Integrated optimization and simulation models for the locomotive refueling system configuration problem

by

Lucas George Verschelden

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Industrial and Manufacturing Systems Engineering

College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2017

Approved by:

Co-Major Professor
Dr. Todd W. Easton

Approved by:

Co-Major Professor
Dr. Jessica L. Heier Stamm

Abstract

Locomotives in the U.S. use over 3 billion gallons of fuel each year and faster refueling can increase rail network capacity without the infrastructure cost associated with new terminals or tracks. This thesis introduces the locomotive refueling system configuration problem (LRSCP), which seeks to improve efficiency in refueling yards through new technologies or policies. This research also creates two new methods to solve LRSCP.

The first method uses an integer program to solve the off-line LRSCP and develop a static refueling policy. The train refueling integer program, TRIP, maximizes the weighted number of train combinations that can be refueled without delay. TRIP is optimized and its outputs are used as inputs to a simulation developed in Simio® for testing and validation.

The second method creates an integrated integer program and simulation to solve the on-line LRSCP and produces a dynamic refueling policy. This tool, built in Python, incorporates a different integer program, the strike line integer program (SLIP), into the simulation. SLIP determines the optimal refueling assignment for each incoming train. The simulation incorporates SLIP's solution for testing and validation. This tool is truly integrated and requires approximately 300 instances of SLIP to simulate a single day.

Based on experimental results, solving either TRIP or SLIP and incorporating the optimal refueling policy improves railyard operations by 10 to 30%. This impact is statistically significant and increases the capacity of a railyard. Additionally, it impacts other important parameters such as time spent in the yard and the maximum queue for the railyard. Furthermore, there is a significant decrease in wasted time and an improvement to railyard efficiency. Implementing either method should increase a railyard's capacity and significantly increase revenue opportunities.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to acknowledge the countless friends that have helped to mentor me and push me throughout my college career. I would like to give a special thanks to the KSUMB Alto Saxophones for being a part of my family and always pushing me to do more than I thought was possible. I’d also like to thank my industrial engineering colleagues who have given me more joy and knowledge than anyone could imagine.

I would also like to acknowledge Dr. Todd Easton for being the greatest teacher, mentor, and companion that I could ever ask for. The knowledge I have gained and the advice I have been given have been invaluable and I could never thank you enough. Additionally, Dr. Jessica Heier Stamm has been instrumental throughout my growth as an industrial engineer as both a teacher and a mentor. I have learned so much and been very grateful to the opportunities that you have given to me. Finally, I’d like to thank Dr. Chwen Sheu for agreeing to be on my committee and for believing in me.

Dedication

I dedicate my work to my family and friends. My parents, Mark Verschelden and Katherine Verschelden, raised me to be an industrial engineer and follow in their footprints. I could not be more proud to have done so. I am so grateful to them for their unconditional support and their constant belief in me. I’d also like to thank my siblings, Thomas, Samuel, and Sarah for setting great examples for me and pushing me to try to surpass them. I hope that I can set the same example for you.

Chapter 1 - Introduction

Efficient railroad operations are vital to the economy. Railroads account for nearly 40 percent of freight movement ton-miles in the U.S., and total freight tonnage moved by rail is expected to increase 22 percent between 2010 and 2035 (U.S. Department of Transportation 2010). U.S. Class I railroads generated $67.6 billion in freight revenue in 2012, a value that more than doubled (in inflation-adjusted dollars) in comparison to 2000 (U.S. Department of Transportation 2015).

Refueling yards are vital points on freight rail networks. U.S. Class I railroads used 3.687 billion gallons of fuel in 2014 to power 25,916 locomotives (U.S. Department of Transportation 2016). A typical yard may refuel more than 100 trains per day. At busy times, a refueling station may make concessions, such as slowing train arrivals. The research in this thesis improves the efficiency of railyards through the introduction and optimization of the locomotive refueling system configuration problem.

1.1 Locomotive Refueling System Configuration Problem Description

Refueling yards utilize fuel pumps on fixed platforms and direct-to-locomotive (DTL) fuel trucks. Locomotives at the front of a train (front-end power) are fueled at platforms that are situated alongside the tracks and equipped with multiple fuel pumps. Some large trains have distributed power, meaning there are locomotives at the rear and/or middle of the train in addition to those at the front. Distributed power locomotives are refueled with DTL trucks.

In addition to fueling, other processes, including inspection or maintenance, may happen at these railyards. The typical processes followed upon a train's arrival to the yard are depicted in Figure 1. The inbound train first traverses to the fueling platform and stops at a designated "strike line". A flag indicates that it is safe for fueling to begin and an inspector boards the train.

The fueling process is dependent upon the availability of fuel pumps, whose functional radius is limited, and DTLs, which are dispatched to distributed power, if applicable. Inspection can happen concurrently with fueling. Upon completion of these tasks, the train waits for the crew to finalize preparations for departure and for appropriate track clearance before departing the yard.

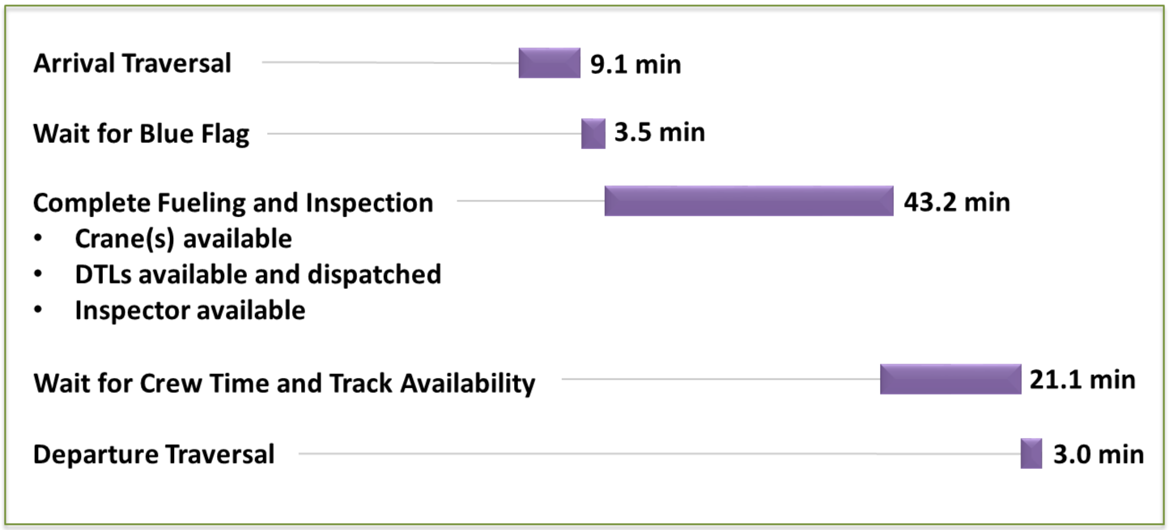

Figure 1: Processes for each train in yard; times are representative of current system

If rail traffic increases, as it is projected to do, adding infrastructure, including tracks and fueling platforms, can increase a yard's capacity, but at considerable expense. A recently-developed upgraded refueling system safely dispenses fuel more than twice as fast as the current industry standard. Upgrading to a new system may represent a more economical option not only to increase an individual yard's capacity, but also to impact throughput across the network.

Refueling speed is important to overall freight railroad operational efficiency, because it should directly impact the time trains spend in the railyard. Decreasing refueling time for each train has the potential to increase the maximum number of trains that can pass through a yard each day, which positively impacts the overall rail network capacity. Regardless of whether the new technology is adopted, a challenge for a given railyard is how to configure existing locomotives with the fuel platform to maximize the associated benefits.

This description led to the creation of the locomotive refueling system configuration problem (LRSCP). LRSCP arises when railroad companies aim to improve efficiency in refueling yards through new policies or technologies. The first decision analyzed by LRSCP is the strike line of a track. The strike line of a track is the point where the train stops when arriving at the platform for refueling. The second decision assigns each front-end locomotive to a pump for refueling. An optimal solution to LRSCP provides strike lines and assigns locomotives to refueling pumps on a given platform, which improves efficiency and increases capacity.

The strike line is important because its sets the location of the locomotives on the train relative to the stationary pumps on the platform. If the strike line was poorly set, the train would stop in front of the beginning of the platform and no pump would be able to reach and fuel the first locomotive. Therefore, having an ineffective strike line would be detrimental to being an efficient refueling process. Additionally, having a poorly placed strike line affects the reach of pumps in the middle of the platform. The reason for this is that the fuel ports on a locomotive are approximately 72 feet apart, the length of a locomotive. However, fuel pumps are only placed 50 feet apart with a 30 foot reach. Therefore, pumps cannot reach the locomotives in the order that they are located on the platform.

Figures 2 and 3 show solutions to a typical LRSCP. This instance has seven pumps and a track on each side. Currently, one track has train 1 with three locomotives and the other track has train 2 with four locomotives. In Figure 2, each track has a strike line of 0 where the train stops at the beginning of the platform. Here, locomotives 5 and 6 cannot be fueled by any pump because the only pumps within reach are currently fueling locomotives 2 and 3. Therefore, there will be a delay for train 2 as it waits for train 1 to finish fueling its locomotives so that the pumps can be used by train 2. Figure 3 shows a strike line of 10 and a strike line of 66. Train 1 uses pumps 1, 3, and 5 while train 2 uses pumps 2, 4, 6, and 7. This is the exact same scenario except that with different strike lines, all the locomotives on both trains can be fueled without delay.

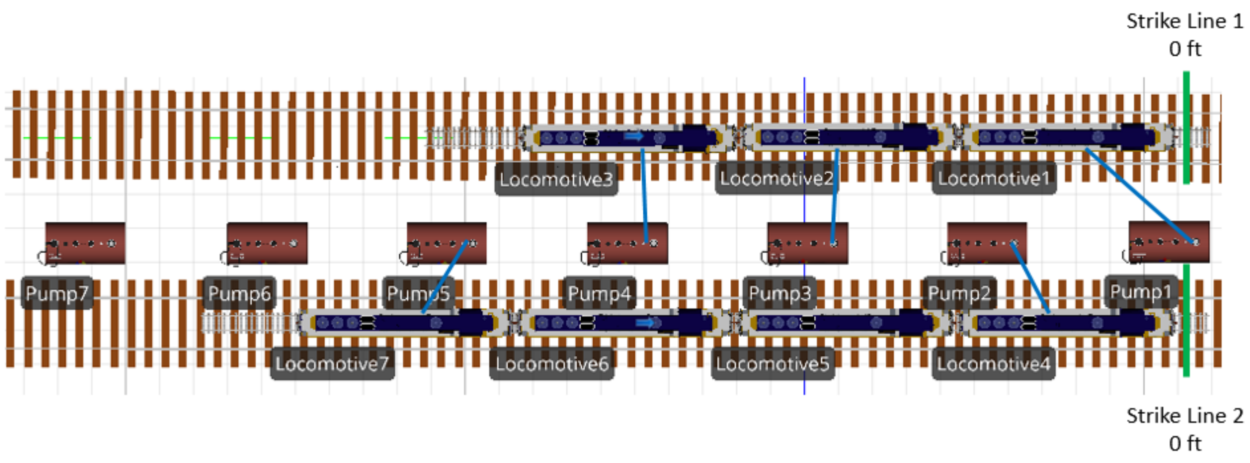

Figure 2: Locomotive Refueling System Configuration Problem Example (0, 0)

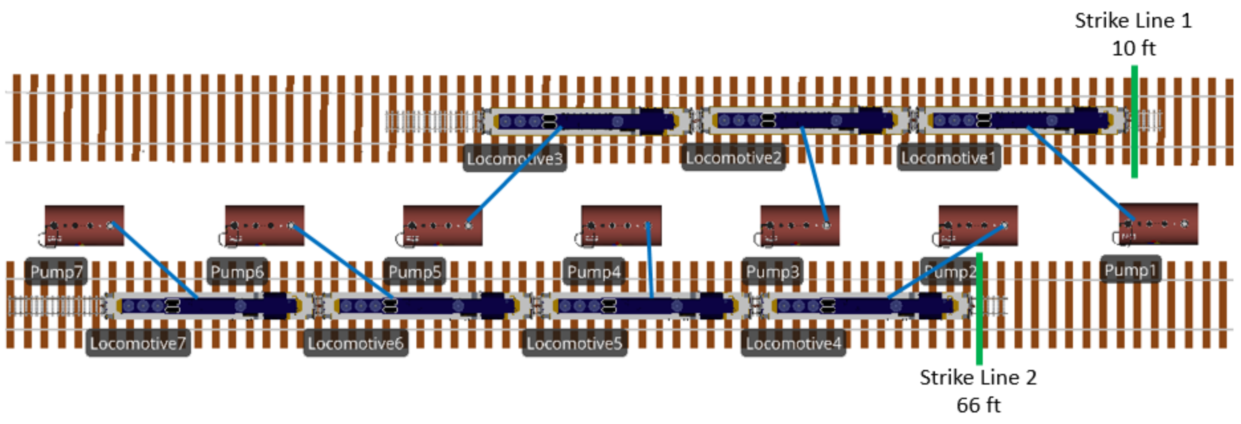

Figure 3: Locomotive Refueling System Configuration Problem Example (10, 66)

This thesis studies two versions of LRSCP, an off-line method and an on-line method. The off-line solution method is a simulation built in Simio®. This computer simulation is considered off-line simulation because an integer program is solved and feeds the strike lines as input to the simulation. However, the two do not interact and the strike line policy is static. The on-line solution method integrates an integer program and a discrete-event simulation model to develop a dynamic strike line policy and I manually coded in Python. In this case, the integer program and the computer simulation interact and both feed as inputs into the other. Both the off-line and on-line methods improve railyard efficiency.

A majority of this chapter and chapter 3 are taken from a paper that will be published as a refereed conference proceedings (Verschelden *et al.* to appear).

1.2 Research Motivation

As previously stated, rail traffic is growing considerably. Rail companies prefer cheaper and easier solutions as opposed to adding infrastructure. One such major railway company was doing this analysis and came to us to analyze the effect of new advances in refueling speeds and what kind of effect it would have on their capacity.

As this problem was being analyzed, an additional problem presented itself. This problem was that the trains had to be correctly aligned with the pumps upon arrival in order to take advantage of the faster refueling time. If the pumps could not reach every locomotive, the train would have to delay and its completion time would return back to a similar value as fueling without using the new refueling technology.

Therefore, the importance of aligning trains to pumps is escalated with the new refueling technology. As more trains enter the system and at faster rates, the more important it is to have platforms performing at high efficiencies and without waste or delay time. Therefore, correct configuration and alignment became a huge component of interest. Additionally, this problem is sufficiently complex and trivial verifiable solutions did not readily present themselves.

1.3 Research Contributions

This thesis makes three major contributions to railroad research optimization. The first contribution is the introduction of a new problem, the locomotive refueling system configuration problem. LRSCP bridges the gap between railroad research problems that examine the entire network and problems which examine only a railyard. LRSCP incorporates both the scheduling of the network and the tactical decisions of running a railyard into a single problem. Both an off-line and on-line version of LRSCP are presented. Furthermore, obtaining and verifying the quality of an LRSCP's solution is challenging due to its inherent properties including randomness, scheduling, and size.

The second primary contribution is a solution and validation methodology to solve the off-line LRSCP. This process begins with an integer programming model, the objective of which maximizes the weighted number of train combinations that can be refueled without delay. This integer program is solved and used as input to a simulation developed in Simio®. The simulation produces two metrics: the average time each train spends in the yard and the average maximum number of trains waiting to enter the yard. Metrics for multiple scenarios provide insight about the impact of refueling system configuration on overall yard operations. Based on the experimental results, solving the integer program and incorporating the optimal strike lines improves the refueling yard operations regardless of whether or not a new refueling technology is adopted. This impact is statistically significant and allows more trains to be processed each day.

The third significant contribution is the creation of a method to solve the on-line LRSCP. This tool, built in Python, dynamically incorporates an integer program into the simulation. The integer program determines the optimal strike lines and pump assignments for each incoming train. The simulation uses the integer programs' output to determine track and pump usage. This is truly an integrated system as approximately 300 integer programs are solved to simulate a single day. Similar to the off-line simulation, this simulation accurately models the railyard and examines the impact of different LRSCP solutions. Using the optimal on-line solutions results in a 30% improvement in wasted time over intuitive polices and similar results to the optimal off-line solution.

1.4 Outline

Chapter 2 describes the background necessary to understand the research. In the beginning, it examines the history of simulation including random numbers, formulation, and the different types of simulation including discrete event and agent-based modeling. It follows with an overview of optimization specifically related to integer programming. It explains how the branch and bound algorithm solves integer programs and highlights integer programming applications. Finally, it reviews related railroad research optimization applications.

Chapter 3 covers the off-line simulation and optimization and provides a full problem description of the locomotive refueling system configuration problem. It then introduces the integer programming formulation for maximizing the weighted number of train combinations that can be refueled without delay. A simulation is discussed including its parameters and complexity. Finally, the chapter reviews the outputs of the off-line simulation with the integer program and compares the results of the current static strike line policies.

Chapter 4 covers the on-line simulation and optimization. In the beginning, it describes the differences between the off-line simulation built in Simio® and the on-line simulation built in Python. Next, it describes the integer program formulation for minimizing the completion time of all incoming trains. It also describes the simulation in detail including its parameters and complexity. Finally, it reviews the outputs of the on-line simulation and integer program and the analysis and results of the dynamic strike line policy.

Chapter 5 draws conclusions from Chapters 3 and 4 with regards to the use of strike lines in the locomotive refueling system configuration problem. It continues with conclusions drawn from the research methods presented in this thesis. It finishes with some topics to consider for future research.

Chapter 2 - Literature Review

The purpose of this chapter is to provide the necessary background knowledge in the realms of simulation, optimization, and railroad applications in order to understand this research. The first section covers discrete event simulation as a tool that improves real-world problems, provides a better understanding of the system, or tests future scenarios. The second section focuses on the importance of integer programming and optimization tools to guarantee the best solution to a complex problem. Lastly, this chapter examines optimization research related to railroads to support the idea that LRSCP is a new problem.

2.1 Simulation

A simulation is best described as a representation of a real-world system and is typically used for training, prediction, and system improvement. For example, a common tactic used by the military is to simulate dangerous situations and complex tactics. Additionally, nearly all sports teams simulate the opposing team with a practice squad or backups to prepare for an upcoming game. Another common real-world simulation is operations done by medical doctors and surgeons to train in a safe environment prior to performing on a live subject. These examples show how simulation can greatly increase the safety, efficiency, and outcome of systems.

As technology has progressed, computer simulations have become increasingly common for differing types of systems. Computer simulations model non-human controlled systems, complex systems, and systems that span long periods of time. These types of systems can provide knowledge about the past, the present, and the future. Some interesting computer simulations include production systems (Huang et al. 1983), weather forecasting (Johnson et al. 1996), the big bang (Shavitt and Tankel 2004), flight (Jones et al. 1999), health-care (Jun et al.

1999; Brailsford and Hilton 2001), and disease spread (Harvey et al. 2007). From this point forward, simulation will refer to computer simulation.

Simulation uses different inputs to model complex decisions in order to provide the user with additional outputs or insights. These insights are difficult to gain while examining the real-world system. Additionally, simulation can be used to forecast expected results and plan for items which cannot currently be monitored (Zhang et al. 2001). Simulation typically tests new designs, explores new operating procedures, identifies bottlenecks, tests hypothesis, and analyzes "what-if" questions (Shannon 1998).

Accurate simulation models can be difficult due to the complexity of the system, gaining reliable input data, and finding qualified individuals to build and analyze the model. Additionally, simulation's main weakness is its inability to optimize problems. Simulation often is used as an analytical tool, which leads to better decision making, but optimization is limited to the tested cases.

Discrete event simulations are models where the system changes only at discrete points in time (Banks et al. 1984). An example of such a system would be a classical queueing model where the queue size changes when a person enters or leaves the system. However, the system does not change its state at any other time besides these discrete events. In this research, discrete event simulation is used to model the railyard where the discrete events are when a train enters or exits the railyard, finishes fueling, etc. Other commonly modeled systems which use discrete event simulation include "manufacturing plants, inventory systems, distribution systems, communication networks, transportation networks, health-care delivery systems", and queueing models (Fishman 2001).

One common application of discrete event simulation analyzes manufacturing processes. One group of researchers used discrete event simulation to analyze the impact of lean manufacturing on a manufacturing process prior to implementation. They used this extra analytical tool to find that lean manufacturing saves “inventory, floor space, transportation, manpower, equipment, lead times, and variability” (Detty and Yingling 2000). Another manufacturing application includes an on-line simulation to increase the flexibility and adaptability of their production line (Drake et al. 1995). In another application, the changes made with use of a discrete event simulation created “tangible savings including a capital avoidance of $5,000,000, reduced daily management time of 12 man-hours/day, and increased tester utilization” (Sivakumar 2001).

Transportation and logistics is another area in which discrete event simulation is commonly used. The operations of seaports have been examined with discrete event simulations by Nevins et al. (1998). Their work states, “It provided a useful planning tool for analyzing more efficient ways in which to conduct seaport operations.” Similarly, the world-wide transportation of crude oil has been analyzed and provided benefits into the logistics industry (Cheng and Duran 2004).

The next section covers how simulation was first used and briefly explains how it has grown since then. Following that, the computational aspects of simulation including the structure and explanation of pseudo-random numbers are described.

2.1.1 A History of Simulation

Simulation is first credited to Georges-Louis Leclerc, Comte de Buffon, when he conducted his needle experiment in 1777 (Buffon 1777). Buffon’s experiment was to examine the probability that a needle randomly tossed onto a floor with parallel lines would intersect a line. The goal was to estimate the value of π and prove his math correct through experimentation. This is documented by Goldsman et al. as “the earliest example of using independent replications of a simulation to approximate an important physical constant” (2010).

Stan Ulam is often referred to as the first modern user of simulation as a major tool. While working on the Manhattan Project, Ulam worked with ENIAC, one of the world’s first electronic computers in 1946. Ulam designed and created the first computer-based simulation, a method called the Monte Carlo Method (Ulam and Metropolis 1949). Ulam’s success led to great increases in the use and creation of simulation in the growing computerized world.

The upgrading of computers from the 1950s to 2010s led to major increases in computer simulations, especially with emphasis on more complex and realistic problems. Richard Conway was the first widely recognized man to create a general framework for developing digital simulations (Goldsman et al. 2010). Conway worked with Johnson and Maxwell at Cornell University to write a paper about the problems of digital systems simulation (Conway et al. 1959). As advances continued to increase computer memory and processing power, digital simulations became more advanced and companies began developing their own SPLs (Simulation Processing Languages) and specific software. One such company and software is called Simio®.

Simio® is a commercially available advanced discrete event simulation software created in the mid-2000s. Simio® focuses on an object orientation while most simulation software promote a process orientation. Yet, Simio® also supports the use of multiple modeling paradigms such as a process or event orientation. Simio® can also simulate both discrete and continuous systems, as well as systems described by agent-based modeling. This mixing of different systems within a single model gives the user more control to model complex systems.

Simio$^{®}$ is used in this research to build an accurate and complex representation of a railyard and includes the ability to show an animated railyard. As summarized by Pegden, Simio$^{®}$ allows for "greater creation of objects with more complex behaviors, specialized features to directly support capacity scheduling models, and supports multiple modeling paradigms with an emphasis on object oriented models" (2008). The next section covers how to build a simulation model with respect to these modeling paradigms.

2.1.2 Computational Aspects of Computer Simulations

The vast majority of computer simulations follow a similar framework, which is described by Banks et. al in their book, *Discrete Event System Simulation* (1984). Simulation models require six main objects: the system, the entities or controllers, the attributes, the activities, the events, and the states (Banks et al. 1984). These six objects can be seen below in Table 1 and allow for the modeling of nearly any real-world system and should be used as a framework when creating a simulation.

Modelers can follow this framework to help them create useful and accurate simulations. However, this remains a job for a highly specialized employee with the knowledge and resources to build a successful model. Simulation experts typically have expertise in mathematics, process analysis, and computer programming. A key component of simulation is the use of random numbers to create repeatable yet random events to analyze many different scenarios. Random numbers are the focus of the next section.

Table 1: Examples of Systems and Components (Banks et al. 1984)

System	*Entities*	*Attributes*	*Activities*	*Events*	*State Variables*
Banking	Customers	Checking-account balance	Making deposits	Arrival; departure	# of busy tellers, customers waiting
Rapid rail	Riders	Origination; destination	Traveling	Arrival at station/ destination	# of riders waiting at station, riders in transit
Production	Machines	Speed; capacity; breakdown rate	Welding; stamping	Breakdown	Status of machines (busy, idle, or down)
Communication	Messages	Length; destination	Transmitting	Arrival at destination	# waiting to be transmitted
Inventory	Warehouse	Capacity	Withdrawing	Demand	Levels of inventory; backlogged demands

2.1.2.1 Random Numbers

The most fundamental principle of simulation is the ability to create truly random events. One or several random number generators are commonly used to create these random numbers. The quality of a random number generator has a huge effect on the quality of a simulation. Random numbers require two important properties, independence and uniformity. Going all the way back to Ulam, random numbers were generated with physical objects such as: drawing a card out of a deck, flipping a coin, white noise produced by electronic circuits, roulette wheels, and others (Niederreiter 1978).

Random numbers are often tied to a system or a method, and therefore, the random numbers could not be replicated and therefore are truly random. The examples listed above are all examples of random number generators. These numbers are random because they are not repeatable and are provably uniform, independent, and equally distributed.

There also exists another type of random number called pseudo-random numbers. These pseudo-random numbers are created using generators and mathematical algorithms. Effective pseudo-random number generators often start with a seed which is input into an equation or algorithm. Effective equations or algorithms have large cycles to prevent falling into continuous repeating of the same numbers. The reason for the seed is so that the random numbers can be duplicated. Changing the seed will lead to a different set of pseudo-random numbers. Pseudo-random numbers often cause issues with the definition of random numbers in that they may not always be provably uniform, independent, or equally distributed. In most cases, they appear and act as if they were entirely random.

One effective pseudo-random number generator is the Mersenne Twister. The Mersenne Twister is a 623-dimensionally equidistributed uniform pseudo-random number generator with a cycle of 2^{19937} (Matsumoto and Nishimura 1998). Another random number generator developed by Wichmann and Hill has a cycle of 2.78×10^{13} (Wichmann and Hill 1982). For reference, “even using 1000 random numbers per second continuously, the sequence would not repeat itself for over 880 years” (Wichmann and Hill 1982). The randomness of these number generators leads to better simulation software capabilities and increases the true randomness of a simulation.

Pseudo-random numbers are most commonly used for computer simulations because the numbers are repeatable. Repeatable random numbers are important for computer simulations because it allows the simulation analyst to recreate scenarios. This is most important when debugging a simulation in the building phase. Without the ability to recreate the same scenario, it becomes challenging to identify or solve a problem. These pseudo-random numbers still are effective at creating the random events and replications necessary for a simulation. Additionally, randomness can be applied to decisions made during simulations, which lead to a type of simulation modeling called agent-based modeling.

2.1.3 Agent-Based Simulations

Agent-based modeling is a popular type of simulation model in today's society due to its ability to handle increasingly complex problems. In reality, agent-based modeling is a slightly more complex form of discrete event simulation with the emphasis placed on action increments instead of discrete time increments. Agent-based modeling focuses on the ability for "agents" to follow decision rules to make independent decisions rather than acting passively (Macal and North 2007). Agent-based simulations emphasize entities' actions as opposed to discrete event, which focus on time. The ability for agents to make decisions in a simulation increases the ability to model complex systems, such as ones where the decision of one party affects the decision of another as well as the outcome of the system. These decisions can be made following different rules by the agent such as being "mutually reactive to current and/or past interactions", "responsive to their future expectations and take decisions in a selfish way", or "responsive to their own and other people's future expectations" (Helbing and Balietti 2012). Agent-based modeling poses various advantages over traditional and discrete event simulations and is summarized in Table 2.

Table 2: Advantages of Agent-Based versus Traditional Modeling (Heppenstall et al. 2012)

Traditional Modeling	Agent-Based Modeling
Deterministic (one future)	Stochastic (multiple futures)
Allocative (top-down)	Aggregative (bottom-up)
Equation based formulas	Adaptive agents
Do not give explanations	Explanatory power
Few parameters	Many parameters
Spatially coarse	Spatially explicit
Environment given	Environment created
You react to them	You learn from them

The complexity of agent-based models has led to numerous applications in various industries. One such application is the humanitarian industry and emergency response. "Designing and managing supply chains for disaster response entail significant challenges due to uncertainty, great urgency, limited or damaged infrastructure, limited resources, dynamic conditions, and the large number of entities involved in the response" (Ergun et al. 2013). The agent-based simulation built by Megan Menth analyzed collaboration between different agencies in response to disaster relief as well as tested various decisions made by humanitarian response agencies to cover as many people as possible with all the resources needed to regain life's essentials (Menth and Heier Stamm 2015). Continued research in the realm of humanitarian logistics helps to increase the speed of relief, the care provided, and the quality of life for disaster victims.

Another common application of agent-based modeling is in disease spread. Various models have been built to model the spread of contagious diseases throughout a population as well as administering a vaccine and curing diseases (Perez and Dragicevic 2009; Eubank et al 2004; Bauer et al. 2010; Easton et al. 2011). Additional applications for agent-based modeling include modelling economies (Tesfatsion 2005), traffic (Chen et al. 2005), and organization management (Bonabeau 2002).

As mentioned previously, simulations lack the ability to optimize problems and only perform case by case analysis. Therefore, other tools are used to optimize problems. One such common tool is integer programming and is the focus of the next section.

2.2 Optimization and Integer Programming

Integer programming is a class of optimization problems in which some or all decision variables are restricted to integer values. Unlike simulation, integer programming solves an objective function restricted by constraints to an optimal mathematical solution (Hoffman and Ralphs 2012). An integer program (IP) has the following form:

Maximize $c^T x$

Subject to: $Ax \leq b$

$x \in \mathbb{Z}^n_+$

where $A \in \mathbb{R}^{mxn}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$ are input parameters and x is a decision variable vector.

A provably optimal solution is output from the integer program as z^* and x^*. This optimal solution is found with different algorithms, the most common of which is called branch and bound.

2.2.1 Branch and Bound

Land and Doig created branch and bound in 1960 to solve integer programs (Land and Doig 1960). Branch and bound solves integer programs' linear relaxations in order to find the optimal solution to the integer program. The linear relaxation is the integer program without the integer constraints. Formally, the linear relaxation is

Maximize $c^T x$

Subject To: $Ax \leq b$

$x \geq 0$

where $A \epsilon \mathbb{R}^{mxn}$, $b \epsilon \mathbb{R}^{m}$, and $c \epsilon \mathbb{R}^{n}$ are input parameters and x is a decision variable vector.

The linear relaxation is a linear program. The most common algorithm used to solve these linear relaxations is the simplex method (Dantzig 1947). The simplex method solves linear relaxations with the addition of a slack variable, which allows it to easily optimize the decision variables (Wolfe 1959). Even though the simplex method runs in exponential time, interior point algorithms exist that run in polynomial time (Boyd and Vandeberg 2004; Potra and Wright 2000).

The first step of branch and bound solves the IP's linear relaxation. This creates the root node. If the solution is integer or infeasible, then the IP is solved. If it is unbounded and there exists an integer solution and rational ray of unboundedness, then the IP is unbounded. If none of these cases exist, then there exists a fractional variable x_i.

Two new nodes are created and become child nodes, which is called branching. One child node adds the constraint $x_i \leq x_i^* - (x_i^* \text{ mod } 1)$ and the other child includes the constraint $x_i \geq x_i^* + 1 - (x_i^* \text{ mod } 1)$, where x_i^* is taken from the linear relaxation solution. This process of solving nodes and branching continues until all nodes are fathomed. A node is fathomed if it is infeasible, integer, or has a worse linear relaxation objective function value than the best found integer feasible solution. This last type of fathoming is the bound portion of the algorithm. The best found integer solution is updated as the algorithm progresses. Once all other nodes are eliminated, the best integer solution (if one exists) is reported.

Integer programs are *NP*-complete problems. This means that all currently known algorithms used to solve integer programs take an exponential amount of time to give a provably optimal solution to at least some class of problems. It is important to note that there is no guarantee that an IP will solve within any realistic amount of time. However, integer programs are commonly used in real-world applications due to the fact that many problems are solved in a reasonable amount of time.

2.2.2 Applications of Integer Programming

Integer programming is commonly used to solve real-world applications. These applications solve problems in sports, scheduling, healthcare, manufacturing, and transportation. A few examples of these problems are examined in this section.

One may be surprised to learn that integer programming is used in the wide world of sports. Sports require scheduling of many different teams and games with constraints on opponents, locations, times, etc. These become complex integer programming problems. One such application of integer programming is the scheduling of games for the Ecuadorian football league (Recalde et al. 2013). In this case, the goal of the problem is to design a schedule for the 12 teams in the Ecuadorian A Series Championship. The constraints for this schedule include the fact that the first two phases of the league are played in an "inverted mirror double round robin tournament" involving 22 rounds. The third phase simply has the first place team of the first phase and the first place team of the second phase play two matches to decide the championship.

The complexity of this problem arrives in the scheduling constraints. The teams must play home-away consistently. Some teams share a stadium, therefore, those two teams cannot be both playing home at the same week. Also, the schedule must account for fan participation as the biggest games of the year should not be played at the beginning or the end of the season. Incorporating all these constraints and formulating this problem as an integer program, Recalde et al. (2013) developed a solution which was presented to the leaders of the Ecuadorian football league and got great support for their ability to develop good schedules. Additional sports applications include the travelling tournament problem (Ribiero 2011), distance minimization (Ribiero 2011), and referee assignment (Alarcon et al. 2012).

Another typical area for integer programming problems is in the medical industry. One particularly interesting application uses integer programming to reduce waste and limit shortages of blood products at hospitals (Gunpinar and Centeno 2014). Blood is obviously a scarce resource and limiting waste is important while the cost of being short of inventory is potentially a human life. Gunpinar and Centeno modeled the supply chain of blood using a hospital and blood center where the demand for blood comes from the hospital and the blood center has a limited supply of blood. This becomes a difficult problem due to the fact that most emergency procedures are not forecastable and therefore the amount of blood needed at a given time is entirely uncertain. Additionally, most doctors ask for more blood than they expect to need in order to be safe. This causes excess blood to go back to reserves and be wasted without an additional demand for the same blood. After modeling, solving, and testing their solution, Gunpinar and Centeno achieved more efficient and cost effective blood resource allocations. Additional medical applications include the scheduling of nurses (Hasegawa and Kosugi 2006) and measuring the impact of client choice on preventative healthcare (Zhang et al. 2011).

One common theme which can be seen even within industries with integer programming is the scheduling problem. Scheduling is commonly modeled and optimized using integer programming, including in this thesis. In this thesis, integer programming is used to solve the scheduling of trains to tracks and of pumps to locomotives. Other scheduling applications include scheduling maintenance for coal chains (Boland et al. 2012), traffic or school bus scheduling (Fugenschuh 2007), and production scheduling for make-to-order manufacturing

(Sawik 2003). Optimization techniques are also common in the railroad industry, which is the focus of the next section.

2.3 Locomotive Research Applications

This section covers some specific examples of optimization research from the railroad industry. The purpose of this section is to show which problems have previously been studied in railroad research and what research techniques are used to solve them. Learning about similar problems makes it clear that LRSCP is different and adds a new problem to railroad optimization research.

Strategic, tactical, and operational decisions arising in the railroad industry are complex, highly combinatorial, and frequently characterized by uncertainty. Researchers and practitioners have developed numerous models to support decision making; Cordeau et al. (1998) and Assad (1980) provide reviews. Integer programming and simulation are among the most common methodologies.

Discrete event simulation is a commonly used tool in railroad logistics. One of the most common uses is train scheduling (Dorfman and Medanic 2004; Sajedinejad et al. 2011). Scheduling efficiently allows multiple trains to travel along the same track and therefore increases railway capacity. Dorfman and Medanic created the TAS (Travel Advance Strategy) which was developed using discrete event simulation (Dorfman and Medanic 2002). Lesyna also used simulation to size industrial rail car fleets to increase efficiency on railways (Lesyna 1999). Additionally, Hill and Bond worked on a blocking and signaling system for railways (Hill and Bond 1995).

Other train scheduling problems consider operational guidance for handling disruptions to planned schedules (Barta et al. 2012; D’Ariano et al. 2007). Researchers have introduced models for assigning locomotives to trains (Vaidyanathan et al. 2008), crew planning (Chahar et al. 2011), scheduling maintenance (Budai et al. 2006), and managing empty railcar movements (Sherali and Suharko 1998).

The research presented in this thesis occurs in railyards as opposed to railways. Therefore, research conducted for railyards is of particular interest in order to see if LRSCP is a new problem. A railyard is a complex series of railroad tracks in a designated area specifically for sorting, loading or unloading, or refueling railroad cars. These processes can be complicated due to the size of trains and the number of changes which need to be done in a railyard. When a train comes into a railyard, it may have cars which need to change trains, cars which need to be unloaded or loaded into unique stationary containers, and locomotives that require refueling. Performing all these tasks at the same time and as quickly as possible poses many interesting problems.

An important tactical decision is blocking, the process of grouping shipments (and their corresponding cars) together so that shipments need not be sorted in every yard (Ahuja et al. 2007; Daganzo 1986; Newton et al. 1998). Blocking is important to limit the amount of work done in each railyard to increase the efficiency of a given train as opposed to a given railyard. Blocking is especially important for trains transporting numerous unique shipments with different sources and destinations due to limiting infrastructure. Given a blocking plan and a train schedule, the block-to-train assignment problem determines which trains will transport each block (Jha et al. 2008).

Additional railyard activities include maintenance, inspection, and refueling. The complex operational processes are important in railroads' overall efficiency. Detailed simulation models depicting railyard operations have been proposed (Lin and Cheng 2009; Lin and Cheng

2011), enabling decision makers to evaluate changes in infrastructure, resource allocation, and operating policies. Similarly, He, Song, and Chaudhry (2003) propose an integer programming model that optimizes yard operations. Others have considered specific components within the framework of yard operations, such as optimal container transfer between trains in a yard where freight moves by container-on-flatcar (Bostel and Dejax 1998).

Solutions to one railroad management problem often serve as inputs to other decision models. The locomotive fleet refueling problem considers a railroad company's decisions about when and where to refuel locomotives. Given a routing and scheduling plan for locomotives in a fleet, the company must decide at which sites to refuel each locomotive and on what schedule (Nourbakhsh and Ouyang 2010; Raviv and Kaspi 2012). This decision is impacted by site-specific fuel costs, contracting costs with fuel suppliers, and the costs for delays incurred while locomotives are being fueled.

Many problems in railroads examine problems within a single railyard. Numerous other problems focus on the efficiency of a train with respect to its movement between multiple railyards. LRSCP makes decisions for both within a railyard and scheduling between railyards. Therefore, LRSCP examines both the scheduling of the incoming trains, and the locations for where to refuel trains within a railyard, bridging the gap between the different research areas.

Unlike models previously described in the literature, the approach proposed in this thesis models the refueling process in detail and specifically supports decisions that occur in the transition between fueling systems. Additionally, this model considers the size and schedule of incoming trains to optimize the refueling of every train incoming or currently in the yard. The contributions of this thesis are the introduction of the locomotive refueling system configuration problem and the methods to solve it. These results are examined in the next two chapters.

Chapter 3 - Off-Line Simulation and Optimization for LRSCP

The purpose of this chapter is to describe the methods used to analyze and solve the locomotive refueling system configuration problem using off-line methods. A substantial portion of this chapter is taken from a paper that will be published as a refereed conference proceedings (Verschelden *et al.* to appear). The first section describes the locomotive refueling system configuration problem. The second section covers the formulation of the integer program to find the optimal static policy solutions. The third section describes the discrete event simulation, which was built to model and test the system. Finally, the results of the off-line simulation and optimization are discussed and analyzed.

3.1 Locomotive Refueling System Configuration Problem

A primary contribution of this thesis is the introduction of the locomotive refueling system configuration problem, LRSCP. LRSCP arose when examining how trains were lined up when they arrived at a refueling platform. This was discovered as research was conducted on examining the effect of changing fueling speeds on refueling platforms. However, the change in refueling speed is not actually necessary for LRSCP to exist. LRSCP has an impact on efficiency of railyards with or without advanced refueling technology.

The aim of LRSCP is to have every train leave the railyard as soon as possible. It places specific emphasis on the configuration of the trains to the tracks and the locomotives to the pumps. The reason for this is the strange dimensions used by refueling platforms and locomotives. For example, the fuel ports on a locomotive are approximately 72 feet apart, the length of a locomotive. However, fuel pumps are only placed 50 feet apart with a 30 foot reach. Therefore, pumps cannot reach the locomotives in the order that they are located on the platform. Therefore, the goal of LRSCP is to line up the trains to the platforms in such a way that every locomotive on either track can immediately start refueling upon arrival and therefore be finished as soon as possible. This alignment is largely dependent upon the strike line.

The strike line of a track is the point where the train stops when arriving at the platform for refueling. This strike line is important because its sets the location of the locomotives on the train relative to the stationary pumps on the platform. A poorly placed strike line affects the reach of pumps and may increase the time of fueling as two locomotives are fueled sequentially by the same pump.

For example, if a strike line of one track is set to 0, then it is not feasible for pumps 1, 2, and 3 to fuel the three front locomotives. It is more likely that pumps 1, 3, and 4 will perform this fueling. An additional complication of the strike lines is that there are two strike lines for each platform, one on the east side of the platform and one on the west side. Therefore, if both strike lines are 0 and two trains with three locomotives are on both tracks, then an added delay will happen. The first locomotive of the second train has to go to the second pump, but then the fifth pump cannot reach the second locomotive. Consequently, the second train must delay fueling until the third pump finishes.

Figure 4 shows a solution to a typical LRSCP. This instance has seven pumps and a track on each side. Currently, one track has a train with three locomotives and the other track has a train with four locomotives. In the figure, the first strike line is at 10 and the second at 66. Now, train 1 uses pumps 1, 3, and 5 while train 2 uses pumps 2, 4, 6, and 7. Thus, these strike lines with this train combination have no added delays. The strike lines and pump assignments clearly have an impact on the efficiency of the system.

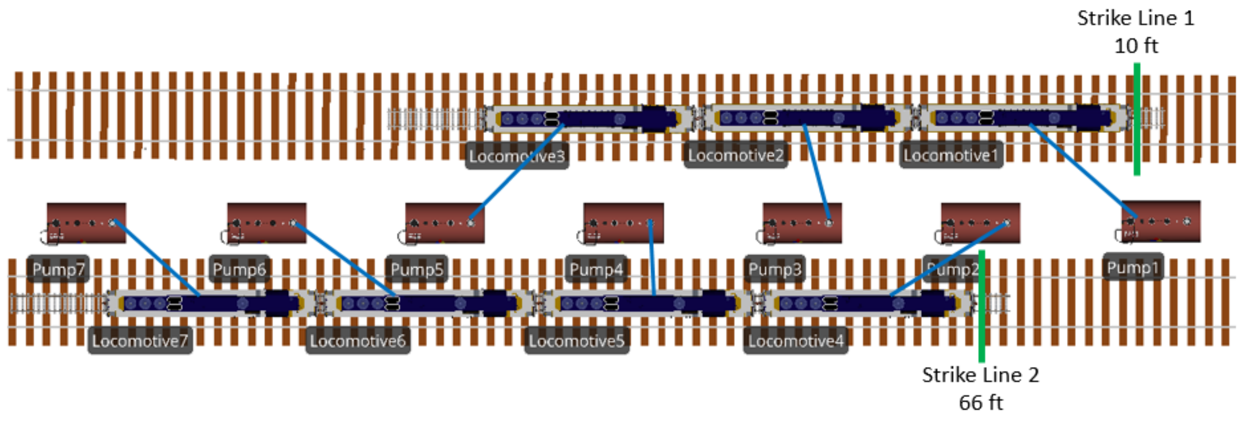

Figure 4: Locomotive Refueling System Configuration Problem Example

Various objectives or measures of efficiency can exist for LCRSP. The primary objective is to minimize the amount of wasted time. Wasted time consists of the time that the train had to delay prior to arriving at the platform plus any fueling delays at the platform. This waste time can be calculated as the difference between when a train leaves the platform minus the earliest time that the train could have arrived at the platform minus the train's longest locomotive refueling time. Note that waste time is not necessarily accrued every time a locomotive does not immediately start fueling due to the pump it needs being in use. It is possible for a locomotive to delay refueling and still have zero waste time because the locomotive which started refueling late required less fuel than some other locomotive that began refueling immediately upon arrival.

This thesis pursues two distinct LCRSP problems, off-line and on-line. The off-line LCRSP problem requires the strike lines to be set on each track; regardless of the train arrivals, these lines never change. Off-line policies provide consistency and regularity at the platform. On-line policies allow the strike lines to be set to different values based upon the arriving trains. Each arriving train can have a different strike line, which should improve refueling efficiency, but could complicate operational execution.

To solve the off-line LRSCP, a simulation is built in Simio® and an integer program is created to determine the optimal strike lines. The computer simulation is referred to as the off-line simulation because the integer program is solved exactly once and feeds the strike lines as a parameter input to the simulation. The integer program evaluates the waste time as simply a yes or no variable. Either a set of two trains can be fueled without waste time (feasible) or with waste time (infeasible). Minimizing the number of infeasible trains is the objective function of the off-line optimization.

To solve the on-line LRSCP, an integrated simulation and integer program is developed. The simulation, manually coded into Python, frequently creates an integer program, which is solved using CPLEX. The solution provides the strike line for the each incoming train, dynamic strike lines, and also determines which pump should be used to refuel each locomotive. Simulating the dynamic strike lines allow for comparison between having a single strike line versus changing strike lines for each train. Thus, the integer program and the computer simulation interact and both feed as inputs into the other in order to improve the efficiency of railyards.

Both off-line and on-line LRSCP instances have similar input. The on-line instance is presented in more detail in Chapter 4. The remainder of this chapter focuses on the off-line problem.

The input to the off-line LRSCP is a set of fuel pumps $P=\{p_1, \dots, p_q\}$, a set of trains $T=\{t_1, ..., t_n\}$, and a fixed number of tracks. The p_l fuel pump has an associated fixed location $d_l \in \mathbb{R}$ and functional radius $r_l \in \mathbb{R}_+$ for all $l \in \{1, \dots, q\}$. Furthermore, each pump can only reach a subset of tracks. Each platform services 2 tracks, one on each side. Due to similarity among the platforms, the solution of one platform is applied to all platforms.

The pumping platforms service n different types of trains. These trains may differ by number of locomotives and also the locomotive model type. Each train, t_i, has m_i front locomotives that should be filled at the platform for all $i \in \{1, \ldots, n\}$. Each locomotive of train t_i has an f_{ik}, which represents the distance from the front of the first locomotive to the fuel port on the k^{th} locomotive for all $i \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, m_i\}$. Furthermore, each locomotive has a g_{ik}, which represents the amount of needed fuel by the k^{th} locomotive for all $i \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, m_i\}$.

The output of LCRSP is a strike line for each track where the strike lines are in $\mathbb{R}$. The strike line is where the front of the train should stop. Another primary output is an assignment of locomotives to pumps. The goal of LCRSP is to assign the strike lines for all the tracks in such a way that locomotives can be fueled without delay. That is, every locomotive is able to be assigned a pump and begin fueling immediately.

The strike lines given by the integer program are then inputs into simulation models which depict the railyard in its entirety with real data gathered from a major railway company. The simulation produces two metrics: the average time each train spends in the yard and the average maximum number of trains waiting to enter the yard. These metrics are compared for multiple strike line scenarios to gain insight about the impact of refueling system configuration on overall yard operations.

Solving LRSCP helps improve efficiency in refueling yards through new policies or technologies. This is done by optimally aligning trains using solutions from the integer program so that the pumps can reach the locomotives in the most efficient manner. The integer program, including its objective function, decision variables, and constraints, is explained in full detail in the following section.

3.2 Off-Line IP Formulation

The train refueling integer program (TRIP) seeks to obtain a strike line for each track such that the maximum weighted number of train combinations can be refueled on the platform without delay. For a given train combination, refueling without delay occurs if every front-end locomotive on each train can be assigned a fuel pump, where each fuel pump can fuel only one locomotive.

This problem is easily generalized to an arbitrary number of tracks or fuel ports. In fact, most locomotives have a front and back fuel port, but only one port will be converted to the new technology. Thus, TRIP assumes only one fuel port per locomotive.

For both brevity and real-world applicability, the number of tracks is limited to two fuel platforms located between them and each platform has two tracks. Thus, there are $n^2 + 2n$ potential train combinations that could simultaneously be at each platform. The train combinations are n different trains on track 1 and n trains on track 2 (n^2) along with n different trains on track 1 with track 2 empty and the n trains on track 2 with track 1 empty.

Define α_r to be the probability that the refueling yard is in the r^{th} train combination for all $r \in \{1,\ldots, (n^2+2n)\}$. Let $E_r = \{1,\ldots, r'\}$ represent the r' locomotives on the r^{th} train combination for all $r \in \{1, \ldots, (n^2+2n)\}$. Obviously, E_r is partitioned into E_r^1 and E_r^2, which represents the locomotives on track one and two, respectively. Thus, if the r^{th} train combination has t_i and t_j on tracks one and two, respectively, then $r' = m_i+m_j$ and $E_r^1 =\{1, \ldots, m_i\}$ and $E_r^2 =\{m_i+1, \ldots, m_i +m_j\}$.

The decision variables for TRIP are:

$$w_r = \begin{cases} 1 & \text{if train combination } r \text{ is filled without delay} \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } r \in \{1,...,n^2+2n\}$$

$$x_{rkl} = \begin{cases} 1 & \text{if locomotive } k \text{ of } E_r \text{ is filled by pump } l \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } r \in \{1,...,n^2+2n\},\ k \epsilon E_r \text{ and } l \epsilon P$$

$$y_{rkl} = \begin{cases} 1 & \text{if locomotive } k \text{ of } E_r \text{ could be filled by pump } l \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } r \in \{1,...,n^2+2n\},\ k \epsilon E_r \text{ and } l \epsilon P$$

$s_1, s_2 \in \mathbb{R}$, the strike lines for tracks 1 and 2, respectively

TRIP is formally defined as follows.

$$\text{Maximize } \sum_{r=1}^{n^2+2n} \alpha_r w_r$$

Subject to:

$$-s_1 - f_{ik} + d_l - r_l \le M(1-y_{rkl}) \text{ for all } r \in \{1,...,n^2+2n\},\ k \in E_r^1,\ l \in P \qquad (1)$$

$$s_1 + f_{ik} - d_l - r_l \le M(1-y_{rkl}) \text{ for all } r \in \{1,...,n^2+2n\},\ k \in E_r^1,\ l \in P \qquad (2)$$

$$-s_2 - f_{jk-m_i} + d_l - r_l \le M(1-y_{rkl}) \text{ for all } r \in \{1,...,n^2+2n\},\ k \in E_r^2,\ l \in P \qquad (3)$$

$$s_2 + f_{jk-m_i} - d_l - r_l \le M(1-y_{rkl}) \text{ for all } r \in \{1,...,n^2+2n\},\ k \in E_r^2,\ l \in P \qquad (4)$$

$$x_{rkl} \le y_{rkl} \text{ for all } r \in \{1,...,n^2+2n\},\ k \in E_r,\ l \in P \qquad (5)$$

$$\sum_{k \in E_r} x_{rkl} \le 1 \text{ for all } r \in \{1,...,n^2+2n\},\ l \in P \qquad (6)$$

$$w_r \le \sum_{l \in P} x_{rkl} \text{ for all } r \in \{1,...,n^2+2n\},\ k \in E_r \qquad (7)$$

$$x_{rkl} \in \{0,1\} \text{ for all } r \in \{1,...,n^2+2n\},\ k \in E_r,\ l \in P$$
$$y_{rkl} \in \{0,1\} \text{ for all } r \in \{1,...,n^2+2n\},\ k \in E_r,\ l \in P$$
$$w_r \in \{0,1\} \text{ for all } r \in \{1,...,n^2+2n\}$$

The objective function maximizes the weighted number of train combinations that can be filled without delay. In the off-line simulation, waste time is simply measured as a yes or no variable. Either a combination of trains may be fueled without waste time (feasible) or with waste time (infeasible). The objective function of the off-line integer program maximizes the weighted number of train combinations that can be refueled without delay by minimizing the number of infeasible trains.

The first constraints force y_{rkl} to be zero when $-s_1-f_{ik}+d_l-r_l > 0$. In other words, if $d_l-r_l > s_1+f_{ik}$, then the fuel port on the kth locomotive of train combination r is too far away (on the negative side) from pump l to be fueled by that pump. Observe that $k \epsilon E_r^1$, which forces this constraint to be implemented only on the locomotives on track 1. Similarly, the second set of constraints requires $y_{rkl} = 0$ whenever $s_1+f_{ik}>d_l+r_l$. Thus, pump l cannot reach on the positive side to fuel the kth locomotive on train combination r. These constraints incorporate the physical dimensions of the problem. They guarantee that the optimal solution follows the true dimensions of both the platforms, pumps, and locomotive models. Extending this logic to model the strike line for track 2 is straightforward and shown in constraints (3) and (4). The only change is to restrict the fuel port locations to be on the jth train and the $k - m_i$th locomotive where $k \epsilon E_r^2$.

The fifth set of constraints only allows a locomotive to be fueled at pump k if the pump could reach (y_{rkl} can be set to 1). This constraint is a simple but necessary constraint which requires that the pump must be able to reach in order for the model to assign that pump to that locomotive.

Each pump can only fuel one locomotive per train combination as shown in (6). This constraint forces each pump to only fuel a train on either side of the track. This constraint prevents a pump from being used by both trains on a given platform.

The final constraint allows w_r=1 only if every locomotive from the r^{th} train combination is fueled by some pump. This constraint proves that all locomotives on a train must be filled in order for the train to be considered feasible. A train is not considered feasible if three out of the four locomotives were filled without delay.

Observe that the strike lines are unrestricted variables and a negative strike line implies that the front of the train stops beyond the beginning of the platform. Additionally, all other variables are binary variables which can only be 0 or 1 where 1 means it is either feasible or assigned to a given pump. Using these constraints and objective function, the model was written into a text file by Python and solved by CPLEX.

TRIP can be large. Currently, the vast majority of trains passing through the yard in this study have at most five front-end locomotives. If one restricts the problem to a single style of locomotive model and seven pumps, then there are only five train types. In this case, TRIP has over 2,500 variables and 4,000 constraints and is solved in less than one second using CPLEX 12.6.2 on a desktop computer. If one allows two different locomotive models, the number of train types expands to 64 and TRIP has over 450,000 variables and over 750,000 constraints. Unfortunately, this instance of TRIP did not solve in two days. Solving large TRIP instances requires additional research, such as implementing cutting planes or advanced branching strategies, which is left as future work. The optimal strike lines from TRIP are input into the simulation model, which is explained in detail in the following section.

3.3 Off-Line Simulation Model

The optimal strike lines identified by TRIP serve as input to a simulation of the refueling yard processes. The simulation processes include arrival traverse time, pre-fueling crew time, time to setup and pump fuel at the platforms and with DTLs, inspection time, post-fueling delays for crew, track clearance, and departure traverse time. The simulation model's purpose is to measure the impact of the strike lines and fueling technology choices on two railyard performance measures: the average time in yard for a train and the maximum number of trains waiting to be assigned a platform during the day. The latter measure captures the brief part of the day when the system is busiest. This is reported as the average maximum queue length. These two measures most reflect the impact that any change would have on the capacity and efficiency of the real-world system. The following two sections describe the model logic for the off-line simulation, which is built in Simio®.

The model is demonstrated using data derived from a Class I railroad, illustrating the capability to use this approach to support decision making about yard operations. The simulation model depicts a refueling yard with eight different tracks, four facing east and four facing west. Inbound trains come from both directions and require refueling at one of four platforms in the yard. Each refueling platform has seven pumps. A train continues along the inbound track until it reaches the refueling platform, where the train waits until all of its locomotives have been fully refueled. After refueling is complete, the train then waits until crew time and inspection processes are completed, after which it leaves the yard. Figure 5 depicts one platform in the simulation model.

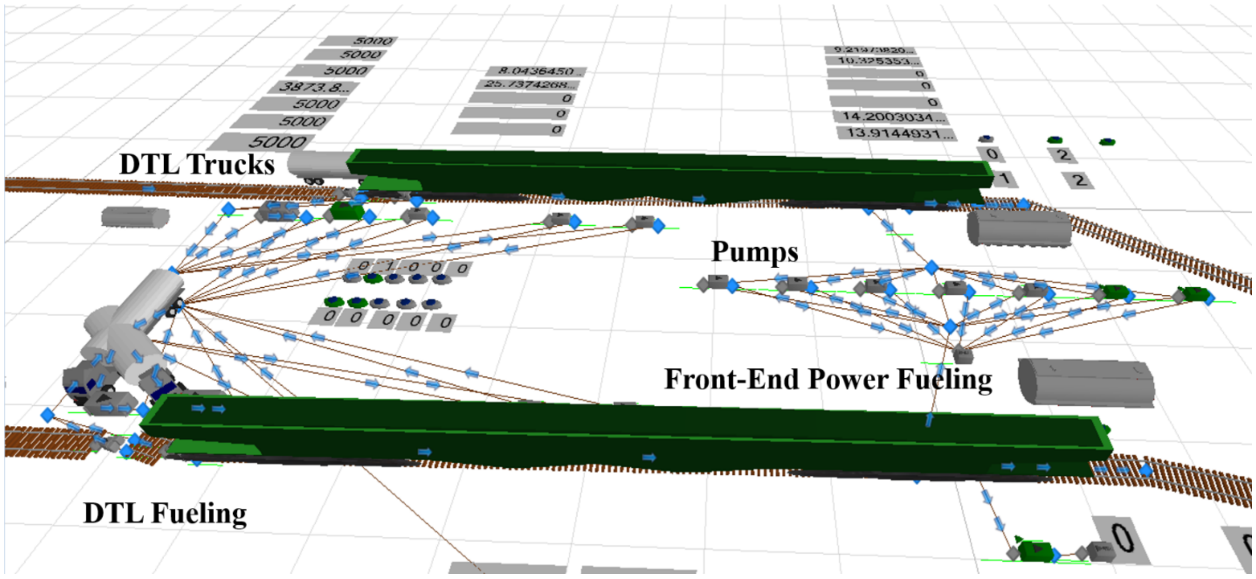

Figure 5: Simulation model visualization of one refueling platform.

3.3.1 DTLs

A key part of this system is the DTL (direct-to-locomotive) refueling. When the train enters the railyard, only the front locomotives are fueled by the platforms. However, some trains also have locomotives in the rear or middle of the train. These locomotives are fueled by DTLs at the same time the front locomotives are being filled by the pumps.

This process is modeled with seven different trucks that travel to the rear of each train while it waits at the platform. A truck refuels one rear locomotive and then returns to its base to refuel its tank before moving on to dispense fuel to another rear locomotive on the same or another train. Seven trucks serve all eight tracks in the railyard, both eastbound and westbound. Additionally, 76% of trains have zero locomotives on the rear of the train, but 18% have two locomotives on the rear. Intuitively, DTL fueling could cause waste time in the system due to a lack of trucks for all rear locomotives in the railyard at a given time. Additionally, these trucks have to refuel after fueling a locomotive and this causes additional time when the pumps are fueling and the DTLs are not.

The modeling of the DTLs proved to be challenging. DTLs created multiple hard-coded processes in order to mediate the problem. The challenge of having a server which requires a travel time as a function of distance from whichever track it was on to the refueling point to another track required them to be vehicles in the model. However, vehicles simply transport objects via Simio®'s logic. Therefore, other processes had to be created and manipulated in order for the DTLs to also refuel a rear locomotive. Additionally, the fuel level in each truck at any given time had to be accounted for because a truck could not fuel a train with more fuel than it possessed in its tank. Accurately modeling DTLs required over 50 different properties and 120 processes.

After analyzing this issue it became obvious that the DTLs do not significantly impact the time spent in the yard by each train. Even if DTLs did, purchasing a few additional DTL units is cheap compared to adding additional tracks and a platform. Therefore, this research focuses primarily on the refueling platforms.

3.3.2 Platform Modeling

Modeling the platform is not straightforward. A primary reason is that the system does not immediately seize a pump resource or join a queue. Instead, it must determine which pump resource to seize upon arrival to the system, delay additional time due to the desired pump being used to seize a resource, and have resources seized by two independent sets of objects. These features and the required model logic provide some insight into the interesting and challenging pieces of this problem.

As mentioned previously, each platform consists of seven pumps, which are created as single servers so that each pump can fuel at most one locomotive at a time. Unlike most simulations, no locomotives are waiting on a pump to finish. Rather, the next train does not begin approaching the platform until the track is free. Once a train is at the platform, the track becomes free after all of the train's locomotives are filled with fuel, the inspection has taken place, and the train has had sufficient time to start and move forward. Thus, trains are queued off of the track and the fuel dispensing pumps never have a queue, unless trains are infeasible.

When a train reaches the platform, an entity is created for each of the front locomotives and each "locomotive" travels to a node where the train and locomotive dimensions are set. This entity calculates f_{ik} for each locomotive. Based upon this distance and the track's strike line, the simulation assigns each locomotive to a pump. This assignment is fairly complex and incorporates 28 different decisions as well as various other assignments through Simio$^{®}$'s add-on processes. This assignment begins with the first locomotive and sequentially assigns locomotives to the first available pump. A pump is available if it is not filling a locomotive and the pump can reach the locomotive's fuel port. This logic follows identically for the second track with the obvious adjustments for the second strike line.

If a locomotive cannot immediately be filled by any of the pumps at its platform, the locomotive is considered infeasible. This scenario calls a process that increases the delay at the next location (the inspection and crew time) to account for 10 minutes plus refueling time. This time accounts for either waiting for the pump to become available or moving the train to a location where the infeasible locomotive can be fueled by an open pump. Then, the process delays the additional time period for actually refueling that locomotive. In other words, if a locomotive is considered infeasible, the time the train spends at the platform refueling nearly doubles. This highlights the importance of optimizing the strike lines to refuel the maximum number of trains and decrease the number of infeasible locomotives.

The simulation model is fairly large. The model has over 1,300 Simio® blocks, which are spread across 234 separate processes. Nevertheless, an individual replication for a 24-hour period required less than 10 seconds. Thus, the model is computationally tractable and experiments with multiple replications can be performed easily.

3.4 Execution and Results of the Off-Line Simulation

The simulation model is demonstrated on a single railyard using data derived from industry sources and literature. The optimal strike lines generated by TRIP are compared against intuitive policies to determine the impact on refueling yard performance measures.

3.4.1 Parameters

The model was built under the following assumptions. Incoming trains are assumed to be equally distributed between east- and west-bound arrivals. All locomotives are refueled to capacity starting from a current fuel volume, which is generated randomly for each locomotive based on a probability distribution derived from historical data. One assumption is that setup times for upgraded and conventional fueling systems are the same, and that adapters can be fitted to upgraded nozzles, if needed, to deliver fuel to conventional receivers. Each simulation replication is one day, and each scenario has 100 replications.

The simulation model is tested in multiple scenarios by changing variables such as refueling speeds, train weight, and the number of arrivals in a day, which can be seen in Table 3. Additionally, three sets of strike lines are examined for this model: 0 and 0, 0 and 150, and 10 and 66. The first two pairs represent intuitive policies. The first policy, denoted 0 and 0, corresponds to each train stopping at the beginning of the platform. This policy also represents a purely static policy since both tracks have the same stationary strike line. The second set of strike lines are 0 and 150, which implies that trains on the first track stop at the first pump and those on the second track stop at the fourth pump. This policy is considered a static policy for each track; while the strike line does not change, the strike line is different for each track. Under this configuration, it is unlikely that a train with fewer than four locomotives is infeasible. The third set of strike lines used are 10 and 66, which are the optimal strike lines from TRIP. This policy is also a static strike line for each track. The goal of optimizing these strike lines is to decrease the number of infeasible combinations of trains in LRSCP and improve the efficiency of the system.

Table 3: Parameter values.

Parameter	Value
Fueling Rates	**gallons per minute**
Platform, Conventional System	220
Platform, Upgraded System	450
DTL, Conventional System	170
DTL, Upgraded System	450
DTL Refueling	280
Train Composition	**number of locomotives (fraction of trains)**
Front	1 (<1%), 2 (32%), 3 (53%), 4 (14%), 5 (<1%)
Middle	0 (94%), 1 (1%), 2 (5%)
Rear	0 (76%), 1 (4%), 2 (18%), 3 (2%)
Other Processes	
Inspection time (minutes)	40
Fraction of trains inspected	80%
Post-fueling delay (minutes)	24.25

Prior to conducting experiments with the strike lines and refueling technologies, the simulation model was validated. Using parameter values for the current system, the simulation results indicate that the average refueling time for a train is 68.87 minutes. The corresponding value from historical data is 70.05 minutes. The simulation suggests that the refueling yard begins to experience delays when there are 95 – 105 trains arriving per day. Personnel familiar with the operation confirmed that it is necessary to adjust arrival rates when more than 100 trains are scheduled to arrive in a day. The distribution is uniform(*x*-*y*, *x*+*y*) where *x* is the designated number of arrivals during a day in minutes and y is 50% of x. Consequently, this simulation model accurately represents the existing system.

To determine the strike lines to test within the simulation model, TRIP is solved with each train combination equally likely: $a_r = 1/(n^2+2n)$ for all $r \in \{1, \ldots, (n^2+2n)\}$. The optimal strike lines identified by TRIP are 10 and 66, which is to stagger the strike lines of the two tracks. However, the staggering is not the anticipated half staggering as a locomotive is about 72 feet long. Rather, the answer is to move the strike line on the second track so the front locomotives overlap by only a quarter. Furthermore, the starting point is adjusted slightly backwards and the second track can never have a locomotive assigned to the first pump.

In TRIP, there were 35 combinations of trains with n being equal to 5 and there being a single locomotive type. Of those combinations, 25 were found to be feasible by TRIP. Of the 10 combinations which were infeasible, eight of them had more than seven locomotives on a platform which would always be infeasible regardless of strike line.

With these parameters, three daily arrival rates: 100, 125, and 166 trains per day, are tested. The first is the current average arrival rate. The second is approximately the current capacity, and the third represents anticipated growth in freight volume.

3.4.2 Off-Line LRSCP Results

The simulation results for different strike lines in different scenarios can be seen in Tables 4-7. The first table shows the system at 100 trains per day and using the refueling system that is currently operational. As can be seen in Table 4, the average train time in the yard decreases from 68.87 to 68.16 minutes, or about 1.0%. A standard two-sample t-test for the 100 replications is used to compare the average train time in the yard for the optimized strike lines to results for intuitive strike lines. The p-value of each test is <0.0001, which shows that for any reasonably desired confidence level, the optimized strike lines decrease the average train time in the yard. It may seem surprising that a small difference yields such a small *p*-value, but this is due to the small variance between replications. Even though there is a statistically significant difference, the practical benefit is marginal at an arrival rate of 100 trains per day. However, it is noted that the average number of infeasible trains throughout the 100 replications is 0.19 and 0.46 lower, respectively, for the optimized strike lines versus the other two strike line configurations.

Table 4: Impact of strike line decisions on refueling yard performance measures using current refueling system at arrival rate of 100 trains per day (off-line simulation)

Strike Lines (Track 1, Track 2)	**Average Maximum Queue (trains)**	**Average Train Time in Yard (min)**	**Average Number of Infeasible Trains**
0, 0	0.09	68.87	0.22
0, 150	0.22	69.99	0.49
10, 66 (TRIP optimal solution)	0.04	68.16	0.03

Table 5 shows the system at 125 trains per day and using the refueling system that is currently operational. In this case, the average train time in the yard decreases from 70.38 to 68.49 minutes, or about 2.7%. A standard two-sample *t*-test for the 100 replications is used to compare the average train time in the yard for the optimized strike lines to results for intuitive strike lines. The *p*-value of each test is <0.0001, which shows that for any reasonably desired confidence level, the optimized strike lines decrease the average train time in the yard. Additionally, the average number of infeasible trains is 0.63 and 0.72 lower, respectively, for the optimized strike lines versus the other two strike line configurations.

Table 5: Impact of strike line decisions on refueling yard performance measures using current refueling system at arrival rate of 125 trains per day (off-line simulation)

Strike Lines (Track 1, Track 2)	Average Maximum Queue (trains)	Average Train Time in Yard (min)	Average Number of Infeasible Trains
0, 0	1.00	70.38	0.89
0, 150	1.11	71.10	0.98
10, 66 (TRIP optimal solution)	0.72	68.49	0.26

Table 6 demonstrates the impact of strike line choice when train arrival rates are elevated (166 trains per day) under the current refueling system. The average train time in the yard decreases from 79.21 to 73.24 minutes, or about 7.5%, with optimal strike lines. A standard two-sample *t*-test for the 100 replications is used to compare the average train time in the yard for the optimized strike lines versus intuitive strike lines. The *p*-value of each test is <0.0001, which shows that for any reasonably desired confidence level, the optimized strike lines decrease the average train time in the yard. The average number of infeasible trains is also 2.48 and 2.14 lower, respectively, for the optimal strike lines when compared to the intuitive values.

Table 6: Impact of strike line decisions on refueling yard performance measures using current refueling system at arrival rate of 166 trains per day (off-line simulation)

Strike Lines (Track 1, Track 2)	Average Maximum Queue (trains)	Average Train Time in Yard (min)	Average Number of Infeasible Trains
0, 0	5.66	79.21	2.98
0, 150	4.81	77.43	2.64
10, 66 (TRIP optimal solution)	3.36	73.24	0.50

Finally, Table 7 summarizes the combined impact of converting the pumping technology to the faster nozzles and optimizing strike lines when the arrival rate is 166 trains per day. The optimized strike lines provide measurable improvement compared to intuitive strike lines even when new refueling technology is adopted. Here, the average train time in the yard decreases from 76.05 to 67.89 minutes, or about 10.7%. A standard two-sample *t*-test for the 100 replications is used to compare the average train time in the yard for the optimized strike lines to results for intuitive strike lines. The *p*-value of each test is <0.0001, which shows that for any reasonably desired confidence level, the optimized strike lines decrease the average train time in the yard. In this case, the average number of infeasible trains is 3.09 and 2.31 lower, respectively, for the optimized strike lines versus the other two strike line configurations.

Table 7: Impact of strike line decisions on refueling yard performance measures using upgraded refueling system at arrival rate of 166 trains per day (off-line simulation)

Strike Lines (Track 1, Track 2)	**Average Maximum Queue (trains)**	**Average Train Time in Yard (min)**	**Average Number of Infeasible Trains**
0, 0	5.12	76.05	4.24
0, 150	4.42	73.98	3.46
10, 66 (TRIP optimal solution)	2.64	67.89	1.15

Based on the experimental results, solving TRIP and incorporating the optimal strike lines will improve the refueling yard operations regardless of whether or not a new refueling technology is adopted. This impact is statistically significant and allows more trains to be processed each day.

Additionally, using a new refueling technology should enable railroad companies to dramatically escalate the number of trains through a refueling yard, which increases system capacity without the enormous expense of laying new track. Therefore, this technology should be researched and engineered to have the desired effect on the railyard's capacity and efficiency.

Furthermore, using optimal strike lines leads to significant decrease in the amount of down time by the pumps and increases the efficiency of the platforms. This would lead to time savings, an additional ability to handle variability, and significant potential revenue opportunities with increased railyard capacity. Having more efficient platforms and the ability to re-solve for optimal strike lines allows change if maintenance was needed on a pump without greatly damaging the efficiency of the platform as a whole.

These static strike line policies still have a large number of infeasible trains when the capacity is increased to the expected future demand. Therefore, an additional method, the on-line simulation and the dynamic strike line policy, is pursued to limit the number of infeasible trains. This method is the topic of the next chapter.

Chapter 4 - On-Line Simulation and Optimization for LRSCP

The purpose of this chapter is to describe the methods used to analyze and solve the on-line locomotive refueling system configuration problem. The first section covers the formulation of the IP that finds the optimal dynamic solutions. The second section describes the discrete event simulation, which models and tests the system. Finally, the results of the on-line simulation and optimization are discussed and analyzed.

4.1 On-Line IP Formulation

The on-line LRSCP differs substantially from the off-line version. Rather than knowing all the train combinations that could exist and optimizing a single static policy, the on-line version of LRSCP changes the strike line for each train. In this case, LRSCP considers trains on the horizon that have not yet arrived at the platform. For example, in the on-line LRSCP, there could be any number of trains in a single train combination. This changes both the input and the output for the integer program. The objective function and constraints are also different.

The input for the platforms is identical to the off-line simulation. The integer program has a set of fuel pumps $P=\{p_1, \ldots, p_q\}$ a fixed number of tracks, $L=\{l_1,\ldots,l_b\}$. Each fuel pump has an associated fixed location $d \in \mathbb{R}$ and functional radius $r \in \mathbb{R}_+$ for all $p \in P$. Furthermore, each pump can only reach a subset of tracks. Define L^k to be all the tracks that are serviced by pump k for all $k \in \{1,\ldots,q\}$. Considered in this thesis, there are 14 pumps and two platforms. Pumps 1-7 serve tracks 1 and 2 and pumps 8-14 serve tracks 3 and 4. Therefore, $L^1 = L^2 = \ldots = L^7 =$ {track 1, track 2} and $L^8 = L^9 = \ldots = L^{14} =$ {track 3, track 4}. Similarly, τ_{lh} to be 0 if track l is occupied at time h and 1 if track l is open at time h for all $l \in L$ and $h \in H$.

For a given instance of the on-line LRSCP, only a finite time horizon is considered and the incoming trains are assumed to be known. Trains $T=\{t_1, \ldots, t_n\}$ considered in the model are only those that could arrive at a platform prior to the end of the time horizon. This set of trains changes as trains appear on the time horizon or arrive at a platform. Each train, t_i, has m_i front locomotives that should be filled at the platform for all $i \in \{1, \ldots, n\}$. Each locomotive of train t_i has an f_{ij}, which represents the distance from the front of the first locomotive to the fuel port on the j^{th} locomotive for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m_i\}$. Furthermore, each locomotive has a g_{ij}, which represents the amount of needed fuel by the j^{th} locomotive for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m_i\}$. Finally, each train has an arrival time $a_i \in \mathbb{R}$, the earliest time that a train could arrive at a platform.

Let H be the time horizon. This time horizon is calculated as the earliest arrival time of all trains in T until the latest arrival of all trains in T plus some constant time. Here this constant time is 30-40 minutes, which should provide sufficient time to fuel the last train. In order to integrate the integer program with the discrete-event simulation, time had to be discretized. Therefore, time h in the integer program is a single minute, from the first minute to the last minute of the time horizon, H.

The output of LCRSP is an assignment of each train to a track and an accompanying strike line, where the train stops on this track. Additionally, the output includes which locomotive on each train is being fueled by which pump and at what times.

The goal of LCRSP is to assign the strike lines and pumps for all the trains in such a way that trains can be fueled without waste time. Waste time for each train is calculated as the train's finishing time at the platform minus the earliest time the train could have arrived at the platform minus the longest fueling time for any locomotive on the train. Observe that a train's earliest arrival time and a locomotive's refueling time are both constant. Thus, minimizing total waste of all trains time is equivalent to minimizing total completion time.

Consequently, LRSCP can be viewed as a sum of total completion time scheduling problem with release times (arrival times at the platform) and multiple machines (tracks and pumps). The pump assignment creates added complexity as some of the multiple machines must share resources. Thus, shortest processing time first, the algorithm which solves basic total completion time scheduling problems, cannot optimally solve this problem. Hence, an integer program is created to solve this problem.

The strike line integer program (SLIP) seeks to obtain a strike line for each train such that the total waste time is minimized. Waste time is greater than zero if any train experiences any delays due to track or pump availability. This problem is easily generalized to an arbitrary number of tracks or fuel ports. In fact, most locomotives have a front and back fuel port, but only one port will be converted to the new technology. Thus, SLIP assumes only one fuel port per locomotive.

The problem's on-line nature allows for trains to be occupying tracks and pumps at the start of the time horizon. To develop SLIP, define the parameters q_{ikh} to be 0 if h is less than the arrival time of the train or if the kth pump is being used by an existing train at the platform at time h. If not, q_{ikh} is 1 and represents that a locomotive on train i may be fueled by pump k.

The decision variables for SLIP are:

$$x_{ijkh} = \begin{cases} 1 & \text{if train } i, \text{ locomotive } j \text{ is fueled by pump } k \text{ at time } h \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in T,\ j \in M_i,\ k \in P, \text{ and } h \in H$$

$$y_{ijkh} = \begin{cases} 1 & \text{if train } i, \text{ locomotive } j \text{ can be reached by pump } k \text{ at time } h \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in T,\ j \in M_i,\ k \in P, \text{ and } h \in H$$

$$z_{ijkh} = \begin{cases} 1 & \text{if pump } k \text{ starts fueling locomotive } j \text{ on train } i \text{ at time } h \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in T,\ j \in M_i,\ k \in P, \text{ and } h \in H$$

$$v_{ilh} = \begin{cases} 1 & \text{if train } i \text{ is on track } l \text{ at time } h \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in T,\ l \in L,\ h \in H$$

$$w_{il} = \begin{cases} 1 & \text{if train } i \text{ is on track } l \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in T,\ l \in L$$

$u_i =$ the time at which train i finishes fueling for all $i \in T$

$s_i \in \mathbb{R}$ for all $i \in T$

SLIP is formally defined as follows.

Minimize $\sum_{i \in T} u_i$

Subject to:

$$u_i - h\, x_{ijkh} \geq 0 \quad \text{for all } i \in T,\ j \in M_i,\ k \in P, \text{ and } h \in H \tag{1}$$

$$\sum_{j \in M_i} x_{ijkh} \leq q_{ikh} \quad \text{for all } i \in T,\ k \in P, \text{ and } h \in H \tag{2}$$

$$\sum_{k \in P} \sum_{h \in H} x_{ijkh} = g_{ij} \quad \text{for all } i \in T \text{ and } j \in M_i \tag{3}$$

$$\sum_{k \in P} \sum_{h \in H} z_{ijkh} = 1 \quad \text{for all } i \in T \text{ and } j \in M_i \tag{4}$$

$$x_{ijkh} - x_{ijk(h-1)} - z_{ijkh} \leq 0 \quad \text{for all } i \in T,\ j \in M_i,\ k \in P,\ h \geq 2, \text{ and } h \in H \tag{5}$$

$$-s_i - f_{ij} + d_k - r_k \leq M(1 - y_{ijk}) \quad \text{for all } i \in T,\ j \in M_i, \text{ and } k \in P \tag{6}$$

$$s_i + f_{ij} + d_k - r_k \leq M(1 - y_{ijk}) \quad \text{for all } i \in T,\ j \in M_i, \text{ and } k \in P \tag{7}$$

$$-My_{ijk} + \sum_{h \in H} x_{ijkh} \leq 0 \quad \text{for all } i \in T,\ j \in M_i, \text{ and } k \in P \tag{8}$$

$$\sum_{l \in L} v_{ilh} \leq 1 \quad \text{for all } i \in T \text{ and } h \in H \tag{9}$$

$$\sum_{i \in T} v_{ilh} \leq 1 \quad \text{for all } l \in L \text{ and } h \in H \tag{10}$$

$$\sum_{l \in L} w_{il} \leq 1 \quad \text{for all } i \in T \tag{11}$$

$$v_{ilh} - w_{il} \leq 0 \quad \text{for all } i \in T, l \in L, \text{ and } h \in H \tag{12}$$

$$\sum_{i \in T} v_{ilh} \leq \tau_{lh} \quad \text{for all } l \in L, \text{ and } h \in H \tag{13}$$

$$x_{ijkh} - \sum_{c \in L^k} v_{ich} \leq 0 \quad \text{for all } i \in T,\ j \in M_i, k \in P, \text{ and } h \in H \tag{14}$$

$$x_{ijkh} \in \{0,1\} \text{ for all } i \in T,\ j \in M_i, k \in P, \text{ and } h \in H$$
$$y_{ijkh} \in \{0,1\} \text{ for all } i \in T,\ j \in M_i, k \in P, \text{ and } h \in H$$
$$z_{ijkh} \in \{0,1\} \text{ for all } i \in T,\ j \in M_i, k \in P, \text{ and } h \in H$$
$$v_{ilh} \in \{0,1\} \text{ for all } i \in T,\ j \in M_i, k \in P, \text{ and } h \in H$$
$$w_{il} \in \{0,1\} \text{ for all } i \in T \text{ and } l \in L$$
$$u_i \text{ is integer for all } i \in T$$
$$s_i \text{ is integer for all } i \in T$$

The objective function minimizes the finishing time, u_i, of each train in an effort to eliminate waste time. Waste time is calculated as the difference between when a train leaves the platform minus the earliest time that the train could have arrived at the platform minus the train's longest locomotive refueling time. This value is calculated as input by the simulation not the IP, but is optimized in the IP by minimizing the completion time.

Constraints (1) – (8) deal with assigning locomotives to pumps. The first constraint assigns the variable u_i to be equal to the last time in which one of its locomotives is assigned to a pump. This is the time when the train is completely finished fueling and therefore waits for crew time or inspection to finish before leaving the platform. This sets the values used by the objective function.

The second constraint forces a pump to be assigned to only a single locomotive at a single time. The pump can obviously be unused and therefore, the left hand side may be zero, but it cannot be greater than 1. This constraint is necessary because trains, which are currently being fueled by the platform, are not a part of the IP, but are still using pumps and tracks at a given time *t.* Additionally, it prevents a locomotive from being assigned to a pump if the train has not yet arrived at the platform, which is why q_{ikh} was created.

The third constraint states that each locomotive must be filled for long enough to completely fuel the locomotive. This is based off an input from the simulation called g_{ij} for each locomotive. The time necessary to fuel that locomotive is then calculated and the sum of the times that locomotive is being fueled must be equal to the time needed to fuel it.

The fourth constraint ensures that every locomotive has a time at which it begins fueling. This variable, z_{ijkh}, is important to prevent locomotives from switching pumps and from starting and stopping. The assumption is that when a train starts fueling, it will continue to fuel until finished without interruption. The fifth constraint enforces this assumption by ensuring that each time that a locomotive is being fueled is consecutive with the time prior. This enforces that a pump cannot stop fueling at any time until it is finished.

The sixth constraint forces y_{ijk} to be zero when $–s_i-f_{ijk}+d_l-r_l > 0$. In other words, if $d_k-r_k > s_i+f_{ijk}$, then the fuel port on the jth locomotive of train i is too far away (on the negative side) from pump k to be fueled by that pump. Similarly, the seventh constraint requires $y_{ijk} = 0$ whenever $s_i+f_{ijl}>d_k+r_k$. Thus, pump k cannot reach on the positive side to fuel the jth locomotive on train *i*. These constraints incorporate the problem's physical dimensions and the optimal strike line. They guarantee that the optimal solution follows the true dimensions of the platforms, pumps, and locomotive models. The eighth constraint forces the accessible variable to be less than or equal to the assign variable. In other words, a pump cannot be assigned to a locomotive if it cannot reach that locomotive.

Constraints (9) - (14) focus on train assignment to tracks. The ninth constraint ensures that any train can be assigned to at most one track at any time *t*. The tenth constraint requires that at most a single train can be assigned to any track at a given time *t*. This prevents two trains from ever occupying the same track. The eleventh constraint forces a train to be assigned to one and only one track. This prevents trains from being on track 1 and switching to track 2, while still using the same pumps.

The twelfth constraint prevents a train from being assigned to a track at time *t* unless it is assigned to that track through the entire time horizon. This states that trains cannot switch tracks during the time horizon represented by the IP. The thirteenth constraint ensures that a train cannot be assigned to a track if there is a train already on that track and stopped at the platform. This is similar to constraint two, where a train already at the platform is no longer a part of the IP being solved for the current time horizon, but still occupies space and time in this horizon.

The fourteenth constraint ensure that if a train is using pumps 1-7, then it must be on track 1 or 2 and vice versa. Similarly any train on tracks 3 or 4 can only use pumps 8-14 and any train using pumps 8-14 must be on tracks 3 or 4. This is done to model the platform and track alignment of the real system.

Observe that the strike lines and finish times are unrestricted variables and a negative strike line implies that the front of the train stops beyond the beginning of the platform.

Additionally, all other variables are binary variables. Using these constraints and objective function, the model was written into a text file by Python and solved by CPLEX. The optimal variables were returned back into Python and the simulation.

SLIP can be large and is dependent upon the time horizon and number of trains in the system. The time horizon plays a major factor in the solving time of SLIP because it is broken down into minutes and multiple constraints have for all $h \in H$. The time horizon is set so that the first train arriving at the platform begins it and runs until the final train arrives and has enough time to fuel. As more trains enter the system, more locomotives are considered in the time horizon. One interesting aspect of the time horizon is that it can be set so that only a certain number of trains will be incorporated into SLIP. For example, it would be possible to set the time horizon so that at most the next four trains would be considered in a SLIP instance regardless of how near or far from the platform. Additionally, if the value is set to 1, then SLIP would simply optimize the incoming train regardless of the impact it may have on the trains following it, which would be a greedy solution.

As the time horizon increases, the speed of solving SLIP and of running the simulation increases substantially. With the time horizon being at most four trains, SLIP is written out and solved in around two minutes. This instance of SLIP had over 43,000 constraints and 32,000 variables. Solving large SLIP instances or solving these instances faster requires additional research, such as implementing cutting planes or advanced branching strategies, which is left as future work. The outputs from SLIP are input into the simulation model, which is explained in the following section.

4.2 On-Line Simulation Model

The on-line simulation model is integrated with SLIP to analyze the impact of a dynamic strike line policy. This on-line simulation is built entirely as computer code in Python and is a discrete-event simulation. The on-line simulation integrated with SLIP takes over 1,000 lines of computer code in Python to perform all the necessary processes. It follows three distinct time events: when a train appears in the time horizon, when a train arrives at a platform, and when a train finishes refueling and leaves the system. At the beginning of any of these three events, a SLIP instance is printed into a text file and solved by CPLEX to determine the optimal strike line and pump configuration. These optimal values are inputs into the simulation and then time advances to the next step. At the end of the simulation, the average time that a train spends in the yard and the maximum queue are output to analyze the effect of the dynamic strike line policy.

The model is demonstrated using data derived from a Class I railroad, illustrating the capability to use this approach to support decision making about yard operations. The simulation model depicts a refueling yard with four different tracks facing in the same direction, as opposed to the off-line simulation model which has eight tracks in both directions. Inbound trains require refueling at one of two platforms in the yard. Each refueling platform has seven pumps. A train appears in the system and requires 40 minutes until it arrives at a platform. Once arriving at a platform, the train waits until all of its locomotives have been fully refueled. After refueling is complete, the train waits for a constant time, which is equivalent to crew time and inspection, after which it leaves the yard.

The simulation is run by using a while loop, which runs for the chosen length, 24 hours or one day. The first step of the simulation is to determine which of the three events will happen first, a train arrival, a platform arrival, or a train leaving. This is done by calculating the next time for each of these events. The system starts empty and the first event is always a train appearing on the time horizon. The events and SLIP integration can be seen in Figure 6.

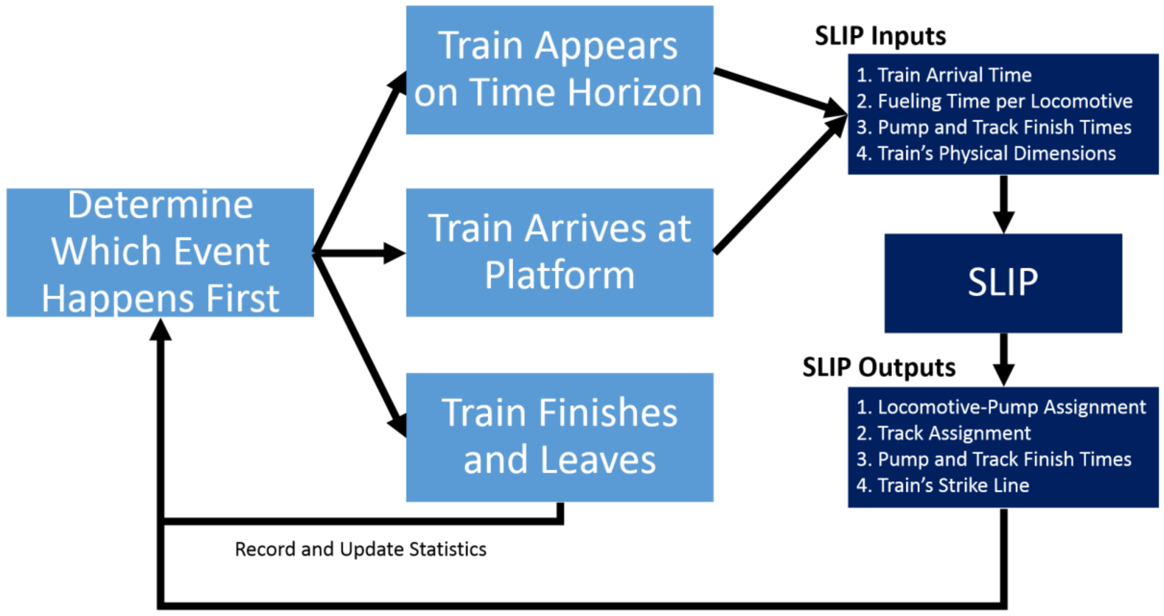
Train Appears on Time Horizon
SLIP Inputs
1. Train Arrival Time
2. Fueling Time per Locomotive
3. Pump and Track Finish Times
4. Train's Physical Dimensions
Determine Which Event Happens First
Train Arrives at Platform
SLIP
SLIP Outputs
1. Locomotive-Pump Assignment
2. Track Assignment
3. Pump and Track Finish Times
4. Train's Strike Line
Train Finishes and Leaves
Record and Update Statistics

Figure 6: On-Line Simulation Integration

Whenever the next event is a train appearing in the time horizon, multiple processes are performed. The first process reads a data file with various locomotive dimensions and then creates random train combinations. For example, train 1 may consist of any locomotive in the file with any number of one to five front locomotives. Therefore, train 1 may have two locomotives of type 2 and one of type 3 on its train. These combinations are set for all trains and chosen to be the next incoming train. The number and type of locomotives on a train is decided from historical distributions taken from real data. Additionally, this process assigns various values to characteristics which will be used throughout the simulation including the train number, the arrival time to the platform, the fuel needed by each locomotive, the fuel port locations, and other physical characteristics.

Next, time advances, then SLIP is written out and solved. Its solution assigns the timing, pump configuration, and strike line for all incoming trains. Each incoming train now has a track decided and an arrival time at the platform. Finally, the next incoming train that will appear on the time horizon is determined. The next incoming train comes into the system using the same distribution as the off-line simulation. Thus, trains appear on the horizon with an interarrival time that is random uniformly distributed integer between α - $\alpha/2$ and $\alpha + \alpha/2$. Here α is the number of minutes that would generate the designated number of arrivals during a day, i.e. α equals 1440 minutes /the number of daily arrivals. This concludes the train arrival event and the simulation loops back to the calculation of the next event.

The second event which could take place is a platform arrival. In this case, the next event that happens is that a train which has been in the system and driving towards the railyard enters the railyard and stops at the platform according to the previous SLIP's solution. The process that runs this event is one which determines specifically when the train arrives at the platform with respect to other trains being on that track. For example, even if a train can arrive at time 60, if the train on that track is not finished until time 65 then the train cannot truly arrive at the platform until time 70, the previous train's completion time plus five minutes for the train to leave the track. This is the identical arrival time as returned by SLIP. More importantly, this process changes the state of the train to platform arrival so that the track location, pump configuration, and timing of the refueling for this train can no longer be changed when running SLIP. This prevents SLIP from re-optimizing a train which is already stopped and fueling at a platform. This is accomplished by assigning a finish time on the pump. Thus, a locomotive finishing refueling is not an event. After this process is completed, SLIP is optimized to account for this train at the platform and outputs the optimal values for the remaining incoming trains. This concludes the platform arrival event and the simulation loops back to the calculation of the next event.

The third event that can occur is when a train finishes refueling and leaves the system. This process calculates the necessary values for simulation output for this train, including the time spent in the station, the strike line used, and the waste time incurred. Next, the train is moved to a state where it is no longer part of the system. Finally, the results such as objective function, strike line, and train time are summarized. This concludes the train leaving event and the simulations loops back to the calculation of the next event.

The simulation runs for a fixed time window and time increments with each event. At its conclusion, the simulations outputs (average time spent by the train in the railyard, maximum queue, average strike lines, and amount of waste time) are calculated, printed and summarized. These results are presented in the following section.

4.3 Execution and Results of the On-Line Simulation

Using the integer program and simulation, the effect of the different strike line policies can be tested and validated. The tool is analyzed for various possible combinations of train arrivals per day and strike line policies. The parameters used in this simulation are the same as those used in Chapter 3 in terms of pump speed, weighted number of locomotives on a given train, inspection time and percentage, and post-fueling delay. Each scenario is presented as averages of ~30 replications and each replication takes approximately 45 minutes when 100 trains arrive per day and over three hours when 166 trains arrive per day.

The integrated simulation also has the ability to input static strike line policies to compare directly with the dynamic strike line policy, due to the fact that the off-line simulation and on-line simulation will have slightly different results. These results are different because the off-line simulation includes separate random processes for inspection and DTL fueling, whereas the on-line simulation focuses only on refueling of the front-end locomotives and uses constant estimates to account for inspection, post-fueling delay, and travel time.

These static policies are used in the simulation by adding an additional constraint to the IP. This constraint checks which track the train is on and sets the strike line accordingly. For example, this constraint follows the IP formulation, $s_1 w_{i1} + s_2 w_{i2} + s_1 w_{i3} + s_2 w_{i4} = s_i$ where s_1 is the static strike line for tracks one and three, s_2 is the static strike line for tracks two and four, and s_i is the strike line for train i for all trains $i \in T$.

The simulation results for different strike lines in different scenarios can be seen in Tables 8-11. The first table shows the system at 100 trains per day and using the refueling system that is currently operational. Table 8, the average train time in the yard increases from 60.52 to 60.72 minutes when comparing a static intuitive policy to the SLIP solution. The strike line policy has limited effect on average time in the railyard due to the simulations' randomness and the fact that wasted time is still minimal with 100 trains per day.

Table 8: Impact of strike line decisions on refueling yard performance measures using current refueling system at arrival rate of 100 trains per day (on-line simulation)

Strike Lines (Track 1, Track 2)	Average Maximum Queue (trains)	Average Train Time in Yard (min)	Average Number of Infeasible Trains	Average Total Number of Minutes Wasted (min)
0, 0	0.07	60.52	0.10	0.41
0, 150	0.45	60.52	0.45	4.21
10, 66 (TRIP optimal solution)	0.10	60.76	0.17	0.38
Dynamic (SLIP optimal solution)	0.00	60.72	0.03	0.07

A standard two-sample *t*-test for the 30 replications is used to compare the average number of minutes wasted for the dynamic optimized strike line policy to results for intuitive strike line policies and TRIP optimal solution. The *p*-value of the 0, 0 strike line policy is 0.252 and thus there is not a statistically significant difference in the means of the average number of minutes wasted. The *p*-value of the 10, 66 strike line policy is 0.145 and thus there is not a statistically significant difference in the means of the average number of minutes wasted. In contrast, the *p*-value of the 0, 150 strike line policy is 0.002, which shows that for any reasonably desired confidence level, the dynamically optimized strike lines decrease the average wasted time. It can also be seen that TRIP performs 7% and SLIP performs 83% better than 0, 0 in average wasted time. It can be seen that at an arrival rate of 100 trains per day, the effect of the strike line is minimal because it is rare that many trains will be in the system simultaneously. It is noted that the average number of infeasible trains throughout the 30 replications is 0.07 and 0.42 lower, respectively, for the dynamic optimized strike lines versus the other two strike line configurations and 0.31 lower as compared to the TRIP optimal solution.

Table 9: Impact of strike line decisions on refueling yard performance measures using current refueling system at arrival rate of 125 trains per day (on-line simulation)

Strike Lines (Track 1, Track 2)	**Average Maximum Queue (trains)**	**Average Train Time in Yard (min)**	**Average Number of Infeasible Trains**	**Average Total Number of Minutes Wasted (min)**
0, 0	0.07	60.83	0.07	0.26
0, 150	0.38	60.72	0.72	5.28
10, 66 (TRIP optimal solution)	0.03	60.76	0.03	0.24
Dynamic (SLIP optimal solution)	0.00	60.59	0.00	0.00

Table 9 shows the system at 125 trains per day and using the refueling system that is currently operational. As can be seen in Table 9, the average train time in the yard decreases from 60.72 to 60.59 minutes when comparing a static intuitive policy to the SLIP solution. The strike line policy has limited effect on average time in the railyard due to the simulations' randomness and the fact that wasted time is still minimal with 125 trains per day.

A standard two-sample *t*-test for the 30 replications is used to compare the average number of minutes wasted for the dynamic optimized strike line policy to results for intuitive strike line policies and TRIP optimal solution. The *p*-value of the 0, 0 strike line policy is 0.246 and thus there is not a statistically significant difference in the means of the average number of minutes wasted. The *p*-value of the 10, 66 strike line policy is 0.326 and thus there is not a statistically significant difference in the means of the average number of minutes wasted. In contrast, the *p*-value of the 0, 150 strike line policy is 0.004, which shows that for any reasonably desired confidence level, the dynamically optimized strike lines decrease the average wasted time. It can also be seen that TRIP performs 8% and SLIP performs 100% better than 0, 0 in average wasted time. It can be seen that at an arrival rate of 125 trains per day, the effect of the strike line is minimal because it is rare that many trains will be in the system simultaneously. It is noted that the average number of infeasible trains throughout the 30 replications is 0.34 and 4.14 lower, respectively, for the dynamic optimized strike lines versus the other two strike line configurations and 0.24 lower compared to the TRIP optimal strike lines.

Table 10 shows the system at 166 trains per day and using the refueling system that is currently operational. As can be seen in Table 10, the average train time in the yard decreases from 61.55 to 61.03 minutes when comparing a static intuitive policy to the SLIP solution. The strike line policy has limited effect on average time in the railyard due to the simulations' randomness.

Table 10: Impact of strike line decisions on refueling yard performance measures using current refueling system at arrival rate of 166 trains per day (on-line simulation)

Strike Lines (Track 1, Track 2)	**Average Maximum Queue (trains)**	**Average Train Time in Yard (min)**	**Average Number of Infeasible Trains**	**Average Total Number of Minutes Wasted (min)**
0, 0	1.07	61.55	8.97	50.93
0, 150	1.14	62.24	10.45	88.83
10, 66 (TRIP optimal solution)	1.10	61.24	6.66	34.07
Dynamic (SLIP optimal solution)	0.90	61.03	4.41	22.55

A standard two-sample *t*-test for the 30 replications is used to compare the average number of minutes wasted for the dynamic optimized strike line policy to results for intuitive strike line policies and TRIP optimal solution. The *p*-value of the 10, 66 strike line policy is 0.192 and thus there is not a statistically significant difference in the means of the average number of minutes wasted. The *p*-value of the 0, 0 strike line policy is 0.005 and thus there is a statistically significant difference in the means of the average number of minutes wasted. Similarly, the *p*-value of the 0, 150 strike line policy is 0.0001, which shows that for any reasonably desired confidence level, the dynamically optimized strike lines decrease the average wasted time. It can also be seen that TRIP performs 33% and SLIP performs 56% better than 0, 0 in average wasted time. It can be seen that at an arrival rate of 166 trains per day, the effect of the strike line is much more significant because it is common that many trains will be in the system simultaneously. It is noted that the average number of infeasible trains throughout the 30 replications is 4.56 and 6.04 lower, respectively, for the dynamic optimized strike lines versus the other two strike line configurations and 2.25 lower compared to the TRIP optimal strike lines.

Table 11 shows the system at 166 trains per day and using the upgraded refueling system. As can be seen in Table 11, the average train time in the yard increases from 57.00 to 57.79 minutes when comparing a static intuitive policy to the SLIP solution. The strike line policy has limited effect on average time in the railyard due to the simulations' randomness.

Table 11: Impact of strike line decisions on refueling yard performance measures using upgraded refueling system at arrival rate of 166 trains per day (on-line simulation)

Strike Lines (Track 1, Track 2)	Average Maximum Queue (trains)	Average Train Time in Yard (min)	Average Number of Infeasible Trains	Average Total Number of Minutes Wasted (min)
0, 0	1.00	57.10	3.59	21.52
0, 150	1.14	58.93	7.34	67.00
10, 66 (TRIP optimal solution)	0.89	57.76	3.49	19.95
Dynamic (SLIP optimal solution)	0.90	57.79	3.38	15.90

A standard two-sample *t*-test for the 30 replications is used to compare the average number of minutes wasted for the dynamic optimized strike line policy to results for intuitive strike line policies and TRIP optimal solution. The *p*-value of the 0, 0 strike line policy is 0.615 and thus there is not a statistically significant difference in the means of the average number of minutes wasted. The *p*-value of the 10, 66 strike line policy is 0.458 and thus there is not a statistically significant difference in the means of the average number of minutes wasted. In contrast, the *p*-value of the 0, 150 strike line policy is 0.0001, which shows that for any reasonably desired confidence level, the dynamically optimized strike lines decrease the average wasted time. It can also be seen that TRIP performs 7% and SLIP performs 26% better than 0, 0 in average wasted time.

It can be seen that at an arrival rate of 166 trains per day, the effect of the strike line is significant because it is common that many trains will be in the system simultaneously. However, that effect is limited because pumps free themselves much quicker due to faster refueling speeds. This means that even though tracks are occupied, the pumps are often free and available, leading to a decrease in strike line impact. It is noted that the average number of infeasible trains throughout the 30 replications is 0.00 and 3.96 lower, respectively, for the dynamic optimized strike lines versus the other two strike line configurations and 0.11 lower compared to the TRIP optimal strike lines.

Based on the experimental results, solving SLIP and incorporating the dynamic optimal strike line policy will improve the refueling yard operations regardless of whether or not a new refueling technology is adopted. This impact is statistically significant against static strike line policies of 0, 150 and intuitively better than 0, 0 and TRIP's optimal policy 10, 66. This allows a dynamic policy to process more trains each day. It can be seen from these results that in all cases, the dynamic strike line policy performs better in aligning trains than using any of the static policies. This proves the significance of being able to plan for each train according to the situation as opposed to using a constant policy.

These results using this integrated simulation show that the TRIP optimal solution still performs better than the intuitive strike line policies. This further demonstrates the effectiveness of the TRIP optimal static policy compared with the intuitive strike line policies. It also proves that either method would perform better in the real-world than an intuitive strike line policy.

Chapter 5 - Conclusion

Well-run railroad operations are vital to the economy. Rail transportation accounts for nearly 40 percent of freight movement ton-miles in the U.S. More importantly, total freight tonnage moved by rail is expected to increase 22 percent between 2010 and 2035 (U.S. Department of Transportation 2010). Refueling yards are critical to the efficiency of rail networks as these yards pump over 3 billion gallons of fuel each year (U.S. Department of Transportation 2016). If rail traffic increases, adding infrastructure, including tracks and fueling platforms, can increase a yard's capacity, but at considerable expense and time. Upgrading to a new refueling system may represent a more economical option to increase an individual yard's capacity. In order to take full advantage of this new technology, railyards must become more efficient and eliminate any waste time. The research in this thesis improves the efficiency of railyards through the introduction and solution techniques of the locomotive refueling system configuration problem (LRSCP).

A primary contribution of this thesis is a solution and validation methodology to solve the off-line LRSCP. This process begins with TRIP, an integer program that maximizes the weighted number of train combinations that can be refueled without delay. TRIP's outputs are used as inputs to a simulation developed in Simio®. The off-line simulation examines the impact of intuitive static strike line policies in comparison to the optimized strike lines.

Another significant contribution of this thesis is the creation of a method to solve the on-line LRSCP. This tool, built in Python, dynamically incorporates SLIP into the simulation. SLIP determines the optimal strike lines and pump assignments for each incoming train. The simulation uses SLIP's output to determine track and pump usage. This is truly an integrated system as approximately 300 integer programs are solved to simulate a single day. Similar to the off-line simulation, the on-line simulation models the railyard and examines the impact of different LRSCP solutions.

Based on the experimental results, incorporating the optimal strike lines from TRIP or SLIP will improve the refueling yard operations by 10% to 30% regardless of whether or not a new refueling technology is adopted. This impact is statistically significant and allows more trains to be processed each day as well as impacting other important parameters, such as time spent in the yard and the maximum queue for the railyard. Additionally, using a new refueling technology should enable railroad companies to dramatically increase system capacity without the enormous expense of laying new track. Furthermore, using optimal strike lines leads to a significant decrease in the amount of waste time and increases the efficiency of the platforms. This would lead to time savings, an additional ability to handle variability, and significant potential revenue opportunities with increased railyard capacity.

5.1 Recommendation for Practice

Since the strike line decision was found to be influential on railyard efficiency, I recommend that railroad companies implement either SLIP's or TRIP's optimal strike line policy into their day-to-day practice. This will allow railyards to improve efficiency, increase capacity, and decrease the time trains spend in the railyard. These variables are intuitively important for all railroad companies and the implementation of TRIP would be relatively simple with significant potential revenue due to increased capacity. Implementing SLIP would also improve the efficiency of any railyard and would be an upgrade over intuitive policies. TRIP should be implemented in any railyard in need of a quick and easy solution with little impact on the entire network or one with operational difficulty that would result from a changing strike line for each train. However, SLIP should be implemented for all railyards that play a major role in the rail network or can easily implement a dynamic policy. SLIP's dynamic policy will always perform at least as well as TRIP's optimal static policy with the capability to further reduce wasted time.

5.2 Future Work

There are many opportunities for future work related to this research. The first area for future work focuses on increasing the size of LRSCP to incorporate multiple railyards and eventually the entire rail network. While LRSCP in this thesis revolves around increasing the efficiency inside a single railyard, LRSCP and similar problems could be combined to increase the efficiency across the entire rail network. These gains would lead to substantial revenue opportunities for railroad companies.

The second area of future research is examining LRSCP and its similar relationship to scheduling problems. LRSCP can be viewed as a sum of total completion time scheduling problem with release times (arrival times at the platform) and multiple machines (tracks and pumps). The pump assignment creates added complexity as some of the multiple machines have shared resources. Thus, the shortest processing time first algorithm cannot optimally solve this problem. However, researching different sum of total completion time scheduling problems and examining algorithms to solve for optimality could lead to the creation of an algorithm, which outputs a provably optimal solution to LRSCP without the time-intensive use of an integer program. Creating such an algorithm will drastically increase the solving speed of TRIP and SLIP. Alternatively, LRSCP could be proven to be *NP*-Complete.

A third area of future work could focus on identifying problems and applications for integrating simulation and optimization. Most simulation software, such as Simio®, does not have the capability to be integrated with optimization software, such as CPLEX. This limits the types of problems that can be solved using either software, as well as the quality of solutions gained for either method. Most simulation software includes what they call an optimization function, but instead of being provably optimal, these functions simply choose a variable to change when trying to maximize a certain output. Every possible parameter is tested and the parameters that have the greatest output are returned as best-case solutions. However, due to the randomness of simulations, these solutions are not provably optimal. Therefore, creating a software with the functionality to perform integrated optimization and simulation would have a great impact on the ability to improve complex real-world applications.

Appendix A - References

Ahuja, R.K., K.C. Jha, and J. Liu. 2007. “Solving Real-Life Railroad Blocking Problems.” *Interfaces* 37(5):404–419.

Alarcón, Fernando, Guillermo Durán, and Mario Guajardo. 2014. "Referee Assignment in the Chilean Football League Using Integer Programming and Patterns." *International Transactions in Operational Research* 21.3: 415-438.

Assad, A.A. 1980. “Models for Rail Transportation.” *Transportation Research Part A: General* 14(3):205–220.

Barkan, C.P.L. 2004. “Cost Effectiveness Of Railroad Fuel Spill Prevention Using A New Locomotive Refueling System.” *Transportation Research Part D: Transport and Environment* 9(4):251–262.

Barta, J., A.E. Rizzoli, M. Salani, and L.M. Gambardella. 2012. “Statistical Modelling Of Delays in a Rail Freight Transportation Network.” In *Proceedings of the 2012 Winter Simulation Conference,* edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, 3250–3261. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Bauer, Amy L., Catherine AA Beauchemin, and Alan S. Perelson. 2009. "Agent-Based Modeling of Host–Pathogen Systems: The Successes and Challenges." *Information sciences* 179.10: 1379-1389.

Boland, Natashia, et al. 2013. "Mixed Integer Programming Based Maintenance Scheduling For the Hunter Valley Coal Chain." *Journal of Scheduling* 16.6: 649-659.

Bonabeau, Eric. 2002. "Agent-Based Modeling: Methods And Techniques For Simulating Human Systems." *Proceedings of the National Academy of Sciences* 99.suppl 3: 7280-7287.

Bostel, N., and P. Dejax. 1998. “Models and Algorithms for Container Allocation Problems on Trains in a Rapid Transshipment Shunting Yard.” *Transportation Science* 32(4):370–379.

Brailsford, S. C., and N. A. Hilton. 2001. "A Comparison of Discrete Event Simulation and System Dynamics for Modelling Health Care Systems.”

Budai, G., D. Huisman, and R. Dekker. 2006. “Scheduling Preventive Railway Maintenance Activities.” *Journal of the Operational Research Society* 57(9):1035–1044.

Buffon, G. 1777. Essai D’arithmétique Morale. Histoire Naturelle, Générale Et Particulière, Supplément 4:46–123.

Chahar, K., C. Cheng, and Y. Pranoto. 2011. “Strategic Crew Planning Tool In Railroad: A Discrete Event Simulation.” In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, 3693–3703. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Chen, Xuwei, John W. Meaker, and F. Benjamin Zhan. 2006. "Agent-Based Modeling and Analysis of Hurricane Evacuation Procedures for the Florida Keys." *Natural Hazards* 38.3: 321.

Cheng, Lifei, and Marco A. Duran. 2004. "Logistics for World-Wide Crude Oil Transportation Using Discrete Event Simulation and Optimal Control." *Computers & chemical engineering* 28.6: 897-911.

Conway, R. W., B. M. Johnson, and W. L. Maxwell. 1959. “Some Problems of Digital Systems Simulation”. *Management Science* 6 (1): 92–110.

Cordeau, J.-F., P. Toth, and D. Vigo. 1998. “A Survey of Optimization Models for Train Routing and Scheduling.” *Transportation Science* 32(4):380–404.

Daganzo, Carlos F. 1986. “Static Blocking At Railyards: Sorting Implications and Track Requirements.” *Transportation Science* 20(3):189–199.

D'Ariano, A., D. Pacciarelli, and M. Pranzo. 2007. “A Branch and Bound Algorithm for Scheduling Trains in a Railway Network.” *European Journal of Operational Research* 183(2):643–657.

Detty, Richard B., and Jon C. Yingling. 2000. "Quantifying Benefits Of Conversion To Lean Manufacturing With Discrete Event Simulation: A Case Study." *International Journal of Production Research* 38.2: 429-445.

Dorfman, M. J., and J. Medanic. 2004. "Scheduling Trains on a Railway Network Using a Discrete Event Model of Railway Traffic." *Transportation Research Part B: Methodological* 38.1: 81-98.

Dorfman, M.J., and J. Medanic. 2004. “Scheduling Trains on a Railway Network Using a Discrete Event Model of Railway Traffic.” *Transportation Research Part B: Methodological* 38(1):81–98.

Dantzig, G. B. 1947. “Maximization of a Linear Function of Variables Subject To Linear Inequalities.” *Activity Analysis of Production and Alloction, New York, Wiley*: 339-347.

Drake, Gleen R., Jeffrey S. Smith, and Brett A. Peters. 1995. "Simulation as a Planning and Scheduling Tool for Flexible Manufacturing Systems." *Simulation Conference Proceedings, 1995. Winter*. IEEE, 1995.

Easton, Todd, et al. 2011. "Simulating the Spread of an Epidemic in a Small Rural Kansas Town." *International Journal of Artificial Life Research (IJALR)* 2.2: 95-104.

Ergun, Özlem, et al. 2014. "Improving Humanitarian Operations through Technology-Enabled Collaboration." *Production and Operations Management*23.6: 1002-1014.

Eubank, Stephen, et al. 2004. "Modelling Disease Outbreaks in Realistic Urban Social Networks." *Nature* 429.6988: 180.

Fishman, George. 2013. "Discrete-Event Simulation: Modeling, Programming, and Analysis." *Springer Science & Business Media*.

Fügenschuh, Armin. 2009. "Solving a School Bus Scheduling Problem with Integer Programming." *European Journal of Operational Research* 193.3: 867-884.

Goldsman, David, Richard E. Nance, and James R. Wilson. 2010. "A Brief History Of Simulation Revisited." *Proceedings of the Winter Simulation Conference*. Winter Simulation Conference, 2010.

Gunpinar, Serkan, and Grisselle Centeno. 2015. "Stochastic Integer Programming Models for Reducing Wastages and Shortages of Blood Products at Hospitals." *Computers & Operations Research* 54: 129-141.

Harvey, Neil, et al. 2007. "The North American Animal Disease Spread Model: A Simulation Model to Assist Decision Making In Evaluating Animal Disease Incursions." *Preventive veterinary medicine* 82.3: 176-197.

Hasegawa, Seiya, and Yukio Kosugi. 2006. "Solving Nurse Scheduling Problem by Integer-Programming-Based Local Search." *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*. Vol. 2. IEEE, 2006.

He, S., R. Song, and S.S. Chaudhry. 2003. "An Integrated Dispatching Model for Rail Yards Operations." *Computers and Operations Research* 30(7):939–966.

Helbing, Dirk. 2012. "Agent-Based Modeling." *Social self-organization*. Springer Berlin Heidelberg. 25-70.

Heppenstall, Alison J., et al., 2011. eds. "Agent-Based Models of Geographical Systems". *Springer Science & Business Media*.

Hill, R. John, and Louisa J. Bond. 1995. "Modelling Moving-Block Railway Signalling Systems Using Discrete-Event Simulation." *Railroad Conference, 1995., Proceedings of the 1995 IEEE/ASME Joint*. IEEE, 1995.

Hoffman, Karla L., and Ted K. Ralphs. 2013. "Integer and Combinatorial Optimization." *Encyclopedia of Operations Research and Management Science*. Springer US. 771-783.

Huang, Philip Y., Loren P. Rees, and Bernard W. Taylor. 1983. "A Simulation Analysis of the Japanese Just-In-Time Technique (With Kanbans) For a Multiline, Multistage Production System." *Decision Sciences* 14.3: 326-344.

Jerry, Banks. 1984. “Discrete-Event System Simulation”. *Pearson Education India*.

Jha, K.C., R.K. Ahuja, and G. Şahin. 2008. “New Approaches for Solving the Block-To-Train Assignment Problem.” *Networks* 51(1):48–62.

Johnson, Gregory L., et al. 1996. "Stochastic Weather Simulation: Overview and Analysis of Two Commonly Used Models." *Journal of Applied Meteorology* 35.10: 1878-1896.

Jones, Randolph M., et al. 1999. "Automated Intelligent Pilots For Combat Flight Simulation." *AI magazine* 20.1: 27.

Jun, J. B., Sheldon H. Jacobson, and James R. Swisher. 1999. "Application of Discrete-Event Simulation in Health Care Clinics: A Survey." *Journal of the operational research society*: 109-123.

Land, A. H., A. G. Doig. 1960. “An Automatic Method of Solving Discrete Programming Problems”. *Econometrica* 28, 497–520.

Lesyna, William R. 1999. "Sizing Industrial Rail Car Fleets Using Discrete-Event Simulation." *Simulation Conference Proceedings, 1999 Winter*. Vol. 2. IEEE, 1999.

Lin, E., and C. Cheng. 2009. “Yardsim: A Rail Yard Simulation Framework and Its Implementation in a Major Railroad in the U.S.” In *Proceedings of the 2009 Winter Simulation Conference,* edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 2532–2541. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Lin, E., and C. Cheng. 2011. “Simulation and Analysis of Railroad Hump Yards in North America.” In *Proceedings of the 2011 Winter Simulation Conference,* edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, 3710–3718. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Macal, Charles M., and J. Michael. 2007. "North. Agent-Based Modeling And Simulation: Desktop Abms." *the 2007 Winter Simulation Conference. Washington, DC, USA*. 2007.

Matsumoto, Makoto, and Takuji Nishimura. 1998. "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator." *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1: 3-30.

Menth, Megan, and Jessica L. Heier Stamm. 2015. "An Agent-Based Modeling Approach to Improve Coordination between Humanitarian Relief Providers." *Winter Simulation Conference (WSC), 2015*. IEEE, 2015.

Metropolis, Nicholas, and Stanislaw Ulam. 1949. "The Monte Carlo Method." *Journal of the American statistical association* 44.247: 335-341.

Nevins, Michael R., Charles M. Macal, and Joseph C. Joines. 1998. "A Discrete-Event Simulation Model for Seaport Operations." *Simulation* 70.4: 213-223.

Newton, H.N., C. Barnhart, and P.H. Vance. 1998. "Constructing Railroad Blocking Plans to Minimize Handling Costs." *Transportation Science* 32(4):330–345.

Niederreiter, Harald. 1978. "Quasi-Monte Carlo Methods and Pseudo-Random Numbers." *Bulletin of the American Mathematical Society* 84.6: 957-1041.

Nourbakhsh, S.M., and Y. Ouyang. 2010. "Optimal Fueling Strategies for Locomotive Fleets In Railroad Networks." *Transportation Research Part B: Methodological* 44(8-9):1104–1114.

Pegden, C. Dennis. 2008. "Introduction to SIMIO." *Simulation Conference, 2008. WSC 2008. Winter*. IEEE.

Perez, Liliana, and Suzana Dragicevic. 2009. "An Agent-Based Approach For Modeling Dynamics Of Contagious Disease Spread." *International journal of health geographics* 8.1: 50.

Raviv, T., and M. Kaspi. 2012. "The Locomotive Fleet Fueling Problem." *Operations Research Letters* 40(1):39–45.

Recalde, Diego, Ramiro Torres, and Polo Vaca. 2013. "Scheduling the Professional Ecuadorian Football League by Integer Programming." *Computers & Operations Research* 40.10: 2478-2484.

Ribeiro, Celso C. 2012. "Sports Scheduling: Problems and Applications." *International Transactions in Operational Research* 19.1-2: 201-226.

Sajedinejad, A., S. Mardani, E. Hasannayebi, S.A.R. Mir Mohammadi K., and A. Kabirian. 2011. "SIMARAIL: Simulation Based Optimization Software For Scheduling Railway Network." In *Proceedings of the 2011 Winter Simulation Conference,* edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, 3735–3746. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sawik, Tadeusz. 2005. "Integer Programming Approach to Production Scheduling For Make-To-Order Manufacturing." *Mathematical and computer modelling*41.1: 99-118.

Shannon, Robert E. 1998. "Introduction to the Art and Science of Simulation." *Proceedings of the 30th conference on Winter simulation*. IEEE Computer Society Press.

Shavitt, Yuval, and Tomer Tankel. 2004. "Big-Bang Simulation for Embedding Network Distances In Euclidean Space." *IEEE/ACM Transactions on Networking (TON)* 12.6: 993-1006.

Sherali, H.D., and A.B. Suharko. 1998. "A Tactical Decision Support System for Empty Railcar Management." *Transportation Science* 32(4):306–329.

Sivakumar, A. I. 2001. "Multiobjective Dynamic Scheduling Using Discrete Event Simulation." *International Journal of Computer Integrated Manufacturing* 14.2: 154-167.

Tesfatsion, Leigh. 2006. "Agent-Based Computational Economics: A Constructive Approach to Economic Theory." *Handbook of computational economics* 2: 831-880.

U.S. Department of Transportation, Bureau of Transportation Statistics. 2015. *Freight Facts and Figures*.https://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/data_and_statistics/by_subject/freight/freight_facts_2015.

U.S. Department of Transportation, Bureau of Transportation Statistics. 2016. *National Transportation Statistics*. http://www.bts.gov/publications/national_transportation_statistics/.

U.S. Department of Transportation, Federal Railroad Administration. 2010. *National Rail Plan Progress Report*. https://www.fra.dot.gov/eLib/Details/L02696.

Vaidyanathan, B., R.K. Ahuja, and J.B. Orlin. 2008. "The Locomotive Routing Problem." *Transportation Science* 42(4):492–507.

Verschelden, L., J. Heier Stamm, and T. Easton. (To appear). "Integrated Optimization and Simulation Models for the Locomotive Refueling System Configuration Problem". *In Proceedings of the 2017 Winter Simulation Conference*, edited by W.K.V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wichmann, Brian A., and I. David Hill. 1982. "Algorithm AS 183: An Efficient and Portable Pseudo-Random Number Generator." *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 31.2: 188-190.

Wolfe, Philip. 1959. "The Simplex Method for Quadratic Programming." *Econometrica: Journal of the Econometric Society*: 382-398.

Zhang, G. Peter, B. Eddy Patuwo, and Michael Y. Hu. 2001. "A Simulation Study of Artificial Neural Networks for Nonlinear Time-Series Forecasting." *Computers & Operations Research* 28.4: 381-396.

Zhang, Yue, Oded Berman, and Vedat Verter. 2012. "The Impact of Client Choice on Preventive Healthcare Facility Network Design." *OR spectrum* 34.2: 349-370.