Performance Analysis of a Double Crane with Finite Interoperational Buffer Capacity with Multiple Fidelity Simulations

by

Sundaravaradhan Rengarajan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2018 by the
Graduate Supervisory Committee:

Giulia Pedrielli, Chair
Feng Ju
Teresa Wu

ARIZONA STATE UNIVERSITY

August 2018

ABSTRACT

With trends of globalization on rise, predominant of the trades happen by sea, and experts have predicted an increase in trade volumes over the next few years. With increasing trade volumes, container ships' upsizing is being carried out to meet the demand. But the problem with container ships' upsizing is that the sea port terminals must be equipped adequately to improve the turnaround time otherwise the container ships' upsizing would not yield the anticipated benefits. This thesis focus on a special type of a double automated crane set-up, with a finite interoperational buffer capacity. The buffer is placed in between the cranes, and the idea behind this research is to analyze the performance of the crane operations when this technology is adopted. This thesis proposes the approximation of this complex system, thereby addressing the computational time issue and allowing to efficiently analyze the performance of the system. The approach to model this system has been carried out in two phases. The first phase consists of the development of discrete event simulation model to make the system evolve over time. The challenges of this model are its high processing time which consists of performing large number of experimental runs, thus laying the foundation for the development of the analytical model of the system, and with respect to analytical modeling, a continuous time markov process approach has been adopted. Further, to improve the efficiency of the analytical model, a state aggregation approach is proposed. Thus, this thesis would give an insight on the outcomes of the two approaches and the behavior of the error space, and the performance of the models for the varying buffer capacities would reflect the scope of improvement in these kinds of operational set up.

ACKNOWLEDGMENTS

I am thankful to my advisor Dr. Giulia Pedrielli for giving me a wonderful opportunity to solve this interesting problem. I am equally thankful to Dr. Feng Ju for providing me valuable feedbacks during the meeting sessions on improving the research. I am thankful to Dr. Teresa Wu for being the committee member of this thesis. I would like to thank Mr. Brint Macmillan for assisting me with the software installations. I am thankful to Mr. Logan Mathesan, Mr. Cong Liu, Mr. Gokul Chandrasekaran, Mrs. Carly Metcalfe for their valuable suggestions. Additionally, I would like to thank my parents, Mr. S.P Rengarajan and Mrs. Revathi Rengarajan for their constant love and support. I would also like to thank my friends, well-wishing relatives, and other well-wishers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Maritime trade constitutes an important part of world economy. As of 2017, over 80% of global trade by volume occurs through maritime and it is estimated that the volumes would grow at a compound annual growth rate of 3.2 percent between 2017 and 2022 [33]. With the growing trade volume, it is important to ensure that the seaports handle the demand with ease. In the last few years, the vessels deployed in the transpacific trade lanes nearly doubled their capacity while the terminals- which were sized for much smaller ships- have relatively remained unchanged [14]. This mismatch between the capacities and the demand could cause ripples in the supply chain and prior to any changes made in the seaports, it is very important to carefully analyze the proposed new systems owing to the huge amount of costs involved. In this thesis, a special type of crane operation called automated double crane with interoperational finite buffer capacity have been considered to assess the productivity. The objective of this study is to analyze the performance of this complex system by addressing the computational issue to run the simulations leading to efficient analysis.

The approach to analyze the system has been done in two phases. The first phase consists of the development of discrete event simulation models and the second phase consists of the development of analytical models. The idea of developing a discrete event simulation model is to model the system that is close to reality as certain complex operations are better modeled through this approach [34]. But it turned out that the discrete event simulation model resulted in higher computation times leading to the development of analytical modeling. And with respect to analytical modeling, a continuous time markov approach has been used to model the system. The reason for using continuous time markov process was that the system being modeled had current set of events depending on their corresponding immediate previous set of events, and not on the events beyond that. So this makes us realize that the system is memoryless, hence continuous time markov process modeling approach is used. Further, the two models have been compared to observe the error space distribution to gain useful insights about the gap between the two models. The distribution of error space, and the factors influencing the process are discussed as well.

CHAPTER 2

LITERATURE REVIEW

The objective of the literature review is to understand the researches with respect to sea port terminals. Thus far, the research for the terminal operations can be classified into two categories, assessment of the productivity of the terminal operations, and optimization of the terminal operations. And the latter classification could be further classified into two- Quay Crane Scheduling Problem and a Quay Crane Assignment problem.

To the best of my knowledge, there are numerous studies deploying discrete event simulation modeling for the analysis of sea port terminals. Discrete event simulation concerns the modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time which are classified as events[34]. [15] considers double trolley quay cranes and supertainer quay cranes for their analysis and have generalized the picking times as normally distributed with a specific mean and standard distribution and developed a statistical model out of that. [18] discusses advanced quay cranes that include multi lift spreaders, double trolley Q/C's, double sided operating systems. [3] modeled the seaport terminal at Gioia Tauro in Southern Italy using discrete event simulation. This research considered the discharge process of vessel considering the straddle carriers, cranes as some of the operational components but the location of containers on vessels have not been considered explicitly. Moreover, the operational set up was entirely different from our case where we consider platform(s) as additional component(s). Numerous other studies have simulated a specific sea port terminal considering the various factors associated with it but to my utmost knowledge, model of double crane with platform(s) where one crane serving the vessel and the other serving the yard has not been analyzed so far. [32] discusses the development of a queuing model for the seaport terminal, and considers the ships' arrival and the servicing of the ships as M/M/1 queuing system. [10] minimizes the processing time of the vessel using asymmetric travelling salesman problem and Johnson's algorithm. [12] developed a queueing model to analyze the congestion of the system. Queuing models were used by [13] as well but the objective of this study was to assess the impacts of tug services on harbor congestion. [8] captures the system through continuous time markov process and focusses on the performance of two quay crane operations. This research generalizes the service and arrival rates which in reality would be a state dependent parameter. By state dependent parameter, it is intended that the rates depend on the respective states of the system which include the location of containers as well. [20] discusses a two phase integrated approach that consists of optimizing the container processing, and the output of the optimization algorithm which consists of the optimal schedule and other decision variables would be the input for the simulation model. The objective of this study was to optimize the container processing whereas in our case, the objective is to analyze the container processing for various parameters. [5] discusses the throughput analysis of passing dual yard cranes by a continuous time markov chain approach. One of the assumptions made in evaluating the service times was that the picking of containers from the stack buffer lanes would follow a uniform distribution. This assumption is not realistic because, in reality while a picking strategy is used, the probability of picking containers from certain buffer lanes would be close to zero.

Based on the literature review, it is understood that with respect to analysis of the sea port terminals, much of the researches have been carried out considering a specific set of terminal setups as majority of the researchers studied a particular sea port terminal and have reported out their analysis. So, the analysis of the sea port terminals should not be generalized as there are multiple terminals with different layout design. Also, researches considering crane systems with interoperational buffer capacity are few to the best of my knowledge.

CHAPTER 3

METHODOLOGY

This work looks into the efficient analysis of the productivity of an automated double crane system with finite decoupling buffer capacity. The system constitutes of quay crane on the vessel side and yard crane on the yard side, and platform of either size one or size two is placed at the middle of these cranes which acts as temporary buffer. As a vessel arrives, unloading and loading operations occur, and the process evolves over time until completion. And the containers being processed are homogeneous which implies that they could be twenty foot equivalent unit containers or forty foot equivalent unit containers. Below is the overview of the approach taken to analyze the system.

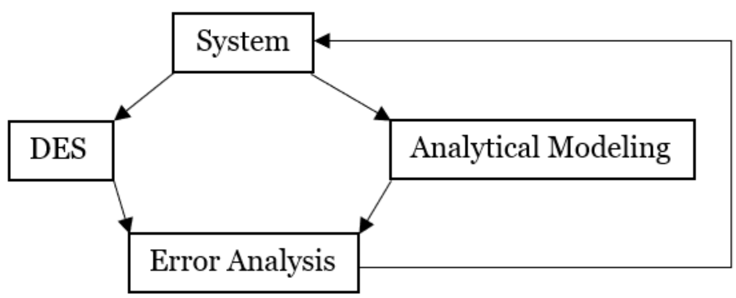
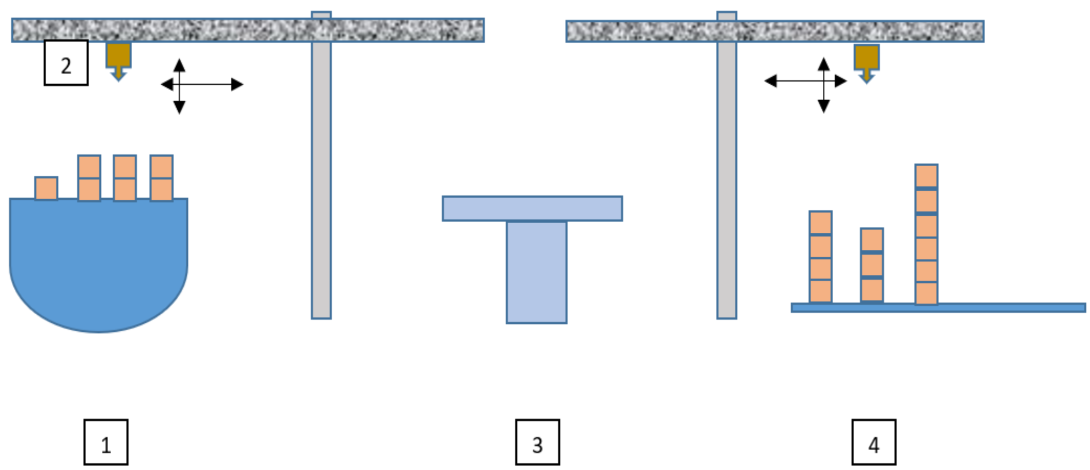


Figure 1: Methodology overview

So, the approach consists of the development of the discrete event simulation model, and to improve the efficiency further, an analytical model is developed. Further, the error space of the analytical model is explored to identify its performance with respect to simulation model.

Description of the system

This problem simulates the process of double crane operation which loads and unloads containers off vessel. Below is the schematic representation of the process. In this type of operation, there are two cranes, one that processes the vessel, and the other that processes the yard, and these cranes have crane spreaders that hold the containers, and these spreaders traverse in three directions(x, y and z) for the pick and drop operation of the containers. The platform which bisects the entire operation acts as a temporary buffer and relays the entire operation.



1- Vessel, 2- Crane Spreader, 3- Platform, 4- Yard

Figure 2: Schematic representation of the processes

The objective of the simulation study is to assess the performance of this system for a platform of size one and platform of size two separately. The performance measure of this process is the makespan. Makespan is the time required to complete all jobs [25].

3.1 Discrete event simulation modeling

The simulation model simulates the above system with various level of values and user inputs. These are the location of the containers and empty spaces on vessel and on yard, time taken by the yard crane and quay crane to process, number of loading and number of unloading jobs, loading and unloading strategy, and number of platform.

3.1.1 Model logic

Event based approach

To give an intuitive understanding of the system, an event graph is constructed.

An event graph consists of nodes and directed arcs which depict how the events are scheduled from other events and from themselves [34].

Appendix A contains the notation for event graph.

In the event graph, $QC_j$and YC_j take values of either zero, one, or two which represent no jobs, loading job, and unloading job respectively.

QC_p and YC_p take values of either one or zero. Zero implies that the quay crane spreader is at the vessel point, and yard crane spreader is at the yard point. A value of one implies that the quay crane spreader or yard crane spreader is at the location of platform.

The event graph of the system is displayed below,

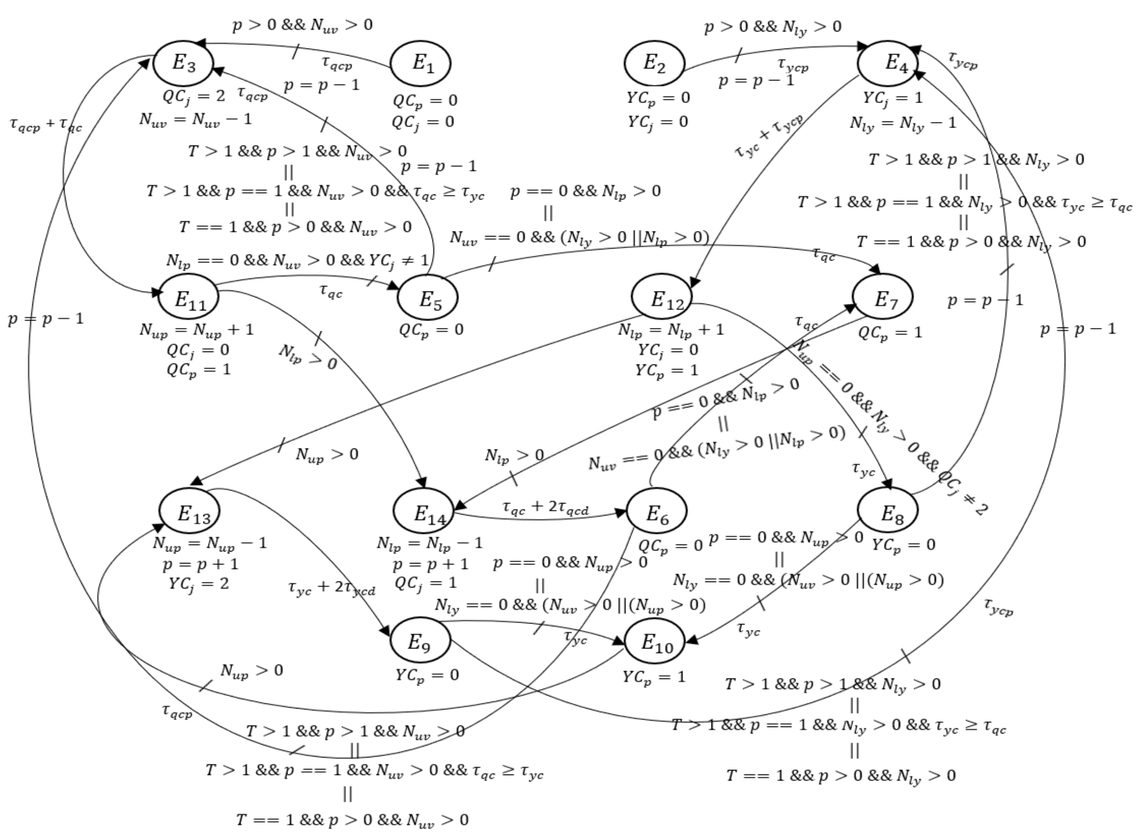


Figure 3: Event Graph of the simulation models

To help you understand the event graph, occurrences of events E1 and E3 are explained below, and rest of the events work based on similar logic.

Initializing event1 corresponds to the beginning of unloading process. Initially, before the event begins the position of quay crane would be zero, and this could be inferred that the quay crane spreader is located at the vessel point. This event would schedule the picking of unloading job from the vessel (E3) on the condition that at least one platform is available, and there is at least one unloading job on the vessel. Based on the satisfaction of the above condition, the picking of unloading job from the vessel occurs after τ_{qcp} time units.

As E3 begins to occur, number of platform gets decremented by one as one platform has been reserved for this unloading operation, quay crane's job status gets changed to two which infers that quay crane has been occupied with an unloading job, and number of unloading jobs on vessel is decremented by one as an unloading job is being picked by the occurrence of E3.

For the occurrences of certain events, no time unit has been explicitly mentioned near the arrows in the event graph which implies that the events occur instantly based on the condition getting satisfied.

The event graph shows the way the events are triggered leading to the process evolution in time. But the simulation has been performed using a process based approach, and let's get into the detail of this modeling by understanding each set of operation separately.

Process based approach

Figure 4 represents the system from the perspective of a process based approach.

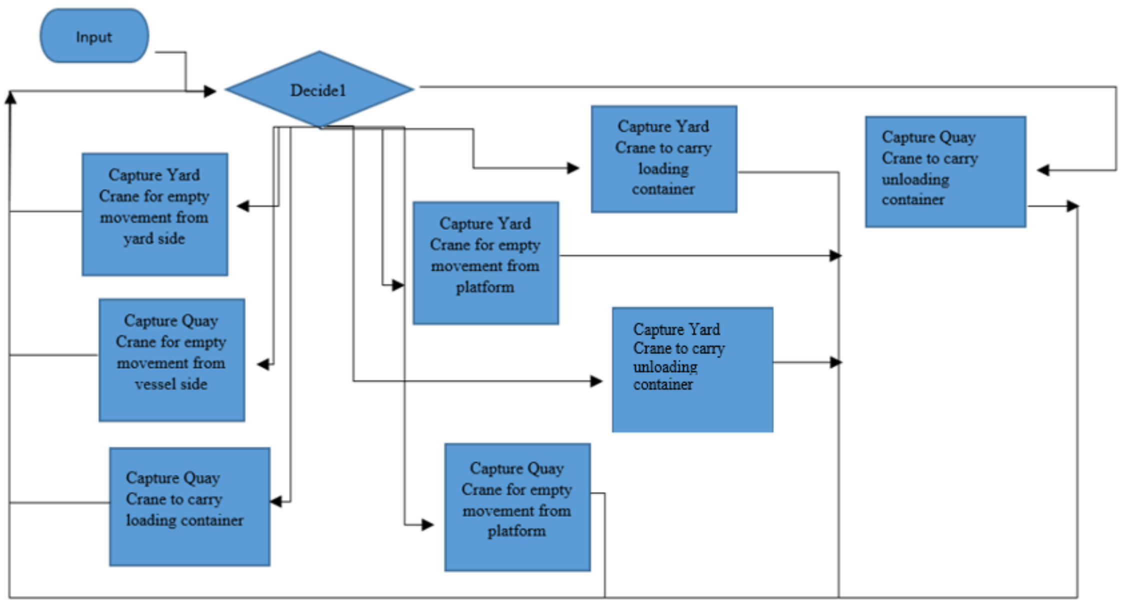


Figure 4: Process based approach flow diagram of the simulation models

The input node in Figure 4 initializes the simulation. Either of the process could begin, and arbitrarily the unloading process is chosen to start with.

The possible set of tasks that are involved in the entire process are capture of the quay crane with an unloading or loading container, capture of the yard crane with an unloading or loading container, empty movements of quay crane, and empty movements of yard crane. These tasks are performed depending on the conditions that are set in the decide1 node.

The decide1 node in Figure 4 comprises various conditions which check for the availability of platforms, time taken by the quay crane to reach the vessel, time taken by the quay crane to reach the platform, time taken by the yard crane to reach the yard, time taken by the yard crane to reach the platform, number of unloading jobs to be performed, number of loading jobs to be performed. Based on the conditions that are getting satisfied, the decide1 node dictates the flow of the process, and one of the important things that it performs is the aid in avoiding deadlock situation. For instance, when only one platform exists, and both the loading and unloading processes are equally likely to occur, there are chances that both the processes could simultaneously occur resulting in making both the cranes carry their respective containers and reach the platform, and that would be an unfortunate incident as the platform could handle only one container. To prevent such kind of events, the decide1 node would be very helpful.

The times the entire operation is concerned with are time to reach the start position of quay crane, time to reach the end position of quay crane, time to reach the start position of yard crane, and time to reach the end position of yard crane. The start position for the quay crane and yard crane has been set to be the vessel side point and yard side point respectively. The end position of the quay crane and yard crane are set to be the platform side. Apart from these, the times to pick and drop containers are incorporated in the modeling as well. As we are analyzing an automated crane system, the movement of the crane times are deterministic, and the elements of variability are included in the location of containers on vessel and on yard, and location of empty space on yard and on vessel. Further, the simulation clock takes the most recent time and gets updated as the process evolves.

Platform

The platform has two possible states, an available state and a busy state. Whenever the platform is seized, it gets changed to busy state, and whenever it is released, the state becomes available. The capacity of the platform is finite and can take smaller positive integer values starting from one. But as part of our analysis, we have considered platform of buffer sizes one and two only.

Dynamic creation of empty spaces

The location of containers as well as the empty slots either on the yard or on the vessel are defined by the user in the simulation model, so the location is not deterministic. And the system is modeled in such a way that the generation of empty slots also evolves over time. For instance, if a vessel contains non-empty slots with "to be" unloaded containers, then as the unloading process happens, empty slots would be generated on the vessel. This applies to the yard side as well.

Container's pick-up and drop-off policy

The configuration of the containers on the vessel and on the yard are balanced. The containers are stacked one over the other and are arranged along rows and bays. There are four different strategies for picking and dropping the containers. Firstly, the containers could be picked or dropped across the bay and then across the stack. Secondly, the containers could be picked or dropped across the bay and then across the row. Thirdly, the containers could be picked or dropped across the row and then across the stack. Finally, the containers could be picked or dropped across the row and then across the bay. These strategies are same for the vessel and yard.

Using the approach stated above, the simulation model has been programmed and run but it turned out that the simulation modeling took more time to run leading to the development of an alternative model.

3.2 Analytical Modeling

It is generally believed that complex processes are not analytically tractable. But by considering the right set of assumptions, the development of analytical model is possible.

And the reason to consider analytical modeling is because of the computational inefficiency of the simulation model. So, this section discusses the analytical framework of the system.

The system which includes the one with buffer of size one, and with buffer of size two are modeled using continuous time markov process approach. This section discusses the creation of state space for the two models, and the implication of the state space created with respect to computational tractability is mentioned. Further, an aggregated state space approach has been proposed culminating in the stochastic modeling of the system under consideration.

3.2.1 Definition of the stochastic process

Before getting into the computation method, let's define the stochastic process that we are modeling.

The continuous time stochastic process $\{X(t), t \geq 0\}$ that we are modeling is markovian as it satisfies the property [27],

$P(X(t_n) = s_n \mid X(t_{n-1}) = s_{n-1}, X(t_{n-2}) = s_{n-2}, \dots , X(t_0) = s_0) = P(X(t_n) = s_n \mid X(t_{n-1}) = s_{n-1})$, where $s_n, s_{n-1}, \dots, s_0$ are the states indexed by the order of occurrences n, and $t_n > t_{n-1} > t_{n-2} >, \dots > t_0$

3.2.2 State set definition

System with buffer of size one

The states of the system with a buffer of size one are represented in the fashion - $\{(N_u, N_l, S_q, S_b, S_y), (N_u - 1, N_l, S_q, S_b, S_y), (N_u, N_l - 1, S_q, S_b, S_y), \dots, (0,0,0,0,0)\}$ where N_u, N_l represent the number of containers to be unloaded and the number of containers to be loaded respectively.

S_q, S_b , S_y represent the state of the quay crane, state of the buffer, and state of the yard crane respectively, and they could take values of $\{0,1,2\}$ where 0 means that the system is idle, 1 means that the system is occupied by a loading job, and 2 means that the system is occupied by an unloading job.

As we know the possible set of values that each sub state takes, it may seem that the total number of states are $(N_u + 1) * (N_l + 1) * 3 * 3 * 3$, but that's not the case. For instance, a state of $\{N_u, N_l, 2, S_b, 1\}$ is not possible as quay crane has an unloaded container on it and yard crane has a loaded container on it, and this would result in a deadlock situation that we had discussed earlier in Chapter 3.1.1.

Example

To make you understand the intuition of the stateset created, consider a statestet of $\{N_u - 3, N_l, 2,2,2\}$. The sub states of this set S_q, S_b, S_y are filled by 2, and this means that quay crane, buffer, and yard crane are processing unloading jobs. We are not certain about the total number of unloading jobs that have been completed but we are certain that three unloading jobs are already in process as indicated by the sub states S_q, S_b, S_y, so the sub state N_u could only take values starting from $N_u - 3$ and can go down till zero. If N_u takes a value of zero, it means that there are no unloading jobs on the vessel, but unloading jobs are under process. With respect to N_l , this stateset does not provide any information about the loading jobs, so N_l could take any values starting from N_l and could go down till zero.

System with buffer of size two

Now let's look into the system with buffer of size two. The states of this system are represented in the fashion,

$\{(N_u, N_l, S_q, S_{b1}, S_{b2}, S_y), (N_u - 1, N_l, S_q, S_{b1}, S_{b2}, S_y), (N_u, N_l - 1, S_q, S_{b1}, S_{b2}, S_y), .., (0,0,0,0,0,0)\}$. The only change is that we have two buffers, and their respective states are represented by S_{b1} and S_{b2}, and just like the earlier case, each of these sub states could take values of $\{0,1,2\}$

The possible set of states in this case are mentioned in Appendix C.

Now having created the state sets and based on the number of containers to be loaded and unloaded, the required statespace of the system could be generated.

This is a terminating reducible markov process with absorbing states of $(0,0,0,0,0)$ and $(0,0,0,0,0,0)$ for the case of buffer size of one and for the case of buffer of size two respectively. The rest of the states are transient.

3.2.3 Creation of state dependent transition rate matrix

We need to compute the makespan of the process, and in terms of a markov process, we need to compute the total duration of the process to get absorbed. The parameters in our process are the times taken by the quay crane, yard crane, platform, pick and drop times in the vessel, pick and drop times in the yard. So, the parameters for quay crane, yard crane, and platform remain fixed throughout the process whereas the pick and drop times alone vary. It would be intuitive to corroborate this with an example.

Example

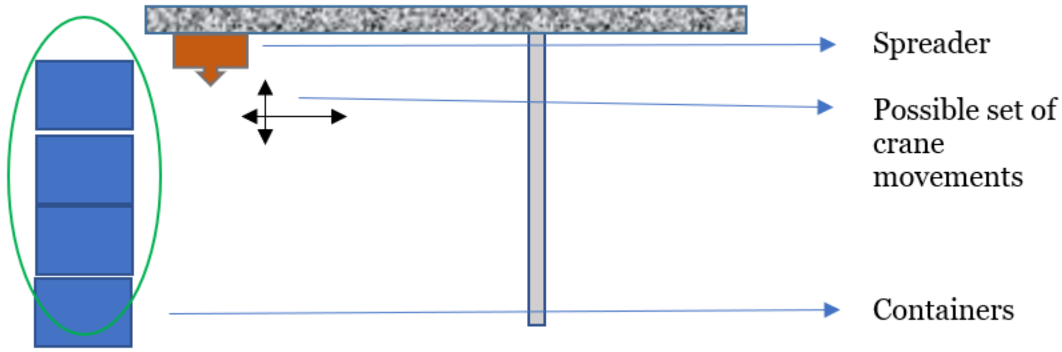


Figure 5: Illustration of container processing

From the Figure 5, we could see that the spreader would take less amount of time to pick the topmost container and would take more amount of time to pick the bottommost container. A pick operation involves both to and from movement of spreader, and same is the case with drop operation. The vertical black arrow represents the times to pick containers either from their respective vessel point or yard point. The horizontal black arrow represents the transfer of containers from platform to vessel point, or from vessel point to platform, and the time taken to perform these movements is same for the quay crane for a given set of process. The same applies to yard side as well.

From the above example, the movement illustrated by the horizontal black arrow comprises the fixed parameters, and the movement illustrated by the vertical black arrow comprises the variable parameters. The idea of vertical arrow movement was for illustration purpose but the pick and drop operation involves both horizontal and vertical movements as well. Using this information, the transition rate matrix is defined. So, the transition rate matrix consists of variable and fixed parameters. As illustrated by figure-5, each of the containers have different variable parameters defined by their location on vessel or on yard.

In the state-set $\{N_u, N_l, S_q, S_b, S_y\}$ of the system with buffer of size one, a change of state of S_b occurs at a fixed transition rate as this rate is defined by the parameter of platform time which is fixed. A change of state of N_u occurs at a variable transition rate as the change of this state implies that an unloading job has been picked from the vessel. As said earlier, the picking times differ based on the location of containers, so the transition rate is variable. Similarly, a change of state of N_l occurs at a variable transition rate. In the case of S_q and S_y, their transition rates depend on the type of operation. If the state of S_q represents an unloading job, the transition rate is fixed as it is the movement of container to the platform. On the other hand, if the state of S_q represents a loading job, the transition rate comprises of fixed parameter which is the movement of container to the platform and the variable parameter which is the movement of container to the empty location on vessel for dropping the container. Same applies to the case of S_y. So, the transition rate matrix comprises the rates that are fixed and variable. In other words, the transition rate matrix is state dependent. This case is extended to the system of buffer size two.

3.2.4 Analytical model simplification

With the information of the states, the transition matrix has been computed. But the implication of the state sets that we have created is that the statespace gets bigger and bigger as the number of loading and unloading jobs increase. For instance, a total of around 370 million states are created to process 5000 containers for the system with buffer of size two. And the transition rate matrix' size would be the square of 370 million. Such a high number of states would require an intense computational procedure and time, and the model would be inefficient.

This problem could be tackled by aggregating the statespace. The system consists of fixed and variable parameters. If we look at the state-set, the change of states of S_q, S_b, S_y occur at a fixed rate except for the dropping operation. So, it is not a bad idea to eliminate these states and reduce our state-set to $\{(N_u, N_l), (N_u - 1, N_l), (N_u, N_l - 1), .., (0,0)\}$. With this aggregated statespace, the total number of states for processing 5000 containers is close to 6.2 million and the transition rate is state dependent. But this number is still too high to be efficient.

Another approach to solve this problem is to model the states in the fashion $\{(N_u), (N_l), (N_u - 1), (N_l - 1), .., (0\ unloading\ jobs), (0\ loading\ jobs), (b_l),(\ b_u)\}$ where b_l indicates that the buffer location is occupied by a loading job and b_u indicates that the buffer location is occupied by an unloading job, and the rest of the states represent the number of loading and unloading jobs to be processed. This is the case for system with buffer size one. For system with buffer of size two, the states would be in the fashion,

$\{(N_u), (N_l), (N_u - 1), (N_l - 1), .., (0\ unloading\ jobs), (0\ loading\ jobs), (b1_l),(\ b1_u), (b2_l),(\ b2_u)\}$. Here b1 and b2 represent the two buffer locations and rest of the state logic is the same as the system with buffer of size one.

In this case, for a total of 5000 containers, the total number of states are 5006 for the system with buffer of size two and 5004 for the system with buffer of size one. Although the number of states is in thousands, this is a significant reduction. The transition diagrams using the aggregated state space are given below,

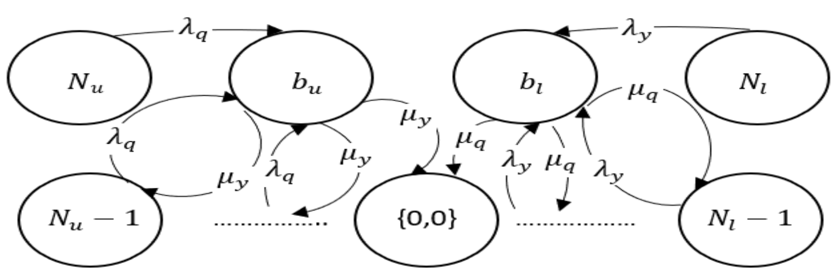


Figure 6: Transition diagram of system with buffer of size one

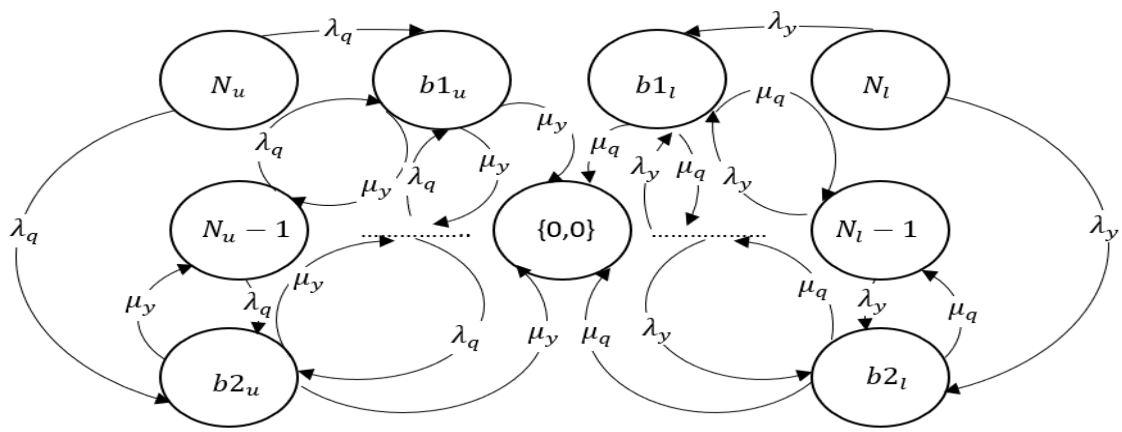


Figure 7: Transition diagram of system with buffer of size two

Building on this idea of state space, considering the fact that there are fixed and variable parameters, further state space reduction could be done by averaging the variable parameters. Therefore, the idea is to compute the average total dropping and picking times and include the averaged value in measuring transition rate. The reason for averaging the dropping and picking times is to convert the state dependent transition rate matrix into a matrix consisting of fixed set of rates throughout.

Then we consider the states ,

$\{(1\ loading\ job), (1\ unloading\ job), (0\ loading\ job\ \&\ 0\ unloading\ job), (b_l), (b_u)\}$ for the system with buffer of size one and,

$\{(1\ loading\ job), (1\ unloading\ job), (0\ loading\ job\ /$
$0\ unloading\ job), (b1_l), (b1_u), (b2_l), (b2_u)\}$ for the system with buffer of size two, and compute the time to load and unload a container for each of the two models. By this method, the number of states is reduced to 5 for the system with buffer of size one, and 7 for the system with buffer of size two. With the calculation of loading time and unloading time for a single container, the processing times for the total number of containers could be computed.

Example

You might wonder the idea of using the above states for computing the loading and unloading times, and to give an intuition to this, look into Table 1 , which consists of the unloading and loading jobs as represented by (N_u, N_l). Starting from the cell with value (3,3), a transition is made to the next immediate cell situated either to its right or to its bottom. In this fashion the cell with value (0,0) is reached. The arrows represent the possible set of transitions and the transitions have constant rates throughout for any number of loading and unloading jobs. So, if we consider the cell with value (1,1), the transition rate to (1,0) is the same as the transition rate to (0,1), and so is the case for any other state transitions that you could possibly imagine.

Thus, by computing the time taken for reaching the next immediate cell, the total time to process the entire set of jobs could be determined. This idea is mathematically easier to compute but significant amount of information is lost by averaging the variable parameters which in our case were the dropping and picking times. But fortunately, the simulation models that have been developed earlier would help us know the information that have been lost through error space. And I defer further discussion on this to next chapter.

(3,3) ↓ →	↓ (3,2) →	↓ (3,1) →	(3,0) ↓
↓ (2,3) →	↓ (2,2) ↓ →	↓ (2,1) ↓ →	↓ (2,0) ↓
(1,3) ↓ →	↓ (1,2) ↓ →	↓ (1,1) ↓ →	↓ (1,0) ↓
(0,3) ↓ →	↓ (0,2) →	↓ (0,1) →	(0,0) ↓

Table 1: State space for 3 loading and 3 unloading containers

3.2.5 Modeling the process:

To give a mathematical idea for the aggregated state space models with one loading and one unloading job, let's consider,

$$\lambda_q = \frac{1}{(2 * quaycrane\ time) + average\ picking\ time\ of\ one\ container\ from\ vessel}$$

$$\lambda_y = \frac{1}{(2 * yardcrane\ time) + average\ picking\ time\ of\ one\ container\ from\ yard}$$

$$\mu_q = \frac{1}{(2 * quaycrane\ time) + average\ dropping\ time\ of\ one\ container\ on\ vessel}$$

$$\mu_y = \frac{1}{(2 * yardcrane\ time) + average\ dropping\ time\ of\ one\ container\ on\ yard}$$

Where,

quaycrane_time is the time taken by quay crane to move from platform to vessel or from vessel to platform and does not take the pickup and drop times into account.

yardcrane_time is the time taken by quay crane to move from platform to yard or from yard to platform and does not take the pickup and drop times into account.

quaycrane_time and *yardcrane_time* are constant throughout the entire set of operations.

The notations for the states of the system with buffer of size one are,

1- $1\ unloading\ job$
2- 1 *loading job*
3- b_l
4- b_u
5- Absorption state (0 loading job/0 unloading job)

Similarly, the notations for the states of system with buffer of size two are,

1- $1\ unloading\ job$
2- 1 *loading job*

3- b_{l1}

4- b_{l2}

5- b_{u1}

6- b_{u2}

7- Absorption state (0 loading job/0 unloading job)

For the system with buffer of size one, the aggregated transition rate matrix would be,

$$
\begin{array}{c} \text{States} \\ \Lambda_{buffer1} = \end{array}
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}
\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{bmatrix} -\lambda_q & 0 & 0 & \lambda_q & 0 \\ 0 & -\lambda_y & \lambda_y & 0 & 0 \\ 0 & 0 & -\mu_q & 0 & \mu_q \\ 0 & 0 & 0 & -\mu_y & \mu_y \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}
$$

The mean time to get absorbed into the state 5 could be solved by the following linear equations [27],

$$0 = a_{1,5} + \lambda_{1,1}a_{1,5}m_{1,5} + \lambda_{1,2}a_{2,5}m_{2,5} + \lambda_{1,3}a_{3,5}m_{3,5} + \lambda_{1,4}a_{4,5}m_{4,5}$$

$$0 = a_{2,5} + \lambda_{2,1}a_{1,5}m_{1,5} + \lambda_{2,2}a_{2,5}m_{2,5} + \lambda_{2,3}a_{3,5}m_{3,5} + \lambda_{2,4}a_{4,5}m_{4,5}$$

$$0 = a_{3,5} + \lambda_{3,1}a_{1,5}m_{1,5} + \lambda_{3,2}a_{2,5}m_{2,5} + \lambda_{3,3}a_{3,5}m_{3,5} + \lambda_{3,4}a_{4,5}m_{4,5}$$

$$0 = a_{4,5} + \lambda_{4,1}a_{1,5}m_{1,5} + \lambda_{4,2}a_{2,5}m_{2,5} + \lambda_{4,3}a_{3,5}m_{3,5} + \lambda_{4,4}a_{4,5}m_{4,5}$$

$a_{i,j}$ is the absorption probability from transient state i to absorbing state j

$\lambda_{i,j}$ is the rate of reaching state i from state j

$m_{i,j}$ is the mean time to reach absorbing state j from transient state i

Further substituting the rates would lead into,

$$0 = 1 - \lambda_q m_{1,5} + \lambda_q m_{4,5}$$

$$0 = 1 - \lambda_y m_{2,5} + \lambda_y m_{3,5}$$

$$0 = 1 - \mu_q m_{3,5}$$

$$0 = 1 - \mu_y m_{45}$$

For the system with buffer size of size two, the aggregated transition rate matrix would be,

$$\Lambda_{buffer2} = \begin{array}{c} \text{States} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} -2\lambda_q & 0 & 0 & 0 & \lambda_q & \lambda_q & 0 \\ 0 & -2\lambda_y & \lambda_y & \lambda_y & 0 & 0 & 0 \\ 0 & 0 & -\mu_q & 0 & 0 & 0 & \mu_q \\ 0 & 0 & 0 & -\mu_q & 0 & 0 & \mu_q \\ 0 & 0 & 0 & 0 & -\mu_y & 0 & \mu_y \\ 0 & 0 & 0 & 0 & 0 & -\mu_y & \mu_y \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Similarly, the mean time to get absorbed could be calculated as,

$0 = a_{1,7} + \lambda_{1,1}a_{1,7}m_{1,7} + \lambda_{1,2}a_{2,7}m_{2,7} + \lambda_{1,3}a_{3,7}m_{3,7} + \lambda_{1,4}a_{4,7}m_{4,7} + \lambda_{1,5}a_{5,7}m_{5,7} + \lambda_{1,6}a_{6,7}m_{6,7}$

$0 = a_{2,7} + \lambda_{2,1}a_{1,7}m_{1,7} + \lambda_{2,2}a_{2,7}m_{2,7} + \lambda_{2,3}a_{3,7}m_{3,7} + \lambda_{2,4}a_{4,7}m_{4,7} + \lambda_{2,5}a_{5,7}m_{5,7} + \lambda_{2,6}a_{6,7}m_{6,7}$

$0 = a_{3,7} + \lambda_{3,1}a_{1,7}m_{1,7} + \lambda_{3,2}a_{2,7}m_{2,7} + \lambda_{3,3}a_{3,7}m_{3,7} + \lambda_{3,4}a_{4,7}m_{4,7} + \lambda_{3,5}a_{5,7}m_{5,7} + \lambda_{3,6}a_{6,7}m_{6,7}$

$0 = a_{4,7} + \lambda_{4,1}a_{1,7}m_{1,7} + \lambda_{4,2}a_{2,7}m_{2,7} + \lambda_{4,3}a_{3,7}m_{3,7} + \lambda_{4,4}a_{4,7}m_{4,7} + \lambda_{4,5}a_{5,7}m_{5,7} + \lambda_{4,6}a_{6,7}m_{6,7}$

$0 = a_{5,7} + \lambda_{5,1}a_{1,7}m_{1,7} + \lambda_{5,2}a_{2,7}m_{2,7} + \lambda_{5,3}a_{3,7}m_{3,7} + \lambda_{5,4}a_{4,7}m_{4,7} + \lambda_{5,5}a_{5,7}m_{5,7} + \lambda_{5,6}a_{6,7}m_{6,7}$

$0 = a_{6,7} + \lambda_{6,1}a_{1,7}m_{1,7} + \lambda_{6,2}a_{2,7}m_{2,7} + \lambda_{6,3}a_{3,7}m_{3,7} + \lambda_{6,4}a_{4,7}m_{4,7} + \lambda_{6,5}a_{5,7}m_{5,7} + \lambda_{6,6}a_{6,7}m_{6,7}$

$a_{i,j}$ is the absorption probability from transient state i to absorbing state j

$\lambda_{i,j}$ is the rate of reaching state i from state j

$m_{i,j}$ is the mean time to reach absorbing state j from transient state i

Substituting the rate values into these equations would yield,

$$0 = 1 - 2\lambda_q m_{1,7} + \lambda_q m_{5,7} + \lambda_q m_{6,7}$$

$$0 = 1 - 2\lambda_y m_{2,7} + \lambda_y m_{3,7} + \lambda_y m_{4,7}$$

$$0 = 1 - \mu_q m_{3,7}$$

$$0 = 1 - \mu_q m_{4,7}$$

$$0 = 1 - \mu_y m_{5,7}$$

$$0 = 1 - \mu_y m_{6,7}$$

From the above set of equations,

$m_{1,5}$- Mean time to unload a job in the system with buffer of size one

$m_{2,5}$- Mean time to load a job in the system with buffer of size one

$m_{1,7}$- Mean time to unload a job in the system with buffer of size two

$m_{2,7}$- Mean time to load a job in the system with buffer of size two

The average time to load or unload a job is,

$$\tau = \frac{m_{1,5}+m_{2,5}}{2}, \text{ for system with buffer size one}$$

$$\tau = \frac{m_{1,7}+m_{2,7}}{2}, \text{ for system with buffer size two}$$

Thus, the total processing time would be,

$$total\ processing\ time = \tau * (N_u + N_l)$$

This method of aggregating the statespace with one loading and one unloading job has improved the computational efficiency.

CHAPTER 4

EMPIRICAL ANALYSIS

4.1 Experimental settings

The experiments have been carried out for the simulation and analytical model for the system with platform of size one and for the system with platform of size two in Python using PyCharm environment. The objective of the experiments was to identify the parameters that contribute significantly to the makespan and analyze the behavior of error space thereby gaining insights on the performance of the analytical models with aggregated statespace. As described earlier, makespan is the time required to complete all jobs [25]. The parameters of interest in this study are the number of unloading jobs, number of loading jobs, the quay crane time, yard crane time, the location of containers on vessel, and the location of containers on yard. The experiments were conducted using the following factor levels,

<table>
<tr><th>S.No</th><th>Parameters</th><th>Parameter levels</th><th>Description</th></tr>
<tr><td>1</td><td>Number of unloading jobs</td><td>2500, 6000</td><td></td></tr>
<tr><td>2</td><td>Number of loading jobs</td><td>2500, 6000</td><td></td></tr>
<tr><td>3</td><td>Quay Crane Time[time units]</td><td>150, 300</td><td rowspan="2"></td></tr>
<tr><td>4</td><td>Yard Crane Time[time units]</td><td>150, 300</td></tr>
<tr><td>5</td><td>Location of containers on vessel[Nominal factor]</td><td>1, 2</td><td rowspan="2">1 represents 10 containers across the bays and 10 containers across the stacks
2 represents 10 containers across the rows and 10 containers across the stacks</td></tr>
<tr><td>6</td><td>Location of containers on yard[Nominal factor]</td><td>1, 2</td></tr>
</table>

Table 2: Configuration details of the experiments

A total of 64 runs were performed for each set of experiments under a constant strategy of picking and dropping containers represented by 1 for simulation model and 0 for analytical model in the program.

The experimental details for each of the models are mentioned in Appendix D and E.

4.2 Statistical Analysis

The results constitute the makespan from the simulation models and the error space due to the differences in simulation and analytical models. The analysis which was done in JMP software shows the distribution of the makespan and error space across different parameter settings, and the graphs depict the behavior of the makespan and error space in different parameter regions. Based on the behavior exhibited by the results, inferences are drawn reflecting the parameter regions taken into consideration.

4.2.1 Simulation model analysis

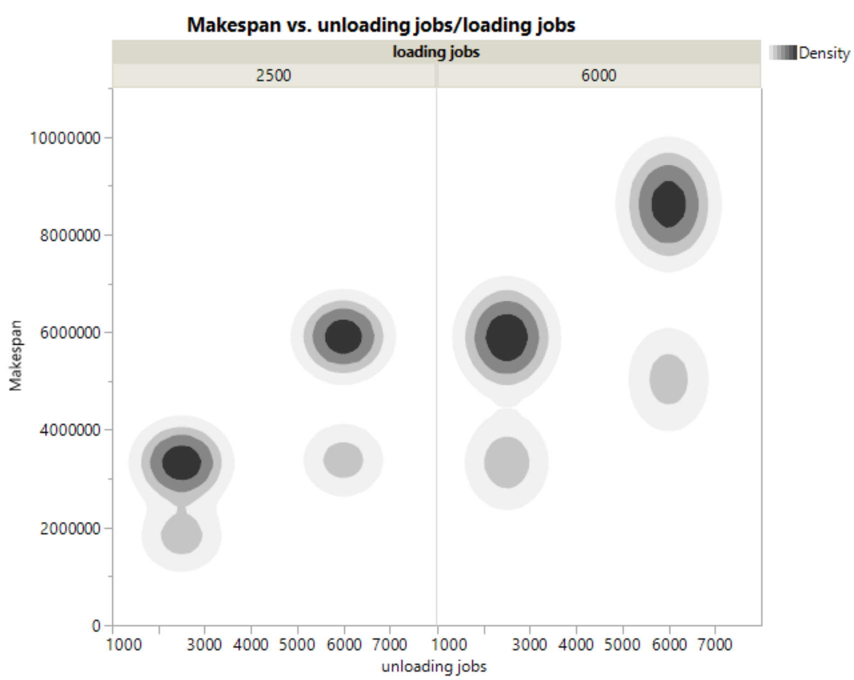


Figure 8(a)

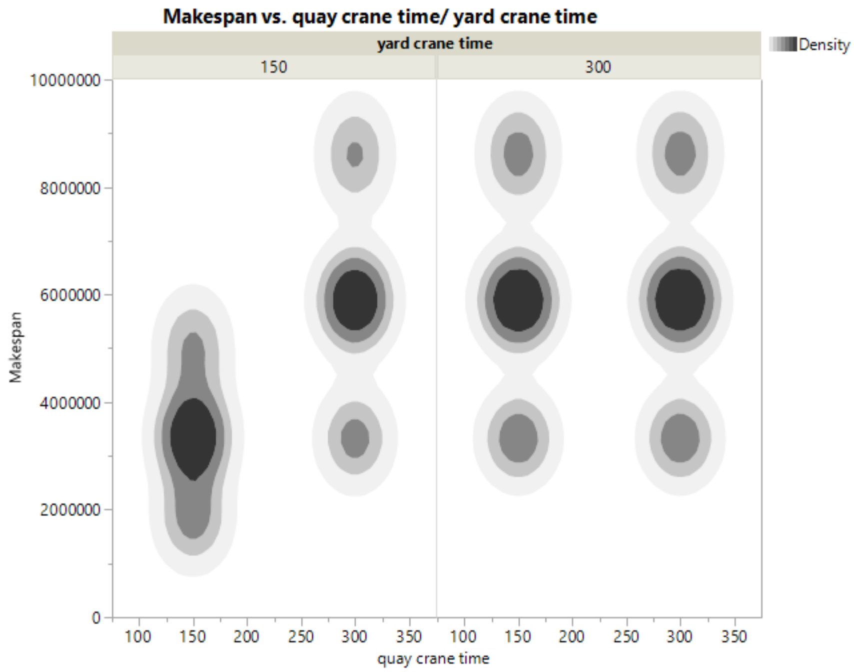


Figure 8(b)

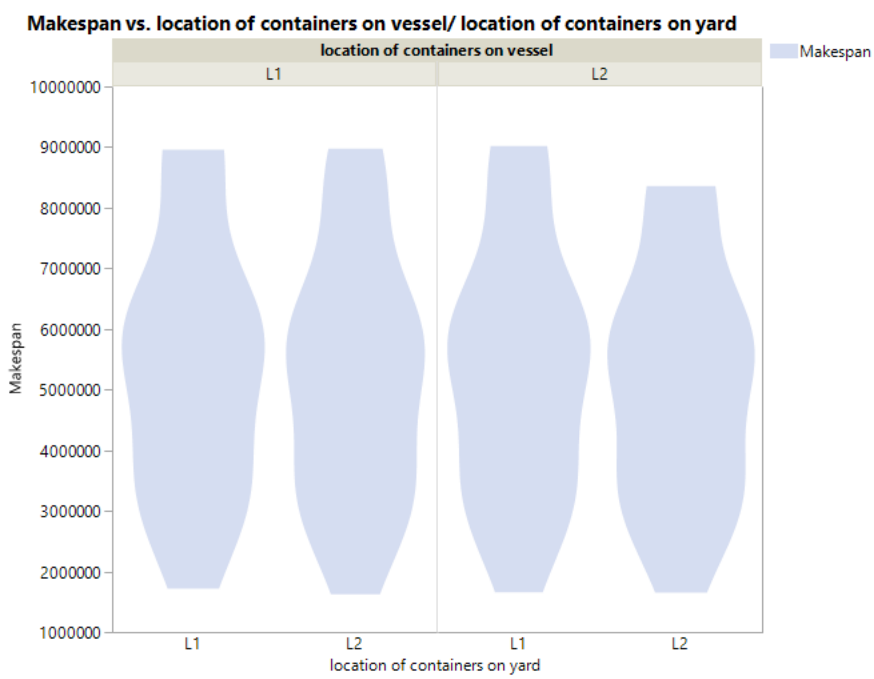


Figure 8(c)

Figure 8: Makespan distribution of simulation model of platform of size one system

Figure 8(a) shows that as the total number of containers which constitute the number of loading jobs and number of unloading jobs increase, the makespan increases and whenever they decrease, the makespan decreases. And for a particular set of parameter settings, the figure shows darker and lighter regions which indicate that the distribution of makespan is bimodal. The darker regions are the ones in which more number of makespan observations fall into compared to that of the lighter regions. This behavior can be attributed to the contribution of other parameter values which affect the makespan values. So, when at a particular parameter level of number of unloading jobs and number of loading jobs, the other parameters of our experiments vary.

S.No	Parameter levels of yard crane time and quay crane time
1	150,150
2	150,300
3	300,150
4	300,300

Table 3: Parameter levels of yard crane time and quay crane time

Table 3 indicates the different parameter levels of yard crane time and quay crane time that vary at a particular parameter level of number of loading jobs and number of unloading jobs. As the weaker link sets the pace of the process, either of the cranes sets the pace of the process depending on their times. From table 3, whenever quay crane time and yard crane time are equal resulting from both of them taking values of either 150 time units or 300 time units, both the cranes equally set the pace of the operations. But in the case of unequal times which result in the quay crane time being 150 time units and yard crane time being 300 units, or quay crane time being 300 time units and yard crane time being 150 time units, the weaker link sets the pace. We can see from table 3 that there are three set of parameters settings in which atleast one of the time is 300 time units. These three parameter settings have contributed to the darker region, and only one parameter setting has contributed to the lighter region. Hence, the distribution of the makespan in Figure 8(a) has resulted in bimodalities across the different parameter settings of the number of loading jobs and number of unloading jobs resulting in darker and lighter regions.

Figure 8(b) shows that as either of the yard crane time or quay crane time takes 300 time units, the makespan increases, or else it decreases. An observation into the figure 8(b) reveals that the distribution is trimodal. A deeper exploration into the makespan values for a particular set of yard crane time and quay crane time reveals that the number of loading and number of unloading jobs influence the makespan values.

S.No	Parameter levels of number of loading jobs and number of unloading jobs	Total Number of containers to be processed
1	2500,2500	5000
2	2500,6000	8500
3	6000,2500	8500
4	6000,6000	12000

Table 4: Parameter levels of number of loading jobs and number of unloading jobs

Table 4 indicates the total number of containers which vary at a particular parameter level of yard crane time and quay crane time. From table 4, one of the parameter settings of the number of loading and number of unloading jobs has a total of 5000 containers, two other parameter settings of the number of loading and number of unloading jobs have a total of 8500 containers, and the remaining one parameter setting of number of loading jobs and number of unloading jobs has a total of 12000 containers. As more number of containers take a higher makespan compared to less number of containers, the distribution in figure 8(b) has resulted in a trimodal one, where the darker regions are because of the contribution of the two parameter settings resulting in a total of 8500 containers, and the lighter ones are due to the parameter settings that result in a total of either 5000 containers or 12000 containers. Also, in figure 8(b), as the quay crane time and yard crane time each take 150 time units, the distribution is not a trimodal but a unimodal with a higher variability, and the reason for this variability is due to the number of containers that we have discussed.

Figure 8(c) shows the spread of the makespan at a particular parameter setting of location of containers on vessel and on yard which indicate the contribution of the other parameters that we have discussed earlier. Apart from this, figure 8(c) does not give any meaningful insights, hence the parameters of location of containers on vessel and on yard were not considered in our earlier inferences of figure 8(a) and figure 8(b).

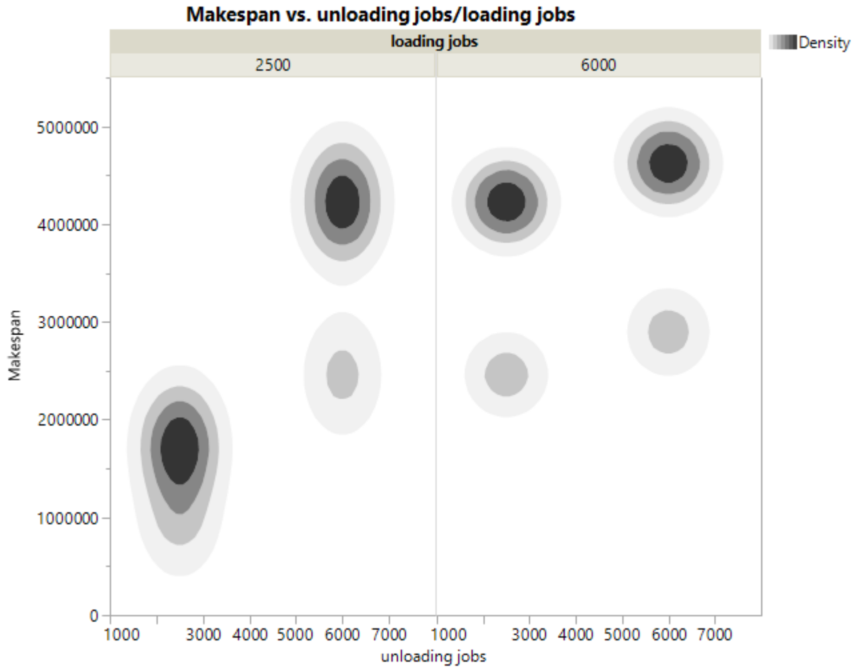


Figure 9(a)

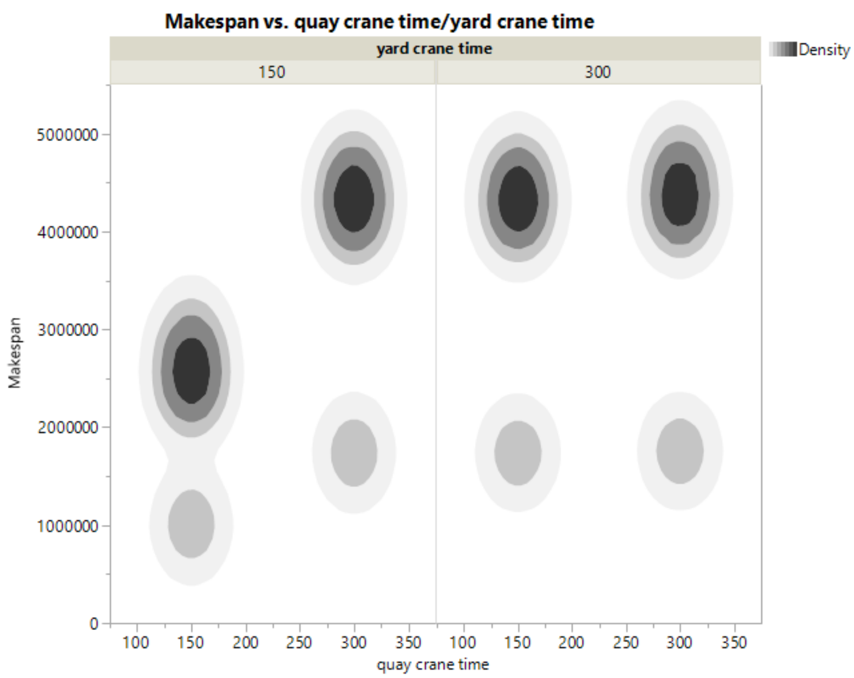


Figure 9(b)

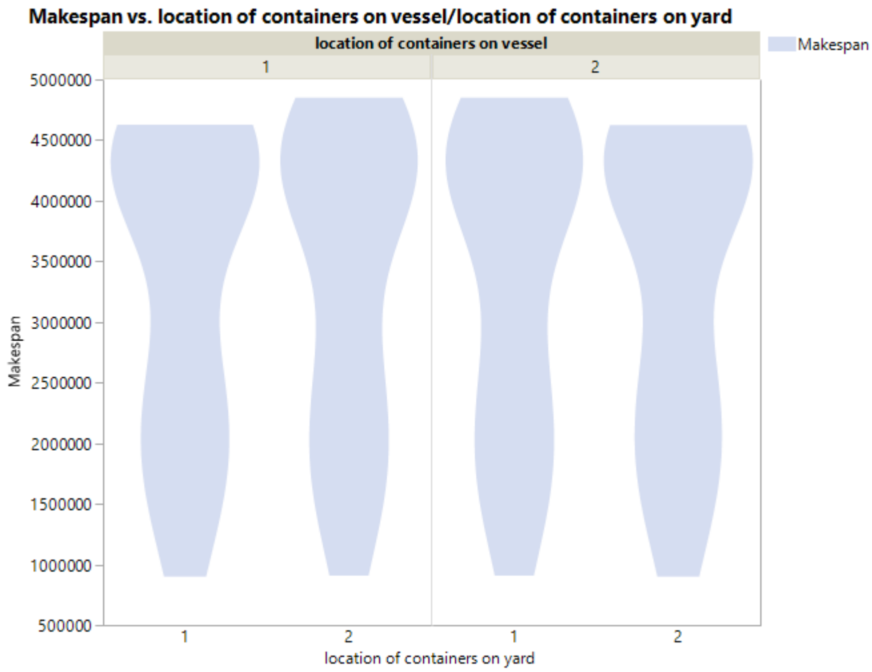


Figure 9(c)

Figure 9: Makespan distribution of simulation model of platform of size two system

Figure 9(a) shows that the distribution of the makespan of the simulation model with platform of size two is multi modal as well. The distribution has resulted in darker and lighter regions, and this is again attributed to the yard crane time and quay crane time which gets varied throughout resulting in this kind of distribution. But as the number of unloading jobs and number of loading jobs take 2500 each, the distribution is unimodal whereas the variability is high. The higher variability is due to the yard crane time and quay crane time which gets varied throughout resulting in higher makespan values when atleast either of the crane time is 300 units, and resulting in lower makespan values for rest of the parameter levels.

Figure 9(b) shows the distribution of makespan of the simulation model with platform of size two which is multi modal as well. In the earlier case, for simulation model with platform of size one, the distribution was tri modal owing to different set of total number of containers. But in this case, the distribution is bi modal. On a closer look into the figure 9(b) reveals that the variability around the higher mode value is higher than the variability around the lower mode value, and this higher variability is attributed to the different set of total number of containers, which in our case are 5000, 8500, and 12000 containers. So, the lower mode value occurs when the total number of containers are equal to 5000, and from table 4, there is only one parameter setting resulting in a total number of containers of 5000, thereby leading to a lighter region with a comparatively lesser variability. On the other hand, the higher variability is due to the rest of the parameter settings of table 4 which takes into account the total number of containers of 8500 and 12000. So, for total number of containers of 12000, the lower region of makespan values overlap with the higher region of makespan values for total number of containers of 8500. This overlap has resulted in a higher variability region rather than getting distributed around two different modes distinctly. The reason for this overlap can be attributed to the location of containers on vessel and on yard. When the parameter level of location of containers on vessel and on yard is 1, the makespan increases and decreases when the parameter level of containers on vessel and on yard is 2. So, this variability in the makespan for different parameter levels of containers on vessel and on yard could have resulted in the higher variability of darker region. And adding to this, the makespan in this system is lower than that of the system with buffer of size one, and this is due to the contribution of increased platform size.

Figure 9(c) shows the distribution of the makespan across different levels of the number of containers on vessel and on yard, and the spread signifies the variability of the makespan for different levels of other parameter settings that we have considered.

4.2.2 Error space analysis

Figure 10(a)

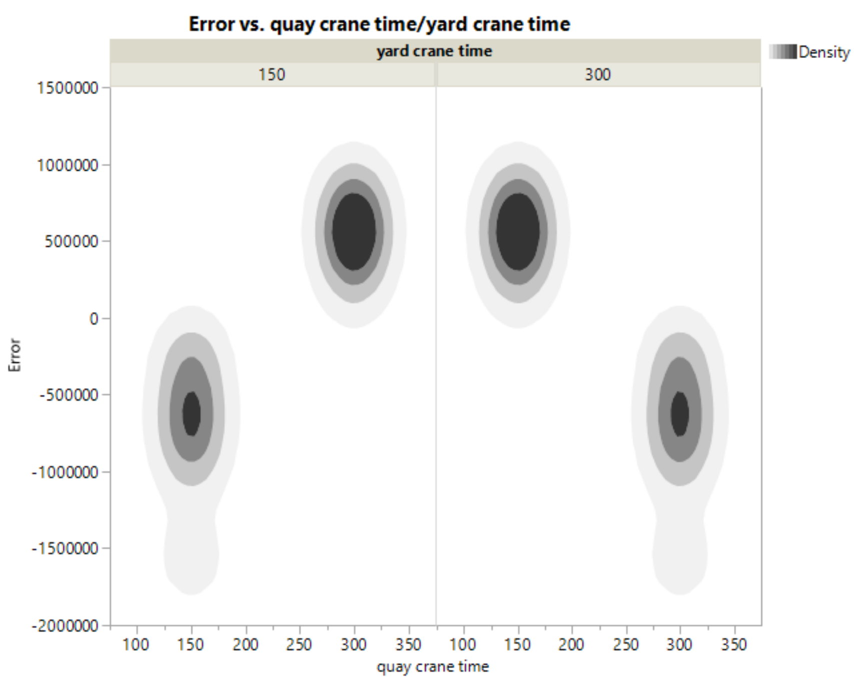


Figure 10(b)

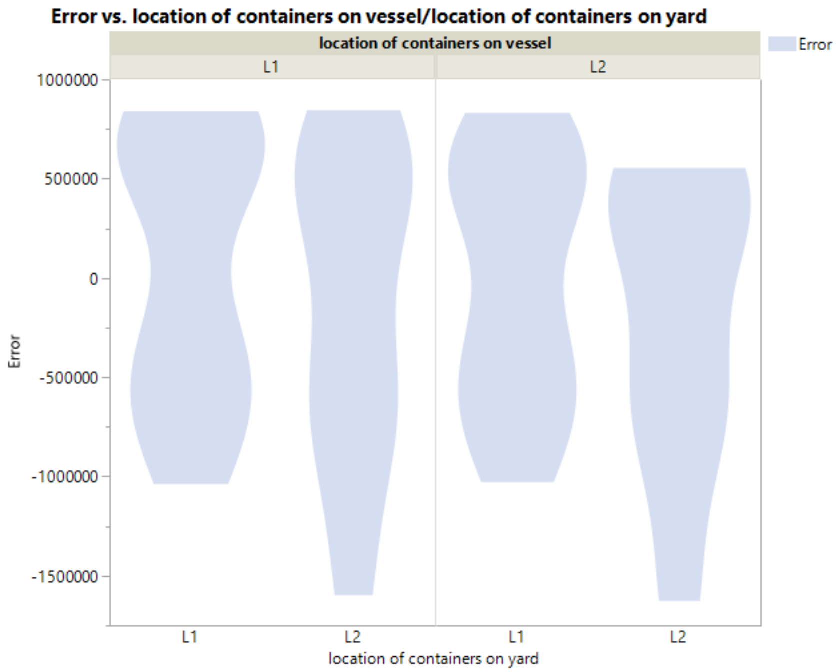


Figure 10(c)

Figure 10: Distribution of error space of system with platform of size one

Figure 10(a) shows the error space for the system with buffer of size one across different levels of number of unloading jobs and number of loading jobs. The distribution is bimodal in this case, and we could see that the distribution is even around the two set of mode values. This reflects that the analytical model does produce results in consistent with the simulation model for all the parameter levels.

Figure 10(b) reveals an interesting observation which makes us realize that whenever the factor levels of yard crane time and quay crane time were both 150 time units or 300 time units, the analytical model overestimated the makespan of its corresponding simulation model, and whenever one of the crane's time is 150 time units and the other crane's time is 300 units, the analytical model underestimated the makespan of its corresponding simulation model.

Figure 10(c) is similar to the earlier distribution of the simulation model, and the variability is throughout, reflecting the error space for different regions of yard crane time and quay crane time, and for different regions of number of loading jobs and number of unloading jobs.

The type of distributions from figure 10(a) and figure 10(b) could be attributed to the way the analytical modeling works. If you recollect the discussion in previous chapter, the analytical model has been aggregated to a single loading job and a single unloading job. The makespan is the average completion time of both the jobs for the two different starting states multiplied by the total number of containers to be processed. And these starting states are the commencement of the process by picking a loading job first or the commencement of the process by picking an unloading job first. Whenever both the crane times are either 150 time units or 300 time units, the completion times to process single loading job and single unloading job irrespective of the starting states are same for equal number of actual loading and unloading jobs to be processed, and different for different number of actual loading and unloading jobs to be processed. The actual loading and unloading jobs in our case are 2500 and 6000, which are the parameter levels. Whenever one of the crane's time is 150 time units and the other crane's time is 300 units, the completion time to process single loading job and single unloading job from one starting state is different from the other, and the possible causes to this could be the aggregated state space where the number of states have been reduced to 5, or to the exponential distribution of service times which the continuous time markov process uses.

Figure 11(a)

Figure 11(b)

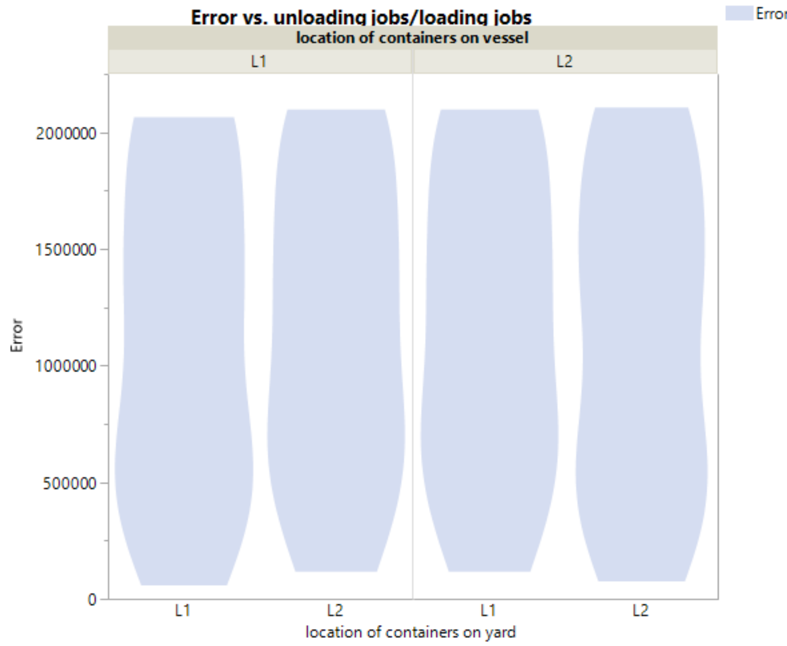


Figure 11(c)

Figure 11: Distribution of error space of system with platform of size two

Figure 11(a) shows the distribution of the error space of the system with platform of size two across different parameter levels of number of loading jobs and number of unloading jobs. The distribution is multimodal for two set of parameter level regions, bimodal for one parameter region and unimodal for other parameter level region. With respect to multimodality, one region is comparatively darker than the other reflecting the earlier inference that we had drawn for the simulation model related to the contribution of the yard crane time and quay crane time. Also, the behavior of the analytical model influences the error space which we will discuss shortly.

Figure 11(b) shows the distribution of the error space of the system with platform of size across different parameter levels of yard crane time and quay crane time. And this distribution is trimodal in two parameter level regions, and bimodal with high variability in the other parameter level regions. With respect to trimodality, the same explanation that we gave for simulation model considering the total number of containers is applicable here too. An additional consideration is the behavior of the analytical model which influences the distribution of error space. And for bimodality, the overlapping explanation of makespan given earlier for the simulation model is applicable apart from the consideration of the influence of analytical model.

In the case of system with platform of size two, the analytical model underestimated the makespan of its corresponding simulation model for all factor levels. This is evident from the region of error which is greater than zero for all factor levels. The reasons for such kind of behavior could be attributed to the aggregated state space where the number of states have been reduced to 7, or to the distribution of the service times which are exponential.

Remarks

- As the makespan depends on the crane time that takes higher amount of time, a reduction in this time would lead to a reduction in the makespan. And with respect to the location of containers on yard and on vessel, significant inferences could not be drawn based on the factor levels taken into consideration.
- The error space comprises random error component and model error component. The model error component is evident from the underestimation and overestimation of the results of the simulation model by the analytical model. For the case of the system with buffer of size one, whenever the factor levels of yard crane time and quay crane time were both 150 time units or 300 time units, the analytical model overestimated the makespan of its corresponding simulation model, and whenever one of the crane's time is 150 time units and the other crane's time is 300 units, the analytical model underestimated the makespan of its corresponding simulation model. On the other hand, for the system with buffer of

size two, the analytical model underestimated the makespan of its corresponding simulation model for all factor levels, and the reasons for the analytical model to exhibit such behavior could be attributed to the aggregated state space method or to the exponential distribution of service times which the markov process uses.

- With respect to error space prediction, as the simulation model is computationally slower than the analytical model, the error space is necessary for us to predict the makespan of the simulation model using the analytical model. And the way the distribution has turned out for different factor levels of the simulation and analytical models, it is important to consider the same set of factor levels for the both the models to predict an error space that would be more accurate rather than a generalized error space. Hence, predicting the error space distinctly for separate parameters would bridge the gap between the simulation model and the analytical model.

CHAPTER 5

CONCLUSION

This research gave deeper insights to the problem that we had considered. Based on the results, we could identify the importance of the number of containers and speed of the cranes in determining the makespan. This signifies the importance of these parameters which cannot be overlooked as the direction of change of these parameters impact the makespan. Also, the makespan depends on the crane that is slower, and in this regard, any improvement to be done on the cranes with respect to their speed has to be on both of them in such a way that they operate at the same levels of speed, otherwise the improvement will not yield any significant benefits with respect to makespan.

With respect to the model, the analytical model is inaccurate compared to the simulation model but computationally faster. For the system with buffer of size one, as the factor levels of yard crane time and quay crane time are both 150 time units or 300 time units, the analytical model overestimates the makespan of its corresponding simulation model, and as one of the crane's time is 150 time units and the other crane's time is 300 units, the analytical model underestimates the makespan of its corresponding simulation model. On the other hand, for the system with buffer of size two, the analytical model underestimates the makespan of its corresponding simulation model for all factor levels, and the reasons for the analytical model to exhibit such behavior could be attributed to the aggregated state space method or to the exponential distribution of service times which the markov process uses. On the other hand, the simulation model is highly accurate but computationally inefficient. Owing to these attributes, the simulation could be used as a surrogate model to aid in the prediction of error space. And this error space coupled with the analytical model could yield results that are faster and accurate.

Going forward, the research can be explored into two phases. One is the estimation of the error space distribution of the simulation and analytical models for all parameter levels distinctly. The parameter levels to be considered should be a multiple combination of containers to be processed, crane times, and the platform sizes. So, given a certain set of input parameters, this error space distribution would aid in the prediction of the accurate makespan coupled with the results of the analytical models. The second phase is the identification of optimal parameters through simulation optimization methods. Based on the prediction of error space for the parameter levels, the parameter region that yields lower makespan for certain fixed levels of parameters can be obtained, and this can be coupled into the simulation optimization methods to identify the optimal running of the system which should involve additional parameters like pick policy, drop policy as well into the modeling.

REFERENCES

[1]
F. Contu, A. Di Febrraro, and N. Sacco, "A model for performance evaluation and sensitivity analysis of seaport container terminals," in *Proceedings of the 18th World Congress*, Milano, Italy.

[2]
A. Krishnamurthy, R. Suri, and M. Vernon, "A New Approach for Analyzing Queuing Models of Material Control Strategies in Manufacturing Systems."

[3]
P. Canonaco, P. Legato, R. M. Mazza, and R. Musmanno, "A queuing network model for the management of berth crane operations."

[4]
W. Young Yun and Y. Seok Choi, "A simulation model for container-terminal operation analysis using an object-oriented approach."

[5]
S. Saini, D. Roy, and R. de Koster, "A stochastic model for the throughput analysis of passing dual yard cranes."

[6]
K. Ronald Studer, "A study of ship size and turnaround time in the port of vancouver," The University of British Columbia, Vancouver, Canada.

[7]
C. Bierwirth and F. Meisel, "A survey of berth allocation and quay crane scheduling problems in container terminals."

[8]
D. Roy, V. Dhingra, and R. de Koster, "A Two-level Stochastic Model to Estimate Vessel Throughput Time."

[9]
S.-K. Kang, H. Jung, I. H. Im, K.-Y. Chung, and J.-H. Lee, "Active Discrete Event Simulation using Probability Distribution of Shipbuilding Process."

[10]
A. S and T. K, "An Analysis on the Modeling of Container Terminal Operations."

[11]
G. Assadipour, G. Y. Ke, and M. Verma, "An analytical framework for integrated maritime terminal scheduling problems with time windows."

[12]
N. Saeed and O. I. Larsen, "Application of queuing methodology to analyze congestion: A case study of the Manila International Container Terminal, Philippines."

[13]
S. M. Easa, "APPROXIMATE QUEUEING MODELS FOR ANALYZING HARBOR TERMINAL OPERATIONS."

[14]
N. Hacegaba, "Bold solutions: how seaports can conquer congestion."

[15]
M.-H. Phan-Thi, K. Ryu, and K. Hwan Kim, “Comparing Cycle Times of Advanced Quay Cranes in Container Terminals,” vol. Industrial Engineering & Management Systems Vol 12, No 4, December 2013, pp.359–367.

[16]
I. Borovits and P. Ein-Dor, “COMPUTER SIMULATION OF A SEAPORT CONTAINER TERMINAL.”

[17]
D. C. Montgomery, *Design and analysis of experiments*. John Wiley & Sons, Inc.

[18]
S.-L. Chao and Y.-J. Lin, “Evaluating advanced quay cranes in container terminals.”

[19]
A. M. Law and Averill M. Law & Associates, Inc., “HOW TO BUILD VALID AND CREDIBLE SIMULATION MODELS,” in *Proceedings of the 2009 Winter Simulation Conference M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.*

[20]
Q. Zeng and Z. Yang, “Integrating simulation and optimization to schedule loading operations in container terminals.”

[21]
D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*. Hoboken: John Wiley & SonsIncorporated, 2012. Accessed May 3, 2018. ProQuest Ebook Central.

[22]
S. Ross, *Introduction to Probability models*, Eleventh. Elsevier.

[23]
S. Du, R. Xu, D. Huang, and X. Yao, “Markov modeling and analysis of multi-stage manufacturing systems with remote quality information feedback.”

[24]
V. Bhaskar and P. Lallement, “Modeling a supply chain using a network of queues.”

[25]
R. G. Askin and C. R. Standdridge, *Modeling and analysis of manufacturing systems*. John Wiley & Sons, Inc.

[26]
R. Mohamed, “Modeling and Optimization of Decision-Making Process During Loading and Unloading Operations at Container Port,” *Springerlink.com*.

[27]
J. J. Solberg, *Modeling random processes for engineers and managers*. John Wiley & Sons, Inc.

[28]
Q. Zeng, Z. Yang, and L. Lai, “Models and algorithms for multi-crane oriented scheduling method in container terminals.”

[29]
S. Kang, J. C. Medina, and Y. Ouyang, “Optimal operations of transportation fleet for unloading activities at container ports.”

[30]
K. Wu, L. F. McGinnis, and B. Zwart, "QUEUEING MODELS FOR SINGLE MACHINE MANUFACTURING SYSTEMS WITH INTERRUPTIONS," presented at the Proceedings of the 2008 Winter Simulation Conference.

[31]
M. Manitz, "Queueing-model based analysis of assembly lines with finite buffers and general service times."

[32]
M. Hess, S. Kos, and S. Hess, "QUEUING SYSTEM IN OPTIMIZATION FUNCTION OF THE PORT'S BULK UNLOADING TERMINAL."

[33]
"Review of Maritime Transport, 2017," presented at the United Nations Conference on Trade and Development.

[34]
A. M. Law, *Simulation modeling and analysis*, Fourth. McGrawHill.

APPENDIX A

EVENT GRAPH NOTATION DETAILS

S.no	Description	Notation
1	Event of quay crane to begin moving from vessel point to pick an unloading job (Initializing event1)	E_1
2	Event of yard crane to begin moving from yard point to pick a loading job (Initializing event2)	E_2
3	Event of quay crane to pick an unloading job from vessel after reaching the location of that unloading job	E_3
4	Event of yard crane to pick a loading job from yard after reaching the location of that loading job	E_4
5	Event of quay crane to reach the vessel point from platform	E_5
6	Event of quay crane to reach the vessel point after dropping a loading job on vessel	E_6
7	Event of quay crane to reach the platform from vessel point	E_7
8	Event of yard crane to reach the yard point from platform	E_8
9	Event of yard crane to reach the yard point after dropping an unloading job on yard	E_9
10	Event of yard crane to reach the platform from yard point	E_{10}
11	Event to drop an unloading job on platform by quay crane	E_{11}
12	Event to drop a loading job on platform by yard crane	E_{12}
13	Event to pick an unloading job from platform by yard crane	E_{13}
14	Event to pick a loading job from platform by quay crane	E_{14}
15	State of quay crane with respect to the job being processed	QC_j
16	State of yard crane with respect to the job being processed	YC_j
17	State of quay crane with respect to its position	QC_p
18	State of yard crane with respect to its position	YC_p
19	The number of unloading jobs on vessel	N_{uv}
20	The number of loading jobs on yard	N_{ly}
21	The number of platforms that are available	p
22	The number of loading jobs on platform	N_{up}
23	The number of unloading jobs on platform	N_{lp}
24	Time taken by quay crane to move from vessel point to platform, or from platform to vessel point	τ_{qc}
25	Time taken by yard crane to move from yard point to platform, or from platform to yard point	τ_{yc}
26	Time taken by quay crane to move from vessel point to unloading job's location for picking, or from unloading job's location to vessel point	τ_{qcp}
27	Time taken by quay crane to move from yard point to loading job's location for picking, or from loading job's location to yard point	τ_{ycp}
28	Time taken by quay crane to move from vessel point to loading job's location for dropping, or from loading job's location to vessel point	τ_{qcd}
29	Time taken by quay crane to move from yard point to unloading job's location for dropping, or from unloading job's location to yard point	τ_{ycd}
30	Total number of platforms	T

APPENDIX B

STATESET OF SYSTEM WITH BUFFER OF SIZE ONE

Stateset Number	N_u	N_l	S_q	S_b	S_y
1	$N_u, N_u - 1, .., 0$	$N_l, N_l - 1, .., 0$	0	0	0
2	$N_u, N_u - 1, .., 0$	$N_l - 1, N_l - 2, .., 0$	0	0	1
3	$N_u, N_u - 1, .., 0$	$N_l - 1, N_l - 2, .., 0$	0	1	0
4	$N_u, N_u - 1, .., 0$	$N_l - 1, N_l - 2, .., 0$	1	0	0
5	$N_u, N_u - 1, .., 0$	$N_l - 2, N_l - 3, .., 0$	0	1	1
6	$N_u, N_u - 1, .., 0$	$N_l - 2, N_l - 3, .., 0$	1	0	1
7	$N_u, N_u - 1, .., 0$	$N_l - 2, N_l - 3, .., 0$	1	1	0
8	$N_u, N_u - 1, .., 0$	$N_l - 3, N_l - 4, .., 0$	1	1	1
9	$N_u - 1, N_u - 2, .., 0$	$N_l, N_l - 1, .., 0$	2	0	0
10	$N_u - 1, N_u - 2, .., 0$	$N_l, N_l - 1, .., 0$	0	2	0
11	$N_u - 1, N_u - 2, .., 0$	$N_l, N_l - 1, .., 0$	0	0	2
12	$N_u - 2, N_u - 3, .., 0$	$N_l, N_l - 1, .., 0$	2	2	0
13	$N_u - 2, N_u - 3, .., 0$	$N_l, N_l - 1, .., 0$	2	0	2
14	$N_u - 2, N_u - 3, .., 0$	$N_l, N_l - 1, .., 0$	0	2	2
15	$N_u - 3, N_u - 4, .., 0$	$N_l, N_l - 1, .., 0$	2	2	2

APPENDIX C

STATESET OF SYSTEM WITH BUFFER SIZE TWO

Stateset Number	N_u	N_l	S_q	S_{b1}	S_{b2}	S_y
1	$N_u, N_u-1,..,0$	$N_l, N_l-1,..,0$	0	0	0	0
2	$N_u, N_u-1,..,0$	$N_l-1, N_l-2,..,0$	0	0	0	1
3	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	2	0	0	1
4	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	2	0	1	0
5	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	2	1	0	0
6	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	0	2	0	1
7	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	0	0	2	1
8	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	0	1	2	0
9	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	0	2	1	0
10	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	1	2	0	0
11	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	1	0	2	0
12	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	0	1	0	2
13	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	0	0	1	2
14	$N_u-1, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	1	0	0	2
15	$N_u-1, N_u-2,..,0$	$N_l, N_l-1,..,0$	0	0	0	2
16	$N_u, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	1	0	0	0
17	$N_u, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	0	1	0	0
18	$N_u, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	0	0	1	0
19	$N_u, N_u-2,..,0$	$N_l-1, N_l-2,..,0$	0	0	0	1
20	$N_u-1, N_u-2,..,0$	$N_l, N_l-1,..,0$	2	0	0	0
21	$N_u-1, N_u-2,..,0$	$N_l, N_l-1,..,0$	0	2	0	0
22	$N_u-1, N_u-2,..,0$	$N_l, N_l-1,..,0$	0	0	2	0
23	$N_u-2, N_u-3,..,0$	$N_l-1, N_l,..,0$	2	2	0	1
24	$N_u-2, N_u-3,..,0$	$N_l-1, N_l,..,0$	2	0	2	1
25	$N_u, N_u-1,..,0$	$N_l-2, N_l-3,..,0$	0	0	1	1
26	$N_u, N_u-1,..,0$	$N_l-2, N_l-3,..,0$	0	1	0	1
27	$N_u, N_u-1,..,0$	$N_l-2, N_l-3,..,0$	0	1	1	0
28	$N_u, N_u-1,..,0$	$N_l-2, N_l-3,..,0$	1	0	1	0
29	$N_u, N_u-1,..,0$	$N_l-2, N_l-3,..,0$	1	1	0	0
30	$N_u, N_u-1,..,0$	$N_l-3, N_l-4,..,0$	1	1	0	1
31	$N_u, N_u-1,..,0$	$N_l-3, N_l-4,..,0$	1	1	1	0
32	$N_u-2, N_u-3,..,0$	$N_l, N_l-1,..,0$	2	0	2	0
33	$N_u-2, N_u-3,..,0$	$N_l, N_l-1,..,0$	0	2	2	0
34	$N_u-2, N_u-3,..,0$	$N_l, N_l-1,..,0$	0	2	0	2
35	$N_u-2, N_u-3,..,0$	$N_l, N_l-1,..,0$	0	0	2	2
36	$N_u-3, N_u-3,..,0$	$N_l, N_l-1,..,0$	2	0	2	2
37	$N_u-3, N_u-3,..,0$	$N_l, N_l-1,..,0$	0	2	2	2
38	$N_u-3, N_u-3,..,0$	$N_l, N_l-1,..,0$	2	2	0	2
39	$N_u, N_u-1,..,0$	$N_l-2, N_l-3,..,0$	1	0	0	1
40	$N_u-2, N_u-1,..,0$	$N_l, N_l-1,..,0$	2	0	0	2
41	$N_u, N_u-1,..,0$	$N_l-3, N_l-4,..,0$	1	0	1	1
42	$N_u-2, N_u-3,..,0$	$N_l-1, N_l-2,..,0$	2	1	0	2
43	$N_u-2, N_u-3,..,0$	$N_l-1, N_l-2,..,0$	2	0	1	2
44	$N_u-1, N_u-2,..,0$	$N_l-2, N_l-3,..,0$	1	2	0	1
45	$N_u-1, N_u-2,..,0$	$N_l-2, N_l-3,..,0$	1	0	2	1

Stateset Number	N_u	N_l	S_q	S_{b1}	S_{b2}	S_y
46	$N_u - 1, N_u - 2, \ldots, 0$	$N_l - 2, N_l - 3, \ldots, 0$	2	1	0	1
47	$N_u - 1, N_u - 2, \ldots, 0$	$N_l - 2, N_l - 3, \ldots, 0$	2	0	1	1
48	$N_u - 1, N_u - 2, \ldots, 0$	$N_l - 2, N_l - 3, \ldots, 0$	1	1	0	2
49	$N_u - 1, N_u - 2, \ldots, 0$	$N_l - 2, N_l - 3, \ldots, 0$	1	0	1	2
50	$N_u - 2, N_u - 1, \ldots, 0$	$N_l - 1, N_l - 2, \ldots, 0$	1	2	0	2
51	$N_u - 2, N_u - 1, \ldots, 0$	$N_l - 1, N_l - 2, \ldots, 0$	1	0	2	2
52	$N_u - 1, N_u - 2, \ldots, 0$	$N_l - 2, N_l - 3, \ldots, 0$	0	1	1	2
53	$N_u - 2, N_u - 1, \ldots, 0$	$N_l - 1, N_l - 2, \ldots, 0$	1	2	2	0
54	$N_u - 2, N_u - 1, \ldots, 0$	$N_l - 1, N_l - 2, \ldots, 0$	0	2	1	2
55	$N_u - 2, N_u - 1, \ldots, 0$	$N_l - 1, N_l - 2, \ldots, 0$	0	1	2	2
56	$N_u - 1, N_u - 2, \ldots, 0$	$N_l - 2, N_l - 3, \ldots, 0$	1	1	2	0
57	$N_u - 1, N_u - 2, \ldots, 0$	$N_l - 2, N_l - 3, \ldots, 0$	1	2	1	0

APPENDIX D

EXPERIMENT DETAILS OF SYSTEM WITH BUFFER OF SIZE ONE

Total runs	Number of unloading jobs	Number of loading jobs	Yard Crane time [time units]	Quay Crane time [time units]	Location of containers on Vessel [nominal factor]	Location of containers on Yard [nominal factor]	Simulation result [time units]	Analytical result [time units]	Error [time units]
1	2500	2500	150	150	1	1	1865088	2100034.2	-234946.2
2	2500	2500	150	150	1	2	1791484	2100038.7	-308554.7
3	2500	2500	150	150	2	1	1865088	2100038.7	-234950.7
4	2500	2500	150	150	2	2	1790088	2100043.1	-309955.1
5	2500	2500	150	300	1	1	3364938	2850034.2	514903.8
6	2500	2500	150	300	1	2	3364938	2850038.7	514899.3
7	2500	2500	150	300	2	1	3289938	2850038.7	439899.3
8	2500	2500	150	300	2	2	3289938	2850043.1	439894.9
9	2500	2500	300	150	1	1	3364940	2850034.2	514905.8
10	2500	2500	300	150	1	2	3289940	2850038.7	439901.3
11	2500	2500	300	150	2	1	3364940	2850038.7	514901.3
12	2500	2500	300	150	2	2	3289940	2850043.1	439896.9
13	2500	2500	300	300	1	1	3365088	3600034.2	-234946.2
14	2500	2500	300	300	1	2	3290956	3600038.7	-309082.7
15	2500	2500	300	300	2	1	3365088	3600038.7	-234950.7
16	2500	2500	300	300	2	2	3290088	3600043.1	-309955.1
17	2500	6000	150	150	1	1	3475088	4016300.9	-541212.9
18	2500	6000	150	150	1	2	3295802	4016298.4	-720496.4
19	2500	6000	150	150	2	1	3370088	4016304.9	-646216.9
20	2500	6000	150	150	2	2	3191178	4016302.4	-825124.4
21	2500	6000	150	300	1	1	5954938	5291300.9	663637.1
22	2500	6000	150	300	1	2	5954898	5291298.4	663599.6
23	2500	6000	150	300	2	1	5774938	5291304.9	483633.1
24	2500	6000	150	300	2	2	5774898	5291302.4	483595.6
25	2500	6000	300	150	1	1	6017990	5291300.9	726689.1
26	2500	6000	300	150	1	2	5835990	5291298.4	544691.6
27	2500	6000	300	150	2	1	6017960	5291304.9	726655.1
28	2500	6000	300	150	2	2	5835960	5291302.4	544657.6
29	2500	6000	300	300	1	1	6025088	6566300.9	-541212.9
30	2500	6000	300	300	1	2	5845502	6566298.4	-720796.4
31	2500	6000	300	300	2	1	5920088	6566304.9	-646216.9
32	2500	6000	300	300	2	2	5740728	6566302.4	-825574.4
33	6000	2500	150	150	1	1	3405412	4016288.6	-610876.6
34	6000	2500	150	150	1	2	3253808	4016293.7	-762485.7
35	6000	2500	150	150	2	1	3486870	4016293.9	-529423.9
36	6000	2500	150	150	2	2	3362422	4016299	-653877
37	6000	2500	150	300	1	1	6017990	5291288.6	726701.4
38	6000	2500	150	300	1	2	6017960	5291293.7	726666.3

Total runs	Number of unloading jobs	Number of loading jobs	Yard Crane time [time units]	Quay Crane time [time units]	Location of containers on Vessel [nominal factor]	Location of containers on Yard [nominal factor]	Simulation result [time units]	Analytical result [time units]	Error [time units]
39	6000	2500	150	300	2	1	5835990	5291293.9	544696.1
40	6000	2500	150	300	2	2	5835960	5291299	544661
41	6000	2500	300	150	1	1	5954940	5291288.6	663651.4
42	6000	2500	300	150	1	2	5774940	5291293.7	483646.3
43	6000	2500	300	150	2	1	5954900	5291293.9	663606.1
44	6000	2500	300	150	2	2	5774900	5291299	483601
45	6000	2500	300	300	1	1	5955238	6566288.6	-611050.6
46	6000	2500	300	300	1	2	5803106	6566293.7	-763187.7
47	6000	2500	300	300	2	1	6036008	6566293.9	-530285.9
48	6000	2500	300	300	2	2	5910922	6566299	-655377
49	6000	6000	150	150	1	1	5316088	6300042	-983954
50	6000	6000	150	150	1	2	4769274	6300053.2	-1530779.2
51	6000	6000	150	150	2	1	5316088	6300053.2	-983965.2
52	6000	6000	150	150	2	2	4716088	6300064.5	-1583976.5
53	6000	6000	150	300	1	1	8916008	8100042	815966
54	6000	6000	150	300	1	2	8916008	8100053.2	815954.8
55	6000	6000	150	300	2	1	8316008	8100053.2	215954.8
56	6000	6000	150	300	2	2	8316008	8100064.5	215943.5
57	6000	6000	300	150	1	1	8916010	8100042	815968
58	6000	6000	300	150	1	2	8316010	8100053.2	215956.8
59	6000	6000	300	150	2	1	8916010	8100053.2	815956.8
60	6000	6000	300	150	2	2	8316010	8100064.5	215945.5
61	6000	6000	300	300	1	1	8916088	9900042	-983954
62	6000	6000	300	300	1	2	8368224	9900053.2	-1531829.2
63	6000	6000	300	300	2	1	8916088	9900053.2	-983965.2
64	6000	6000	300	300	2	2	8316088	9900064.5	-1583976.5

APPENDIX E

EXPERIMENT DETAILS OF SYSTEM WITH BUFFER OF SIZE TWO

Total runs	Number of unloading jobs	Number of loading jobs	Yard Crane time [time units]	Quay Crane time [time units]	Location of containers on Vessel [nominal factor]	Location of containers on Yard [nominal factor]	Simulation result [time units]	Analytical result [time units]	Error [time units]
1	2500	2500	150	150	1	1	989838	870077.59	119760.41
2	2500	2500	150	150	1	2	1020678	870077.59	150600.41
3	2500	2500	150	150	2	1	1020678	870088.84	150589.16
4	2500	2500	150	150	2	2	989568	870088.84	119479.16
5	2500	2500	150	300	1	1	1739838	1245077.6	494760.4
6	2500	2500	150	300	1	2	1739838	1245077.6	494760.4
7	2500	2500	150	300	2	1	1739568	1245088.8	494479.2
8	2500	2500	150	300	2	2	1739568	1245088.8	494479.2
9	2500	2500	300	150	1	1	1739838	1245077.6	494760.4
10	2500	2500	300	150	1	2	1739568	1245077.6	494490.4
11	2500	2500	300	150	2	1	1739838	1245088.8	494749.2
12	2500	2500	300	150	2	2	1739568	1245088.8	494479.2
13	2500	2500	300	300	1	1	1739838	1620077.6	119760.4
14	2500	2500	300	300	1	2	1770678	1620077.6	150600.4
15	2500	2500	300	300	2	1	1770678	1620088.8	150589.2
16	2500	2500	300	300	2	2	1739568	1620088.8	119479.2
17	2500	6000	150	150	1	1	2425950	1553498.2	872451.8
18	2500	6000	150	150	1	2	2582832	1553466.2	1029365.8
19	2500	6000	150	150	2	1	2354856	1553509.5	801346.5
20	2500	6000	150	150	2	2	2480712	1553477.5	927234.5
21	2500	6000	150	300	1	1	4225146	2190998.2	2034147.8
22	2500	6000	150	300	1	2	4225062	2190966.2	2034095.8
23	2500	6000	150	300	2	1	4120146	2191009.5	1929136.5
24	2500	6000	150	300	2	2	4120062	2190977.5	1929084.5
25	2500	6000	300	150	1	1	4218458	2190998.2	2027459.8
26	2500	6000	300	150	1	2	4285918	2190966.2	2094951.8
27	2500	6000	300	150	2	1	4218428	2191009.5	2027418.5
28	2500	6000	300	150	2	2	4285888	2190977.5	2094910.5
29	2500	6000	300	300	1	1	4226098	2828498.2	1397599.8
30	2500	6000	300	300	1	2	4382982	2828466.2	1554515.8
31	2500	6000	300	300	2	1	4155988	2828509.5	1327478.5
32	2500	6000	300	300	2	2	4280862	2828477.5	1452384.5
33	6000	2500	150	150	1	1	2425918	1553450.8	872467.2
34	6000	2500	150	150	1	2	2354596	1553464	801132
35	6000	2500	150	150	2	1	2582832	1553470.1	1029361.9
36	6000	2500	150	150	2	2	2480712	1553483.4	927228.6
37	6000	2500	150	300	1	1	4218458	2190950.8	2027507.2
38	6000	2500	150	300	1	2	4218428	2190964	2027464

Total runs	Number of unloading jobs	Number of loading jobs	Yard Crane time [time units]	Quay Crane time [time units]	Location of containers on Vessel [nominal factor]	Location of containers on Yard [nominal factor]	Simulation result [time units]	Analytical result [time units]	Error [time units]
39	6000	2500	150	300	2	1	4285918	2190970.1	2094947.9
40	6000	2500	150	300	2	2	4285888	2190983.4	2094904.6
41	6000	2500	300	150	1	1	4225146	2190950.8	2034195.2
42	6000	2500	300	150	1	2	4120146	2190964	1929182
43	6000	2500	300	150	2	1	4225062	2190970.1	2034091.9
44	6000	2500	300	150	2	2	4120062	2190983.4	1929078.6
45	6000	2500	300	300	1	1	4226098	2828450.8	1397647.2
46	6000	2500	300	300	1	2	4155988	2828464	1327524
47	6000	2500	300	300	2	1	4382982	2828470.1	1554511.9
48	6000	2500	300	300	2	2	4280862	2828483.4	1452378.6
49	6000	6000	150	150	1	1	2795838	2298101.8	497736.2
50	6000	6000	150	150	1	2	2998938	2298101.8	700836.2
51	6000	6000	150	150	2	1	2998938	2298130.5	700807.5
52	6000	6000	150	150	2	2	2794938	2298130.5	496807.5
53	6000	6000	150	300	1	1	4595838	3198101.8	1397736.2
54	6000	6000	150	300	1	2	4595838	3198101.8	1397736.2
55	6000	6000	150	300	2	1	4594938	3198130.5	1396807.5
56	6000	6000	150	300	2	2	4594938	3198130.5	1396807.5
57	6000	6000	300	150	1	1	4595838	3198101.8	1397736.2
58	6000	6000	300	150	1	2	4594938	3198101.8	1396836.2
59	6000	6000	300	150	2	1	4595838	3198130.5	1397707.5
60	6000	6000	300	150	2	2	4594938	3198130.5	1396807.5
61	6000	6000	300	300	1	1	4595838	4098101.8	497736.2
62	6000	6000	300	300	1	2	4798938	4098101.8	700836.2
63	6000	6000	300	300	2	1	4798938	4098130.5	700807.5
64	6000	6000	300	300	2	2	4594938	4098130.5	496807.5

APPENDIX F

DISCRETE EVENT SIMULATION MODEL CODE

```
import numpy as np

total_platform=input("Enter the number of platforms")
unload_containers=input("Enter the number of containers to be unloaded")
load_containers=input("Enter the number of containers to be loaded")
vesselloadjob_schedule_strategy=input("Enter loading job strategy at vessel
side")
yardloadjob_schedule_strategy=input("Enter loading job strategy at yard side")
vesselunloadjob_schedule_strategy=input("Enter unloading job strategy at vessel
side")
yardunloadjob_schedule_strategy=input("Enter unloading job strategy at yard
side")

length_vessel=input("enter the number of containers in vessel across length")
width_vessel=input("enter the number of containers in vessel across width")
height_vessel=input("enter the number of containers in vessel that could be
stacked including the bottom one")
length_yard=input("enter the number of containers in yard across length")
width_yard=input("enter the number of containers in yard across width")
height_yard=input("enter the number of containers in yard that could be stacked
including the bottom one")
vessel_load=np.zeros((length_vessel,width_vessel))
truck_load=np.zeros((length_yard,width_yard))
vessel_empty=np.zeros((2*length_vessel,width_vessel))
truck_empty=np.zeros((2*length_yard,width_yard))

for lengt in range(0,length_vessel):
    for widtt in range(0,width_vessel):
        vessel_load[lengt,widtt]=height_vessel

for lengt in range(0, length_yard):
    for widtt in range(0, width_yard):
        truck_load[lengt,widtt]=height_yard

for leng in range(0,length_vessel):
    for widt in range(0, width_vessel):
        vessel_empty[leng,widt]=2*height_vessel

for leng in range(0, length_yard):
    for widt in range(0, width_yard):
        truck_empty[leng,widt]=2*height_yard

#Variables in the process
previous_operation=1#If the previous iteration was an unloading process, this
value will take 1, if it was a loading process, it will take 2. for
initializing we set to 2
truck_unloadtime=1#Time it takes for yard crane in an unloading process to
place container on yard from platform. This value is kept for initialization
purpose
vessel_emptytime=150#Time it takes for quay crane to perform an empty-container
movement
truck_emptytime=150#Time it takes for yard crane to perform an empty-container
movement
vessel_loadtime=1#Time it takes for quay crane in a loading process to place
container on vessel from platform.This value is kept for initialization purpose
vessel_unloadtime=1#Time it takes for quay crane in an unloading process to
place container on platform from vessel.This value is kept for initialization
purpose
truck_loadtime=1#Time it takes for yard crane in a loading process to place
container on platform from vessel.This value is kept for initialization purpose
#total_containers is the total number of containers to be loaded & unloaded
platform_time=0#The minimum amount of time a container would wait on platform
```

```
is 20.
i=0# i and j are the indexes for location of containers on vessel to be
unloaded
j=0

k=25# k and j are indexes for empty location on vessel where containers would
be placed during loading operation
l=0

a=0# a and b are the indexes for location of containers on yard to be loaded
b=0

c=25# c and d are the indexes for empty location on yard where containers would
be placed during unloading operation
d=0
unload_time=0#Time taken for a container to get unloaded(includes the movement
time on cranes and waiting time on platform)
load_time=0#Time taken for a container to get loaded(includes the movement time
on cranes and waiting time on platform)
plat=[]
no_platform=total_platform#Total number of platforms that are available or
unoccupied
process_indicator=1#1 for unloading, 2 for loading
unload_count=0
load_count=0
time_to_reach_startposition_yardcrane=0
time_to_reach_startposition_quaycrane=0
time_to_reach_endposition_quaycrane=0
time_to_reach_endposition_yardcrane=0
time_quaycrane=0
time_yardcrane=0
unload_waiting=0
load_waiting=0
position_yardcrane=0
position_quaycrane=0
platform_handling_time_forcrane=0
container_handling_time_forcrane=0
alignment_time=0
time_emptyquaycrane=150
time_emptyyardcrane=150
position_yardcranedummy=2
position_quaycranedummy=2
time_to_reach_endposition_yardcrane_dummy=0
time_to_reach_endposition_quaycrane_dummy=0
time_to_reach_startposition_quaycrane_dummy=0
time_to_reach_startposition_yardcrane_dummy=0
dummy_123=0

simclock=0
iteration_range=(unload_containers+load_containers)*20
count_truck1=0
count_truck2=0
count_vessel1=0
count_vessel2=0

for process in range(0,iteration_range):
    print "position of yard crane",position_yardcrane
    print "position of quay crane",position_quaycrane
    print process

    def unload_quaycrane():
        global vessel_unloadtime,vessel_load,i,j,
```

```
truck_unloadtime,c,d,platform_time,unload_time,process_indicator,unloading_coun
t,previous_operation,position_quaycrane
        global
time_quaycrane,time_yardcrane,no_platform,unload_waiting,time_to_reach_endposit
ion_quaycrane,time,
platform_handling_time_forcrane,alignment_time,time_to_reach_endposition_quaycr
ane_dummy#process_indicator=1
        global
time_emptyquaycrane,container_handling_time_forcrane,count_vessel1#previous_ope
ration=1
        count_vessel1=count_vessel1+1

        dummy1=1

        if vesselunloadjob_schedule_strategy==1:#vertical and across j
            for i in range(0,length_vessel):
                for j in range(0,width_vessel):
                    for k in range(0,height_vessel):
                        if vessel_load[i,j]>0:
                            vessel_unloadtime = (2 * (i + 1)) + (2 * (j + 1)) +
(2 * vessel_load[i, j]) + time_emptyquaycrane
                            vessel_load[i,j]=vessel_load[i,j]-1
                            dummy1=2
                            break
                    if dummy1==2:
                        if vessel_load[i,j]==0:
                            vessel_empty[i, j] = vessel_load[i, j]
                        else:
                            pass
                        break
                if dummy1 == 2:
                    break

        elif vesselunloadjob_schedule_strategy==2:#horizontal and across j
            for i in range(0,length_vessel):
                for k in range(0,height_vessel):
                    for j in range(0,width_vessel):
                        if vessel_load[i,j]>0:
                            vessel_unloadtime = (2 * (i + 1)) + (2 * (j + 1)) +
(2 * vessel_load[i, j]) + time_emptyquaycrane
                            vessel_load[i,j]=vessel_load[i,j]-1
                            dummy1=2
                            break
                    if  dummy1==2:
                        if vessel_load[i,j]==0:
                            vessel_empty[i, j] = vessel_load[i, j]
                        else:
                            pass
                        break
                if dummy1 == 2:
                    break

        elif vesselunloadjob_schedule_strategy==3:#vertical across i
            for j in range(0, width_vessel):
                for i in range(0, length_vessel):
                    for k in range(0,height_vessel):
                        if vessel_load[i, j] > 0:
                            vessel_unloadtime = (2 * (i + 1)) + (2 * (j + 1)) +
(2 * vessel_load[i, j]) + time_emptyquaycrane
                            vessel_load[i, j] = vessel_load[i, j] - 1
                            dummy1 = 2
                            break
```

```
                    if dummy1 == 2:
                        if vessel_load[i, j] == 0:
                            vessel_empty[i, j] = vessel_load[i, j]
                        else:
                            pass
                        break
                if dummy1 == 2:
                    break

        elif vesselunloadjob_schedule_strategy==4:#horizontal across i
            for j in range(0,width_vessel):
                for k in range(0,height_vessel):
                    for i in range(0,length_vessel):
                        if vessel_load[i,j]>0:
                            vessel_unloadtime = (2 * (i + 1)) + (2 * (j + 1)) +
(2 * vessel_load[i, j]) + time_emptyquaycrane
                            vessel_load[i,j]=vessel_load[i,j]-1
                            dummy1=2
                            break
                    if  dummy1==2:
                        if vessel_load[i,j]==0:
                            vessel_empty[i, j] = vessel_load[i, j]
                        else:
                            pass
                        break
                if dummy1 == 2:
                    break

        position_quaycrane=1

time_quaycrane=vessel_unloadtime+time_quaycrane+platform_handling_time_forcrane
+alignment_time+container_handling_time_forcrane
        no_platform=no_platform-1
        unload_waiting=unload_waiting+1
        time_to_reach_endposition_quaycrane=time_quaycrane
        time_to_reach_endposition_quaycrane_dummy=time_quaycrane

        time=timing_routine()


    def unload_yardcrane():
        global vessel_unloadtime, vessel_load, i, j, truck_unloadtime, c, d,
platform_time, unload_time, process_indicator, unloading_count,
previous_operation,unload_count,position_yardcrane,unload_waiting
        global time_quaycrane,
time_yardcrane,no_platform,time_to_reach_startposition_yardcrane,time,platform_
handling_time_forcrane,alignment_time
        global
time_emptyyardcrane,container_handling_time_forcrane,count_truck1
       dummy=1


        if yardunloadjob_schedule_strategy==1:#vertical and across d
            for c in range(0,2*length_yard):
                for d in range(0,width_yard):
                    for e in range(0,height_yard):
                        if truck_empty[c, d] < height_yard:
                            truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) +
(2 * (truck_empty[c, d]+1)) + time_emptyyardcrane
                            truck_empty[c, d] = truck_empty[c, d] + 1
                            dummy=2
                            break
```

```
                    if dummy==2:
                        break
                if dummy == 2:
                    break

        elif yardunloadjob_schedule_strategy==2:#Horizontal and across d
            for c in range(0,2*length_yard):
                for e in range(0,height_yard):
                    for d in range(0,width_yard):
                        if truck_empty[c, d] < height_yard:
                            truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) +
(2 * (truck_empty[c, d]+1)) + time_emptyyardcrane
                            truck_empty[c, d] = truck_empty[c, d] + 1
                            dummy=2
                            break
                    if dummy==2:
                        break
                if dummy == 2:
                    break

        elif yardunloadjob_schedule_strategy==3:#Vertical and across c
            for d in range(0,width_yard):
                for c in range(0,2*length_yard):
                    for e in range(0,height_yard):
                        if truck_empty[c, d] < height_yard:
                            truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) +
(2 * (truck_empty[c, d]+1)) + time_emptyyardcrane
                            truck_empty[c, d] = truck_empty[c, d] + 1
                            dummy=2
                            break
                    if dummy==2:
                        break
                if dummy == 2:
                    break

        elif yardunloadjob_schedule_strategy==4:#Horizontal and across c
            for d in range(0,width_yard):
                for e in range(0,height_yard):
                    for c in range(0,2*length_yard):
                        if truck_empty[c, d] < height_yard:
                            truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) +
(2 * (truck_empty[c, d]+1)) + time_emptyyardcrane
                            truck_empty[c, d] = truck_empty[c, d] + 1
                            dummy=2
                            break
                    if dummy==2:
                        break
                if dummy == 2:
                    break

        unload_count=unload_count+1
        position_yardcrane=0
        unload_waiting=unload_waiting-1
        no_platform=no_platform+1


        if time_to_reach_endposition_yardcrane<=
time_to_reach_endposition_quaycrane :
            platform_time = 0+platform_time
            time_yardcrane=time_to_reach_endposition_quaycrane
        else:
```

```
            platform_time=time_to_reach_endposition_yardcrane-
time_to_reach_endposition_quaycrane+platform_time
        time_yardcrane=time_yardcrane +
truck_unloadtime+platform_handling_time_forcrane+alignment_time+container_handl
ing_time_forcrane
        time_to_reach_startposition_yardcrane=time_yardcrane


        timing_routine()


    def load_yardcrane():
        global vessel_loadtime,truck_load,a,b,
truck_loadtime,k,l,platform_time,load_time,process_indicator,loading_count,prev
ious_operation,position_yardcrane
        global time_quaycrane,
time_yardcrane,no_platform,load_waiting,time_to_reach_endposition_yardcrane,tim
e,platform_handling_time_forcrane,alignment_time,time_to_reach_endposition_yard
crane_dummy
        global
time_emptyyardcrane,container_handling_time_forcrane,count_truck2#process_indic
ator=2
        global dummy_123


        dummy2=1

        if yardloadjob_schedule_strategy==1:#Vertical and across b
            for a in range(0,length_yard):
                for b in range(0,width_yard):
                    for c in range(0,height_yard):
                        if truck_load[a,b]>0:
                            truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2
* truck_load[a, b]) + time_emptyyardcrane
                            truck_load[a,b]=truck_load[a,b]-1
                            dummy2=2
                            break
                    if dummy2==2:
                        if truck_load[a,b]==0:
                            truck_empty[a, b] = truck_load[a, b]
                        else:
                            pass
                        break
                if dummy2 == 2:
                    break

        elif yardloadjob_schedule_strategy == 2:#Horizontal and across b
            for a in range(0, length_yard):
                for c in range(0, height_yard):
                    for b in range(0, width_yard):
                        if truck_load[a, b] > 0:
                            truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2
* truck_load[a, b]) + time_emptyyardcrane
                            truck_load[a, b] = truck_load[a, b] - 1
                            dummy2 = 2
                            break
                    if dummy2 == 2:
                        if truck_load[a, b] == 0:
                            truck_empty[a, b] = truck_load[a, b]
                        else:
                            pass
                        break
```

```
                if dummy2 == 2:
                    break

        elif yardloadjob_schedule_strategy == 3:#Vertical and across a
            for b in range(0, width_yard):
                for a in range(0, length_yard):
                    for c in range(0, height_yard):
                        if truck_load[a, b] > 0:
                            truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2
* truck_load[a, b]) + time_emptyyardcrane
                            truck_load[a, b] = truck_load[a, b] - 1
                            dummy2 = 2
                            break
                    if dummy2 == 2:
                        if truck_load[a, b] == 0:
                            truck_empty[a, b] = truck_load[a, b]
                        else:
                            pass
                        break
                if dummy2 == 2:
                    break

        elif yardloadjob_schedule_strategy == 4:#Horizontal and across a
            for b in range(0, width_yard):
                for c in range(0, height_yard):
                    for a in range(0, length_yard):
                        if truck_load[a, b] > 0:
                            truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2
* truck_load[a, b]) + time_emptyyardcrane
                            truck_load[a, b] = truck_load[a, b] - 1
                            dummy2 = 2
                            break
                    if dummy2 == 2:
                        if truck_load[a, b] == 0:
                            truck_empty[a, b] = truck_load[a, b]
                        else:
                            pass
                        break
                if dummy2 == 2:
                    break

        position_yardcrane=1

        time_yardcrane=time_yardcrane +
truck_loadtime+platform_handling_time_forcrane+alignment_time+container_handlin
g_time_forcrane
        no_platform=no_platform-1
        load_waiting=load_waiting+1
        time_to_reach_endposition_yardcrane=time_yardcrane
        time_to_reach_endposition_yardcrane_dummy=time_yardcrane
        dummy_123=dummy_123+truck_loadtime

        timing_routine()

    def load_quaycrane():
        global vessel_loadtime, truck_load, a, b, truck_loadtime, k, l,
platform_time, load_time, process_indicator, loading_count,
previous_operation,load_count,position_quaycrane,load_waiting
        global time_quaycrane,
time_yardcrane,no_platform,load_waiting,time_to_reach_startposition_quaycrane,t
ime,platform_handling_time_forcrane,alignment_time
        global
```

```
time_emptyquaycrane,container_handling_time_forcrane,count_vessel2
        dummy3=1#Its used for this for block alone

        if vesselloadjob_schedule_strategy==1:#Vertical and across l
            for k in range(0,2*length_vessel):
                for l in range(0,width_vessel):
                    for m in range(0,height_vessel):
                        if vessel_empty[k, l] < height_vessel:
                            vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) +
(2 * (vessel_empty[k, l]+1)) + time_emptyquaycrane
                            vessel_empty[k, l] = vessel_empty[k, l] + 1
                            dummy3=2
                            break
                    if dummy3==2:
                        break
                if dummy3 == 2:
                    break

        elif vesselloadjob_schedule_strategy==2:#Horizontal and across l
            for k in range(0,2*length_vessel):
                for m in range(0,height_vessel):
                    for l in range(0,width_vessel):
                        if vessel_empty[k, l] < height_vessel:
                            vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) +
(2 * (vessel_empty[k, l]+1)) + time_emptyquaycrane
                            vessel_empty[k, l] = vessel_empty[k, l] + 1
                            dummy3=2
                            break
                    if dummy3==2:
                        break
                if dummy3 == 2:
                    break

        elif vesselloadjob_schedule_strategy==3:#Vertical and across k
            for l in range(0,width_vessel):
                for k in range(0,2*length_vessel):
                    for m in range(0,height_vessel):
                        if vessel_empty[k, l] < height_vessel:
                            vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) +
(2 * (vessel_empty[k, l]+1)) + time_emptyquaycrane
                            vessel_empty[k, l] = vessel_empty[k, l] + 1
                            dummy3=2
                            break
                    if dummy3==2:
                        break
                if dummy3 == 2:
                    break

        elif vesselloadjob_schedule_strategy==4:#Horizontal and across k
            for l in range(0,width_vessel):
                for m in range(0,height_vessel):
                    for k in range(0,2*length_vessel):
                        if vessel_empty[k, l] < height_vessel:
                            vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) +
(2 *(vessel_empty[k, l]+1)) + time_emptyquaycrane
                            vessel_empty[k, l] = vessel_empty[k, l] + 1
                            dummy3=2
                            break
                    if dummy3==2:
                        break
                if dummy3 == 2:
                    break
```

```
        load_count=load_count+1
        load_waiting=load_waiting-1

        load_time=truck_loadtime+vessel_loadtime+platform_time+load_time
        position_quaycrane=0

        no_platform=no_platform+1

        if
time_to_reach_endposition_quaycrane<=time_to_reach_endposition_yardcrane:
            platform_time = 0+platform_time
            time_quaycrane=time_to_reach_endposition_yardcrane
        else:

            platform_time=time_to_reach_endposition_quaycrane-
time_to_reach_endposition_yardcrane+platform_time

        time_quaycrane = time_quaycrane + vessel_loadtime +
platform_handling_time_forcrane +
alignment_time+container_handling_time_forcrane
        time_to_reach_startposition_quaycrane=time_quaycrane
        timing_routine()



    def timing_routine():
        global time_yardcrane, time_quaycrane,simclock
        if time_yardcrane>time_quaycrane:
            simclock=time_yardcrane
        else:
            simclock=time_quaycrane

    if unload_count<unload_containers and load_count<load_containers:
        if no_platform>0:
            if position_yardcrane == 0 and position_quaycrane==0:
                if time_to_reach_startposition_yardcrane <
time_to_reach_startposition_quaycrane:
                    load_yardcrane()
                    if no_platform>0: #and unload_waiting>0:
                        unload_quaycrane()

                    else:
                        time_quaycrane = time_quaycrane + time_emptyquaycrane +
alignment_time
                        position_quaycrane=1
                        time_to_reach_endposition_quaycrane=time_quaycrane

#time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                        timing_routine()

                else:
                    print "2.0"
                    unload_quaycrane()
                    print "pos quay", position_quaycrane

                    if no_platform>0:
                        load_yardcrane()

                    else:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
```

```
                        position_yardcrane=1
                        time_to_reach_endposition_yardcrane=time_yardcrane
##time_to_reach_endposition_yardcrane_dummy=time_yardcrane
                        timing_routine()


            elif position_yardcrane == 0 and position_quaycrane == 1:
                print "test2"
                print "start
pos_yardcrane",time_to_reach_startposition_yardcrane
                print "end pos_quay",time_to_reach_endposition_quaycrane
                if
time_to_reach_startposition_yardcrane<time_to_reach_endposition_quaycrane:
                    load_yardcrane()


                    if load_waiting>0:
                        load_quaycrane()
                    else:

time_quaycrane=time_quaycrane+time_emptyquaycrane+alignment_time
                        position_quaycrane=0
                        time_to_reach_startposition_quaycrane=time_quaycrane
                        timing_routine()

                else:
                    ##time_to_reach_endposition_quaycrane = 10000000
                    ##time_to_reach_startposition_yardcrane = 10000000
                    if load_waiting>0:
                        load_quaycrane()
                    else:

time_quaycrane=time_quaycrane+time_emptyquaycrane+alignment_time
                        position_quaycrane=0
                        time_to_reach_startposition_quaycrane=time_quaycrane
                        timing_routine()
                    #load_yardcrane()
                    #print "Test2.1"
                    #position_yardcranedummy = position_yardcrane
                    #position_yardcrane = 2

            elif position_yardcrane == 1 and position_quaycrane == 0:
                print "test3"
                if
time_to_reach_endposition_yardcrane<time_to_reach_startposition_quaycrane:
                    ##time_to_reach_endposition_yardcrane=10000000
                    ##time_to_reach_startposition_quaycrane=10000000
                    if unload_waiting>0:
                        print "subtest3"
                        unload_yardcrane()
                    else:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                        position_yardcrane=0
                        time_to_reach_startposition_yardcrane=time_yardcrane
                        timing_routine()
                else:
                    ##time_to_reach_endposition_yardcrane = 10000000
                    ##time_to_reach_startposition_quaycrane = 10000000
                    unload_quaycrane()
                    if unload_waiting>0:
```

```
                        unload_yardcrane()

                    else:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                        position_yardcrane=0
                        time_to_reach_startposition_yardcrane=time_yardcrane
                        timing_routine()

            elif position_yardcrane == 1 and position_quaycrane == 1:
                print "test"
                if
time_to_reach_endposition_yardcrane<time_to_reach_endposition_quaycrane:
                    if unload_waiting>0 and load_waiting>0:
                        unload_yardcrane()
                    elif unload_waiting>0 and load_waiting==0:
                        unload_yardcrane()
                    elif unload_waiting==0 and load_waiting>0:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                        position_yardcrane=0
                        time_to_reach_startposition_yardcrane=time_yardcrane
                        timing_routine()
                    else:
                        unload_yardcrane()

                else:
                    if unload_waiting>0 and load_waiting>0:
                        load_quaycrane()
                    elif unload_waiting>0 and load_waiting==0:

time_quaycrane=time_quaycrane+alignment_time+time_emptyquaycrane
                        position_quaycrane=0
                        time_to_reach_startposition_quaycrane=time_quaycrane
                        timing_routine()
                    elif unload_waiting==0 and load_waiting>0:
                        load_quaycrane()

                    else:
                        load_quaycrane()

        else:
            print "position yardcrane",position_yardcrane
            print "position quaycrane",position_quaycrane
            if position_yardcrane==0 and position_quaycrane==0:

                if
time_to_reach_startposition_yardcrane<time_to_reach_startposition_quaycrane:
                    ##time_to_reach_startposition_quaycrane=10000000
                    ##time_to_reach_startposition_yardcrane=10000000
                    if unload_waiting==total_platform:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                        position_yardcrane=1
                        time_to_reach_endposition_yardcrane=time_yardcrane

##time_to_reach_endposition_yardcrane_dummy=time_yardcrane
                        timing_routine()
                    elif load_waiting==total_platform:

time_quaycrane=time_quaycrane+time_emptyquaycrane+alignment_time
                        position_quaycrane=1
```

```
                        time_to_reach_endposition_quaycrane=time_quaycrane

##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                        timing_routine()
                    else:
                        time_yardcrane = time_yardcrane + time_emptyyardcrane +
alignment_time
                        position_yardcrane = 1
                        time_to_reach_endposition_yardcrane = time_yardcrane
                        timing_routine()
                        position_yardcranedummy = position_yardcrane

                else:
                    ##time_to_reach_startposition_quaycrane = 10000000
                    ##time_to_reach_startposition_yardcrane = 10000000
                    if unload_waiting==total_platform:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                        position_yardcrane=1
                        time_to_reach_endposition_yardcrane=time_yardcrane

##time_to_reach_endposition_yardcrane_dummy=time_yardcrane
                        timing_routine()
                    elif load_waiting==total_platform:

time_quaycrane=time_quaycrane+time_emptyquaycrane+alignment_time
                        position_quaycrane=1
                        time_to_reach_endposition_quaycrane=time_quaycrane

##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                        timing_routine()
                    else:
                        time_quaycrane = time_quaycrane + time_emptyquaycrane +
alignment_time
                        position_quaycrane = 1
                        timing_routine()
                        time_to_reach_endposition_quaycrane=time_quaycrane

##time_to_reach_endposition_quaycrane_dummy=time_quaycrane

            elif position_yardcrane == 0 and position_quaycrane == 1:
                if
time_to_reach_startposition_yardcrane<time_to_reach_endposition_quaycrane:
                    ##time_to_reach_startposition_yardcrane=10000000
                    ##time_to_reach_endposition_quaycrane=10000000
                    if unload_waiting==total_platform:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                        position_yardcrane=1
                        timing_routine()
                        time_to_reach_endposition_yardcrane=time_yardcrane

##time_to_reach_endposition_yardcrane_dummy=time_yardcrane
                    elif load_waiting==total_platform:
                        time_yardcrane=time_quaycrane
                        load_quaycrane()
                    else:
                        time_yardcrane = time_yardcrane + time_emptyyardcrane +
alignment_time
                        position_yardcrane = 1
                        timing_routine()
                        time_to_reach_endposition_yardcrane=time_yardcrane
```

```
##time_to_reach_endposition_yardcrane_dummy=time_yardcrane

                else:
                    #time_to_reach_startposition_yardcrane = 10000000
                    #time_to_reach_endposition_quaycrane = 10000000
                    if load_waiting>0:
                        load_quaycrane()
                    else:
                        time_quaycrane = time_quaycrane + time_emptyquaycrane +
alignment_time
                        position_quaycrane = 0
                        time_to_reach_endposition_quaycrane=time_quaycrane

##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                        timing_routine()

            elif position_yardcrane == 1 and position_quaycrane == 0:
                if time_to_reach_endposition_yardcrane <
time_to_reach_startposition_quaycrane:
                    ##time_to_reach_endposition_yardcrane = 10000000
                    ##time_to_reach_startposition_quaycrane = 10000000
                    if unload_waiting>0:
                        unload_yardcrane()
                    else:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                        position_yardcrane=0
                        time_to_reach_startposition_yardcrane=time_yardcrane

                        timing_routine()

                else:
                    ##time_to_reach_endposition_yardcrane = 10000000
                    ##time_to_reach_startposition_quaycrane = 10000000
                    if unload_waiting == total_platform:
                        time_quaycrane=time_yardcrane
                        unload_yardcrane()
                    elif load_waiting == total_platform:
                        time_quaycrane = time_quaycrane + time_emptyquaycrane +
alignment_time
                        position_quaycrane = 1
                        time_to_reach_endposition_quaycrane=time_quaycrane

##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                        timing_routine()
                    else:
                        time_quaycrane = time_quaycrane + time_emptyquaycrane +
alignment_time
                        position_quaycrane = 1
                        time_to_reach_endposition_quaycrane=time_quaycrane

##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                        timing_routine()

            elif position_yardcrane == 1 and position_quaycrane == 1:
                print "test1"
                if time_to_reach_endposition_yardcrane <
time_to_reach_endposition_quaycrane:
                    ##time_to_reach_endposition_yardcrane = 10000000
                    ##time_to_reach_endposition_quaycrane = 10000000
                    if unload_waiting >0:
```

```
                        unload_yardcrane()
                    else:

                        time_yardcrane = time_yardcrane + time_emptyyardcrane +
alignment_time
                        position_yardcrane = 0
                        time_to_reach_startposition_yardcrane = time_yardcrane

##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                        timing_routine()

                else:

                    ##time_to_reach_endposition_yardcrane = 10000000
                    ##time_to_reach_endposition_quaycrane = 10000000
                    if load_waiting >0:
                        load_quaycrane()
                    else:
                        time_quaycrane = time_quaycrane + time_emptyquaycrane +
alignment_time
                        position_quaycrane = 0
                        time_to_reach_startposition_quaycrane = time_quaycrane

##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                        timing_routine()


    elif unload_count==unload_containers and load_count<load_containers:
        if no_platform>0:
            if position_quaycrane==1 and position_yardcrane==1 and
load_waiting>0:
                if
time_to_reach_endposition_yardcrane<time_to_reach_endposition_quaycrane:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                    position_yardcrane=0
                    time_to_reach_startposition_yardcrane=time_yardcrane
                    timing_routine()
                else:
                    load_quaycrane()
                ##time_to_reach_endposition_yardcrane=10000000
                ##time_to_reach_endposition_quaycrane=10000000
            elif position_quaycrane==1 and position_yardcrane==1 and
load_waiting==0:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                position_yardcrane=0
                time_to_reach_startposition_yardcrane=time_yardcrane
                timing_routine()
            elif position_quaycrane==1 and position_yardcrane==0 and
load_waiting>0:
                if
time_to_reach_startposition_yardcrane<time_to_reach_endposition_quaycrane:
                    load_yardcrane()
                else:
                    load_quaycrane()
                ##time_to_reach_startposition_yardcrane=10000000
                ##time_to_reach_endposition_quaycrane=10000000
            elif position_quaycrane==1 and position_yardcrane==0 and
load_waiting==0:
                load_yardcrane()
                ##time_to_reach_startposition_yardcrane=10000000
```

```
                ##time_to_reach_endposition_quaycrane=10000000
            elif position_quaycrane==0 and position_yardcrane==0:#It does not
matter if there is a loading job waiting or not, since both of them are at 0th
positions
                if time_to_reach_startposition_yardcrane <
time_to_reach_startposition_quaycrane:
                    load_yardcrane()
                else:

time_quaycrane=time_quaycrane+time_emptyquaycrane+alignment_time
                    position_quaycrane=1
                    time_to_reach_endposition_quaycrane=time_quaycrane
                    ##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                    timing_routine()
                ##time_to_reach_startposition_yardcrane=10000000
                ##time_to_reach_startposition_quaycrane=10000000
            elif position_quaycrane==0 and position_yardcrane==1:
                if
time_to_reach_endposition_yardcrane<time_to_reach_startposition_quaycrane:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                    timing_routine()
                    position_yardcrane=0
                    time_to_reach_startposition_yardcrane=time_yardcrane
                else:

time_quaycrane=time_quaycrane+time_emptyquaycrane+alignment_time
                    position_quaycrane=1
                    time_to_reach_endposition_quaycrane=time_quaycrane
                    ##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                    timing_routine()
                ##time_to_reach_startposition_quaycrane=10000000
                ##time_to_reach_endposition_yardcrane=10000000

        else:
            if position_quaycrane == 1 and position_yardcrane == 1:# and
load_waiting > 0:
                if time_to_reach_endposition_yardcrane <
time_to_reach_endposition_quaycrane:
                    time_yardcrane = time_yardcrane + time_emptyyardcrane +
alignment_time
                    position_yardcrane = 0
                    time_to_reach_startposition_yardcrane = time_yardcrane
                    timing_routine()
                else:
                    load_quaycrane()
                ##time_to_reach_endposition_yardcrane = 10000000
                ##time_to_reach_endposition_quaycrane = 10000000
            elif position_quaycrane == 1 and position_yardcrane == 0:# and
load_waiting > 0:
                time_yardcrane=time_quaycrane
                load_quaycrane()
            elif position_quaycrane == 0 and position_yardcrane == 0:# It does
not matter if there is a loading job waiting or not, since both of them are at
0th positions
                time_quaycrane = time_quaycrane + time_emptyquaycrane +
alignment_time
                position_quaycrane = 1
                time_to_reach_endposition_quaycrane = time_quaycrane
                ##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                timing_routine()
            elif position_quaycrane == 0 and position_yardcrane == 1:
```

```
                 if time_to_reach_endposition_yardcrane <
time_to_reach_startposition_quaycrane:
                     time_yardcrane = time_yardcrane + time_emptyyardcrane +
alignment_time
                     timing_routine()
                     position_yardcrane = 0
                     time_to_reach_startposition_yardcrane = time_yardcrane
                 else:
                     time_quaycrane = time_quaycrane + time_emptyquaycrane +
alignment_time
                     position_quaycrane = 1
                     time_to_reach_endposition_quaycrane = time_quaycrane
                     ##time_to_reach_endposition_quaycrane_dummy=time_quaycrane
                     timing_routine()
                 ##time_to_reach_startposition_quaycrane = 10000000
                 ##time_to_reach_endposition_yardcrane = 10000000

    elif load_count == load_containers and unload_count < unload_containers:
        if no_platform > 0:
            if position_quaycrane == 1 and position_yardcrane == 1 and
unload_waiting > 0:
                if time_to_reach_endposition_yardcrane <
time_to_reach_endposition_quaycrane:
                    unload_yardcrane()
                else:
                    time_quaycrane = time_quaycrane + time_emptyquaycrane +
alignment_time
                    position_quaycrane = 0
                    time_to_reach_startposition_quaycrane = time_quaycrane
                    timing_routine()
                ##time_to_reach_endposition_yardcrane = 10000000
                ##time_to_reach_endposition_quaycrane = 10000000
            elif position_quaycrane == 1 and position_yardcrane == 1 and
unload_waiting == 0:
                time_quaycrane = time_quaycrane + time_emptyquaycrane +
alignment_time
                position_quaycrane = 0
                time_to_reach_startposition_quaycrane = time_quaycrane
                timing_routine()
            elif position_quaycrane == 1 and position_yardcrane == 0:
                if time_to_reach_startposition_yardcrane <
time_to_reach_endposition_quaycrane:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                    position_yardcrane=1
                    time_to_reach_endposition_yardcrane=time_yardcrane
                    ##time_to_reach_endposition_yardcrane_dummy=time_yardcrane
                    timing_routine()
                else:

time_quaycrane=time_quaycrane+time_emptyquaycrane+alignment_time
                    position_quaycrane=0
                    time_to_reach_startposition_quaycrane=time_quaycrane
                    timing_routine()
                ##time_to_reach_startposition_yardcrane = 10000000
                ##time_to_reach_endposition_quaycrane = 10000000
            elif position_quaycrane == 0 and position_yardcrane == 0:   # It
does not matter if there is a loading job waiting or not, since both of them
are at 0th positions
                if time_to_reach_startposition_yardcrane <
time_to_reach_startposition_quaycrane:
                    time_yardcrane = time_yardcrane + time_emptyyardcrane +
```

```
alignment_time
                    position_yardcrane = 1
                    time_to_reach_endposition_yardcrane = time_yardcrane
                    ##time_to_reach_endposition_yardcrane_dummy=time_yardcrane
                    timing_routine()
                else:
                    unload_quaycrane()
                ##time_to_reach_startposition_yardcrane = 10000000
                ##time_to_reach_startposition_quaycrane = 10000000
            elif position_quaycrane == 0 and position_yardcrane == 1 and
unload_waiting>0:
                if time_to_reach_endposition_yardcrane <
time_to_reach_startposition_quaycrane:
                   unload_yardcrane()
                else:
                    unload_quaycrane()
                ##time_to_reach_startposition_quaycrane = 10000000
                ##time_to_reach_endposition_yardcrane = 10000000
            elif position_quaycrane == 0 and position_yardcrane == 1 and
unload_waiting== 0:
                unload_quaycrane()
                ##time_to_reach_startposition_quaycrane = 10000000
                ##time_to_reach_endposition_yardcrane = 10000000
        else:
            if position_quaycrane == 1 and position_yardcrane == 1:# and
unload_waiting > 0:
                if time_to_reach_endposition_yardcrane <
time_to_reach_endposition_quaycrane:
                    unload_yardcrane()
                else:

time_quaycrane=time_quaycrane+time_emptyquaycrane+alignment_time
                    position_quaycrane=0
                    time_to_reach_startposition_quaycrane=time_quaycrane
                    timing_routine()
                ##time_to_reach_endposition_yardcrane = 10000000
                ##time_to_reach_endposition_quaycrane = 10000000
            elif position_quaycrane == 1 and position_yardcrane == 0:# and
unload_waiting > 0:
                if
time_to_reach_startposition_yardcrane<time_to_reach_endposition_quaycrane:

time_yardcrane=time_yardcrane+time_emptyyardcrane+alignment_time
                    position_yardcrane=1
                    time_to_reach_endposition_yardcrane=time_yardcrane
                    ##time_to_reach_endposition_yardcrane_dummy=time_yardcrane
                    timing_routine()
                else:

time_quaycrane=time_quaycrane+time_emptyquaycrane+alignment_time
                    position_quaycrane=0
                    time_to_reach_startposition_quaycrane=time_quaycrane
                    timing_routine()
            elif position_quaycrane == 0 and position_yardcrane == 0:  # It
does not matter if there is a loading job waiting or not, since both of them
are at 0th positions
                time_yardcrane = time_yardcrane + time_emptyyardcrane +
alignment_time
                position_yardcrane = 1
                time_to_reach_endposition_yardcrane = time_yardcrane
                ##time_to_reach_endposition_yardcrane_dummy=time_yardcrane
                timing_routine()
```

```
            elif position_quaycrane == 0 and position_yardcrane == 1:# and
unload_waiting>0:
                time_quaycrane = time_yardcrane
                unload_yardcrane()


    else:
        print "process over"


print simclock
```

APPENDIX G

ANALYTICAL MODEL CODE FOR SYSTEM WITH BUFFER OF SIZE ONE

```
import numpy as np
nu=input("no. of containers to be unloaded")
nl=input("no of containers to be loaded")
vesseload_X1=input("enter the vesselload strategy from 0 to 3")
yardload_X2=input("enter the yardload strategy from 0 to 3")
vesselunload_X3=input("enter the vessel unload strategy from 0 to 3")
yardunload_X4=input("enter the yard unload strategy from 0 to 3")

variable_vessel=1
variable_truck=1
length_vessel=input("enter the number of containers in vessel across length")
width_vessel=input("enter the number of containers in vessel across width")
height_vessel=input("enter the number of containers in vessel that could be
stacked including the bottom one")
length_yard=input("enter the number of containers in yard across length")
width_yard=input("enter the number of containers in yard across width")
height_yard=input("enter the number of containers in yard that could be stacked
including the bottom one")
vessel_load=np.zeros((length_vessel,width_vessel))
truck_load=np.zeros((length_yard,width_yard))
vessel_empty=np.zeros((2*length_vessel,width_vessel))
truck_empty=np.zeros((2*length_yard,width_yard))


for lengt in range(0,length_vessel):
    for widtt in range(0,width_vessel):
        vessel_load[lengt,widtt]=height_vessel

for lengt in range(0, length_yard):
    for widtt in range(0, width_yard):
        truck_load[lengt,widtt]=height_yard

for leng in range(0,length_vessel):
    for widt in range(0, width_vessel):
        vessel_empty[leng,widt]=2*height_vessel

for leng in range(0, length_yard):
    for widt in range(0, width_yard):
        truck_empty[leng,widt]=2*height_yard

vesselside_time=0
yardside_time=0
vesselside_rate=0
yardside_rate=0
dummy=0
dummy1=0
dummy2=0
dummy3=0

count_load=0
count_unload=0
total_vesselunload_time=0
total_truckunload_time=0
total_vesselload_time=0
total_truckload_time=0

def vesselunload():
    global vessel_unloadrate,
dummy1,vessel_unloadtime,i,j,variable_vessel,vessel_load,vessel_empty,rate,vess
el_unloadjob_time_matrix,nth_container_beingunloaded,total_vesselunload_time
    dummy1=0
    if vesselunload_X3==0:#vertical across v
```

```
        for u in range(0, length_vessel):
            for v in range(0, width_vessel):
                for w in range(0, height_vessel):
                    if vessel_load[u, v] > 0:
                        vessel_unloadtime = (2 * (u + 1)) + (2 * (v + 1)) + (2
* vessel_load[u, v])
                        vessel_load[u, v] = vessel_load[u, v] - 1
                        dummy1 = 2
                        break
                if dummy1 == 2:
                    if vessel_load[u, v] == 0:
                        vessel_empty[u, v] = vessel_load[u, v]
                    else:
                        pass
                    break
            if dummy1==2:
                break

    elif vesselunload_X3==1:#horizontal across v
        for u in range(0, length_vessel):
            for w in range(0, height_vessel):
                for v in range(0, width_vessel):
                    if vessel_load[u, v] > 0:
                        vessel_unloadtime = (2 * (u + 1)) + (2 * (v + 1)) + (2
* vessel_load[u, v])
                        vessel_load[u, v] = vessel_load[u, v] - 1
                        dummy1 = 2
                        break
                if dummy1 == 2:
                    if vessel_load[u, v] == 0:
                        vessel_empty[u, v] = vessel_load[u, v]
                    else:
                        pass
                    break
            if dummy1==2:
                break

    elif vesselunload_X3==2:#vertical across u
        for v in range(0, width_vessel):
            for u in range(0, length_vessel):
                for w in range(0, height_vessel):
                    if vessel_load[u, v] > 0:
                        vessel_unloadtime = (2 * (u + 1)) + (2 * (v + 1)) + (2
* vessel_load[u, v])
                        vessel_load[u, v] = vessel_load[u, v] - 1
                        dummy1 = 2
                        break
                if dummy1 == 2:
                    if vessel_load[u, v] == 0:
                        vessel_empty[u, v] = vessel_load[u, v]
                    else:
                        pass
                    break
            if dummy1==2:
                break

    elif vesselunload_X3==3:#horizontal across u
        for v in range(0, width_vessel):
            for w in range(0, height_vessel):
                for u in range(0, length_vessel):
                    if vessel_load[u, v] > 0:
                        vessel_unloadtime = (2 * (u + 1)) + (2 * (v + 1)) + (2
```

```
* vessel_load[u, v])
                        vessel_load[u, v] = vessel_load[u, v] - 1
                        dummy1 = 2
                        break
                if dummy1 == 2:
                    if vessel_load[u, v] == 0:
                        vessel_empty[u, v] = vessel_load[u, v]
                    else:
                        pass
                    break
            if dummy1==2:
                break

    vessel_unloadjob_time_matrix[nth_container_beingunloaded-
1,0]=vessel_unloadtime
    total_vesselunload_time=total_vesselunload_time+vessel_unloadtime
    #vessel_unloadrate = 1.0 / vessel_unloadtime
    #rate[i,j]=vessel_unloadrate
    #variable_vessel=2

def vesselload():
    global
vessel_loadrate,dummy3,vessel_loadtime,i,j,variable_vessel,vessel_loadtime,vess
el_empty,rate,nth_container_beingloaded,vessel_loadjob_time_matrix,total_vessel
load_time
    dummy3=0
    if vesseload_X1==0:# vertical across l
        for k in range(0, 2*length_vessel):
            for l in range(0, width_vessel):
                for m in range(0, height_vessel):
                    if vessel_empty[k, l] < height_vessel:
                        vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) + (2 *
(vessel_empty[k, l]+1))
                        vessel_empty[k, l] = vessel_empty[k, l] + 1
                        dummy3 = 2
                        break
                if dummy3 == 2:
                    break
            if dummy3 == 2:
                break

    elif vesseload_X1==1:#horizontal across l
        for k in range(0, 2*length_vessel):
            for m in range(0, height_vessel):
                for l in range(0, width_vessel):
                    if vessel_empty[k, l] < height_vessel:
                        vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) + (2 *
(vessel_empty[k, l]+1))
                        vessel_empty[k, l] = vessel_empty[k, l] + 1
                        dummy3 = 2
                        break
                if dummy3 == 2:
                    break
            if dummy3 == 2:
                break

    elif vesseload_X1==2:#vertical across k
        for l in range(0, width_vessel):
            for k in range(0, 2*length_vessel):
                for m in range(0, height_vessel):
                    if vessel_empty[k, l] < height_vessel:
                        vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) + (2 *
```

```
(vessel_empty[k, l]+1))
                        vessel_empty[k, l] = vessel_empty[k, l] + 1
                        dummy3 = 2
                        break
                if dummy3 == 2:
                    break
            if dummy3 == 2:
                break

    elif vesseload_X1==3:#horizontal acroos k
        for l in range(0, width_vessel):
            for m in range(0, height_vessel):
                for k in range(0, 2*length_vessel):
                    if vessel_empty[k, l] < height_vessel:
                        vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) + (2 *
(vessel_empty[k, l]+1))
                        vessel_empty[k, l] = vessel_empty[k, l] + 1
                        dummy3 = 2
                        break
                if dummy3 == 2:
                    break
            if dummy3 == 2:
                break

    vessel_loadjob_time_matrix[nth_container_beingloaded-1,0]=vessel_loadtime
    total_vesselload_time=total_vesselload_time+vessel_loadtime
    #vessel_loadrate = 1.0 / vessel_loadtime
    #rate[i,j]=rate[i,j]+vessel_loadrate
    #variable_vessel=1

def truckload():
    global dummy2,
truck_loadtime,truck_loadrate,i,j,variable_truck,truck_empty,truck_load,rate,co
unt,nth_container_beingloaded,truck_loadjob_time_matrix,total_truckload_time
    dummy2=0
    if yardload_X2 == 0:  # Vertical and across b
        for a in range(0, length_yard):
            for b in range(0, width_yard):
                for c in range(0, height_yard):
                    if truck_load[a, b] > 0:
                        truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2 *
truck_load[a, b])
                        truck_load[a, b] = truck_load[a, b] - 1
                        dummy2 = 2
                        break
                if dummy2 == 2:
                    if truck_load[a, b] == 0:
                        truck_empty[a, b] = truck_load[a, b]
                    else:
                        pass
                    break
            if dummy2==2:
                break

    elif yardload_X2 == 1:  # Horizontal and across b
        for a in range(0, length_yard):
            for c in range(0, height_yard):
                for b in range(0, width_yard):
                    if truck_load[a, b] > 0:
                        truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2 *
truck_load[a, b])
                        truck_load[a, b] = truck_load[a, b] - 1
```

```
                        dummy2 = 2
                        break
                if dummy2 == 2:
                    if truck_load[a, b] == 0:
                        truck_empty[a, b] = truck_load[a, b]
                    else:
                        pass
                    break
            if dummy2==2:
                break

    elif yardload_X2 == 2:  # Vertical and across a
        for b in range(0, width_yard):
            for a in range(0, length_yard):
                for c in range(0, height_yard):
                    if truck_load[a, b] > 0:
                        truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2 *
truck_load[a, b])
                        truck_load[a, b] = truck_load[a, b] - 1
                        dummy2 = 2
                        break
                if dummy2 == 2:
                    if truck_load[a, b] == 0:
                        truck_empty[a, b] = truck_load[a, b]
                    else:
                        pass
                    break
            if dummy2==2:
                break

    elif yardload_X2 == 3:  # Horizontal and across a
        for b in range(0, width_yard):
            for c in range(0, height_yard):
                for a in range(0, length_yard):
                    if truck_load[a, b] > 0:
                        truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2 *
truck_load[a, b])
                        truck_load[a, b] = truck_load[a, b] - 1
                        dummy2 = 2
                        break
                if dummy2 == 2:
                    if truck_load[a, b] == 0:
                        truck_empty[a, b] = truck_load[a, b]
                    else:
                        pass
                    break
            if dummy2==2:
                break

    truck_loadjob_time_matrix[nth_container_beingloaded-1,0]=truck_loadtime
    total_truckload_time=total_truckload_time+truck_loadtime
    #truck_loadrate = 1.0 / truck_loadtime
    #rate[i,j]=truck_loadrate
    #variable_truck=1

def truckunload():
    global dummy,
truck_unloadtime,truck_unloadrate,i,j,variable_truck,truck_empty,rate,count,nth
_container_beingunloaded,truck_unloadjob_time_matrix,total_truckunload_time
    dummy=0
    if yardunload_X4 == 0:  # vertical and across d
        for c in range(0, 2*length_yard):
```

```
            for d in range(0, width_yard):
                for e in range(0, height_yard):
                    if truck_empty[c, d] < height_yard:
                        truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) + (2 *
(truck_empty[c, d]+1))
                        truck_empty[c, d] = truck_empty[c, d] + 1
                        dummy = 2
                        print c,d
                        print truck_empty[c,d]
                        break
                if dummy == 2:
                    break
            if dummy==2:
                break

    elif yardunload_X4 == 1:  # Horizontal and across d
        for c in range(0, 2*length_yard):
            for e in range(0, height_yard):
                for d in range(0, width_yard):
                    if truck_empty[c, d] < height_yard:
                        truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) + (2 *
(truck_empty[c, d]+1))
                        truck_empty[c, d] = truck_empty[c, d] + 1
                        dummy = 2
                        break
                if dummy == 2:
                    break
            if dummy==2:
                break

    elif yardunload_X4== 2:  # Vertical and across c
        for d in range(0, width_yard):
            for c in range(0, 2*length_yard):
                for e in range(0, height_yard):
                    if truck_empty[c, d] < height_yard:
                        truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) + (2 *
(truck_empty[c, d]+1))
                        truck_empty[c, d] = truck_empty[c, d] + 1
                        dummy = 2
                        break
                if dummy == 2:
                    break
            if dummy==2:
                break

    elif yardunload_X4 == 3:  # Horizontal and across c
        for d in range(0, width_yard):
            for e in range(0, height_yard):
                for c in range(0, 2*length_yard):
                    if truck_empty[c, d] < height_yard:
                        truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) + (2 *
(truck_empty[c, d]+1))
                        truck_empty[c, d] = truck_empty[c, d] + 1
                        dummy = 2
                        break
                if dummy == 2:
                    break
            if dummy==2:
                break

    truck_unloadjob_time_matrix[nth_container_beingunloaded-
1,0]=truck_unloadtime
```

```
    total_truckunload_time=total_truckunload_time+truck_unloadtime
    #truck_unloadrate = 1.0 / truck_unloadtime
    #rate[i,j]=rate[i,j] + truck_unloadrate
    #variable_truck=2


nth_container_beingloaded=0
nth_container_beingunloaded=0

truck_loadjob_time_matrix=np.zeros((nl,1))
truck_unloadjob_time_matrix=np.zeros((nu,1))
vessel_loadjob_time_matrix=np.zeros((nl,1))
vessel_unloadjob_time_matrix=np.zeros((nu,1))

for nth_container_beingloaded in range(nl,0,-1):
    truckload()
    vesselload()

for nth_container_beingunloaded in range(nu,0,-1):
    truckunload()
    vesselunload()

if nl>0:
    for nth_container_beingloaded in range(nl,0,-1):
        truckload()
        vesselload()
if nu>0:
    for nth_container_beingunloaded in range(nu,0,-1):
        truckunload()
        vesselunload()

if nu==0:
    total_truckunload_time=0
    total_vesselunload_time=0
    nu=1#To make the average value to be zero

if nl==0:
    total_truckload_time=0
    total_vesselload_time=0
    nl=1##To make the average value to be zero

state=np.zeros((5,1))
state[2,0]=1
state[1,0]=2
state[0,0]=3
state[3,0]=4
state[4,0]=5


rate=np.zeros((5,5))


quaycrane_time=300
yardcrane_time=300
global total_time
global total_time_processing

total_time=0
total_time_processing=0
```

```
total_time=(total_time_processing+0.5*(total_truckload_time+total_vesselload_ti
me+total_vesselunload_time+total_truckunload_time))/(nu+nl)

print "total time is",total_time

time_yard_drop_pick=0.5*(total_truckload_time+total_truckunload_time)/(nu+nl)
time_vessel_drop_pick=0.5*(total_vesselload_time+total_vesselunload_time)/(nu+n
l)

length_state=5

for i in range(0,length_state):
    for j in range(0,length_state):
# Refer the following 2 codes
        if state[i,0]==1 and state[j,0]==4:#(nu,nl,x,x,x)'s next transition
provided nu and nl remain the same in next transition ie, transitions like
(2,2,0,1,0) to (2,2,1,0,0)
            rate[i,j]=(1.0 /((2*yardcrane_time)+(total_truckload_time/nl)))
        elif state[i,0]==2 and state[j,0]==3:
            rate[i,j]=(1.0/((2*quaycrane_time)+(total_vesselunload_time/nu)))
        elif state[i,0]==3 and state[j,0]==5:
            rate[i, j] = (1.0 / (2*yardcrane_time)+(total_truckunload_time/nu))
        elif state[i,0]==4 and state[j,0]==5:
            rate[i, j] = (1.0 / (2*quaycrane_time)+(total_vesselload_time/nl))
        else:
            pass


print "below is rate"


for i in range(0,length_state):
    for j in range(0,length_state):
        if i!=j:
            rate[i][i] = (rate[i][i] + rate[i][j])
        else:
            pass

for i in range(0,length_state):
       rate[i][i]=-(rate[i][i])

print rate

print state

print length_state

q=np.zeros((length_state-1,length_state-1))
for i in range(0,length_state-1):
    for j in range(0,length_state-1):
        q[i][j]=rate[i][j]

print "q"
print q
#print del_opn


q_neg=-q

r=np.zeros((4,1))

j_r=0
```

```
j=4
for i in range(0,length_state-1):
    r[i][0]=rate[i,j]


print q_neg
from numpy.linalg import inv

e=inv(q_neg)
#ident_matrix=np.identity(length_state-1)

#e=solve(q_neg,ident_matrix)
duration=np.zeros((length_state,1))

for i in range(0,length_state-2):
    for j in range(0,length_state-2):
        duration[i,0]=duration[i,0]+e[i,j]

print duration

print e

a_matrix=np.matmul(e,r)

a_neg=np.zeros((len(a_matrix),1))

for i in range(0,len(a_matrix)):
    a_neg[i,0]=-a_matrix[i,0]

print a_neg

q_mult_a=np.matmul(q,a_matrix)

print "q_mult_a"
print q_mult_a
print a_matrix

total_time_processing=0

mean_time=np.linalg.solve(q,a_neg)


print mean_time[1,0]
print mean_time[2,0]
```

APPENDIX H

ANALYTICAL MODEL CODE FOR SYSTEM WITH BUFFER OF SIZE TWO

```python
import numpy as np
nu=input("no. of containers to be unloaded")
nl=input("no of containers to be loaded")
vesseload_X1=input("enter the vesselload strategy from 0 to 3")
yardload_X2=input("enter the yardload strategy from 0 to 3")
vesselunload_X3=input("enter the vessel unload strategy from 0 to 3")
yardunload_X4=input("enter the yard unload strategy from 0 to 3")


variable_vessel=1
variable_truck=1
length_vessel=input("enter the number of containers in vessel across length")
width_vessel=input("enter the number of containers in vessel across width")
height_vessel=input("enter the number of containers in vessel that could be
stacked including the bottom one")
length_yard=input("enter the number of containers in yard across length")
width_yard=input("enter the number of containers in yard across width")
height_yard=input("enter the number of containers in yard that could be stacked
including the bottom one")
vessel_load=np.zeros((length_vessel,width_vessel))
truck_load=np.zeros((length_yard,width_yard))
vessel_empty=np.zeros((2*length_vessel,width_vessel))
truck_empty=np.zeros((2*length_yard,width_yard))
## The following 4 blocks load excel and import that data into array.

for lengt in range(0,length_vessel):
    for widtt in range(0,width_vessel):
        vessel_load[lengt,widtt]=height_vessel

for lengt in range(0, length_yard):
    for widtt in range(0, width_yard):
        truck_load[lengt,widtt]=height_yard

for leng in range(0,length_vessel):
    for widt in range(0, width_vessel):
        vessel_empty[leng,widt]=2*height_vessel

for leng in range(0, length_yard):
    for widt in range(0, width_yard):
        truck_empty[leng,widt]=2*height_yard

vesselside_time=0
yardside_time=0
vesselside_rate=0
yardside_rate=0
dummy=0
dummy1=0
dummy2=0
dummy3=0

count_load=0
count_unload=0
total_vesselunload_time=0
total_truckunload_time=0
total_vesselload_time=0
total_truckload_time=0

def vesselunload():
    global vessel_unloadrate,
dummy1,vessel_unloadtime,i,j,variable_vessel,vessel_load,vessel_empty,rate,vess
el_unloadjob_time_matrix,nth_container_beingunloaded,total_vesselunload_time
    dummy1=0
```

```
    if vesselunload_X3==0:#vertical across v
        for u in range(0, length_vessel):
            for v in range(0, width_vessel):
                for w in range(0, height_vessel):
                    if vessel_load[u, v] > 0:
                        vessel_unloadtime = (2 * (u + 1)) + (2 * (v + 1)) + (2
* vessel_load[u, v])
                        vessel_load[u, v] = vessel_load[u, v] - 1
                        dummy1 = 2
                        break
                if dummy1 == 2:
                    if vessel_load[u, v] == 0:
                        vessel_empty[u, v] = vessel_load[u, v]
                    else:
                        pass
                    break
            if dummy1==2:
                break

    elif vesselunload_X3==1:#horizontal across v
        for u in range(0, length_vessel):
            for w in range(0, height_vessel):
                for v in range(0, width_vessel):
                    if vessel_load[u, v] > 0:
                        vessel_unloadtime = (2 * (u + 1)) + (2 * (v + 1)) + (2
* vessel_load[u, v])
                        vessel_load[u, v] = vessel_load[u, v] - 1
                        dummy1 = 2
                        break
                if dummy1 == 2:
                    if vessel_load[u, v] == 0:
                        vessel_empty[u, v] = vessel_load[u, v]
                    else:
                        pass
                    break
            if dummy1==2:
                break

    elif vesselunload_X3==2:#vertical across u
        for v in range(0, width_vessel):
            for u in range(0, length_vessel):
                for w in range(0, height_vessel):
                    if vessel_load[u, v] > 0:
                        vessel_unloadtime = (2 * (u + 1)) + (2 * (v + 1)) + (2
* vessel_load[u, v])
                        vessel_load[u, v] = vessel_load[u, v] - 1
                        dummy1 = 2
                        break
                if dummy1 == 2:
                    if vessel_load[u, v] == 0:
                        vessel_empty[u, v] = vessel_load[u, v]
                    else:
                        pass
                    break
            if dummy1==2:
                break

    elif vesselunload_X3==3:#horizontal across u
        for v in range(0, width_vessel):
            for w in range(0, height_vessel):
                for u in range(0, length_vessel):
                    if vessel_load[u, v] > 0:
```

```
                        vessel_unloadtime = (2 * (u + 1)) + (2 * (v + 1)) + (2
* vessel_load[u, v])
                        vessel_load[u, v] = vessel_load[u, v] - 1
                        dummy1 = 2
                        break
                if dummy1 == 2:
                    if vessel_load[u, v] == 0:
                        vessel_empty[u, v] = vessel_load[u, v]
                    else:
                        pass
                    break
            if dummy1==2:
                break

    vessel_unloadjob_time_matrix[nth_container_beingunloaded-
1,0]=vessel_unloadtime
    total_vesselunload_time=total_vesselunload_time+vessel_unloadtime
    #vessel_unloadrate = 1.0 / vessel_unloadtime
    #rate[i,j]=vessel_unloadrate
    #variable_vessel=2

def vesselload():
    global
vessel_loadrate,dummy3,vessel_loadtime,i,j,variable_vessel,vessel_loadtime,vess
el_empty,rate,nth_container_beingloaded,vessel_loadjob_time_matrix,total_vessel
load_time
    dummy3=0
    if vesseload_X1==0:# vertical across l
        for k in range(0, 2*length_vessel):
            for l in range(0, width_vessel):
                for m in range(0, height_vessel):
                    if vessel_empty[k, l] < height_vessel:
                        vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) + (2 *
(vessel_empty[k, l]+1))
                        vessel_empty[k, l] = vessel_empty[k, l] + 1
                        dummy3 = 2
                        break
                if dummy3 == 2:
                    break
            if dummy3 == 2:
                break

    elif vesseload_X1==1:#horizontal across l
        for k in range(0, 2*length_vessel):
            for m in range(0, height_vessel):
                for l in range(0, width_vessel):
                    if vessel_empty[k, l] < height_vessel:
                        vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) + (2 *
(vessel_empty[k, l]+1))
                        vessel_empty[k, l] = vessel_empty[k, l] + 1
                        dummy3 = 2
                        break
                if dummy3 == 2:
                    break
            if dummy3 == 2:
                break

    elif vesseload_X1==2:#vertical across k
        for l in range(0, width_vessel):
            for k in range(0, 2*length_vessel):
                for m in range(0, height_vessel):
                    if vessel_empty[k, l] < height_vessel:
```

```
                        vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) + (2 * 
(vessel_empty[k, l]+1))
                        vessel_empty[k, l] = vessel_empty[k, l] + 1
                        dummy3 = 2
                        break
                if dummy3 == 2:
                    break
            if dummy3 == 2:
                break

    elif vesseload_X1==3:#horizontal acroos k
        for l in range(0, width_vessel):
            for m in range(0, height_vessel):
                for k in range(0, 2*length_vessel):
                    if vessel_empty[k, l] < height_vessel:
                        vessel_loadtime = (2 * (k + 1)) + (2 * (l + 1)) + (2 * 
(vessel_empty[k, l]+1))
                        vessel_empty[k, l] = vessel_empty[k, l] + 1
                        dummy3 = 2
                        break
                if dummy3 == 2:
                    break
            if dummy3 == 2:
                break

    vessel_loadjob_time_matrix[nth_container_beingloaded-1,0]=vessel_loadtime
    total_vesselload_time=total_vesselload_time+vessel_loadtime
    #vessel_loadrate = 1.0 / vessel_loadtime
    #rate[i,j]=rate[i,j]+vessel_loadrate
    #variable_vessel=1

def truckload():
    global dummy2, 
truck_loadtime,truck_loadrate,i,j,variable_truck,truck_empty,truck_load,rate,co
unt,nth_container_beingloaded,truck_loadjob_time_matrix,total_truckload_time
    dummy2=0
    if yardload_X2 == 0:  # Vertical and across b
        for a in range(0, length_yard):
            for b in range(0, width_yard):
                for c in range(0, height_yard):
                    if truck_load[a, b] > 0:
                        truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2 * 
truck_load[a, b])
                        truck_load[a, b] = truck_load[a, b] - 1
                        dummy2 = 2
                        break
                if dummy2 == 2:
                    if truck_load[a, b] == 0:
                        truck_empty[a, b] = truck_load[a, b]
                    else:
                        pass
                    break
            if dummy2==2:
                break

    elif yardload_X2 == 1:  # Horizontal and across b
        for a in range(0, length_yard):
            for c in range(0, height_yard):
                for b in range(0, width_yard):
                    if truck_load[a, b] > 0:
                        truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2 * 
truck_load[a, b])
```

```
                        truck_load[a, b] = truck_load[a, b] - 1
                        dummy2 = 2
                        break
                if dummy2 == 2:
                    if truck_load[a, b] == 0:
                        truck_empty[a, b] = truck_load[a, b]
                    else:
                        pass
                    break
            if dummy2==2:
                break

    elif yardload_X2 == 2:  # Vertical and across a
        for b in range(0, width_yard):
            for a in range(0, length_yard):
                for c in range(0, height_yard):
                    if truck_load[a, b] > 0:
                        truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2 *
truck_load[a, b])
                        truck_load[a, b] = truck_load[a, b] - 1
                        dummy2 = 2
                        break
                if dummy2 == 2:
                    if truck_load[a, b] == 0:
                        truck_empty[a, b] = truck_load[a, b]
                    else:
                        pass
                    break
            if dummy2==2:
                break

    elif yardload_X2 == 3:  # Horizontal and across a
        for b in range(0, width_yard):
            for c in range(0, height_yard):
                for a in range(0, length_yard):
                    if truck_load[a, b] > 0:
                        truck_loadtime = (2 * (a + 1)) + (2 * (b + 1)) + (2 *
truck_load[a, b])
                        truck_load[a, b] = truck_load[a, b] - 1
                        dummy2 = 2
                        break
                if dummy2 == 2:
                    if truck_load[a, b] == 0:
                        truck_empty[a, b] = truck_load[a, b]
                    else:
                        pass
                    break
            if dummy2==2:
                break

    truck_loadjob_time_matrix[nth_container_beingloaded-1,0]=truck_loadtime
    total_truckload_time=total_truckload_time+truck_loadtime
    #truck_loadrate = 1.0 / truck_loadtime
    #rate[i,j]=truck_loadrate
    #variable_truck=1

def truckunload():
    global dummy,
truck_unloadtime,truck_unloadrate,i,j,variable_truck,truck_empty,rate,count,nth
_container_beingunloaded,truck_unloadjob_time_matrix,total_truckunload_time
    dummy=0
    if yardunload_X4 == 0:  # vertical and across d
```

```
        for c in range(0, 2*length_yard):
            for d in range(0, width_yard):
                for e in range(0, height_yard):
                    if truck_empty[c, d] < height_yard:
                        truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) + (2 *
(truck_empty[c, d]+1))
                        truck_empty[c, d] = truck_empty[c, d] + 1
                        dummy = 2
                        print c,d
                        print truck_empty[c,d]
                        break
                if dummy == 2:
                    break
            if dummy==2:
                break

    elif yardunload_X4 == 1:  # Horizontal and across d
        for c in range(0, 2*length_yard):
            for e in range(0, height_yard):
                for d in range(0, width_yard):
                    if truck_empty[c, d] < height_yard:
                        truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) + (2 *
(truck_empty[c, d]+1))
                        truck_empty[c, d] = truck_empty[c, d] + 1
                        dummy = 2
                        break
                if dummy == 2:
                    break
            if dummy==2:
                break

    elif yardunload_X4== 2:  # Vertical and across c
        for d in range(0, width_yard):
            for c in range(0, 2*length_yard):
                for e in range(0, height_yard):
                    if truck_empty[c, d] < height_yard:
                        truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) + (2 *
(truck_empty[c, d]+1))
                        truck_empty[c, d] = truck_empty[c, d] + 1
                        dummy = 2
                        break
                if dummy == 2:
                    break
            if dummy==2:
                break

    elif yardunload_X4 == 3:  # Horizontal and across c
        for d in range(0, width_yard):
            for e in range(0, height_yard):
                for c in range(0, 2*length_yard):
                    if truck_empty[c, d] < height_yard:
                        truck_unloadtime = (2 * (c + 1)) + (2 * (d + 1)) + (2 *
(truck_empty[c, d]+1))
                        truck_empty[c, d] = truck_empty[c, d] + 1
                        dummy = 2
                        break
                if dummy == 2:
                    break
            if dummy==2:
                break

    truck_unloadjob_time_matrix[nth_container_beingunloaded-
```

```
1,0]=truck_unloadtime
    total_truckunload_time=total_truckunload_time+truck_unloadtime
    #truck_unloadrate = 1.0 / truck_unloadtime
    #rate[i,j]=rate[i,j] + truck_unloadrate
    #variable_truck=2

print "time"

nth_container_beingloaded=0
nth_container_beingunloaded=0

truck_loadjob_time_matrix=np.zeros((nl,1))
truck_unloadjob_time_matrix=np.zeros((nu,1))
vessel_loadjob_time_matrix=np.zeros((nl,1))
vessel_unloadjob_time_matrix=np.zeros((nu,1))

if nl>0:
    for nth_container_beingloaded in range(nl,0,-1):
        truckload()
        vesselload()
if nu>0:
    for nth_container_beingunloaded in range(nu,0,-1):
        truckunload()
        vesselunload()

if nu==0:
    total_truckunload_time=0
    total_vesselunload_time=0
    nu=1#To make the average value to be zero

if nl==0:
    total_truckload_time=0
    total_vesselload_time=0
    nl=1##To make the average value to be zero

state=np.zeros((7,1))
state[2,0]=1
state[1,0]=2
state[0,0]=3
state[3,0]=4
state[4,0]=5
state[5,0]=6
state[6,0]=7



rate=np.zeros((7,7))

#vesselside_rate=sch_rate
#yardside_rate=sch_rate
quaycrane_time=300
yardcrane_time=300

global total_time
global total_time_processing

total_time=0
total_time_processing=0


total_time=(total_time_processing+0.5*(total_truckload_time+total_vesselload_ti
me+total_vesselunload_time+total_truckunload_time))/(nu+nl)
```

```
print "total time is",total_time

length_state=7

#print rate
for i in range(0,length_state):
    for j in range(0,length_state):
# Refer the following 2 codes
        if state[i,0]==1 and state[j,0]==5:#(nu,nl,x,x,x)'s next transition
provided nu and nl remain the same in next transition ie, transitions like
(2,2,0,1,0) to (2,2,1,0,0)
            rate[i,j]=(1.0 /((2*yardcrane_time)+(total_truckload_time/nl)))
        if state[i,0]==1 and state[j,0]==6:#(nu,nl,x,x,x)'s next transition
provided nu and nl remain the same in next transition ie, transitions like
(2,2,0,1,0) to (2,2,1,0,0)
            rate[i,j]=(1.0 /((2*yardcrane_time)+(total_truckload_time/nl)))
        elif state[i,0]==2 and state[j,0]==3:
            rate[i,j]=(1.0/((2*quaycrane_time)+(total_vesselunload_time/nu)))
        elif state[i,0]==2 and state[j,0]==4:
            rate[i,j]=(1.0/((2*quaycrane_time)+(total_vesselunload_time/nu)))
        elif state[i,0]==3 and state[j,0]==7:
            rate[i, j] = (1.0 / (2*yardcrane_time)+(total_truckunload_time/nu))
        elif state[i,0]==4 and state[j,0]==7:
            rate[i, j] = (1.0 / (2*yardcrane_time)+(total_truckunload_time/nu))
        elif state[i,0]==5 and state[j,0]==7:
            rate[i, j] = (1.0 / (2*quaycrane_time)+(total_vesselload_time/nl))
        elif state[i, 0] == 6 and state[j, 0] == 7:
            rate[i, j] = (1.0 / (2 * quaycrane_time) + (total_vesselload_time /
nl))
        else:
            pass

print "below is rate"



for i in range(0,length_state):
    for j in range(0,length_state):
        if i!=j:
            rate[i][i] = (rate[i][i] + rate[i][j])
        else:
            pass

for i in range(0,length_state):
       rate[i][i]=-(rate[i][i])

print rate

print state

print length_state

q=np.zeros((length_state-1,length_state-1))
for i in range(0,length_state-1):
    for j in range(0,length_state-1):
        q[i][j]=rate[i][j]

print q
#print del_opn
```

```python
q_neg=-q

print q_neg
from numpy.linalg import inv

e=inv(q_neg)
#ident_matrix=np.identity(length_state-1)

#e=solve(q_neg,ident_matrix)
duration=np.zeros((length_state,1))

for i in range(0,length_state-1):
    for j in range(0,length_state-1):
        duration[i,0]=duration[i,0]+e[i,j]

print duration

print e

r=np.zeros((6,1))

j_r=0
j=6
for i in range(0,length_state-1):
    r[i][0]=rate[i,j]

print "total processing time",total_time_processing

a_matrix=np.matmul(e,r)

a_neg=np.zeros((len(a_matrix),1))

for i in range(0,len(a_matrix)):
    a_neg[i,0]=-a_matrix[i,0]

print a_neg

q_mult_a=np.matmul(q,a_matrix)

print "q_mult_a"
print q_mult_a
print a_matrix

total_time_processing=0

mean_time=np.linalg.solve(q,a_neg)

print mean_time

print mean_time[1,0]
print mean_time[2,0]
```