INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Algorithms for improved efficiency in transportation models

Moyer, Christopher Allen, Ph.D.

The Ohio State University, 1992
ALGORITHMS FOR IMPROVED EFFICIENCY IN 
TRANSPORTATION MODELS 

DISSERTATION 

Presented in Partial Fulfillment of the Requirements for the 
Degree Doctor of Philosophy in the Graduate School of the Ohio 
State University 

By 

Christopher Allen Moyer  B.S., M.B.A., M.A. 

* * * * * 

The Ohio State University 
1992
To my Parents and Kelly Brunarski
ACKNOWLEDGEMENTS

I express sincere appreciation to Dr. Rakesh Vohra for his guidance and insight throughout the research. Thanks go to the other members of my advisory committee, Drs. David Schilling and Wan Soo Rhee, for their suggestions and comments.
VITA

June 11, 1963........................ Born - Dayton, Ohio

1985 .................................. B.S., Wright State
     University, Dayton, Ohio

1986 .................................. M.B.A., Wright State
     University, Dayton, Ohio

1991 - Present ........................ Lecturer, The Ohio State
     University

FIELDS OF STUDY

Major Field: Business Administration

Studies in: Scheduling Algorithms
            Transportation Modeling
            Integer Programming Applications
# TABLE OF CONTENTS

DEDICATION ........................................................................... ii  
ACKNOWLEDGEMENTS ..................................................... iii  
VITA .................................................................................... iv 
LIST OF TABLES .................................................................... vii 
LIST OF FIGURES ............................................................. viii 

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. LITERATURE REVIEW</td>
<td>5</td>
</tr>
<tr>
<td>The Signal Setting Problem</td>
<td>5</td>
</tr>
<tr>
<td>Scheduling the Loading of Crates into the Hold of a Ship</td>
<td>6</td>
</tr>
<tr>
<td>Crane Scheduling for Efficient Processing of Berthed Ships</td>
<td>9</td>
</tr>
<tr>
<td>III. A FAST ALGORITHM FOR SIGNAL SETTING AT TRAFFIC JUNCTIONS</td>
<td>12</td>
</tr>
<tr>
<td>Introduction</td>
<td>12</td>
</tr>
<tr>
<td>The Models</td>
<td>15</td>
</tr>
<tr>
<td>The Solution Procedure</td>
<td>18</td>
</tr>
<tr>
<td>A Simple Heuristic</td>
<td>22</td>
</tr>
<tr>
<td>Examples &amp; Computational Results</td>
<td>24</td>
</tr>
<tr>
<td>IV. CRANE SCHEDULING TO MINIMIZE CRATE SHIFTS AT A PORT</td>
<td>28</td>
</tr>
<tr>
<td>Introduction</td>
<td>28</td>
</tr>
<tr>
<td>The Model</td>
<td>29</td>
</tr>
<tr>
<td>Solution Algorithm</td>
<td>32</td>
</tr>
<tr>
<td>Examples &amp; Computational Results</td>
<td>34</td>
</tr>
<tr>
<td>V. AN OPTIMAL CRANE SCHEDULING ALGORITHM</td>
<td>38</td>
</tr>
<tr>
<td>Introduction</td>
<td>38</td>
</tr>
<tr>
<td>An Integer Programming Formulation</td>
<td>39</td>
</tr>
<tr>
<td>Computational Results</td>
<td>42</td>
</tr>
</tbody>
</table>
VI. CONCLUSIONS AND FURTHER RESEARCH ....................... 45

The Signal Setting Problem .......................... 45
Scheduling the Loading of Crates into the Hold of a Ship .......................... 46
Crane Scheduling for Efficient Processing of Berthed Ships .......................... 47

LIST OF REFERENCES ....................................... 50

APPENDICES

A. Figures ........................................ 53

vi
<table>
<thead>
<tr>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Incompatibility matrix for the junction in Figure 1 .... 13</td>
</tr>
<tr>
<td>2. Sample data for a junction where n=12 .................. 25</td>
</tr>
<tr>
<td>3. Performance comparison: BMILP vs. FSPACE ............... 26</td>
</tr>
<tr>
<td>4. Test results for the crate assignment algorithm (Stability criteria relatively loose.) ........................ 36</td>
</tr>
<tr>
<td>5. Test results for the crate assignment algorithm (Stability criteria relatively tight.) ...................... 37</td>
</tr>
<tr>
<td>6. Performance comparison of crane scheduling algorithms .. 43</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Traffic junction where ( n=6 )</td>
<td>54</td>
</tr>
<tr>
<td>2. Incompatibility graph for the junction illustrated in Figure 1</td>
<td>55</td>
</tr>
<tr>
<td>3. The circle graph associated with the junction in Figure 1</td>
<td>56</td>
</tr>
<tr>
<td>4. The overlap graph associated with the circle graph in Figure 3</td>
<td>57</td>
</tr>
<tr>
<td>5. Traffic junction where ( n=4 )</td>
<td>58</td>
</tr>
<tr>
<td>6. Traffic junction where ( n=6 )</td>
<td>59</td>
</tr>
<tr>
<td>7. Traffic junction where ( n=12 )</td>
<td>60</td>
</tr>
<tr>
<td>8. Traffic junction where ( n=20 )</td>
<td>61</td>
</tr>
<tr>
<td>9. Traffic junction where ( n=30 )</td>
<td>62</td>
</tr>
<tr>
<td>10. Cross-section of Ship</td>
<td>63</td>
</tr>
<tr>
<td>11. Relationship between yard stack and transtainer</td>
<td>64</td>
</tr>
<tr>
<td>12. Overhead view of port layout</td>
<td>65</td>
</tr>
<tr>
<td>13. Cranes for unloading berthed ships</td>
<td>66</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

In recent years there has been an increased awareness of the need for efficient solution procedures for the wide spectrum of transportation models. A primary motivation for this increased awareness is the ever-increasing worldwide competition in the transportation industry. To remain competitive, companies, governments, and military branches must operate their transport systems as efficiently as possible.

More efficient operation of traffic junctions is one area of transportation that could benefit many parties. As the operation of traffic junctions becomes more efficient, fewer motor vehicles will experience lengthy delays. Reduced delays mean that transporters of goods and services will be able to reduce costs. Consumers would then benefit from lower prices. And there would be environmental benefits. Fewer vehicles experiencing lengthy delays means that less fuel will be consumed and less air pollution will be produced.

Improving efficiency is a major goal of the maritime community as well. In 1988 shipping accounted for 79.5% by dollars and 99.8% by tonnage of all U.S. overseas trade. In recent years there have been numerous technological changes that
have improved the efficiency of shipping operations. Our focus will be on developing operational improvements. Specifically, we discuss algorithms to increase the efficiency of port operations.

Planning the loading and unloading of cargo ships is one aspect of port operations where increased efficiency could greatly reduce costs. The more efficient the load planning scheme, the shorter the delays encountered by vessels at a port. Reducing delays will allow for a reduction in costs and help transporters maintain their competitive position. Military branches would also benefit greatly from more efficient load planning schemes. A reduction in port delays means better responsiveness to global activities, which is important to the military, just as it is to private industry.

The literature on mathematical modeling for transportation problems is, in general, extensive. There are, however, some topics in transportation for which the literature is sparse. Little work has been done on the efficient scheduling of signals at traffic junction. Much of what has been done is heuristic in nature. Very little work has been done on the efficient operation of dock-side cranes at ports. In practice, most decisions about the operation of dock-side cranes are based either on simulation or the practical experience of the decision makers.

In this paper we develop algorithms for a variety of transportation problems which have not yet been widely studied. The first problem we study is that of scheduling signal lights at a traffic junction. The signal setting problem for a traffic
junction involves setting the cycle time and the green times and sequencing them so as to maximize a measure of the flow thru the junction. The solution procedure we develop takes advantage of structural characteristics within the junction to find the best signal sequence and the optimal time for each green signal.

The second transportation problem we study is that of assigning dockside cargo crates to positions in a hold of a designated cargo ship. Cargo are initially stacked at a holding yard adjacent to the ship's berth. They must be loaded using a Gantry crane or transtainer system, one at a time, onto a flatbed truck which takes them to a position adjacent to the ship, where they are loaded into the hold of the ship. The objective of this problem is to find the order in which to transfer the crates from the yard stack to the ship. Our algorithm determines the loading schedule which optimizes port efficiency, while maintaining an acceptable level of ship stability. Our solution procedure involves solving a specially structured assignment problem repeatedly until we have a solution which satisfies stability requirements and maximizes port efficiency.

The third transportation problem we study is that of scheduling port cranes in an attempt to load or unload ships in the most efficient manner. The problem we study starts with several ships berthed at a port. Each ship has multiple holds which will require some work time to be loaded or unloaded. There are multiple shore cranes which must be shared among the ships and their holds. Our objective is to determine the crane work
schedule which minimizes the total weighted delay times (and, therefore, costs) for the ships currently berthed. We develop an integer programming model for this problem.
CHAPTER II
LITERATURE REVIEW

As previously noted, there are areas within the broad field of transportation that have received little attention in the literature. This is true of the problems we study. The work that has been done in these areas often is heuristic in nature and/or requires limiting assumptions that restrict the usefulness of the models presented. There have been some effective solution approaches to the problems we study, which we will discuss here and evaluate later.

2.1 The Signal Setting Problem

Initially, the signal setting problem was studied assuming the sequencing of the green times, expressed by a stage matrix, to be known (Allsop, 1971; Yagar, 1974). The stage matrix lists the order in which groups will receive their green signal. Such an approach is not guaranteed to yield an optimal solution.

Subsequently, these models were generalized so as to remove the requirement that the stage matrix be fixed (Allsop and Murchland, 1978). These models could not guarantee optimal solutions for large junctions within acceptable computational guidelines.
In a more recent paper (Improta and Cantarella, 1984) this problem is formulated as a binary-mixed-integer-linear program (BMILP). This model does not require the elements of the stage matrix as explicit variables, and therefore removes some of the limiting assumptions of the previous models. In our computational tests (the results of which are shown in Table 3) it is seen that this BMILP solves in several seconds for smaller problems, but requires more than two minutes of CPU time for problems with five or more inroads.

More recently a solution method based on graph theory has been presented together with the previous BMILP model (Cantarella and Improta, 1988). Their solution method is based on an idea first described by Stoffers (1968), where the minimum cycle time can be determined by considering sets (cliques) of mutually incompatible groups. This graph theory-based approach cannot assure an optimal solution because it assumes, for example, the absence of odd holes in the incompatibility graph. An odd hole is a set of nodes, odd in number and greater in size than three, that are connected in a loop without any chords (the degree of each node is two).

2.2 Scheduling the Loading of Crates into the Hold of a Ship

In modern sea transportation systems, it is critically important to process ships efficiently while they are berthed. The potential for cost savings is large; an average cargo liner spends 60% of its time in port (Imakita, 1978). Further, ship owners realize costs on the order of $1000 for each hour that a
ship spends in port (Sabria, 1986), whether it is being processed or not.

This problem, and ones similar to it, have been studied in the past. First note that the problem here is different from the one studied by Cojeen and Van Dyke (1976). They studied a similar problem where the crate handling was done by a straddle carrier system. This differs from the transtainer system in that straddle carriers can move about in the yard and pull crates from different stacks. Cojeen and Van Dyke consider ship stability in their model, but only the concern of being top-heavy. The primary objective of their model is to minimize the crane traffic interference when there is more than one straddle carrier working in the yard. They develop a load planning heuristic, which is then implemented in what is to date the most complete computer assisted load planning system available. In his dissertation, Imai (1988) uses an objective function which is a measure of the expected number of shifts required for a particular assignment. His model minimizes the expected number of shifts for a straddle carrier-based system which employs one straddle carrier to process crates in one stack.

Some work has been done in the area of transtainer-based systems, though all of the previous solution methods are based on heuristics. Martin et. al. (1988) use a heuristic, in the form of a flow chart, to make the load planning decisions. The logic for the flow chart is very much the same as the logic used in the Masters Thesis presented by Martin (1982). The heuristic
considers the size of the crates, restrictions for refrigerated cargo, and other relevant considerations. It seeks to minimize the number of crate "rehandles," which include shifts at the current yard, and shifts necessary when a crate to be unloaded first is placed below a crate to be unloaded later. It ignores the matter of ship stability, reasoning that most ships now have elaborate ballast systems to aide in the ship's stability. This is true for fore vs. aft and port vs. starboard stability, but not for vertical center of mass, radial stability. The heuristic is tested on two ships, on four voyages each, based at Portland, Oregon. It is noted that the heuristic could have reduced a "composite handling measure" by an average of 4.8%. It is unclear how it would have performed in a general setting.

Other works have tried to formulate this problem as an integer program. Beliech (1984) initially attempted to model this problem with an integer program, but dismissed the idea because of difficulties encountered in problem formulation and the anticipation of unacceptable solution times. Beliech (1984) then implemented a heuristic less complete than that of Martin et. al. Cho (1981) formulated this problem as a mixed integer program with assignment variables relating the current crate position and the eventual position in the ship. Cho quickly concluded that his model was too large to solve as an integer program. He further claimed that the model was too large to solve as a linear program as well. Scott and Chen (1978) formulated this problem with a constrained assignment model similar to the one mentioned later
no attempt was ever made to solve their model, and they too concluded that the model was unsolvable.

2.3 Crane Scheduling for Efficient Processing of Berthed Ships

Many past studies have dealt with other aspects of the port environment. Some work has been done on the efficiency of operations at the port holding yard. Cojeen and Van Dyke (1976), and Martin et al. (1982) study the problem of scheduling cranes at a holding yard adjacent to the berthing area. Other studies address other aspects of the port operation. Frankel (1974), among others, studies the problem of efficient port design. The berthing system is among the most widely studied aspect of port operations, perhaps because it fits nicely into the realm of queueing theory. Sabria (1986) surveys the literature on port operations in general.

Very little previous work, though, addresses the problem of scheduling specialized shore cargo-handling equipment. While some ships may carry their own cranes for loading and unloading cargo, often a ship's port time is largely affected by the availability of port-side shore cranes. This paper studies the scheduling of these shore cranes with respect to the overall efficiency of the port system.

Past work on this particular problem is scarce. Daganzo (1989) solves this problem using an integer program for the assignment of cranes for a series of time periods. One drawback to the IP they present is that the number of variables and constraints in their model grows linearly with the total work.
time required on the ships that are currently berthed. They also assume integer work requirements for each hold, which may not be appropriate. They note that their model will provide a solution within a reasonable time for models with no more than four ships. While it is typically the case that no more than four to six ships will be berthed at any time, it would be useful to have a model that could accommodate a larger number of ships if necessary. They then discuss a heuristic based on some very simple crane scheduling principles and some common sense observations. The heuristic can find solutions for larger problems in a reasonable amount of time, but it cannot guarantee that the solution found is optimal. They discuss a dynamic version of the problem, where ship arrivals at a later date must be considered in the scheduling process. This problem is typically referred to as the "release date" problem, with deterministic work requirements. They explain how their heuristic can be modified to solve such a problem.

A Branch and Bound solution procedure for this problem is presented by Peterkofsky and Daganzo (1990). Their procedure utilizes some observations about the physical characteristics of the model to develop both branching and fathoming criteria. The model size still grows linearly with the total work time requirement, and still assumes that the work required to complete a hold is integer. They provide computational results for problems with anywhere from two to ten ships. Comparisons with the results of their testing, using the same data, are provided
as a basis for evaluation of the solution procedure presented in Chapter V.
CHAPTER III
A FAST ALGORITHM FOR SIGNAL SETTING AT TRAFFIC JUNCTIONS

The efficiency of nearly every land-based company will be affected to some degree by the efficiency of the roadway transport system in its region. An important part of the effectiveness of most roadway transport systems is the operation of the traffic lights at signalized junctions within the system. In this chapter we develop a procedure that allows us to determine the most efficient signal schedule for a junction.

3.1 Introduction

A traffic junction consists of a number of roads leading into a junction. These are referred to as inroads. For each inroad there will be one or more streams, or lanes, of traffic. A set of traffic streams that receives the same signal from the control system is defined as a group. Roads that allow for traffic to flow away from the junction are called outroads. Each group is assigned a starting time and an ending time for its green-amber period. The sequence of these periods will constitute a schedule. The total elapsed time from when one green period begins to when it begins again is known as the cycle time. The junction capacity factor is a measure of the worst case flow for any group. That is, it is the minimum ratio of the green time for a group divided
by a flow factor for that group, which is given. The flow factor for a group is a constant representing the volume of traffic for that group.

Groups that cannot proceed through the junction safely at the same time are said to be incompatible. Figure 1 illustrates a small junction, and Table 1 is its associated incompatibility matrix. This will be a square matrix with entries $a_{i,j}$ equal to 1 if group $i$ and group $j$ are incompatible, 0 otherwise.

Associated with the incompatibility matrix is an incompatibility graph. This graph will have a node for each group. There will be an arc between two nodes if the groups represented by the nodes are incompatible. Figure 2 illustrates the associated incompatibility graph.

**Table 1**

Incompatibility matrix for the junction in Figure 1

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Our solution procedure will determine for us the sequence in which each group gets a green signal and the length of the green signal. Two objectives are typically considered for this problem. One is to maximize a measure of flow thru the junction; the other is to minimize the cycle time for the signal. Here, as in most past works, we fix the cycle time and maximize the flow of traffic thru the junction.

In this paper we propose a solution procedure that does not make the restrictive assumptions of earlier work. This solution procedure is more efficient than past procedures, very often requiring only a simple heuristic to find the optimal solution. A more efficient solution procedure is important for several reasons. If one wishes to analyze the tradeoff between the cycle time and the junction flow factor, the model will need to be solved for each of several cycle times. It is also likely a different optimal signal setting will need to be determined periodically throughout the day as traffic flow varies.

In Section 3.2 we state the problem in terms of the model introduced in Cantarella and Improta (1988). We also discuss the manipulations that lead to the reduced model we solve and the complexity of the problem to be solved. In Section 3.3 we give an outline of our solution procedure. Section 3.4 discusses the scheduling heuristic used as a first step in solving the model, and Section 3.5 provides computational results.
3.2 THE MODELS

In this section we present a variation of the BMILP discussed in Cantarella and Improta (1988). The objective is to maximize the junction capacity factor subject to a fixed cycle time. The decision variables are the starting time of the green and the ending time of the amber for each group, the total cycle time, and a binary precedence variable for each pair of incompatible groups. The cycle time is measured in seconds and except for the lag times all other variables relative to time are expressed as a fraction of the cycle time.

Variables

- \( n \) = number of groups of the junction;
- \( A = (a_{ij}) \) = incompatibility matrix, \( a_{ij} = a_{ji} = 1 \) if groups \( i, j \) are incompatible, 0 otherwise (will be given)
- \( z = \) the inverse of the cycle time \( c \) (will be given)
- \( f \) = the junction capacity factor
- \( y_k \) = the flow factor for group \( k \) (given)
- \( v_k \) = the ending time of amber for group \( k \) (dec. variable)
- \( u_k \) = the starting time of green for group \( k \) (dec. variable)
- \( g_k = \) the effective green time for group \( k = v_k - u_k - l_kz \)
- \( l_k = \) the lost (lag) time for group \( k \) (given)
- \( w_{ij} = 0 \) if the green of group \( i \) proceeds the green of group \( j \), 1 otherwise) (dec. variable)
- \( r_{\text{max},k} = \) the maximum red time for group \( k \) (given)
\( g_{\text{min},k} \) = the minimum green time for group \( k \) (given)

**The Model (P1)**

\[
\text{max. } f
\]

\[
s.t. \ g_k \geq f_k \quad k = 1, \ldots, n \tag{1}
\]

\[
v_k - u_k = g_k + \lambda_k z \quad k = 1, \ldots, n \tag{2}
\]

\[
v_i - u_j \geq -1 \quad \text{for all incompatible pairs } i, j \tag{3}
\]

\[
v_i - u_j \leq 1 \quad \text{for all incompatible pairs } i, j \tag{4}
\]

\[
v_i - u_j \leq w_{ij} \quad \text{for all incompatible pairs } i, j \tag{5}
\]

\[
v_j - u_i \leq 1 - w_{ij} \quad \text{for all incompatible pairs } i, j \tag{6}
\]

\[
w_{ij} + w_{ji} = 1 \quad \text{for all incompatible pairs } i, j \tag{7}
\]

\[
0 \leq v_k \leq 1, -1 \leq u_k \leq 1, w_{ij} = 0/1
\]

Constraint set (1) holds for all groups and insures that the junction capacity factor will be the minimum ratio of a group's green time to its flow factor. Constraint sets (3) and (4) guarantee that there will not be any overlap of the green times for any incompatible groups. Constraint sets (5), (6), and (7) combine to guarantee that for any two incompatible groups, one must follow the other.

The above model can be modified to a form that can be solved more efficiently. To do this we first eliminate dependent constraints, as noted in Cantarella and Improta (1988). We then note that \( u_k = v_k - g_k - \lambda_k z \), and we define a new variable \( \alpha_k \) by the following: \( \alpha_k = 1 - (g_k + \lambda_k z) \). We can now substitute for \( u_k \).
in the model. We fix the cycle time at some value based on our
d prior knowledge of the junction. This can easily be varied later
if one wishes to evaluate the trade-off between the cycle time
and the junction capacity factor. Under these conditions problem
(P1) takes on the form:

The Model (P2)

\[
\begin{align*}
\text{max. } f \\
\text{s.t. } g_k &\geq f y_k \quad \text{for all groups } k \\
\nu_i - \nu_j &\leq w_{ji} + \alpha_j - 1 \quad \text{for all incompatible } (i,j) \\
\nu_j - \nu_i &\leq \alpha_i - w_{ij} \quad \text{for all incompatible } (i,j) \\
w_{ij} + w_{ji} &- 1 \quad \text{for all incompatible } (i,j) \\
0 &\leq \nu_k \leq 1; \ w_{ij} = 0/1
\end{align*}
\]

We will solve this problem using binary search over the
range of possible values for \( f \). For the moment suppose that we
have fixed the value of \( f \). (The procedure for doing this is
discussed in section 3.3.) This in turn fixes the values of \( g_k \);
that is, the green times are fixed. This fixes the \( \alpha_k \) as well.

Now our problem reduces to:

'Is there a sequence of the traffic groups that 'fits'
within the cycle time and does not violate any incompatibility
restrictions?'

This problem can be formulated as:

The Model (P3)

\[
\begin{align*}
\nu_i - \nu_j &\leq w_{ji} + \alpha_j - 1 \quad \text{for all incompatible } (i,j)
\end{align*}
\]
\[ v_j - v_i \leq \alpha_i - w_{ij} \quad \text{for all incompatible } (i,j) \]  
\[ w_{ij} + w_{ji} = 1 \quad \text{for all incompatible } (i,j) \]  
\[ 0 \leq v_k \leq 1; w_{ij} = 0/1 \]

This representation of (P3) illustrates that it is a special case of a scheduling problem discussed in Balas (1985).

The problem of finding a feasible solution to a similar scheduling problem is shown to be NP-complete in Serafini and Ukovich (1989). Similar arguments can be used to show that (P3) is NP-complete as well.

3.3 THE SOLUTION PROCEDURE

The solution procedure described below will be referred to as FSPACE. We first find lower and upper bounds on \( f \). To find a lower bound consider each inroad to the junction separately. At a particular inroad determine the group that would require the longest green time for a fixed \( f \). Call these the critical groups. This will be a function of the lag time and the \( y_i^\_ \) for each group. Define the set of these critical groups from each inroad as \( G \). Now schedule these critical groups in series. Assign green times such that \( g_i/y_i = \sum_j l_j \) for all groups in \( G \). To get a good lower bound we set the green times of the groups in \( G \) so that:

\[ \sum (g_i \mid i \in G) = 1 - \sum (l_i \mid i \in G) \]  

(15)

To set the green times of the groups in \( G \), first select some group \( i \) from \( G \). For all other groups in \( G \) set \( g_j = (y_j/y_i)(g_i) \). All coefficients of \( g_i \) will be positive, so when we substitute
for all $g_j$ in (15), we are assured a positive value for $g_i$ if the sum of the lag times for groups in $G$ is less than the cycle time. This also assures that the other groups in $G$ will have positive green times. Now schedule all other groups at each inroad for the same green time as the critical group at that inroad. The lower bound on the optimal $f$, $f_1$, will be $g_i/y_i$ for any group $i \in G$.

Theorem 3.1 - The above procedure results in a feasible schedule with a lower bound on the optimal value of $f$.

Proof - The proof of feasibility is based on the fact that all groups at any inroad are compatible with one another. The procedure schedules each of the 'largest' groups in series. We now schedule all other groups at a particular inroad within the same time interval as the 'largest' group from that inroad. Since all groups at an inroad are compatible, we necessarily have a feasible setting. And any feasible setting gives us a lower bound on $f$. □

In establishing an upper bound on $f$ we consider any clique in the incompatibility graph. For a clique we schedule the groups represented by the clique in series and maximize $f$ given that this series must fit within the set cycle time. We again set the ratio of $g/y$ to be equal for all groups in the clique. (This maximizes the minimum ratio.) The sum of the green times and lag times (as a percentage of the cycle time) of groups in the clique are set equal to 1. The $f_u = g/y$ ratio for any group in the clique now determines an upper bound on $f$. 
Theorem 3.2 - The value of \( f_u \) determined by the above procedure, for any clique in the incompatibility graph, represents an upper bound on the optimal value of \( f \).

Proof - Assume that \( f_u \) is not an upper bound on \( f \). This implies that for some group \( i \) in the clique, we could increase \( f \) by increasing the \( g_i \) for the group that has the smallest \( g/y \) ratio. Note, though, that we have already set the \( g/y \) ratios for groups in the clique to be equal, and the green times have been set as large as possible given the current cycle time. To increase the green time of one group and maintain feasibility we must decrease the green time of at least one other group in the clique. This will lower \( f \), which is a contradiction. □

While any clique will give us an upper bound on \( f \), the best upper bound using this method will be given by a maximal, or critical clique. Cliques can be compared in terms of the sum of the \( y_i \) for the members of the clique. Due to the special structure of this problem, it can be shown that determining the critical clique can be done in \( O(n^2 + m \log \log n) \) time, where \( n \) is the number of vertices and \( m \) the number of edges in the incompatibility graph.

To verify that finding the maximum weight clique in our problems can be done in polynomial time, we first consider the following transformation of a problem with 6 groups, as seen in Figure 1. The junction can be represented by the circle graph.
displayed in Figure 3. The corresponding overlap graph is displayed in Figure 4. In the circle graph each S represents an inroad and each T represents an outroad. The arcs represent traffic flow. Any intersection of the arcs represents an incompatibility of flow. Further, at any outroad $T_i$ the arcs leading into the outroad must be viewed as being incompatible.

We make this transformation because of the availability of polynomial time algorithms for finding a maximum weight clique in a circle graph, (or the associated overlap graph) (Apostolico and Hambrusch, 1986; Hsu, 1985). One additional adjustment needs to be made. We wish to consider incompatible any two groups that flow into the same outroad of the junction. To assure that these flows are represented in the overlap graph by overlap, and not containment, we increase the length of arcs that flow into some outroad $T$ in the overlap graph by a multiple of some small value $\delta$. We add $\delta$ to the length of the longest arc that terminates at $T$. We add $2\delta$ to the length of the second longest arc that terminates at $T$, and so on. This will not create any artificial intersections, yet it will cause the desired overlap for any two groups which flow into the same outroad.

Having now determined $f_l$ and $f_u$, we run a scheduling heuristic (see Section 3.4) on a set of green times that correspond to $f_u$. To determine the green times we set $g_i = f_u y_i$. In many instances, particularly with smaller junctions, the heuristic finds a feasible schedule, indicating that the upper
bound is tight, and we have found the optimal solution. When the heuristic fails to find a feasible schedule, we run the reduced integer program (P3). Two things may occur. If the integer program finds a feasible setting for $f_u$ and the cycle time, then we have found the optimal $f$ and we are done. If the IP does not find a feasible setting, then the upper bound was not tight. We now need to perform a binary search on the range over $[f_1, f_u]$.

We find the midpoint $f_m$ of the original range on $f$. At $f_m$ we again run the scheduling heuristic. If it finds a feasible setting, then we reset the lower bound at $f_m$ and continue the search. If the heuristic does not find a feasible setting, we again run the integer program. If it provides a feasible setting, we reset the lower bound at the current value of $f_m$. If the integer program does not find a feasible setting, we establish $f_m$ as the new upper bound, and continue the search. The search ends when the difference between the upper and lower bounds, $\xi$, becomes sufficiently small.

In testing it was observed that including the scheduling heuristic in the procedure reduces the total computing time by almost fifty percent, on average, when compared to a similar procedure which does not make use of the heuristic.

3.4 A SIMPLE HEURISTIC

In this section we describe a heuristic that attempts to schedule the groups so as to fit within the cycle time, i.e.,
solve (P3). Once we have fixed the green time for each group we need to determine whether there is a feasible schedule within the set cycle time. The effective green times are listed as a fraction of the cycle time. If we also convert the lag times into percentages of the cycle time, our task is to find a feasible schedule in the interval \([0,1]\). The first attempt at determining a feasible scheduling will involve the use of the simple heuristic which follows.

**SCHEDGROUPS**

*Step 0.* Determine the critical clique. Call this clique \(Q\). Add the lag times to the effective green times to get the net green time needed for each group: \(g_i' = g_i + zl_i\)

*Step 1.* Find the largest job in \(Q\). Start this job at time 0. Also start every other group at its inroad at time 0.

*Step 2.* Go clockwise around the junction to the next inroad. Set the start time of the group in \(Q\) from this inroad at the ending time of the group in \(Q\) from the previous inroad. Start all other groups from this inroad at the same time as the member of \(Q\).

*Step 3.* For all groups at the current inroad (except the member of \(Q\)) adjust the starting time by either:

a. decreasing it until it encounters an incompatible group in the previous inroad

b. increasing it due to an incompatible group in the previous inroad that requires a finishing time after the finishing time of the member of \(Q\) from that previous inroad
Step 4. If not all groups have been scheduled, return to 2. If the jobs have all been scheduled check for feasibility. If all finishing times are less than or equal to 1, we have found a feasible setting. If some finishing time is greater than 1 the setting we have found is infeasible. The problem of incompatible groups overlapping has been taken care of in 3b, so we need not include this condition in our feasibility check.

For each group not in Q, we make comparisons in Step 3 with groups in neighboring inroads. In the extreme case we would have to compare times with (n-2) other groups. This implies that the heuristic is $O(n^2)$.

If the feasibility check determines that we have a feasible schedule, we are done. If not, then we will have to run the integer program, (P3), described in section 3.2. Results of implementing the above procedures are described in the next section.

3.5 EXAMPLES & COMPUTATIONAL RESULTS

We tested procedure ESPACE on several junctions which have physical structure typical to junctions in the American midwest. The five different junctions considered are sketched in Figure 5 thru Figure 9.

For these we used random numbers to generate junction data $(y_i, l_i)$. Sample data for a standard n=12 junction is illustrated in Table 2. The $y_i$ are uniformly distributed on the range [.2,.5]. To determine the $l_i$, we find n numbers uniformly
distributed on the range \([2.5, 6.5]\) and then round them to the nearest integer. Problem data generated in this manner is consistent with the data used by Improta and Cantarella (1984), allowing for a comparison of their procedure and the one described here. The procedure we implemented first finds upper and lower bounds on \(f\), as discussed in section 3.3. We then ran the heuristic on data corresponding to the upper bound on \(f\). In many instances, particularly with smaller junctions, the heuristic finds a feasible signal schedule, verifying that the upper bound on \(f\) is indeed a tight upper bound much of the time. On larger junctions the heuristic did not find a feasible schedule as often.

The results of implementing procedure FSPACE are compared to results obtained by implementing the binary-mixed-integer-linear program introduced in Improta and Cantarella (1984). The findings are listed in Table 3. The comparisons are based on the total CPU time required to obtain the optimal solution. The procedures were implemented on an IBM mainframe, and MPSX was used for the branch-and-bound procedure necessary to solve the integer programs.

Table 2
Sample data for a junction where \(n=12\).

<table>
<thead>
<tr>
<th>GROUP</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y_i)</td>
<td>.35</td>
<td>.32</td>
<td>.45</td>
<td>.44</td>
<td>.43</td>
<td>.33</td>
<td>.34</td>
<td>.47</td>
<td>.22</td>
<td>.29</td>
<td>.36</td>
<td>.48</td>
</tr>
<tr>
<td>(l_i)</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Procedure FSPACE was implemented using $\xi = 0.001$. This guarantees that we will find $f$ to within 0.001. It should be noted that on some of the smaller junctions the total CPU time required for procedure FSPACE is shown as 0. This illustrates that for some small junctions the problem of finding the critical clique and the implementation of the scheduling heuristic can easily be done by hand (or using a simple Lotus template). The values in the table represent the averages for ten implementations of both procedures.

We then tested the model on a larger ($n=42$) junction and found the results to be consistent with those of the smaller junction. We also implemented procedure FSPACE using $\xi = 0.0001$. This increased the total computer time required by, on average, roughly 35 percent for the larger junctions, while having little or no effect for the smaller junctions.

Table 3
Performance comparison: BMILP vs. FSPACE

<table>
<thead>
<tr>
<th></th>
<th>BMILP</th>
<th>Procedure FSPACE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>minimum</td>
<td>average</td>
</tr>
<tr>
<td>n=4</td>
<td>6.28</td>
<td>6.40</td>
</tr>
<tr>
<td>n=6</td>
<td>6.51</td>
<td>6.64</td>
</tr>
<tr>
<td>n=12</td>
<td>24.92</td>
<td>26.85</td>
</tr>
<tr>
<td>n=20</td>
<td>113.21</td>
<td>124.82</td>
</tr>
<tr>
<td>n=30</td>
<td>176.21</td>
<td>189.85</td>
</tr>
</tbody>
</table>
The procedures were also compared on a second set of data. These data were based on uniformly distributed $y_i$, but the lag times were determined differently. Lag times for groups that proceed straight across the junction were set at 3 seconds. Groups turning right were assigned a lag of 4 seconds. And groups turning left were assigned a lag of 5 seconds. These values account for the notion that the delay associated with turning traffic may be larger than that of traffic going straight due to safety considerations. The results of this set of testing are very similar to those seen in the tests involving uniformly distributed lag times and so are not included.
CHAPTER IV
CRANE SCHEDULING TO MINIMIZE CRATE SHIFTS AT A PORT

Many aspects of a sea port's operation contribute to the overall port efficiency. The aspect we study in this chapter is the assignment of cargo crates to positions within the hold of a ship.

4.1 Introduction

Cargo are initially stacked at a holding yard adjacent to the ship's berth. They are loaded sequentially onto a flat-bed truck which takes them to a position adjacent to the ship. There they are loaded into a hold of the ship. The first crate that arrives at the berth-side crane will be placed into the ship in position #1. The second crate is placed in position #2, and so on.

The crates are uniform in size, but will vary in weight. The optimal ordering is the one which requires the fewest crate shifts, while preserving the ship's radial stability. A crate shift is the movement of a crate to a temporary location, not on the flat bed truck, so that a crate currently below it can be loaded onto the flat bed truck. Crate shifts are detrimental to port efficiency. A shift necessarily means that the loading
process will take longer. Crate shifts also bring about the increased chance that cargo will be damaged during handling.

A measure of ship stability commonly used (Imai, 1988) is a radial measure, or a measure of "roll" ability. We seek to find a balance between the conditions of being top-heavy and bottom-heavy. If a ship is top-heavy it could tip in strong winds or rough seas. If a ship is too bottom-heavy, it becomes very difficult to turn. Additionally, a ship that is bottom-heavy will make small rolls, or wobbles, more frequently. This may cause severe damage to the ship's cargo. The model presented here works toward an acceptable middle ground for this "roll" stability.

For this problem we assume the use of a transtainer, or Gantry crane system for handling the crates in the holding yard. For actually loading the crates into the ship, we will assume the use of either a dockside crane or the ship’s own crane when appropriate. Our model studies the loading of a single crate stack into one hold of the ship, where the crates have a single destination. Figure 10 illustrates a cross-section of the ship's hold and the loading sequence. Figure 11 displays the relationship between the transtainer crane and the yard stack. Figure 12 gives a brief description of the layout of the yard and the functions of the transtainer, the flat bed truck, and the berth-side crane.

4.2 THE MODEL

The model we solve has the form of a constrained assignment problem with a non-linear objective function. We need to assign
each crate to exactly one position in the ship. The additional constraints require that the ship displays an acceptable level of radial stability.

**VARIABLES AND CONSTANTS:**

- **N** - the number of crates to be loaded (also the number of positions in the ship to be filled)
- **$T_i$** - the weight of crate $i$, $i = 1, \ldots, N$
- **$D_j$** - the vertical distance from the center of position $j$ to the metacenter of the ship, $j = 1, \ldots, N$

(A ship's metacenter is the intersection of two successive lines representing the force of buoyancy as the ship rolls thru a very small angle. The first line represents the force of buoyancy at rest. The second line represents the buoyant force at a small roll angle required for turning. For purposes of our study we assume the positions and crates have a height of 3 meters and a width of 3 meters. This is a relatively common crate size (Imai, 1988).)

- **$G_{ij}$** - a measure of the center of mass contribution for crate $i$ placed in ship position $j$; $i = 1, \ldots, N$; $j = 1, \ldots, N$
  \[ G_{ij} = \frac{D_j \times T_i}{\bar{D} \times \sum T_i} \]
  where $\bar{D}$ is the average $D_j$

- **$X_{ij}$** - 1 if crate $i$ is placed in position $j$, 0 otherwise;
  $i = 1, \ldots, N$; $j = 1, \ldots, N$
THE MODEL

Min. (the total number of shifts)

\[ \sum_{i} X_{ij} = 1 \quad \text{for } i = 1, \ldots, N \] \hspace{1cm} (16)

\[ \sum_{j} X_{ij} = 1 \quad \text{for } j = 1, \ldots, N \] \hspace{1cm} (17)

\[ .5 \leq \sum_{ij} g_{ij} X_{ij} \leq 1.5 \quad \text{for } i = 1, \ldots, N; \ j = 1, \ldots, N \] \hspace{1cm} (18)

Constraints (16) and (17) clearly have the form of the assignment problem. Constraint (18) assures that our stability measure, which is the vertical distance from the center of mass to the metacenter of the ship, is within an acceptable range. In the industry the common practice is to try to achieve a stability measure as close to 1.0 meter as possible (Imai, 1988).

The number of shifts necessary is a non-linear function of the \( X_{ij} \) variables. The number of shifts associated with a particular assignment is a function of all previous assignments. Imagine, for example, that a crate \( J \), which initially has 2 crates stacked above it is to be loaded into the 10th position in the ship. If neither of the crates initially above crate \( J \) are loaded into positions 1-9, then two shifts are necessary to place \( J \) in position 10. If one, or both, of the crates initially above \( J \) are placed in positions 1-9, then the number of shifts goes down accordingly. This makes determining the actual number of shifts a function of the assignments, which are not known until the model is solved. Unfortunately, this means that the objective function is non-linear.
4.3 SOLUTION ALGORITHM

Our solution procedure first finds the assignment which provides the best stability; that is, the assignment with a stability measure closest to 1.0. If this best assignment requires 0 shifts, then we are done. Otherwise, we compute the next best assignment in terms of stability. The assignment problem we solve takes on the following form:

\[
\begin{align*}
\text{Min.} & \quad \sum_{i,j} C_{ij} X_{ij} \\
\text{s.t.} & \quad \sum_j X_{ij} = 1 \\
& \quad \sum_i X_{ij} = 1
\end{align*}
\]

Here we define \( C_{ij} \) as follows:

\[
C_{ij} = |G_{ij}|
\]

We recall that \( G_{ij} = [D_j T_i] / [D_j T_i] \). We now redefine \( D_j \) as the distance from the center of position \( j \) to a point 1 meter below the ship's metacenter. With this definition, the objective function measures how different the center of gravity is from the optimal point 1 meter below the metacenter. We seek to minimize this difference.

Number the crates, as they are stacked in the yard, so that \( T_1 \leq T_2 \leq \ldots \leq T_N \) (i.e., numbered in increasing weights). Number the positions in the ship such that the bottom row will be numbered first, and the layers above it will be numbered such that if position \( i \) is above position \( j \), then \( i > j \), as is
illustrated in Figure 10. In this numbering scheme, we note that
\[ D_1 \geq D_2 \geq \ldots \geq D_N. \]

We define the model in this manner to illustrate the
similarity in structure between our model and the Factored
Transportation Problem studied by Evans (1984). Evans proves that
if the \( C_{ij} \) have a certain structure, the Northwest Corner rule
will provide the optimal solution to this problem. The structure
of \( C_{ij} \) must be of the form
\[ C_{ij} = U_j V_i \text{ where } U_1 \geq U_2 \geq \ldots \geq U_N \text{ and } V_1 \leq V_2 \leq \ldots \leq V_N. \]

The \( C_{ij} \) for our problem have been defined in a manner
consistent with the structure required by the proof in Evans

Having determined the assignment that provides the best
stability measure, we can now go back and check to see how many
shifts were necessary for this assignment. If it turns out that
no shifts were necessary, then we have found the best possible
assignment and we are done. In the event that the number of
shifts required is not zero, we continue. We then find the second
best assignment in terms of stability. Having found the second
best assignment, we check to see if it requires any crate shifts
in the yard. If it does, we continue. We repeat this procedure
until one of two terminal conditions occur:

1. an assignment with 0 shifts is found
2. we get an assignment with a stability measure outside the allowable range [0.5,1.5]

If condition 1 is met, we know that we have found the best assignment with no shifts required. If there were any better, we would have stopped previously when the better solution was found. If condition 2 is met, we review the solutions that we have generated to this point. We know that each has met the stability criteria, so we now simply choose the solution that requires the fewest number of shifts. An added advantage of this procedure is that it allows us to keep track of the solutions as it runs its course. This allows us to look at the tradeoff between the number of shifts allowed and the best stability that we could achieve using that particular number of shifts.

Murty (1968) provides us with a procedure for ranking all of the assignments with respect to cost. This procedure involves fixing one assignment variable and solving the model again on the N-1 remaining assignments. Fortunately, solving this reduced model is simply another application of the Northwest Corner rule.

4.4 EXAMPLES & COMPUTATIONAL RESULTS

We tested our procedure on a problem size that is typical of a hold for a standard containerized cargo ship (see Imai, 1988). The typical hold will contain thirty-six crates, in six rows and six columns. Typically the crates will be in six by six stacks, one row deep, in the yard. We generate crate weights in four different ways, creating four sets of data. In SET 1 we generate weights which are independent identically distributed and uniform
on \((0,1000]\). The range of crate weights we used is representative of cargo transported on a typical size ship (Imai, 1988). We ran the procedure on fifty different sets of data generated in this manner. The average solution time required, and the variance of these times, is displayed in Table 4. In the second set of data, SET 2, we consider three discrete weights, with each being equally likely to occur. The weights we used for this study were 500, 1000, and 1500 pounds. We implement this data set because it is often the case that many of the crates will have similar contents. In the third set of data, SET 3, each crate is equally likely to be from the set of uniformly weighted crates and crates with three discrete weights, as described above. This situation might arise if the crates were not from a single source. In the fourth set of data, SET 4, eighty percent of the crates have weights which are uniformly distributed on \((0,800]\) pounds. The other twenty percent of the crates weigh 2000 pounds. This fourth set represents weights which are skewed, or non-symmetric.

The results of our computational experiments can be found in Table 4 and Table 5. The results in Table 4 are based on the original stability criteria. That is, the distance between the metacenter and the center of gravity could be anywhere from .5 to 1.5 meters. The results in Table 5 are based on tighter stability criteria. Specifically, the distance between the metacenter and the center of gravity must be between .9 and 1.1 meters. These tables list the average results of fifty runs of the four methods of problem generation described above.
The average number of iterations refers to how many times we had to find the "next best" assignment before we found one that met our stopping criteria. The average stability measure tells us how close we were to the desired stability. The average CPU seconds refers to the total run times required to find the solutions on an IBM 386 personal computer, using FORTRAN code to implement the procedure and handle all necessary record keeping.

It should be noted in Table 4 that the average number of shifts, for the looser stability criteria, is 0 for all data sets. This implies that in each of these sample problems, a solution could be found that met the stability criteria, yet required no shifts of crates in the yard. It is also noted that for the trimodal data set it took, on average, longer to find a solution. This can be explained by the fact that when several crates have the same weight, there will be a high degree of multiple optimality.

Table 4
Test results for the crate assignment algorithm.
(Stability criteria relatively loose.)

<table>
<thead>
<tr>
<th></th>
<th>SET 1</th>
<th>SET 2</th>
<th>SET 3</th>
<th>SET 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. # of Shifts</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Avg. # of Iterations</td>
<td>67.4</td>
<td>179.8</td>
<td>145.9</td>
<td>137.6</td>
</tr>
<tr>
<td>Avg. Stability Measure</td>
<td>1.02</td>
<td>0.99</td>
<td>0.99</td>
<td>1.06</td>
</tr>
<tr>
<td>Avg. CPU seconds</td>
<td>106.4</td>
<td>243.8</td>
<td>220.4</td>
<td>214.9</td>
</tr>
<tr>
<td>Variance of CPU seconds</td>
<td>88.7</td>
<td>218.6</td>
<td>170.9</td>
<td>184.7</td>
</tr>
</tbody>
</table>
In Table 5 we find that it is sometimes necessary to perform crate shifts to obtain an acceptable assignment in terms of stability. Still, our solution procedure finds the acceptable assignment with the fewest shifts required. The average time required to find the optimal solution is lower when we have more restrictive stability criteria. Because of the more restrictive criteria, we more often encounter the second stopping criteria in our procedure.

Table 5

Test results for the crate assignment algorithm.
(Stability criteria relatively tight.)

<table>
<thead>
<tr>
<th></th>
<th>SET 1</th>
<th>SET 2</th>
<th>SET 3</th>
<th>SET 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. # of Shifts</td>
<td>0.0</td>
<td>0.26</td>
<td>0.12</td>
<td>0.88</td>
</tr>
<tr>
<td>Avg. # of Iterations</td>
<td>65.8</td>
<td>156.6</td>
<td>142.7</td>
<td>112.6</td>
</tr>
<tr>
<td>Avg. Stability Measure</td>
<td>0.96</td>
<td>1.02</td>
<td>1.01</td>
<td>0.99</td>
</tr>
<tr>
<td>Avg. CPU seconds</td>
<td>102.6</td>
<td>234.2</td>
<td>217.0</td>
<td>186.6</td>
</tr>
<tr>
<td>Variance of CPU seconds</td>
<td>91.3</td>
<td>256.4</td>
<td>188.3</td>
<td>272.4</td>
</tr>
</tbody>
</table>
Another aspect of port operations that will influence overall port efficiency is the scheduling of shore cranes which load and unload ships currently berthed. Here we study the problem of determining the best crane scheduling in terms of overall port efficiency. The most efficient schedule is identified as the one that minimizes the overall cost of idle time for all ships.

5.1 Introduction

The model we present here deals with an environment in which there are several ships, which require cranes for loading and/or unloading, berthed at a port. Each ship has a number of holds which have individual work requirements. A number of shore cranes must be shared among the ships that require processing.

The cranes typically move from hold to hold or even ship to ship on a rail system. Even the longest crane move at a typical port will require an insignificant amount of time compared to the unloading processes, so the crane shifting time is typically ignored. To make the system understandable, and implementable, the number of crane shifts should not be too large. To facilitate this idea, it is a stated principle that once a crane starts
working on a hold, it will continue to work on that hold until that hold is completed.

When ships have multiple holds, logic dictates that all holds of the ship must be processed before the ship "job" can be considered completed. This obviously delays some departures and dramatically complicates the analysis. The scope of the algorithm presented here only covers the static version of the problem. This involves a fixed number of ships with adequate berthing space. All ships that are to be considered in the analysis are currently berthed.

5.2 AN INTEGER PROGRAMMING FORMULATION

The model we develop is an integer programming model which does not require integer work requirements for each hold. More importantly, the size of our model does not grow as a function of the total work time required in the problem, as past solution procedures have. It does grow in the number of holds that require processing and the number of cranes that are available for use. The key to our model is the assignment of cranes to holds. It is assumed here, as is standard, that a hold can be processed by only one crane at a time. Our model can easily be modified to accommodate multiple cranes per hold. Along with a crane assignment, our model must then determine the order in which each crane will service the holds to which it has been assigned.

Once we have established the crane assignments and the sequencing for each crane, it is an easy task to determine the completion time for each ship. Our objective is to minimize the
total time that the each ships' processing is not complete, weighted by ship. Weights represent the cost of delaying each ship. What follows is a detailed discussion of the model we implement.

**CONSTANTS**

- $C$ - the number of cranes available
- $H$ - the number of holds to be emptied
- $S$ - the number of ships at the dock
- $S_j$ - the set of holds in ship $j$
- $W_i$ - the work time required to process hold $i$
- $M$ - a large number (for this program, $M$ was set equal to the sum of the work requirements)
- $C_j$ - the cost associated with having not completed ship $j$

**VARIABLES**

- $S_i$ - the starting time for unloading hold $i$
- $F_j$ - the finishing time of ship $j$
- $X_{ic}$ = 1 if crane $c$ is assigned to hold $i$, 0 otherwise
- $\delta_{ik}$ = 1 if the work on hold $i$ precedes the work on hold $k$, 0 otherwise

**THE MODEL**

\[
\begin{align*}
\text{Min. } & \sum_{j} C_j F_j \\
\text{s.t. } & F_j \geq S_i + W_i \quad \text{for all } i \in S_j \quad \text{for } j = 1, \ldots, H \quad (23) \\
& \sum_{c} X_{ic} = 1 \quad \text{for all } i = 1, \ldots, H \quad (24)
\end{align*}
\]
The objective is to minimize the weighted delay time for all ships. Constraint set (23) assures that the finish time for each ship is no smaller than the completion time of each of its holds. Constraint set (24) assures that every hold will be assigned exactly one crane. This can easily be modified to allow for multiple cranes processing a single hold. Constraint sets (25), (26), and (27) collectively assure that if two holds are assigned to the same crane, then that crane must process one of them before the other. Constraint set (28) deals with the matter of three holds assigned to the same crane. This set explicitly states that if \( i \) precedes \( j \), and \( j \) precedes \( k \), then \( k \) cannot precede \( i \). Constraint set (28) is not essential in determining the optimal solution to the model. Yet we found that adding these constraints led to quicker solution times. These are valid inequalities that reduce the continuous feasible region without eliminating any feasible integer points. The effect is a reduction in the duality gap between the initial linear programming solution and the eventual integer programming solution we seek.
5.3 COMPUTATIONAL RESULTS

To test our model, we obtained the original data used in Peterkofsky and Daganzo (1990). In their paper they generate data for ten problems for each of two, three, four, five, six, seven, eight, nine, and ten ships. Some of the data was no longer available, so we have comparative results for only problems involving two, three, four, six, and nine ships. For each problem the number of cranes was chosen randomly among two, three, and four. The number of holds in each ship was chosen randomly among two, three, four, and five. The hold workloads were selected randomly as uniformly distributed real numbers over the interval from one to ten. To be effectively analyzed by their procedure, the real numbers representing workload were then truncated to integer values. In the first five problems for each ship size, all ship costs were fixed at 1. In the remaining five problems each ship's costs were set equal to its number of holds. In these latter problems, the costs were roughly proportional to the ship size, and, therefore, its capital cost.

The results of our comparison are displayed in Table 6. As in Peterkofsky and Daganzo (1990), we determined that 15 minutes on an IBM personal computer would be an upper bound on acceptable solution time. What is recorded is the number of times that each procedure found the optimal solution within 15 minutes of CPU time. Reported also is the average time to find a solution for each procedure, recorded to the nearest tenth of a minute.
It should be noted here that the results Peterkofsky and Daganzo obtained were from implementing their algorithm in Pascal on an IBM PC/AT with a 6 MHZ clock. Our results come from using an IBM 386 series machine, with EXPRESS-MP used for the integer programming solutions, and FORTRAN being used for the record management. A more appropriate comparison would have been to run both procedures on the same machine under the same circumstances.

The results for their procedure are as reported in their paper. Our results have used the same data, except where $S=5$. Here the original data was not available, so we generated data in the same manner as the original data was generated.

Table 6
Performance comparison of crane scheduling algorithms.

<table>
<thead>
<tr>
<th># of ships</th>
<th>Peterkofsky &amp; Daganzo</th>
<th>Model Presented</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of times</td>
<td>optimal found</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>2.2</td>
</tr>
<tr>
<td><em>5</em></td>
<td>7</td>
<td>9.3</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
We also tested our model on problems with non-integral workloads for a typical number of ships (S=4). We found that on average the CPU time required to find the optimal solution went up only about 4%. We also ran a series of problem where S=4 and the work times were all set to 9, the previous upper bound on work times. Again we found that this had very little affect on our solution times. Each of these changes would dramatically increase the solution times for the procedure presented by Peterkofsky and Daganzo.
CHAPTER VI
CONCLUSION AND FURTHER RESEARCH

We have developed procedures which will facilitate improved efficiency in a variety of transportation settings. The models we present are implementable and solve the specific problems stated here. Further, this work implies that there are other areas within the field of transportation that mathematical modeling techniques would help to operate more efficiently. Many transportation decision are still made by "rule of thumb." What follows is a discussion of the results of our work and some possible extensions to more general transportation problems.

6.1 The Signal Setting Problem

The results of this work are significant for several reasons. Unlike previous solution methods to this problem, our procedure does not assume knowledge of the stage matrix. Further, our approach guarantees an optimal solution to within a specified $\xi$ for the general case of the problem, without any qualifications for special cases. Procedure FSPACE is also shown to solve much more efficiently than a BMILP for junctions of typical size.

A fast solution to the problem studied here will be useful for several reasons. If one wishes to analyze the tradeoff between the cycle time and the junction flow factor, the model
will need to be solved for each of several cycle times. It is also likely that the junction traffic data will vary throughout the day. The optimal signal setting will need to be determined periodically for the varying traffic data.

Although traffic junctions typically do not exceed 20-30 flow groups, other flow problems requiring the sequencing of incompatible flows may grow much larger. An extension to the single junction would be to consider multiple junctions, either in series or as a network of junctions. A fast optimal solution to the single junction problem could perhaps be used in a heuristic procedure to solve the larger problem.

Another extension to the problem studied here might include accounting for "gap seeking," as in Yagar, 1974. This is the situation where incompatible groups may be allowed to flow at the same time with one group having priority over the other. The group with lower priority could then seek gaps in the flow of traffic from the higher priority group. Another extension would be to include crosswalks in the junction model, or to consider junctions with other special structural characteristics.

6.2 Scheduling the Loading of Crates into the Hold of a Ship

We have introduced a procedure for finding an acceptable assignment of crates currently stacked in a holding yard to positions within the hold of a ship. In each of 150 sample problems our procedure finds an acceptable assignment which requires no crate shifting in the yard. Our procedure addresses the simplified, though not uncommon, case where the crates are
all of the same size, the crates all have the same destination, and there are no other special handling requirements for any of the crates. Our procedure finds the optimal solutions in what can be considered a reasonable amount of time. We have also assumed that the crates were in one stack in the yard, that they all went into the same hold of a single ship, and that "roll" stability was our only concern. It is our intention to next apply the things we have learned about this particular problem structure to expanded problems. Those problems might include having multiple stacks in the yard, multiple holds in the ship, and multiple destinations for the crates. Each of these variations will increase the size of the problem to be solved.

Certain crate restrictions can be dealt with very easily with minor modifications to the current model. If a crate is too heavy to be placed anywhere but the bottom row, then we would simply fix its position in the bottom row and solve the now reduced model. If a crate must be in the top row to take advantage of electrical facilities, we would fix its position somewhere in the top row and solve the reduced problem. We understand that this is likely to increase the solution time, noting that it may now be impossible to find a solution that requires no crate shifts.

6.3 Crane Scheduling for Efficient Processing of Berthed Ships

The model presented in Chapter V solves the static case of the crane scheduling problem. It provided satisfactory results for as many as six ships using a standard microcomputer. Advances
in computer technology will no doubt allow for solution of larger problems in the future.

The model presented in this paper can easily be modified to allow for some extensions to this problem. The model currently accommodates non-integer work times. It could be easily modified to allow for multiple cranes processing a single hold at any given time. There are other complications, however, that the model cannot easily accommodate. We have assumed that we have static demand for the cranes. While we may modify the model to allow for later ship arrivals with deterministic arrival times, the true dynamic model, with stochastic arrival times, cannot be dealt with using the model presented here. One method typically used in the analysis of the true dynamic problems, though, is to solve the static problem each time another ship arrives. For such a procedure it would be very useful to have a static model that solves quickly. If there are also berth space restrictions (something we have assumed away), then some model based in queueing theory will be needed. Perhaps the model developed in this paper would serve as a subroutine within this larger model as well.

The results of this paper, and the insights gained in finding these results, might also be applied to other scheduling problems with similar resource limitations, where a fixed number of machines must process multiple "jobs." Further, insights from this study might also lead us to understand what conditions lead to a necessarily "bad" solution. Knowing this might be useful in
understanding the problem of port terminal design, as well as the design of other processes with similar scheduling difficulties.
LIST OF REFERENCES


APPENDIX A

FIGURES
Figure 1
Traffic junction where n=6.
Figure 2

Incompatibility graph for the junction illustrated in Figure 1.
Figure 3

The circle graph associated with the junction in Figure 1.
Figure 4

The overlap graph associated with the circle graph in Figure 3.
Figure 5

Traffic junction where n=4.

(Group 5 is ignored because it is compatible with all other groups.)
Figure 6

Traffic Junction where n=6.
Figure 7

Traffic junction where $n=12$. 
Figure 8

Traffic junction where n=20.
Figure 9

Traffic junction where n=30.
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

**Figure 10**

Cross-section of Ship
Figure 11

Relationship between yard stack and transtainer.
Figure 12

Overhead view of port layout.
Figure 13

Cranes for unloading berthed ships.