][0] == value_hy[i][l][0] and  count_hy[hy][1] == value_hy[i][l][1]:
        count_hy[hy][2] = count_hy[hy][2]+1
    # -------------------------------------------------------------------------------
    # Insertion of values in to the Hybrid load list
        mehran =3
        if (mehran%2==1):
          temp_hy = []
          temp_hy.append(value_hy[i][l][0])
          temp_hy.append(value_hy[i][l][1])
          temp_hy.append(count_hy[hy][2])
          temp_hy.append(size)
          list_hy.append(temp_hy)
          temp_hy = []

# -------------------------------------------------------------------------------
# Generation of list of loads for Conventional senario
if key_co[i][0] == Node[k] and key_co[i][1]== Node[j]:

  mehran = 0
  for l in range(len(value_co[i])):

    # -------------------------------------------------------------------------------
    # Generate counter values
    # when there is a load from spesific source to specific destination,
    # the value of the counter for that source and destination is incremented

    for co in range(len(count_co)):
      if count_co[co][0] == value_co[i][l][0] and  count_co[co][1] == value_co[i][l][1]:
        count_co[co][2] = count_co[co][2]+1
    # -------------------------------------------------------------------------------
    # Insertion of values in to the conventional load list
        mehran +=1
        if (mehran%2==1):
                temp_co = []
                temp_co.append(value_co[i][l][0])
                temp_co.append(value_co[i][l][1])
                temp_co.append(count_co[co][2])
                temp_co.append(size)
                list_co.append(temp_co)
                temp_co = []

# -------------------------------------------------------------------------------

arc_pi_size = []

for i in range(len(count_pi)):
    if count_pi[i][2] >= 1:
        arc_pi_size.append(count_pi[i])

arc_hy_size = []
for i in range(len(count_hy)):
    if count_hy[i][2] >= 1:
        arc_hy_size.append(count_hy[i])

arc_co_size = []
for i in range(len(count_co)):
    if count_co[i][2] >= 1:
        arc_co_size.append(count_co[i])


b = open('C:/Users/usr1/Desktop/Loads/Arc_PI_size.csv', 'wb')
a = csv.writer(b)
a.writerows(arc_pi_size)
b.close()

b = open('C:/Users/usr1/Desktop/Loads/Arc_HY_size.csv', 'wb')
a = csv.writer(b)
a.writerows(arc_hy_size)
b.close()

b = open('C:/Users/usr1/Desktop/Loads/Arc_CO_size.csv', 'wb')
a = csv.writer(b)
a.writerows(arc_co_size)
b.close()



arc_pi = []
temp_arc_pi = []

for i in range(len(count_pi)):
    if count_pi[i][2] >= 1:
        temp_arc_pi = []
        temp_arc_pi.append(count_pi[i][0])
        temp_arc_pi.append(count_pi[i][1])
        arc_pi.append(temp_arc_pi)

arc_hy = []
temp_arc_hy = []
for i in range(len(count_hy)):
    if count_hy[i][2] >= 1:
        temp_arc_hy = []
        temp_arc_hy.append(count_hy[i][0])
        temp_arc_hy.append(count_hy[i][1])
        arc_hy.append(temp_arc_hy)

arc_co = []
temp_arc_co = []
for i in range(len(count_co)):
    if count_co[i][2] >= 1:
        temp_arc_co = []
        temp_arc_co.append(count_co[i][0])
        temp_arc_co.append(count_co[i][1])
        arc_co.append(temp_arc_co)


b = open('C:/Users/usr1/Desktop/Loads/Act_arc_PI.csv', 'wb')
a = csv.writer(b)
a.writerows(arc_pi)
b.close()

b = open('C:/Users/usr1/Desktop/Loads/Act_arc_HY.csv', 'wb')
a = csv.writer(b)
a.writerows(arc_hy)
b.close()

b = open('C:/Users/usr1/Desktop/Loads/Act_arc_CO.csv', 'wb')
a = csv.writer(b)
a.writerows(arc_co)
b.close()

b = open('C:/Users/usr1/Desktop/Loads/Req_PI.csv', 'wb')
a = csv.writer(b)
a.writerows(list_pi)
b.close()

b = open('C:/Users/usr1/Desktop/Loads/Req_HY.csv', 'wb')
a = csv.writer(b)
a.writerows(list_hy)
b.close()

b = open('C:/Users/usr1/Desktop/Loads/Req_CO.csv', 'wb')
a = csv.writer(b)
a.writerows(list_co)
b.close()
```

Appendix F: GLPK Code of Packing MIP

```
set Node := {'Hal','Tru','Syd','Yar','Cht','Mon','Frd','Stj','Gt1','Riv','Qbc','Gt2','Mte','Gt3','Caw','Mtw','Otn','Ots'};
param q, integer, >=1;
set Req := {1..q};

param k, integer, >=1;
set Cont := {1..k};

set Req_Set, dimen 3;

#set Req_Set:= setof{ i in Node, j in Node, r in Req} (i,j,r);

#------------------------ Parameters-----------------------------------
ax_req_pi
var total;
#------------------------ Obj function-----------------------------------

minimize ObjZ:sum{i in Node, j in Node, c in Cont}con_n[i,j,c];

#------------------------ Constraints-----------------------------------

con1{i in Node, j in Node, r in Req: i <> j }: sum{c in Cont}as[i,j,r,c] = 1;

con2{i in Node, j in Node, c in Cont: i <> j }: sum{r in Req}as[i,j,r,c]*REQ_SIZE[i,j,r] <= CAPACITY*con_n[i,j,c];

con3{i in Node, j in Node}:sum{c in Cont}con_n[i,j,c] = cont[i,j];

con4:  sum{i in Node, j in Node}cont[i,j] = total;

solve;

#for {i in Node, j in Node, r in Req, c in Cont}{

# for {{0}: as[i,j, r, c] == 1} {
 #printf"\n  %s %s %s  %s %s %d %s %d ","From " ,i," to ",j," req ",r," in cont ",c ;
 #}}
#printf{i in Node, j in Node, r in Req, c in Cont} "\n  %s %d %s  %d %s %d %s %d %d "," From " ,i," to ",j," req ",r," in
cont ",c, as[i,j, r, c];
printf "\n\n";

printf{i in Node, j in Node}" \n Num cont from  %s %s %s %s %d",i," to ",j, " -->", sum{c in Cont}con_n[i,j,c];
printf "\n\n";
#printf{i in Node, j in Node, r in Req , c in Cont}REQ_SIZE[i,j,r];
printf " %d \n\n", total;
data;

param q := 66;
param k := 30;

end;
```

Appendix G: Packing Model Calling Algorithm

```
from gurobipy import *
from random import *
import csv

# Nodes in the entire network
Node = ['Hal','Tru','Syd','Yar','Cht','Mon','Frd','Stj','Gt1','Riv','Qbc','Gt2','Mte','Gt3','Caw','Mtw','Otn','Ots']


#from Generator2 import *

#Model 1 --> Consolidation Model

#------------------------------ Data---------------------------------

LP_Debug = "NO"

# Set of Containes
Cont_pi = 95
Cont_hy = 30
Cont_co = 5

#create a new model for PI senario
mpi = Model("model1")

#create a new model for HY senario
mhy = Model("model1")

#create a new model for CO senario
mco = Model("model1")


#create variables
assign = {}

cont_pi = {}
cont_hy = {}
cont_co = {}

#Capacity of the containers
CAPACITY = 1

#----------------Reads DATA from the lists------------------------------

arc_pi = []

with open('Act_arc_PI.csv', 'rb') as f:
   reader = csv.reader(f)
   for row in reader:
      arc_pi.append(row)

arc_hy = []

with open('Act_arc_HY.csv', 'rb') as f:
   reader = csv.reader(f)
   for row in reader:
      arc_hy.append(row)
arc_co = []
```

```
with open('Act_arc_CO.csv', 'rb') as f:
    reader = csv.reader(f)
    for row in reader:
        arc_co.append(row)

list_pi = []

with open('Req_pi.csv', 'rb') as f:
    reader = csv.reader(f)
    for row in reader:
        source = row[0]
        des = row[1]
        req = float(row[2])
        si = float(row[3])
        p = (source, des, req, si)
        list_pi.append(p)

REQ_SET_PI = list_pi

list_hy = []
with open('Req_hy.csv', 'rb') as f:
    reader = csv.reader(f)
    for row in reader:
        source = row[0]
        des = row[1]
        req = float(row[2])
        si = float(row[3])
        p = (source, des, req, si)
        list_hy.append(p)

REQ_SET_HY = list_hy

list_co = []

with open('Req_co.csv', 'rb') as f:
    reader = csv.reader(f)
    for row in reader:
        source = row[0]
        des = row[1]
        req = float(row[2])
        si = float(row[3])
        p = (source, des, req, si)
        list_co.append(p)


REQ_SET_CO = list_co
#---------------------------------------------------------------------

# Final result for calculated number of containers for each senario
Final_Cont_PI = []
Final_Cont_HY = []
Final_Cont_CO = []

#--------------------------------PI Section ---------------------------

L_SIZE = []

def loads_pi (i,j):
    s_d_loads = []
    for p in range(len(REQ_SET_PI)):
        if REQ_SET_PI[p][0] == i and REQ_SET_PI[p][1] == j:
```

```
        s_d_loads.append(REQ_SET_PI[p][3])
        #print s_d_loads
    return s_d_loads


#--------------Calling the optimizer inside the nested loops---------------
num_optimization = 1
for i in Node:
    for j in Node:
        if [i,j] in arc_pi:
            print ""
            print ""
            print "PI Senario"
            print "Optimization number", num_optimization, " of ", len(arc_pi)
            print ""
            print "####################################################"
            print "Min number of containers problem between", i ," to ", j
            print "####################################################"
            L_SIZE = []
            L_SIZE = loads_pi (i,j)
            #print  "Here is the list of loads :", L_SIZE

            print L_SIZE
            #------------------------- Variables---------------------------

            mpi.update()
            # Asignment of request r to container c from node i to node j
            assign = {}

            var1 =0
            for r in range(len(L_SIZE)):
                  for c in range(Cont_pi):
                      assign[r,c] = mpi.addVar(ub = 1, lb = 0 , vtype=GRB.BINARY,  name= 'var1')
                      var1 += 1
            mpi.update()

            cont_pi = {}

            var2 =1
            for c in range(Cont_pi):

                  cont_pi[c] = mpi.addVar(ub = 1, lb = 0 , vtype=GRB.BINARY,  name='var2')
                  var2 +=1

            mpi.update()

            #------------------------- Constraints-------------------------
            counter1 = 0
            for r in range(len(L_SIZE)):
                  for c in range(Cont_pi):

                        mpi.addConstr(quicksum(assign[r,c] for c in range(Cont_pi)), GRB.EQUAL , 1, name = 'counter1' )
                        counter1 +=1
                        #print 'cont_1_%s_%s' % (r,c)

            mpi.update()
            # setting the arc sizes

            counter2 = 0
            for r in range(len(L_SIZE)):
                for c in range(Cont_pi):
```

```
                mpi.addConstr(quicksum(assign[r,c]* L_SIZE[r] for r in range(len(L_SIZE))),GRB.LESS_EQUAL,
CAPACITY* cont_pi[c], name ='counter2')
                counter2 +=1
                #print counter2

            mpi.update()


            #------------------------- ObjZ---------------------------------
            mpi.setObjective(quicksum(cont_pi[c] for c in range(Cont_pi)) , GRB.MINIMIZE)
            mpi.optimize()

            #print "Value of the objective function: ",mpi.objval
            Final_Cont_PI = []
            obj = mpi.objval
            p=(i,j,obj)
            Final_Cont_PI.append(p)

            #------------------Model Clean up Section------------------

            mpi.update()
            num_optimization +=1

            import csv
            b = open('C:/Users/Mehran/Dropbox/Masters/Physical Internet/Simulation Models/Data/Static/M1_PI.csv',
'ab')
            a = csv.writer(b)
            a.writerows(Final_Cont_PI)
            b.close()


#-------------------------Hybrid Section ---------------------------------

L_SIZE = []

def loads_hy (i,j):
    s_d_loads = []
    for p in range(len(REQ_SET_HY)):
        if REQ_SET_HY[p][0] == i and REQ_SET_HY[p][1] == j:
            s_d_loads.append(REQ_SET_HY[p][3])
            #print s_d_loads
    return s_d_loads



#--Calling the optimizer for Hybrid senario inside the nested loops------
num_optimization = 1
for i in Node:
    for j in Node:
        if [i,j] in arc_hy:

            print ""
            print "Hybrid Senario"
            print "Optimization number", num_optimization, " of ", len(arc_hy)
            print ""
            print "####################################################"
            print "Min number of containers problem between", i ," to ", j
            print "####################################################"
            L_SIZEP = L_SIZE
            L_SIZE = []
            L_SIZE = loads_hy (i,j)
            #print  "Here is the list of loads :", L_SIZE


            #------------------------- Variables---------------------------
            #for i in range(Cont):
            #    for j in range(len(L_SIZEP)):
            #        mhy.remove(mhy.getConstrs()[0])


            mhy.update()
            # Asignment of request r to container c from node i to node j
            assign = {}

            var1 =0
            for r in range(len(L_SIZE)):
                  for c in range(Cont_hy):
                      assign[r,c] = mhy.addVar(ub = 1, lb = 0 , vtype=GRB.BINARY,  name='var1')
                      var1 +=1
            mhy.update()

            var2 =0
            cont_hy = {}
            for c in range(Cont_hy):
                cont_hy[c] = mhy.addVar(ub = 1, lb = 0 , vtype=GRB.BINARY,  name='var2')
                var2 +=1

            mhy.update()

            #------------------------- Constraints-------------------------

            counter1 = 0
            for r in range(len(L_SIZE)):
                for c in range(Cont_hy):
                      mhy.addConstr(quicksum(assign[r,c] for c in range(Cont_hy)), GRB.EQUAL , 1, name='counter1')
                      counter1 +=1
                      #print 'cont_1_%s_%s' % (r,c)

            mhy.update()
            # setting the arc sizes

            counter2 = 0
            for r in range(len(L_SIZE)):
               for c in range(Cont_hy):
                mhy.addConstr(quicksum(assign[r,c]* L_SIZE[r] for r in range(len(L_SIZE))),GRB.LESS_EQUAL,
CAPACITY* cont_hy[c], name='counter2')
                counter2 +=1
                #print 'cont_2_%s_%s'% (r,c)

            mhy.update()


            #------------------------- ObjZ--------------------------------
            mhy.setObjective(quicksum(cont_hy[c] for c in range(Cont_hy)) , GRB.MINIMIZE)
            mhy.optimize()

            #print "here is the value of the objective function: ",mhy.objval
            obj = mhy.objval
            Final_Cont_HY = []
            p=(i,j,obj)
            Final_Cont_HY.append(p)
```

```
        #------------------Model Clean up Section------------------
        mhy.update()
        num_optimization +=1

        import csv
        b = open('C:/Users/Mehran/Dropbox/Masters/Physical Internet/Simulation
Models/Data/Static/M1_HY.csv', 'ab')
        a = csv.writer(b)
        a.writerows(Final_Cont_HY)
        b.close()


#-------------------------Conventional Section ---------------------------

L_SIZE = []

def loads_co (i,j):
   s_d_loads = []
   for p in range(len(REQ_SET_CO)):
      if REQ_SET_CO[p][0] == i and REQ_SET_CO[p][1] == j:
         s_d_loads.append(REQ_SET_CO[p][3])
         #print s_d_loads
   return s_d_loads



#--Calling the optimizer for Hybrid senario inside the nested loops------
num_optimization = 1
for i in Node:
   for j in Node:
      if [i,j] in arc_co:

        print ""
        print "Conventional Senario"
        print "Optimization number", num_optimization, " of ", len(arc_co)
        print ""
        print "####################################################"
        print "Min number of containers problem between", i ," to ", j
        print "####################################################"
        L_SIZEP = L_SIZE
        L_SIZE = []
        L_SIZE = loads_co (i,j)
        #print  "Here is the list of loads :", L_SIZE


        #------------------------- Variables---------------------------
        #for i in range(Cont):
        #    for j in range(len(L_SIZEP)):
        #        mco.remove(mco.getConstrs()[0])


        mco.update()
        # Asignment of request r to container c from node i to node j
        assign = {}

        var1 =0
        for r in range(len(L_SIZE)):
              for c in range(Cont_co):
                  assign[r,c] = mco.addVar(ub = 1, lb = 0 , vtype=GRB.BINARY,  name='var1')
                  var1 +=1
        mco.update()

        cont_co = {}
        var2 = 0
        for c in range(Cont_co):
              cont_co[c] = mco.addVar(ub = 1, lb = 0 , vtype=GRB.BINARY,  name='var2')
              var2 +=1

        mco.update()

        #------------------------ Constraints-------------------------

        counter1 = 0
        for r in range(len(L_SIZE)):
              for c in range(Cont_co):
                    mco.addConstr(quicksum(assign[r,c] for c in range(Cont_co)), GRB.EQUAL , 1, name = 'counter1')
                    counter1 +=1
                    #print 'cont_1_%s_%s' % (r,c)

        mco.update()
        # setting the arc sizes

        for r in range(len(L_SIZE)):
           for c in range(Cont_co):
              mco.addConstr(quicksum(assign[r,c]* L_SIZE[r] for r in range(len(L_SIZE))),GRB.LESS_EQUAL,
CAPACITY* cont_co[c], name = 'counter2')
           #print 'cont_2_%s_%s'% (r,c)

        mco.update()


        #------------------------ ObjZ--------------------------------
        mco.setObjective(quicksum(cont_co[c] for c in range(Cont_co)) , GRB.MINIMIZE)
        mco.optimize()

        #print "here is the value of the objective function: ",mco.objval
        obj = mco.objval
        Final_Cont_CO = []
        p=(i,j,obj)
        Final_Cont_CO.append(p)

        #-----------------Model Clean up Section-----------------

        num_optimization +=1
        mco.update()

        import csv
        b = open('C:/Users/Mehran/Dropbox/Masters/Physical Internet/Simulation
Models/Data/Static/M1_CO.csv', 'ab')
        a = csv.writer(b)
        a.writerows(Final_Cont_CO)
        b.close()
```

Appendix H: Calculation of Flow Lower Bond

```
import csv
from Distance import *
from Param import *

key = Distance.keys()
# Puts all the dictionary keys in the key list. There is no need to iterate over the dictionary and collect all the keys

value = Distance.values()
# Puts all the dictionary values int the value list.There is no need iterate over the dictionary and collect all the values


Model_1_PI_LOADS = []
Model_1_HY_LOADS = []
Model_1_CO_LOADS = []

Optimal_Jobs_PI = []
Optimal_Jobs_HY = []
Optimal_Jobs_CO = []

source = []
dest = []

# Read Section
#---------------------------------------------------------------------------
with open('M1_PI.csv', 'rb') as f:

    reader = csv.reader(f)

    for row in reader:
        Model_1_PI_LOADS_TEMP = []

        Model_1_PI_LOADS_TEMP.append(row[0])
        Model_1_PI_LOADS_TEMP.append(row[1])
        Model_1_PI_LOADS_TEMP.append(float(row[2]))
        Model_1_PI_LOADS.append(Model_1_PI_LOADS_TEMP)

Loads_PI = []

#---------------------------------------------------------------------------
with open('M1_HY.csv', 'rb') as f:

    reader = csv.reader(f)

    for row in reader:
        Model_1_HY_LOADS_TEMP = []

        Model_1_HY_LOADS_TEMP.append(row[0])
        Model_1_HY_LOADS_TEMP.append(row[1])
        Model_1_HY_LOADS_TEMP.append(float(row[2]))
        Model_1_HY_LOADS.append(Model_1_HY_LOADS_TEMP)

Loads_HY = []

#---------------------------------------------------------------------------
with open('M1_CO.csv', 'rb') as f:

    reader = csv.reader(f)
    for row in reader:
        Model_1_CO_LOADS_TEMP = []

        Model_1_CO_LOADS_TEMP.append(row[0])
        Model_1_CO_LOADS_TEMP.append(row[1])
        Model_1_CO_LOADS_TEMP.append(float(row[2]))
        Model_1_CO_LOADS.append(Model_1_CO_LOADS_TEMP)

Loads_CO = []
#-------------------------------------------------------------------------

# PI Calculation Section
#-------------------------------------------------------------------------
for pass1 in range(len(Model_1_PI_LOADS)):
    for pass2 in range(len(Model_1_PI_LOADS)):
        if Model_1_PI_LOADS[pass1][0] == Model_1_PI_LOADS[pass2][1] and Model_1_PI_LOADS[pass1][1] ==
Model_1_PI_LOADS[pass2][0]:

                loads = 0
                loads = Model_1_PI_LOADS[pass1][2] - Model_1_PI_LOADS[pass2][2]
                min = 0
                min = Model_1_PI_LOADS[pass1][2]
                if  Model_1_PI_LOADS[pass1][2] > Model_1_PI_LOADS[pass2][2]:
                    min = Model_1_PI_LOADS[pass2][2]
                if loads > 0:
                    p = (Model_1_PI_LOADS[pass1][0], Model_1_PI_LOADS[pass1][1], loads)
                    s = (Model_1_PI_LOADS[pass1][0], Model_1_PI_LOADS[pass1][1], min)
                    Optimal_Jobs_PI.append(s)
                    Loads_PI.append(p)

                if loads == 0:
                    source.append(Model_1_PI_LOADS[pass1][0])
                    dest.append(Model_1_PI_LOADS[pass1][1])
                    if Model_1_PI_LOADS[pass1][1] in source and Model_1_PI_LOADS[pass1][0] in dest:
                        q = (Model_1_PI_LOADS[pass1][0], Model_1_PI_LOADS[pass1][1], loads)
                        s = (Model_1_PI_LOADS[pass1][0], Model_1_PI_LOADS[pass1][1], min)
                        Optimal_Jobs_PI.append(s)
                        Loads_PI.append(q)

total_hours_pi = 0

for i in range(len(Optimal_Jobs_PI)):
    s = Optimal_Jobs_PI[i][0]
    d = Optimal_Jobs_PI[i][1]
    for k in range(len(Distance)):
        if s == key[k][0] and d == key[k][1]:

            total_distance = 2 * value[k]* Optimal_Jobs_PI[i][2]
            job_hours = total_distance/avg_speed

            # Claculation of the total hours of driving for the optimal portion of the PI senario
            total_hours_pi = total_hours_pi + job_hours

            # Test to ensure all the values are read correctly
            #print i , "distance from ", s, " to ", d, " is ", value[k],". Number of jobs ", Loads_PI[i][3], " total job hours ",
job_hours


source = []
dest = []
```

```
# HY Calculation Section
#-------------------------------------------------------------------------
for pass1 in range(len(Model_1_HY_LOADS)):
   for pass2 in range(len(Model_1_HY_LOADS)):
       if Model_1_HY_LOADS[pass1][0] == Model_1_HY_LOADS[pass2][1] and Model_1_HY_LOADS[pass1][1] ==
Model_1_HY_LOADS[pass2][0]:

            loads = 0
            loads = Model_1_HY_LOADS[pass1][2] - Model_1_HY_LOADS[pass2][2]
            min = 0
            min = Model_1_HY_LOADS[pass1][2]
            if  Model_1_HY_LOADS[pass1][2] > Model_1_HY_LOADS[pass2][2]:
               min = Model_1_HY_LOADS[pass2][2]
            if loads > 0:
               p = (Model_1_HY_LOADS[pass1][0], Model_1_HY_LOADS[pass1][1], loads)
               s = (Model_1_HY_LOADS[pass1][0], Model_1_HY_LOADS[pass1][1], min)
               Optimal_Jobs_HY.append(s)
               Loads_HY.append(p)
            if loads == 0:
               source.append(Model_1_HY_LOADS[pass1][0])
               dest.append(Model_1_HY_LOADS[pass1][1])
               if Model_1_HY_LOADS[pass1][1] in source and Model_1_HY_LOADS[pass1][0] in dest:
                  q = (Model_1_HY_LOADS[pass1][0], Model_1_HY_LOADS[pass1][1], loads)
                  s = (Model_1_HY_LOADS[pass1][0], Model_1_HY_LOADS[pass1][1], min)
                  Optimal_Jobs_HY.append(s)
                  Loads_HY.append(q)
total_hours_hy = 0

for i in range(len(Optimal_Jobs_HY)):
   s = Optimal_Jobs_HY[i][0]
   d = Optimal_Jobs_HY[i][1]
   for k in range(len(Distance)):
       if s == key[k][0] and d == key[k][1]:

          total_distance = 2 * value[k]* Optimal_Jobs_HY[i][2]
          job_hours = total_distance/avg_speed

          # Claculation of the total hours of driving for the optimal portion of the PI senario
          total_hours_hy = total_hours_hy + job_hours

          # Test to ensure all the values are read correctly
          #print i , "distance from ", s, " to ", d, " is ", value[k],". Number of jobs ", Loads_HY[i][3], " total job hours ",
job_hours

source = []
dest = []

# CO Calculation Section
#-------------------------------------------------------------------------

for pass1 in range(len(Model_1_CO_LOADS)):
   for pass2 in range(len(Model_1_CO_LOADS)):
       if Model_1_CO_LOADS[pass1][0] == Model_1_CO_LOADS[pass2][1] and Model_1_CO_LOADS[pass1][1] ==
Model_1_CO_LOADS[pass2][0]:

            loads = 0
            loads = Model_1_CO_LOADS[pass1][2] - Model_1_CO_LOADS[pass2][2]
            min = 0
            min = Model_1_CO_LOADS[pass1][2]
            if  Model_1_CO_LOADS[pass1][2] > Model_1_CO_LOADS[pass2][2]:
               min = Model_1_CO_LOADS[pass2][2]
            if loads > 0:
               p = (Model_1_CO_LOADS[pass1][0], Model_1_CO_LOADS[pass1][1], loads)
               s = (Model_1_CO_LOADS[pass1][0], Model_1_CO_LOADS[pass1][1], min)
               Optimal_Jobs_CO.append(s)
               Loads_CO.append(p)

            if loads == 0:
               source.append(Model_1_CO_LOADS[pass1][0])
               dest.append(Model_1_CO_LOADS[pass1][1])
               if Model_1_CO_LOADS[pass1][1] in source and Model_1_CO_LOADS[pass1][0] in dest:
                  q = (Model_1_CO_LOADS[pass1][0], Model_1_CO_LOADS[pass1][1], loads)
                  s = (Model_1_CO_LOADS[pass1][0], Model_1_CO_LOADS[pass1][1], min)
                  Optimal_Jobs_CO.append(s)
                  Loads_CO.append(q)

total_hours_co = 0

for i in range(len(Optimal_Jobs_CO)):
   s = Optimal_Jobs_CO[i][0]
   d = Optimal_Jobs_CO[i][1]
   for k in range(len(Distance)):
      if s == key[k][0] and d == key[k][1]:

         total_distance = 2 * value[k]* Optimal_Jobs_CO[i][2]
         job_hours = total_distance/avg_speed

         # Claculation of the total hours of driving for the optimal portion of the PI senario
         total_hours_co = total_hours_hy + job_hours

         # Test to ensure all the values are read correctly
         #print i , "distance from ", s, " to ", d, " is ", value[k],". Number of jobs ", Loads_CO[i][3], " total job hours ",
job_hours


source = []
dest = []

#--------------------------------------------------------------------------

# Write Section
#--------------------------------------------------------------------------

b = open('M1_PI_REDUCED.csv', 'wb')
a = csv.writer(b)
header = ['s','d','c']
a.writerow(header)
a.writerows(Loads_PI)
b.close()

b = open('M1_PI_JOBS.csv', 'wb')
a = csv.writer(b)
header = ['s','d','jobs']
a.writerow(header)
a.writerows(Optimal_Jobs_PI)
b.close()

b = open('M1_HY_REDUCED.csv', 'wb')
a = csv.writer(b)
header = ['s','d','c']
a.writerow(header)
a.writerows(Loads_HY)
```

```
b.close()

b = open('M1_HY_JOBS.csv', 'wb')
a = csv.writer(b)
header = ['s','d','jobs']
a.writerow(header)
a.writerows(Optimal_Jobs_HY)
b.close()

b = open('M1_CO_REDUCED.csv', 'wb')
a = csv.writer(b)
header = ['s','d','c']
a.writerow(header)
a.writerows(Loads_CO)
b.close()

b = open('M1_CO_JOBS.csv', 'wb')
a = csv.writer(b)
header = ['s','d','jobs']
a.writerow(header)
a.writerows(Loads_CO)
b.close()

b = open('Optimal_Hours.csv', 'wb')
optimals = [["Hours PI", total_hours_pi],["Hours HY", total_hours_hy],["Hours CO", total_hours_co]]
a = optimals
a = csv.writer(b)
a.writerows(optimals)
b.close()
```

Appendix I: GLPK Code of Routing MIP

```
# Written By Mehran Fazili
# Last modified 8 November 2013

/* Set of nodes*/
set Node := {'Hal','Tru','Syd','Yar','Cht','Mon','Frd','Stj','Gt1','Riv','Qbc','Gt2','Mte','Gt3','Caw','Mtw','Otn','Ots'};

param m, integer, >1;
# Number of routes

set Route := {1..m};
#Set of routes

set Cont_Set, dimen 2;

################################
########## Parameters ##############
################################

param FIX_RO_CO{i in Route}, default 20;
#To be investigated - used to calculate the fixed cost of a route

param CONT{i in Node, j in Node}, default 0;
# Number of loads to be transferred from node i to node j

table ara IN "CSV" "M1_CO_REDUCED.csv" :
Cont_Set <- [s, d] , CONT ~ c;


param FTTIME{i in Node, j in Node};
# Loaded travel time from node i to node j

param ETTIME{i in Node, j in Node};
# Deadhead travel time from node i to node  j

param TTTIME, default 50;
# Maximum duration (length) of each route (Set to a large value to relax the corresponding constraint)

param MITIME, default 2;
# Minimum duration (length) of each route (Set to zero to relax the corresponding constraint)

param AVG_R_LEN, default 160;
# Maximum allowed length for average route length (Set to a large value to relax the corresponding constraint)

param NUM_ACT_ROUTE, default 100 ;
# Maximum number of routes allowed to serve the entire demand (Set to a large value to relax the corresponding
constraint)

param MAX_PER_DAED , default 0.4;
# Maximum percentage of deadhead travel length to the total route length (Set to a fractional value close to one to
relax the corresponding constraint)

param M , default 12;
# Auxiliary parameter used in sub tour elmination

################################
########## Variables ###############
```

```
################################

var f_a{i in Node, j in Node, k in Route}, binary;
# Binary, 1 if route k includes the loaded arc from node i to node  j, 0 otherwise

var e_a{i in Node, j in Node, k in Route}, binary;
# Binary, 1 if route k includes the deadhead arc from node i to node  j, 0 otherwise

var route{k in Route}, >=0;
# Total duration (or length) of route k

var s{i in Node, k in Route} ;
# Auxiliary positive integer variable used for sub tour elimination

var act_route{k in Route}, binary ;
# Binary, 1 if rk > 0, 0 otherwise

################################
######### Objective Function ##########
################################
minimize obj:sum{i in Node, j in Node, k in Route}(f_a[i,j,k]*FTTIME[i,j] + e_a[i,j,k]*FTTIME[i,j]) + sum{k in
Route}10*act_route[k];
#Minimizes the total route duration (length)

#minimize obj:sum{k in Route}(route[k]);
#Minimizes the total route duration (length)

################################
######### Technical Const #########
################################

Con1{i in Node, j in Node}: sum{k in Route}f_a[i,j,k] = CONT[i,j];
# Ensures each full travel between source i and destination j is equal to the number of loads from i to j (Determined in
model 1)

Con2{k in Route}:sum{i in Node, j in Node}(f_a[i,j,k]*FTTIME[i,j] + e_a[i,j,k]*ETTIME[i,j]) = route[k];
# Total duration of full and empty assignment of travels to route k is equal to the route duration (length)

Con3{p in Node, k in Route}:sum{i in Node}(f_a[i,p,k] + e_a[i,p,k]) = sum{j in Node}(f_a[p,j,k] + e_a[p,j,k]);
# (Conservation of flow). at each node and for each route, total full and empty entry to a node should be equal to the
total full and empty exit.

con34{k in Route, i in Node}:sum{j in Node}(f_a[i,j,k] + e_a[i,j,k]) <= act_route[k];
#Con37{p in Node, k in Route}:sum{i in Node}(f_a[i,p,k] + e_a[i,p,k]) + sum{j in Node}(f_a[p,j,k] + e_a[p,j,k]) <= 2;
# Every Node is visited only once by each route

#Con31{p in Node, k in Route}:sum{i in Node}(f_a[i,p,k] + e_a[i,p,k]) <= 1;

#Con32{p in Node, k in Route}: sum{j in Node}(f_a[p,j,k] + e_a[p,j,k]) <= 1 ;


Con4{i in Node, j in Node, k in Route}:s[i,k] - s[j,k] + 1 <= M*(1- f_a[i,j,k]);
# Together with cont 5, 6, and 7 is used for sub-tour elimination.

#Con41{i in Node, j in Node, k in Route}:-s[i,k] + s[j,k] - 1 >= -(M*(1- e_a[i,j,k]));
# Together with cont 5, 6, and 7 is used for sub-tour elimination.

Con5{i in Node, k in Route}: s[i,k] <= M;
# Used for sub-tour elimination

Con6{i in Node, k in Route}: s[i,k] >= 1;
```

```
# Used for sub-tour elimination

#Con7{k in Route}: sum{i in Node, j in Node}f_a[i,j,k] <= 4;

con12{k in Route}:sum{i in Node, j in Node}e_a[i,j,k] = 1;
#con121{k in Route}:sum{i in Node, j in Node : i = j}e_a[i,j,k] = 0;

#con121{k in Route}:sum{i in Node, j in Node : i = j}e_a[i,j,k] = 0;
#con121{k in Route}:sum{i in Node, j in Node : i = j}e_a[i,j,k] = 0;

# Sets maximum duration of the deadhead as fraction of the total route duration (length).
#con38{i in Node, j in Node, k in Route}:f_a[i,j,k]+ e_a[i,j,k] + f_a[j,i,k] + e_a[j,i,k] <= 1;

#con39{k in Route}:sum{i in Node, j in Node}(f_a[i,j,k]) <=4;


###############################
########## Policy Constraints ##########
###############################

#con8{k in Route}: route[k] >= act_route[k]*MITIME;
# Sets min route duration (length)

#con81{k in Route}: route[k] <= act_route[k]*100;
# Sets min route duration (length)

cont9{k in Route}: route[k] <= act_route[k]*TTTIME;
# Sets max route duration (length)

#con10: sum{k in Route}route[k] / AVG_R_LEN <= sum{k in Route}act_route[k];
# Sets max average route duration (length)

#con11:sum{k in Route}act_route[k] <= NUM_ACT_ROUTE;
# Sets maximum number of allowed routes

#con12{k in Route}:(sum{i in Node, j in Node}(e_a[i,j,k])*ETTIME[i,j]) / MAX_PER_DAED <= route[k];
# Sets maximum duration of the deadhead as fraction of the total route duration (length).


solve;

printf "\n\nLoaded Arcs and Routes\n\n";
for {i in Node, j in Node, k in Route}{
 for {{0}: f_a[i,j,k] == 1 } {

printf  " %s %d %s %d %s %d \n"," Route", k ," goes from node ",i," to node ",j ;
}}

printf "\n\nDeahead Arcs and veh\n\n";
for {i in Node, j in Node, k in Route}{
for {{0}: e_a[i,j,k] == 1 } {

 printf  " %s %d %s %d %s %d \n","Route", k ," goes from node ",i," to node ",j ;
}}

printf "\n\nDuration of Each Route\n\n";
for {k in Route}{
 for {{0}: route[k] >= 1 } {
 printf " Route  %d %s %f \ Hours \n", k, " is ", route[k];
}}
```

```
printf "\n\n";
data;

param m := 200;

param FTTIME :Hal  Tru      Syd      Yar      Mon      Frd      Stj      Cht      Gt1      Riv      Qbc
   Gt2      Mte      Gt3      Caw      Mtw      Otn      Ots :=
Hal         0.5        1.39   4.49     3.54     3.12     4.86     4.63     3.76     5.77     8.66     10.67
   12.73    12.91    13.3     13.91    13.31    14.85    15.82
Tru         1.39     0.5        3.6      4.43   2.23     3.97     3.74     2.87     4.88     7.77     9.78
   11.84    12.02    12.42    13.02    12.42    13.96    14.93
Syd         4.49     3.6        0.5      7.48   5.26     6.86     6.71     5.83     7.88     10.72    12.7
   14.83    15.06    15.44    16.06    15.46    16.93    17.89
Yar         3.54     4.43     7.48     0.5        5.99   7.72     7.49     6.58     8.81     7.2        13.53
   15.7     15.82    16.06    16.82    16.22    13.38    14.35
Mon         3.12     2.23     5.26     5.99     0.5        2.3        2.08   2.13     3.15     6.11     8.09
   10.2     10.37    10.81    11.37    10.77    12.29    13.28
Frd         4.86     3.97     6.86     7.72     2.3        0.5        1.59   3.81     1.41     4.39     6.33
   8.46     8.76     9.06     9.76     9.16     10.6     11.57
Stj         4.63     3.74     6.71     7.49     2.08     1.59     0.5        3.92   2.53     5.31     7.23
   9.46     9.56     9.96     10.56    9.96     11.54    12.5
Cht         3.76     2.87     5.83     6.58     2.13     3.81     3.92     0.5        4.85   7.63     9.59
   11.74    11.9     12.43    12.9     12.3     13.86    14.79
Gt1         5.77     4.88     7.88     8.81     3.15     1.41     2.53     4.85     0.5        3.44   5.25
   7.46     7.49     8.12     8.49     7.89     9.64     10.62
Riv         8.66     7.77     10.72    7.2      6.11     4.39     5.31     7.63     3.44     0.5        2.45
   6.25     4.74     5.22     5.74     5.14     6.79     7.62
Qbc         10.67    9.78     12.7     13.53    8.09     6.33     7.23     9.59     5.25     2.45     0.5
2.72        2.94     3.45     3.94     3.34     5.02     6.08
Gt2         12.73    11.84    14.83    15.7     10.2     8.46     9.46     11.74    7.46     6.25     2.72
   0.5        2.02   2.39     3.02     2.42     4.09     4.94
Mte         12.91    12.02    15.06    15.82    10.37    8.76     9.56     11.9     7.49     4.74     2.94
   2.02     0.5        1.22   1.5        0.9      2.5      3.4
Gt3         13.3     12.42    15.44    16.06    10.81    9.06     9.96     12.43    8.12     5.22     3.45
   2.39     1.22     0.5        2.45   1.45     3.24     3.85
Caw         13.91    13.02    16.06    16.82    11.37    9.76     10.56    12.9     8.49     5.74     3.94
   3.02     1.5        2.445  0.5      1.5        3.5      4.4
Mtw         13.31    12.42    15.46    16.22    10.77    9.16     9.96     12.3     7.89     5.14     3.34
   2.42     0.9        1.445  1.5      0.5        2.1      3
Otn         14.85    13.96    16.93    13.38    12.29    10.6     11.54    13.86    9.64     6.79     5.02
   4.09     2.5        3.24   3.5        2.1      0.5      2.42
Ots         15.82    14.93    17.89    14.35    13.28    11.57    12.5     14.79    10.62    7.62     6.08
   4.94     3.4        3.85   4.4        3        2.42   0.5
;


param ETTIME :Hal  Tru      Syd      Yar      Mon      Frd      Stj      Cht      Gt1      Riv      Qbc
   Gt2      Mte      Gt3      Caw      Mtw      Otn      Ots :=
Hal         0.00     0.89     3.99     3.04     2.62     4.36     4.13     3.26     5.27     8.16     10.17
   12.23    12.41    12.80    13.41    12.81    14.35    15.32
Tru         0.89     0.00     3.10     3.93     1.73     3.47     3.24     2.37     4.38     7.27     9.28
   11.34    11.52    11.92    12.52    11.92    13.46    14.43
Syd         3.99     3.10     0.00     6.98     4.76     6.36     6.21     5.33     7.38     10.22    12.20
   14.33    14.56    14.94    15.56    14.96    16.43    17.39
Yar         3.04     3.93     6.98     0.00     5.49     7.22     6.99     6.08     8.31     6.70     13.03
   15.20    15.32    15.56    16.32    15.72    12.88    13.85
Mon         2.62     1.73     4.76     5.49     0.00     1.80     1.58     1.63     2.65     5.61     7.59
   9.70     9.87     10.31    10.87    10.27    11.79    12.78
Frd         4.36     3.47     6.36     7.22     1.80     0.00     1.09     3.31     0.91     3.89     5.83
   7.96     8.26     8.56     9.26     8.66     10.10    11.07
```

```
Stj      4.13    3.24    6.21    6.99    1.58    1.09    0.00    3.42    2.03    4.81    6.73
  8.96   9.06    9.46    10.06   9.46    11.04   12.00
Cht      3.26    2.37    5.33    6.08    1.63    3.31    3.42    0.00    4.35    7.13    9.09
  11.24  11.40   11.93   12.40   11.80   13.36   14.29
Gt1      5.27    4.38    7.38    8.31    2.65    0.91    2.03    4.35    0.00    2.94    4.75
  6.96   6.99    7.62    7.99    7.39    9.14    10.12
Riv      8.16    7.27    10.22   6.70    5.61    3.89    4.81    7.13    2.94    0.00    1.95
  5.75   4.24    4.72    5.24    4.64    6.29    7.12
Qbc      10.17   9.28    12.20   13.03   7.59    5.83    6.73    9.09    4.75    1.95    0.00
  2.22   2.44    2.95    3.44    2.84    4.52    5.58
Gt2      12.23   11.34   14.33   15.20   9.70    7.96    8.96    11.24   6.96    5.75    2.22
  0.00   1.52    1.89    2.52    1.92    3.59    4.44
Mte      12.41   11.52   14.56   15.32   9.87    8.26    9.06    11.40   6.99    4.24    2.44
  1.52   0.00    0.72    1.00    0.40    2.00    2.90
Gt3      12.80   11.92   14.94   15.56   10.31   8.56    9.46    11.93   7.62    4.72    2.95
  1.89   0.72    0.00    1.95    0.95    2.74    3.35
Caw      13.41   12.52   15.56   16.32   10.87   9.26    10.06   12.40   7.99    5.24    3.44
  2.52   1.00    1.95    0.00    1.00    3.00    3.90
Mtw      12.81   11.92   14.96   15.72   10.27   8.66    9.46    11.80   7.39    4.64    2.84
  1.92   0.40    0.95    1.00    0.00    1.60    2.50
Otn      14.35   13.46   16.43   12.88   11.79   10.10   11.04   13.36   9.14    6.29    4.52
  3.59   2.00    2.74    3.00    1.60    0.00    1.92
Ots      15.32   14.43   17.39   13.85   12.78   11.07   12.00   14.29   10.12   7.12    5.58
  4.44   2.90    3.35    3.90    2.50    1.92    0.00
;


end;
```

Appendix J: Calculation of the Jobs Enroute Time in Python

```
import csv

from Distance import *
from Param import *

key = Distance.keys()
# Puts all the dictionary keys in the key list. There is no need to iterate over the dictionary and collect all the keys

value = Distance.values()
# Puts all the dictionary values int the value list.There is no need iterate over the dictionary and collect all the values

Model_1_PI_JOBS = []
Model_1_PI_DUR = []
Model_1_PI_LOC = []

Model_1_HY_JOBS = []
Model_1_HY_DUR = []
Model_1_HY_LOC = []

Model_1_CO_JOBS = []
Model_1_CO_DUR = []
Model_1_CO_LOC = []


Model_1_PI_DDUR = []
Model_1_HY_DDUR = []
Model_1_CO_DDUR = []

driving_limit = 14
sleep_time = 8

# Read Section
#-------------------------------------------------------------------------

with open('M1_PI_JOBS.csv', 'rb') as f:

   reader = csv.reader(f)

   for row in reader:
      Model_1_PI_JOBS_TEMP = []
      Model_1_PI_JOBS_TEMP.append(row[0])
      Model_1_PI_JOBS_TEMP.append(row[1])
      Model_1_PI_JOBS_TEMP.append(float(row[2]))
      Model_1_PI_JOBS.append(Model_1_PI_JOBS_TEMP)

with open('M1_HY_JOBS.csv', 'rb') as f:

   reader = csv.reader(f)

   for row in reader:
      Model_1_HY_JOBS_TEMP = []
      Model_1_HY_JOBS_TEMP.append(row[0])
      Model_1_HY_JOBS_TEMP.append(row[1])
      Model_1_HY_JOBS_TEMP.append(float(row[2]))
      Model_1_HY_JOBS.append(Model_1_HY_JOBS_TEMP)

with open('M1_CO_JOBS.csv', 'rb') as f:

   reader = csv.reader(f)
```

```
    for row in reader:
        Model_1_CO_JOBS_TEMP = []
        Model_1_CO_JOBS_TEMP.append(row[0])
        Model_1_CO_JOBS_TEMP.append(row[1])
        Model_1_CO_JOBS_TEMP.append(float(row[2]))
        Model_1_CO_JOBS.append(Model_1_CO_JOBS_TEMP)



#-----------------------------------------------------------------------------
# PI Section
#--------------------------------------------------------------------------
# Jobs from M1 file
job_row = 0
for i in range(len(Model_1_PI_JOBS)):
    s = Model_1_PI_JOBS[i][0]
    d = Model_1_PI_JOBS[i][1]
    for k in range(len(Distance)):
        if s == key[k][0] and d == key[k][1]:

            job_hours = 0
            drive_hours = 0
            total_distance = 0
            total_distance = 2 * value[k]
            #print total_distance
            job_hours = (float(total_distance)/avg_speed)
            drive_hours = job_hours
            number_of_nights = int (job_hours / driving_limit)

            job_hours = job_hours + number_of_nights * sleep_time
            print s,d, drive_hours, number_of_nights,  job_hours

            # print s,d,job_hours
            counter =  int(Model_1_PI_JOBS[i][2])


            for j in range(counter):
                job_row +=1

                LOC = (job_row, s)
                Model_1_PI_LOC.append(LOC)

                DUR = (job_row, job_hours)
                Model_1_PI_DUR.append(DUR)

                DUR_DRIVE = (job_row, drive_hours)
                Model_1_PI_DDUR.append(DUR_DRIVE)

b = open('jobs_info/JOB_LOC_PI.csv', 'wb')
a = csv.writer(b)
header = ['number','node']
a.writerow(header)
a.writerows(Model_1_PI_LOC)
b.close()

b = open('jobs_info/JOB_DUR_PI.csv', 'wb')
a = csv.writer(b)
header = ['number','dur']
a.writerow(header)
a.writerows(Model_1_PI_DUR)
```

```
b.close()

b = open('jobs_info/JOB_DRIVE_DUR_PI.csv', 'wb')
a = csv.writer(b)
header = ['number','dur']
a.writerow(header)
a.writerows(Model_1_PI_DDUR)
b.close()

#---------------------------------------------------------------------------------
# Reading the jobs created in RO optimizer

M2_RAW = []

with open('M2_prep/M2_inventory_pi.csv', 'rb') as f:

    reader = csv.reader(f)

    for row in reader:
        Model_2_TEMP = []
        Model_2_TEMP.append(row[0])
        Model_2_TEMP.append(row[1])
        Model_2_TEMP.append(float(row[2]))
        Model_2_TEMP.append(float(row[3]))
        Model_2_TEMP.append(float(row[4]))
        M2_RAW.append(Model_2_TEMP)

Location = []
Duration = []
DDuration = []

for i in range(len(M2_RAW)):
    for j in range(len(M2_RAW)):
        pass1 = M2_RAW[i][2]
        pass2 = M2_RAW[j][3]
        if pass1 == pass2:
            #print M2_RAW[i][0],M2_RAW[i][1],M2_RAW[i][2],M2_RAW[j][3],M2_RAW[j][4]

            #Entering a social cost into jobs with duration longer than 14 hours
            drive_hours = 0
            job_dur = 0
            job_dur = M2_RAW[j][4]
            drive_hours = job_dur
            #print job_dur
            number_of_nights = int (job_dur / driving_limit)
            #print number_of_nights
            job_dur = job_dur + number_of_nights * sleep_time

            job_row+=1

            loc = (job_row , M2_RAW[i][0])
            Location.append(loc)

            dur = (job_row , job_dur)
            Duration.append(dur)

            ddur = (job_row, drive_hours)
            DDuration.append(ddur)

b = open('jobs_info/JOB_LOC_PI.csv', 'ab')
a = csv.writer(b)
```

```
a.writerows(Location)
b.close()

b = open('jobs_info/JOB_DUR_PI.csv', 'ab')
a = csv.writer(b)
a.writerows(Duration)
b.close()

b = open('jobs_info/JOB_DRIVE_DUR_PI.csv', 'ab')
a = csv.writer(b)
a.writerows(DDuration)
b.close()

#-----------------------------------------------------------------------------
# Hybrid Section

#-----------------------------------------------------------------------------
# Jobs from M1 file
job_row = 0
for i in range(len(Model_1_HY_JOBS)):
    s = Model_1_HY_JOBS[i][0]
    d = Model_1_HY_JOBS[i][1]
    for k in range(len(Distance)):
        if s == key[k][0] and d == key[k][1]:

            job_hours = 0
            total_distance = 0
            total_distance = 2 * value[k]
            #print total_distance
            job_hours = (float(total_distance)/avg_speed)
            drive_hours = job_hours
            number_of_nights = int (job_hours / driving_limit)
            #print s,d,number_of_nights, job_hours
            job_hours = job_hours + number_of_nights * sleep_time


            # print s,d,job_hours
            counter =  int(Model_1_HY_JOBS[i][2])


            for j in range(counter):
                job_row +=1

                LOC = (job_row, s)
                Model_1_HY_LOC.append(LOC)

                DUR = (job_row, job_hours)
                Model_1_HY_DUR.append(DUR)

                DUR_DRIVE = (job_row, drive_hours)
                Model_1_HY_DDUR.append(DUR_DRIVE)

b = open('jobs_info/JOB_LOC_HY.csv', 'wb')
a = csv.writer(b)
header = ['number','node']
a.writerow(header)
a.writerows(Model_1_HY_LOC)
b.close()

b = open('jobs_info/JOB_DUR_HY.csv', 'wb')
a = csv.writer(b)
```

```
header = ['number','dur']
a.writerow(header)
a.writerows(Model_1_HY_DUR)
b.close()

b = open('jobs_info/JOB_DRIVE_DUR_HY.csv', 'wb')
a = csv.writer(b)
header = ['number','dur']
a.writerow(header)
a.writerows(Model_1_HY_DDUR)
b.close()

#-----------------------------------------------------------------------------
# Reading the jobs created in RO optimizer

M2_RAW = []

with open('M2_prep/M2_inventory_hy.csv', 'rb') as f:

    reader = csv.reader(f)

    for row in reader:
        Model_2_TEMP = []
        Model_2_TEMP.append(row[0])
        Model_2_TEMP.append(row[1])
        Model_2_TEMP.append(float(row[2]))
        Model_2_TEMP.append(float(row[3]))
        Model_2_TEMP.append(float(row[4]))
        M2_RAW.append(Model_2_TEMP)

Location = []
Duration = []
DDuration = []

for i in range(len(M2_RAW)):
    for j in range(len(M2_RAW)):
        pass1 = M2_RAW[i][2]
        pass2 = M2_RAW[j][3]
        if pass1 == pass2:
            #print M2_RAW[i][0],M2_RAW[i][1],M2_RAW[i][2],M2_RAW[j][3],M2_RAW[j][4]

            #Entering a social cost into jobs with duration longer than 14 hours
            job_dur = 0
            job_dur = M2_RAW[j][4]
            drive_hours = job_dur
            #print job_dur
            number_of_nights = int (job_dur / driving_limit)
            #print number_of_nights
            job_dur = job_dur + number_of_nights * sleep_time

            job_row+=1
            loc = (job_row , M2_RAW[i][0])
            Location.append(loc)

            dur = (job_row , job_dur)
            Duration.append(dur)

            ddur = (job_row, drive_hours)
            DDuration.append(ddur)

b = open('jobs_info/JOB_LOC_HY.csv', 'ab')
```

```
a = csv.writer(b)
a.writerows(Location)
b.close()

b = open('jobs_info/JOB_DUR_HY.csv', 'ab')
a = csv.writer(b)
a.writerows(Duration)
b.close()

b = open('jobs_info/JOB_DRIVE_DUR_HY.csv', 'ab')
a = csv.writer(b)
a.writerows(DDuration)
b.close()

#-------------------------------------------------------------------------------
# Conventional Section

#-------------------------------------------------------------------------------
# Jobs from M1 file
job_row = 0
for i in range(len(Model_1_CO_JOBS)):
    s = Model_1_CO_JOBS[i][0]
    d = Model_1_CO_JOBS[i][1]
    for k in range(len(Distance)):
        if s == key[k][0] and d == key[k][1]:

            job_hours = 0
            total_distance = 0
            total_distance = 2 * value[k]
            #print total_distance
            job_hours = (float(total_distance)/avg_speed)
            drive_hours = job_hours
            number_of_nights = int (job_hours / driving_limit)
            #print s,d,number_of_nights, job_hours
            job_hours = job_hours + number_of_nights * sleep_time


            # print s,d,job_hours
            counter =  int(Model_1_CO_JOBS[i][2])


            for j in range(counter):
                job_row +=1

                LOC = (job_row, s)
                Model_1_CO_LOC.append(LOC)

                DUR = (job_row, job_hours)
                Model_1_CO_DUR.append(DUR)

                DUR_DRIVE = (job_row, drive_hours)
                Model_1_CO_DDUR.append(DUR_DRIVE)

b = open('jobs_info/JOB_LOC_CO.csv', 'wb')
a = csv.writer(b)
header = ['number','node']
a.writerow(header)
a.writerows(Model_1_CO_LOC)
b.close()

b = open('jobs_info/JOB_DUR_CO.csv', 'wb')
```

```
a = csv.writer(b)
header = ['number','dur']
a.writerow(header)
a.writerows(Model_1_CO_DUR)
b.close()

b = open('jobs_info/JOB_DRIVE_DUR_CO.csv', 'wb')
a = csv.writer(b)
header = ['number','dur']
a.writerow(header)
a.writerows(Model_1_CO_DDUR)
b.close()

#-----------------------------------------------------------------------------
# Reading the jobs created in RO optimizer

M2_RAW = []

with open('M2_prep/M2_inventory_co.csv', 'rb') as f:

    reader = csv.reader(f)

    for row in reader:
        Model_2_TEMP = []
        Model_2_TEMP.append(row[0])
        Model_2_TEMP.append(row[1])
        Model_2_TEMP.append(float(row[2]))
        Model_2_TEMP.append(float(row[3]))
        Model_2_TEMP.append(float(row[4]))
        M2_RAW.append(Model_2_TEMP)

Location = []
Duration = []
DDuration = []

for i in range(len(M2_RAW)):
    for j in range(len(M2_RAW)):
        pass1 = M2_RAW[i][2]
        pass2 = M2_RAW[j][3]
        if pass1 == pass2:
            #print M2_RAW[i][0],M2_RAW[i][1],M2_RAW[i][2],M2_RAW[j][3],M2_RAW[j][4]

            #Entering a social cost into jobs with duration longer than 14 hours
            job_dur = 0
            job_dur = M2_RAW[j][4]
            drive_hours = job_dur

            #print job_dur
            number_of_nights = int (job_dur / driving_limit)
            #print number_of_nights
            job_dur = job_dur + number_of_nights * sleep_time

            job_row+=1
            loc = (job_row , M2_RAW[i][0])
            Location.append(loc)

            dur = (job_row , job_dur)
            Duration.append(dur)

            ddur = (job_row, drive_hours)
            DDuration.append(ddur)
```

```
b = open('jobs_info/JOB_LOC_CO.csv', 'ab')
a = csv.writer(b)
a.writerows(Location)
b.close()

b = open('jobs_info/JOB_DUR_CO.csv', 'ab')
a = csv.writer(b)
a.writerows(Duration)
b.close()

b = open('jobs_info/JOB_DRIVE_DUR_CO.csv', 'ab')
a = csv.writer(b)
a.writerows(DDuration)
b.close()
#--------------------------------------------------------------------------------
```

Appendix K: GLPK Code of Assignment MIP

```
# Developed by Mehran Fazili
# Created on Jan 14, 2014
# Last Updated on Feb 27, 2014
#----------------------------------------------------------------------------------------------------

set Node := {'Hal','Tru','Syd','Yar','Cht','Mon','Frd','Stj','Gt1','Riv','Qbc','Gt2','Mte','Gt3','Caw','Mtw','Otn','Ots'};
/* Set of nodes*/

param K, integer;
set Truck := {1..K} ;

param Q, integer;
set Job :={1..Q};

#-----------------------------------------Parameters------------------------------------------------

param T_L {i in Truck}, symbolic ;
# Location of the truck
set Truck_Set, dimen 1;
table ara IN "CSV" "LP_INFO/TR_LOC.csv" :
Truck_Set <-[trk_num], T_L ~ home;

param J_L {j in Job}, symbolic;
# Location of a job
set Job_Set, dimen 1;
table ara IN "CSV" "JOBS_INFO/JOB_LOC_CO.csv" :
Job_Set <-[number], J_L ~ node;

param J_DUR {j in Job};
# Duration of a job
set JobD_Set, dimen 1;
table ara IN "CSV" "JOBS_INFO/JOB_DUR_CO.csv" :
JobD_Set <-[number], J_DUR ~ dur;

param J_DRIVE_DUR {j in Job};
# Duration of a job
set JobDD_Set, dimen 1;
table ara IN "CSV" "JOBS_INFO/JOB_DRIVE_DUR_CO.csv" :
JobDD_Set <-[number], J_DRIVE_DUR ~ dur;

param FIX_COST{i in Truck};
# Fix cost of entering a new truck into the service
set Truck_Fix_Cost_Set, dimen 1;
table ara IN "CSV" "LP_INFO/FIX_COST.csv" :
Truck_Fix_Cost_Set <-[trk_num], FIX_COST ~ f_cost;

param DELV_SPAN{i in Truck};
# Maximum work time assigned to truck i. 14 for short haul trucks, 72 hours for long haul trucks.
set Del_Span_Set, dimen 1;
table ara IN "CSV" "LP_INFO/DEL_SPAN.csv" :
Del_Span_Set <-[trk_num], DELV_SPAN ~ span;

param DIST{n1 in Node, n2 in Node};
# Distance from one node to other node

param D_DUR {i in Truck , j in Job }  := 2*DIST[T_L[i], J_L[j]] ;
# Distance in hours from node the location of truck i to location of job j

param DAYDRIVE, default 14;
# on duty  time allowed per day

param HOUR_COST{i in Truck}, default 200;
# Averge hourly operational cost of a truck  - from Goverment of Canada   $3.15/ km cost of operation

param SOCIAL_COST, default 60;
```

```
# Average cost of a bed and breakfast, parking by highway.  per night.
#-----------------------------------------Variables------------------------------------------------

var x{i in Truck, j in Job}, binary;
# Binary variable, 1 if truck i is assigned to perform job j

var fix_x{i in Truck}, binary;
# Binary variable, 1 if truck i enter service

var enroute_time{i in Truck}, >=0 ;
#  time truck i is away from home

var fix_cost{i in Truck}, >= 0;
# Fix cost of entering truck i into service

var opr_cost{i in Truck}, >= 0;
# Operation cost of truck i

var soc_cost{i in Truck}, >= 0 ;
# Social cost of truck i

#---------------------------------

var total_soc, >= 0 ;

var total_opr, >= 0 ;

var total_fix, >= 0;

var total_enroute, >=0;

var total_driving_hours, >=0;
#-----------------------------------------Obj Func------------------------------------------------

minimize obj: sum{i in Truck }(opr_cost[i] + soc_cost[i] + fix_cost[i] );

#----------------------------------------Constraints-------------------------------------------------

con1{i in Truck}:sum{j in Job}((D_DUR[i,j] + J_DUR[j])* x[i,j]) = enroute_time[i] ;

con2{j in Job}:sum{i in Truck}x[i,j] = 1;

con3{i in Truck}: enroute_time[i] <= DELV_SPAN[i]*fix_x[i] ;
# Delivery Span for Short Haul Trucks

con4{i in Truck}:sum{j in Job}((D_DUR[i,j] + J_DRIVE_DUR[j])* x[i,j])*HOUR_COST[i] = opr_cost[i] ;
#Operational cost of Short Haul Trucks

cont5{i in Truck}: FIX_COST[i] * fix_x[i] = fix_cost[i];
#Fix cost of Short Haul Trucks

cont6{i in Truck}: sum{j in Job: J_DRIVE_DUR[j] >=14 }((J_DRIVE_DUR[j]* x[i,j]) / DAYDRIVE) * SOCIAL_COST  = soc_cost[i] ;

#cona1{i in Truck: i = 1}: enroute_time[i] <= 93*fix_x[i] ;
# Delivery Span for Short Haul Trucks

#----------------------------------------------------------------------------------------------------

# Following are the help constriant for purpose of collecting statistics. They have nothing to do with the solution of the model.

contax1: sum{i in Truck}soc_cost[i] = total_soc;

contax2: sum{i in Truck}opr_cost[i] = total_opr;

contax3: sum{i in Truck}fix_cost[i] = total_fix;

contax4: sum{i in Truck}enroute_time[i] = total_enroute;

contac5: sum{i in Truck}opr_cost[i]/ HOUR_COST[i] = total_driving_hours;
```

```
#-----------------------------------------------------------------------------------------------------
solve;

for {i in Truck} printf "\n\n %s %d %s %f  \n", "Driving time of truck ", i ," is ", enroute_time[i];

printf "\n\nLoaded Arcs and Routes\n\n";
for {i in Truck , j in Job}{
for {{0}: x[i,j] == 1 } {
printf  " %s %d %s %d  \n\n"," truck ", i ," servce job ",j;
}}
#-----------------------------------------------------------------------------------------------------
data;

param K := 2790;
# Number of trucks in the system

param Q := 451;
# Number of jobs for CO

#param Q := 847;
# Number of jobs for HY

#param Q := 2755;
# Number of jobs for PI

param DIST :Hal              Tru      Syd      Yar      Mon      Frd      Stj      Cht      Gt1      Riv      Qbc
         Gt2      Mte      Gt3      Caw      Mtw      Otn      Ots :=
Hal                        0.00     1.61     5.49     4.30     3.78     5.95     5.66     4.58     7.09     10.70
         13.21    15.79    16.01    16.50    17.26    16.51    18.44    19.65
Tru                        1.61     0.00     4.38     5.41     2.66     4.84     4.55     3.46     5.98     9.59
         12.10    14.68    14.90    15.40    16.15    15.40    17.33    18.54
Syd                        5.49     4.38     0.00     9.23     6.45     8.45     8.26     7.16     9.73     13.28
         15.75    18.41    18.70    19.18    19.95    19.20    21.04    22.24
Yar                        4.30     5.41     9.23     0.00     7.36     9.53     9.24     8.10     10.89    8.88
         16.79    19.50    19.65    19.95    20.90    20.15    16.60    17.81
Mon                        3.78     2.66     6.45     7.36     0.00     2.75     2.48     2.54     3.81     7.51
         9.99     12.63    12.84    13.39    14.09    13.34    15.24    16.48
Frd                        5.95     4.84     8.45     9.53     2.75     0.00     1.86     4.64     1.64     5.36
         7.79     10.45    10.83    11.20    12.08    11.33    13.13    14.34
Stj                        5.66     4.55     8.26     9.24     2.48     1.86     0.00     4.78     3.04     6.51
         8.91     11.70    11.83    12.33    13.08    12.33    14.30    15.50
Cht                        4.58     3.46     7.16     8.10     2.54     4.64     4.78     0.00     5.94     9.41
         11.86    14.55    14.75    15.41    16.00    15.25    17.20    18.36
Gt1                        7.09     5.98     9.73     10.89    3.81     1.64     3.04     5.94     0.00     4.18
         6.44     9.20     9.24     10.03    10.49    9.74     11.93    13.15
Riv                        10.70    9.59     13.28    8.88     7.51     5.36     6.51     9.41     4.18     0.00
         2.94     7.69     5.80     6.40     7.05     6.30     8.36     9.40
Qbc                        13.21    12.10    15.75    16.79    9.99     7.79     8.91     11.86    6.44     2.94
         0.00     3.28     3.55     4.19     4.80     4.05     6.15     7.48
Gt2                        15.79    14.68    18.41    19.50    12.63    10.45    11.70    14.55    9.20     7.69
         3.28     0.00     2.40     2.86     3.65     2.90     4.99     6.05
Mte                        16.01    14.90    18.70    19.65    12.84    10.83    11.83    14.75    9.24     5.80
         3.55     2.40     0.00     1.40     1.75     1.00     3.00     4.13
Gt3                        16.50    15.40    19.18    19.95    13.39    11.20    12.33    15.41    10.03    6.40
         4.19     2.86     1.40     0.00     2.93     1.68     3.93     4.69
Caw                        17.26    16.15    19.95    20.90    14.09    12.08    13.08    16.00    10.49    7.05
         4.80     3.65     1.75     2.93     0.00     1.75     4.25     5.38
Mtw                        16.51    15.40    19.20    20.15    13.34    11.33    12.33    15.25    9.74     6.30
         4.05     2.90     1.00     1.68     1.75     0.00     2.50     3.63
Otn                        18.44    17.33    21.04    16.60    15.24    13.13    14.30    17.20    11.93    8.36
         6.15     4.99     3.00     3.93     4.25     2.50     0.00     2.90
Ots                        19.65    18.54    22.24    17.81    16.48    14.34    15.50    18.36    13.15    9.40
         7.48     6.05     4.13     4.69     5.38     3.63     2.90     0.00;

end;
```

Appendix L: One-Way ANOVA Test

Analysis of variance or One-Way ANOVA is used [46] to investigate the population mean of a one factor problem where there are more than two samples to be compared.

The calculation procedure is explained as follows, however a reader shall refer to [46] for a detailed explanation of the steps. Minitab™16 is used to perform the actual calculation in this thesis.

One-Way ANOVA

Assumption and Hypothesis in One-Way ANOVA

The initial hypothesis in One-Way ANOVA test is shown below. If every observation j form every sample

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_k$$

$$H_1 = at\ least\ two\ of\ the\ means\ are\ not\ equal$$

i is shown by Y_{ji}, then one can show each observation in the form of:

$$Y_{ij} = \mu + \alpha_i + \epsilon_{ij}$$

Where μ, is the overall mean across all samples, α_i is the effect of the treatment i (ex. the effect of policy variation D2D, PI, HY), and ϵ_{ij} is the random error and plays the same role as the error term in regression analysis. Therefore the above hypothesis can be written in the following form:

$$H_0 : \alpha_1 = \alpha_2 = \cdots = \alpha_k = 0$$

$$H_1 = at\ least\ one\ of\ the\ \alpha_i\ is\ not\ equal\ to\ zero$$

The notations are adopted from [46].

$K - 1$	*Degree of freedom*

n	*Number of observations in a sample*
y_{ij}	*The j-th observation of the i-th sample*
$\bar{y}_{i.}$	*The mean value of the i-th sample*
$\bar{y}_{..}$	*Overall mean value across of all samples*
$SST = \sum_{i=1}^{k}\sum_{j=1}^{n}(y_{ij} - \bar{y}_{..})^2$	*Total sum of squares*
$SSA = n\sum_{i=1}^{k}(\bar{y}_{i.} - \bar{y}_{..})^2$	*Treatment sum of squares*
$SSE = \sum_{i=1}^{k}\sum_{j=1}^{n}(y_{ij} - \bar{y}_{i.})^2$	*Error sum of squares*

Table 22 - Analysis of Variance - One-Way ANOVA

Source of Variation	Sum of Squares	Degree of Freedom	Mean Square	Computed *f*
Treatments (Systems)	SSA	$K-1$	$s_1^2 = \frac{SSA}{K-1}$	$f = \frac{s_1^2}{s^2}$
Error	SSE	$K(n-1)$	$s^2 = \frac{SSE}{K(n-1)}$	
Total	SST	$Kn-1$		