AN ABSTRACT OF THE THESIS OF

Wen-Tsao Wang for the degree of Master of Science in Industrial and Manufacturing Engineering presented on April 27, 1989.

Title: Evaluation of Scheduling Rules for Single- and Dual-Dock Automated Storage and Retrieval System

Abstract approved: _____
Sabah U. Randhawa

Automated Storage and Retrieval (AS/R) Systems have had a considerable impact on manufacturing and distribution processes. The configuration of AS/R systems vary considerably, depending on the particular application.

The effect of different retrieval scheduling rules is analyzed and compared for three different unit-load AS/R systems operating under dual-command cycle. The AS/R systems differ in the number of docks (or input/output points) per aisle. The system is based on that there are two storage (S1, S2) and two retrieval (R1, R2) sources. The three AS/R systems are: (1) one dock per aisle handling both storage and both retrievals; (2) two docks, one at each end of the aisle, with storage requests from S1 and S2 retrieved by

either R1 or R2; and (3) two docks, one at each end of the aisle, with storage requests from S1 retrieved by R1 and storage requests from S2 retrieved by R2. The same scheduling rule, first-come-first-serve/closest-open-location, was used for storage. However, retrieval scheduling rules analyzed in the study were first-come-first-serve, nearest-neighbor, and nearest-neighbor with a maximum wait time limit.

A simulation model was developed for each combination of scheduling rule and layout and three performance measures were examined--throughput, mean waiting time, and maximum waiting time. The results obtained from this investigation show that the efficiency of the AS/R system can be improved by the introduction of two docks in each aisle when the input pallets are stored in the closest-open-location, and the input/output pallets for the two docks are independent of each other (layout 3).

In general, each performance measure reflects an independent optimum solution for each layout, or combination of layout and scheduling policies. The use of one of these independent solutions can serve to optimize selected AS/R configurations with regard to the given performance measures, but at the same time other performance measures may be affected adversely. Thus,

there is no one global optima; the optimum is a function of the performance measure used.

Evaluation of Scheduling Rules for Single- and Dual-Dock Automated Storage/Retrieval System

by

Wen-Tsao Wang

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed April 27, 1989

Commencement June 1990

APPROVED:

Assistant Professor of Industrial and Manufacturing Engineering in charge of major

Head of Department of Industrial and Manufacturing Engineering

Dean of Graduate School

Date thesis is presented April 27, 1989

Typed by B. McMechan for Wen-Tsao Wang

Acknowledgements

I wish to express my gratitude and sincere appreciation to my major professor, Dr. Sabah U. Randhawa, for his guidance, support, valuable advice, encouragement and friendship during the preparation of thesis, as well as for what he taught me throughout the course of my study.

I extend my sincere appreciation to Dr. Edward McDowell for his valuable advice, constructive criticism and encouragement during the preparation of this thesis.

Finally, I would like to thank my wife, Poh-Yu, for her encouragement, understanding, patience, and care during these years.

DEDICATION

This Thesis
is dedicated to my parents,

Mr. Shaw-Hua Wang
Mrs. Ing-Kwang Fu Wang

for their strength, and endurance,
since without their support, my graduate
study would never have been completed.

Table of Contents

Table of Contents (continued)

List of Figures

List of Figures (continued)

List of Appendix Figures

List of Tables

EVALUATION OF SCHEDULING RULES FOR SINGLE- AND DUAL-DOCK AUTOMATED STORAGE/RETRIEVAL SYSTEM

I. INTRODUCTION

Background

Automated storage and retrieval (AS/R) systems have had a dramatic impact on storage and warehousing. Through the application of computer controls, handling and storage systems have been successfully integrated into manufacturing and distribution processes. The primary application of these methods has been in the storage and retrieval of finished goods, but recent AS/R focus has been directed to work in process, raw materials and supplies in an effort to assure that material control and manufacturing operations can work in concert to create a cost-efficient, optimum flow of materials with resultant increases in productivity. This requires the integration of the AS/R system into the total matrix of a plant's material handling system and automated processes.

AS/R System Configurations

The configuration of AS/R systems vary considerably depending on the particular application. A typical AS/R system, as shown in Figure 1.1, consists of storage racks, automatic stacker cranes, link conveyors, and input/output (I/O) stations.

The basic function of the automatic stacker crane is to store and retrieve warehoused materials. Typically, each aisle is assigned its own automatic stacker crane (Figure 1.2), consisting of a single- or double-mast frame, a carriage, a shuttle, drives and guidance mechanisms. A stacker crane most often has three mechanical drives: horizontal and vertical drives, which function simultaneously to move the crane diagonally and reduce travel time, and shuttle drive, which transfers the load between the crane and storage locations on the facility's racks. The conveyor system provides the link between the warehouse and incoming and outgoing pallets from a source or to a destination. The input/output (I/O) point is located at the point where pallets are exchanged between the conveyors and the cranes.

In general, four types of AS/R systems have been developed for manufacturing use: unit-load, person-on-board, mini-load and carousel systems. For the unit-

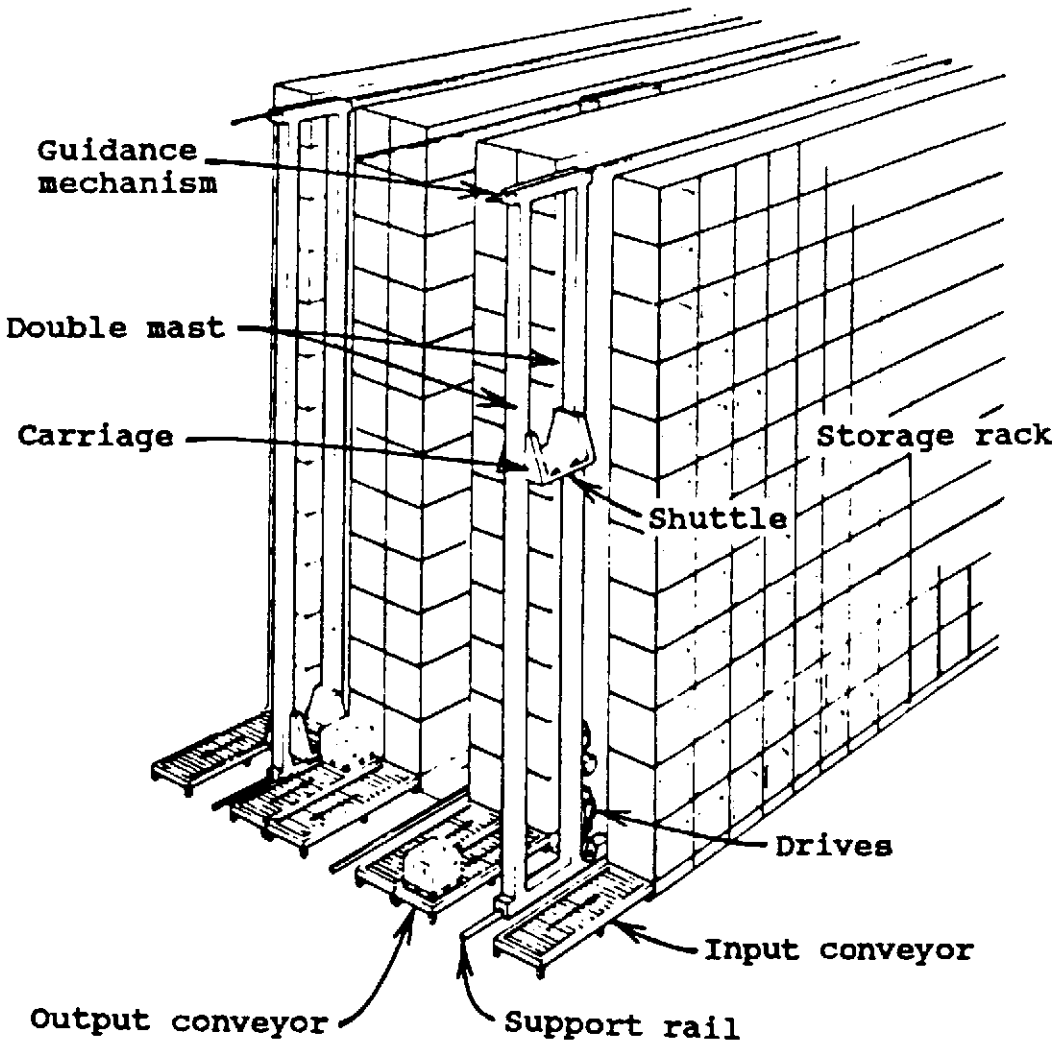


Figure 1.1 Typical AS/R System Layout (*Handbook of Industrial Engineering*, 1982).

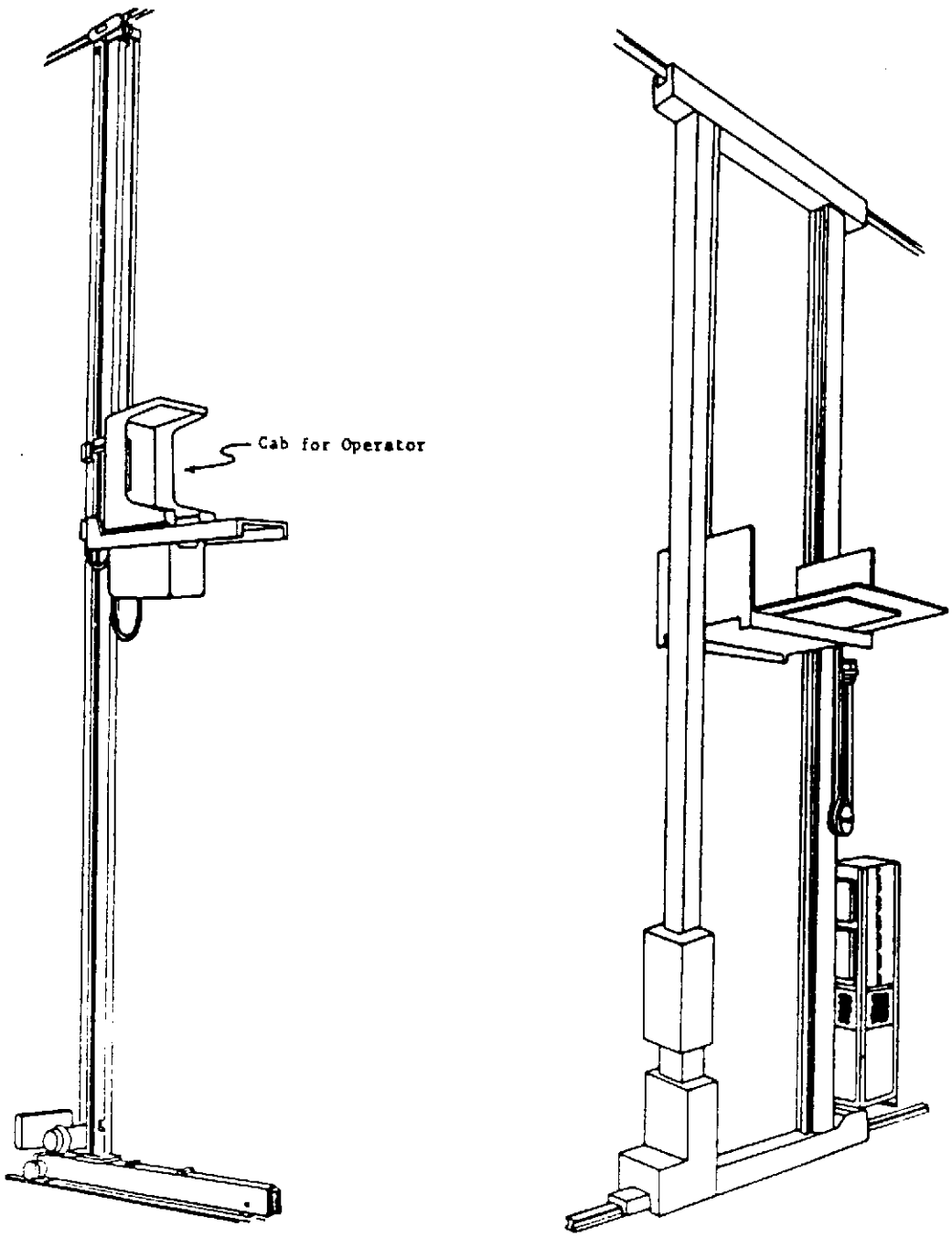


Figure 1.2 Person-On-Board and Unit-Load Automatic Stacker Cranes (*Handbook of Industrial Engineering*, 1982).

load system, loads are palletized or placed in a tote or container. For light loads, the stacker crane capacity varies between 300 to 700 lbs, reaching lifting heights up to 40 ft. For heavier loads, the stacker crane capacity may be from 700 to 8800 lbs, lifting loads to heights of up to 100 ft. Horizontal movement speeds range from 60 to 500 feet per minute, whereas vertical travel speeds range from 60 to 100 feet per minute (*Handbook of Industrial Engineering*, 1984).

The person-on-board AS/R system (Figure 1.2) is used for in-aisle order retrieval. The operator "picks" the items from shelves or bins within the storage structure and places the picked items into totes or modules which are than carried by the S/R machine to the end of the aisle for dispatch. The efficiency of the person-on-board system may be increased by providing the operator with an on-board computer which generates such information as the location and quantities of the item to be retrieved.

The mini-load AS/R system (Figure 1.3) is used for end-of-aisle order retrieval and reshelving. This system provides real-time inventory controls, is generally space efficient and is appropriate for handling small parts that are stored in containers and tote boxes. The last type, the carousel (Figure 1.4), consists of a horizontally revolving bin that moves stored materials to the pick station. It consists of a set of carriers,

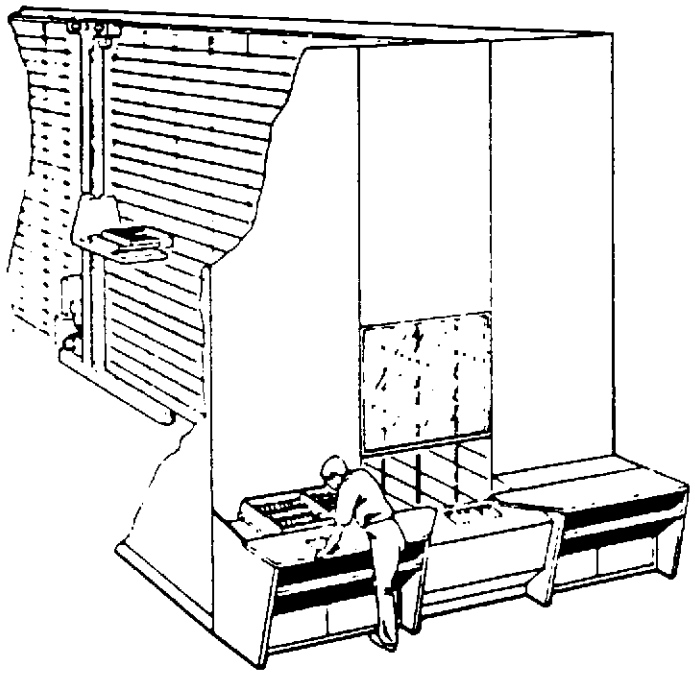


Figure 1.3 Mini-Load Automatic AS/R System (*Handbook of Industrial Engineering*, 1982).

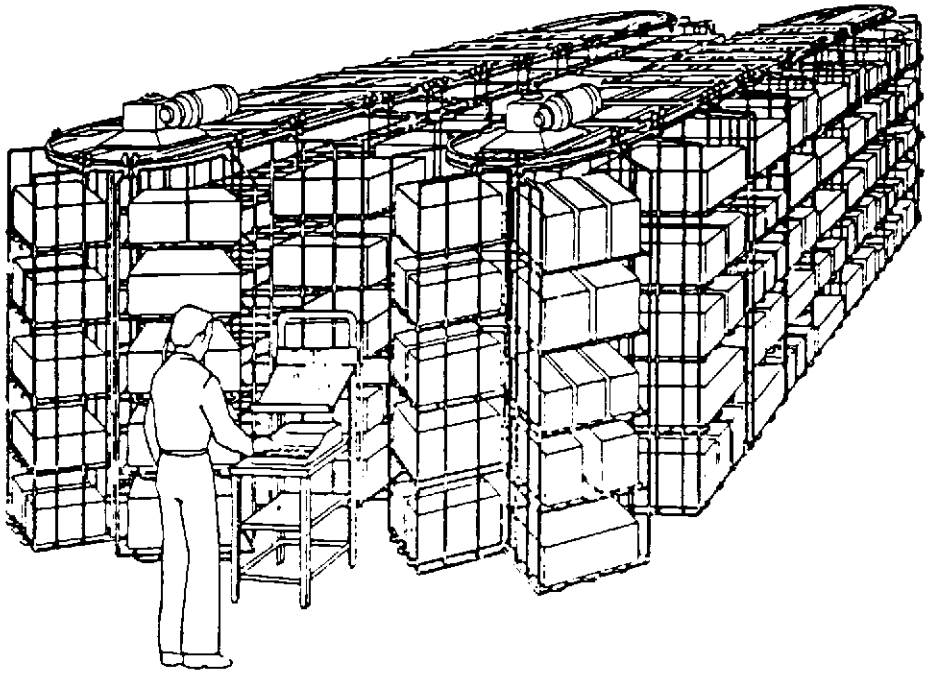


Figure 1.4 Carousel AS/R System (*Handbook of Industrial Engineering*, 1982).

each of which is in turn is composed of vertical rows of multishelf bins. The drive mechanism rotates the carriers to bring the appropriate bin to the "picker".

Existing plant layout, material handling practices, the types of loads handled and desired throughput rates must all be considered in the design or selection of an AS/R system. This type of decision may be formulated by the system supplier, by the user of the system, or by both.

AS/R Improvement Potential

Because of the importance of the stacker crane cycle time in the development of AS/R throughput, considerable attention has been given to the time required for the stacker crane to perform both single-command and dual-command cycles. In the unit-load system, a single-command cycle consists of either storage or retrieval objectives, but not both. A single-command crane storage cycle time is the sum of the times required to "pick" the load at the I/O point, travel to the storage location, place the load in the rack, and return to the I/O point. A single-command retrieval cycle time is similar, operating in the reverse direction. A dual-command cycle involves both storage and retrieval in which cycle time is the sum of time required to "pick" the load at the I/O point, travel to the storage location, place the load in the rack,

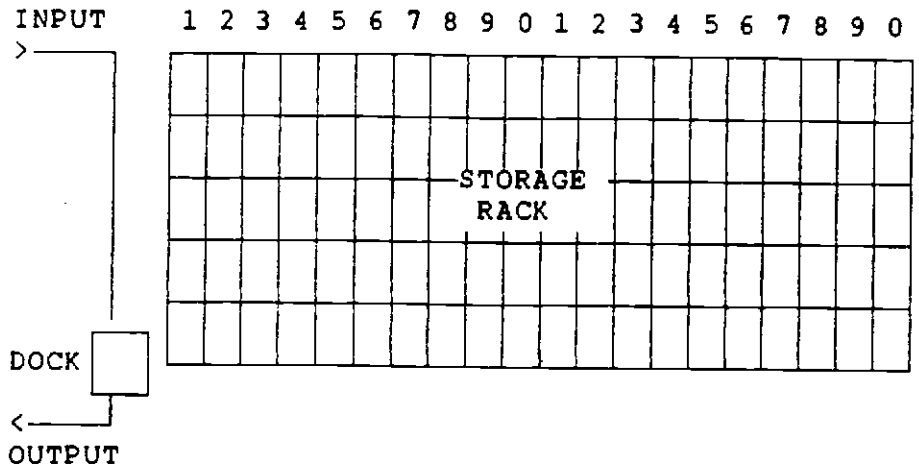
travel empty to the retrieval location, retrieve a load, and return to the I/O location.

For either of stacker crane systems there is a significant potential for reduction of travel time. A time reduction which may be achieved by scheduling three elements: assignment of multiple items to the same pallet; assignment of pallet loads to storage locations; and development of rules for sequencing storage and retrieve requests. For the dual-command unit-load AS/R crane system, Graves and Hausman (1977) demonstrated that system throughput can be dramatically increased by the application of crane interleaving based upon class-base storage location assignments and the nearest-neighbor principle in retrieval queue selections. In addition, Khan (1982) showed that the crane travel time can be reduced by introducing a zone nearest to the I/O point for recycling empty or partially picked pallets. This resulted in throughput improvement in end-of-aisle AS/R systems with some partially picked pallets.

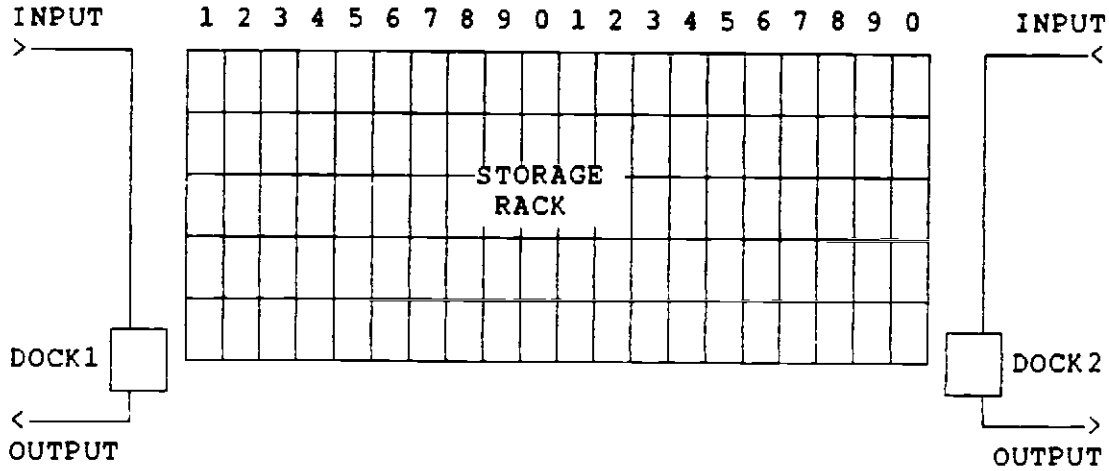
All of the AS/R system research prior to this study has been based on the assumption that each aisle has only one I/O point, which is located at the end of the aisle. Each time the crane leaves the I/O point, the crane must return to the I/O point from which it started independent of the scheduling rules used or the distance the crane travels along the aisle.

The purpose of this study is to introduce a unit-load AS/R system with two I/O points located at opposite ends of each aisle (Figure 1.5). Both I/O points would have identical functions for handling inputs and outputs and for each crane cycle, and the origin or terminus point could be either one of the two I/O points. The result could be a substantial increase in system throughput based upon (1) the intelligent selection of storage locations for input pallets for each I/O point, and (2) rules for sequencing storage and retrieval requests.

The results of this study will be useful to AS/R system designers for whom system objectives are to serve several different input sources and output destinations. Moreover, the results of this study will offer AS/R system users a practical means to improve efficiency since the cost of adding an additional I/O point is much less than the costs of adding an additional automatic stacker crane.



a) Single Dock



b) Dual Dock

Figure 1.5 Single and Dual Dock AS/R Systems.

II. PROBLEM ANALYSIS

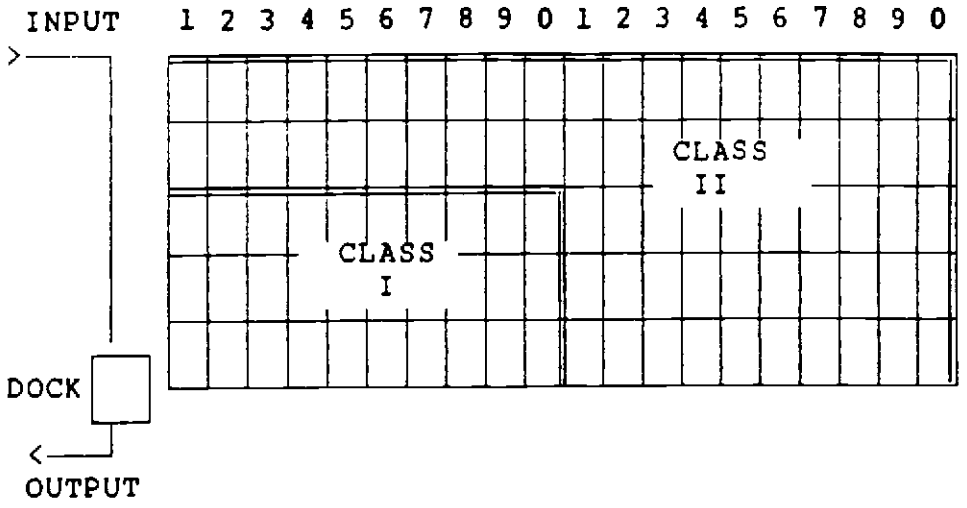
Literature Review

A great variety of research has been conducted in the area of AS/R system throughput improvement. Some approaches have sought to apply mathematical techniques to the formulation of scheduling rules and location assignments policy. Others have been directed at the use of computer simulation techniques to replicate the factors affecting AS/R system efficiency. The common factor in all of the previous research in this area has been the maximization of system throughput by minimizing crane travel time.

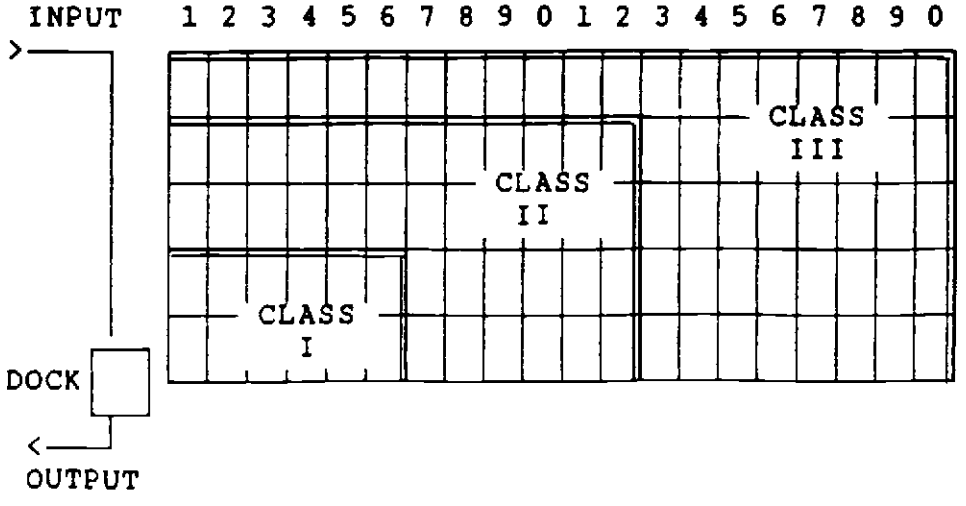
Hausman and Schwarz (1976) presented a study comparing AS/R crane travel time for a randomized storage system with turnover based retrieval. Interleaving or dual-command cycles were not allowed, the storage rack was assumed to be square in time, and the I/O point (the dock) was placed at one corner of the racks. Two storage assignment rules were considered and compared. First, the "Random Storage Assignment" rule approximated the commonly used "Closest-Open-Location" rule, in which pallets are randomly stored in any rack location. Second, a "Turn-Over Based Assignment" rule

(Figure 2.1) was developed, in which the items and storage racks are partitioned into a small number of classes (2 or 3) according to the item turnover rate and the travel distance from the I/O point. For example, the class of items with the highest turnover rate is randomly assigned within the class of locations closest to the I/O point. This second rule generated improvement ranging from 26 to 71 percent for 20/60 to 20/90 turnover distributions taken from an ABC curve in comparison to the first rule. Similarly, throughput improvement was achieved by the use of class-base storage method for two- and three-class storage assignments. The improvement percentages for this storage assignment system were 70 and 85 percent of the potential for a fully turnover-based system for two-class and three-class assignments, respectively.

Graves and Hausman (1977) presented an improvement to the above system. Various combinations of alternative storage assignment rules and scheduling policies were analyzed based upon generation of a dual-command cycle for each trip and comparisons of expected AS/R crane travel time. Results indicated that a reduction in expected AS/R machine travel time could be obtained by the application of turnover-based storage in place of randomized storage. The largest reduction of crane trip time was generated by the use of a full turnover-based storage assignment integrating interleaving and a



a) Two-Class



b) Three-Class

Figure 2.1 Class-Based Storage Location Assignment (Hausman & Schwarz, 1976).

first-come-first-served (FCFS) retrieval queue, followed in turn by two- and three-class turnover-based storage assignments. However, these results were only of theoretical importance since, as was stated in the study, it would be unrealistic to assume that the turnover of every pallet stored in the system could be known and/or constant over time.

Graves and Hausman (1978) then developed a computer simulation model to examine the results of their previous study. Simulation demonstrated that a substantial increase in system throughput resulted from the application of turnover-based storage assignment rules and from interleaving (for a dual-command cycle). However, the actual improvement was generally slightly smaller than predicted. The reason for this discrepancy was that when one queue was empty and interleaving was not possible, the crane could not wait until interleaving could again be applied but proceeded to serve the non-empty queue in a non-interleaving mode. Hence, for simulation purposes, the scheduling policy was actually a mixture of systems based on mandatory interleaving and no interleaving.

Han and McGinnis (1987) have developed a lower boundary for dual-command cycle time, addressing throughput improvement in a conventional unit-load AS/R system by retrieval sequencing when several retrieval requests are presented and dual-command cycles are per-

formed. Taking the FCFS rule as the reference sequencing rule, nearest-neighbor sequencing with a block of 15 to 20 retrievals and 100 percent dual-command cycles reduced travel-between time (that is, travel time between storage and retrieval points) by 60 percent if there is one open location. Moreover, an even greater reduction resulted from the use of more than one open location. Their analysis indicated that for a typical AS/R system, a 60 percent reduction in travel-between time would yield a 12 percent increase in system throughput. For a large, multi-aisle system, a 12 percent increase in throughput could lead to the elimination of an aisle. In addition, Han and McGinnis established a lower boundary on mean dual-command cycle time for systems based on different retrieval block sizes and numbers of open locations. Based on this limit, a 22 percent increase in throughput was the maximum possible and it was concluded that the opportunity for further improvement was slight.

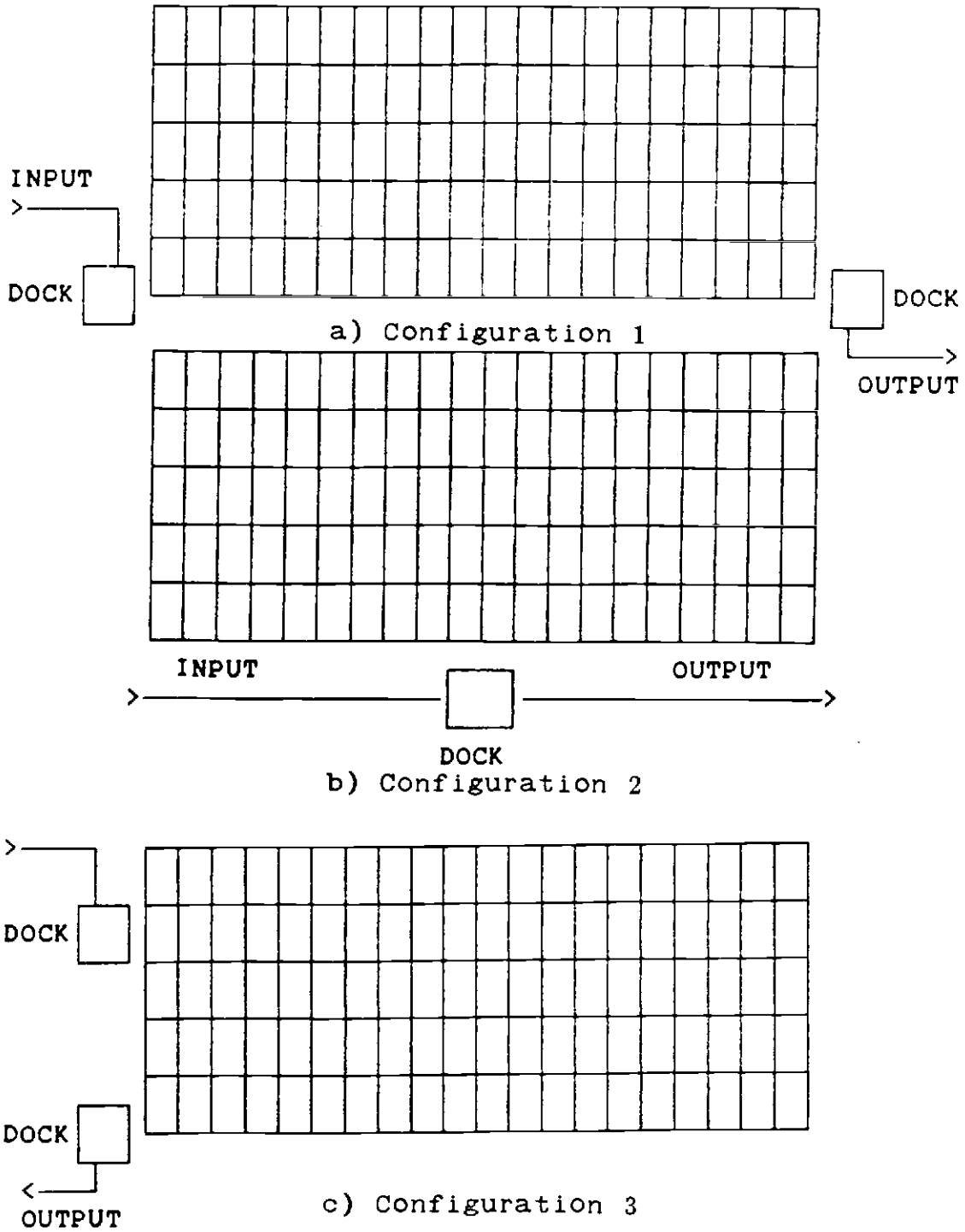
In the studies cited above, the AS/R system I/O point was assumed to be located at either the left or right corners of the storage rack and every trip originated and terminated at this I/O point. In 1984, Bozer and White examined travel times based upon three alternative I/O locations: (1) at the opposite ends of aisle; (2) at the same end of aisle but placed at different elevations; and (3) at the same elevation but at

the aisle midpoint (Figure 2.2). Taking as the basis of comparison the expected dual-command travel time for two random locations, one for a storage point and one for a retrieval point, and restricted to the assumptions defined in their study, Bozer and White concluded that configuration (2) performed 18.3 percent better than configuration (1), and configuration (3) performed 26.2 percent better than configuration (2).

Problem Formulation

A typical unit-load AS/R system with several aisles in each of which storage and retrieval are served by an AS/R crane was considered. A conveyor system provided the link between the system and the destination or source for storage or retrieval, and the I/O point was located at the left-corner of each aisle. The exchange of pallets from the crane to the conveyor occurred at the I/O point.

In this type of AS/R system, one method of improving throughput would involve reduction of mean crane travel time by reducing the dimensions of the rack. However, reducing the dimensions of the rack would incur the loss of storage locations and capacity and an additional aisle would be required to maintain a constant storage capacity for the overall system. This would result in additional AS/R acquisition expense since each aisle is equipped with its own crane and



a) Configuration 1

b) Configuration 2

c) Configuration 3

Figure 2.2 Three Alternative I.O Points (Bozer & White, 1984).

from an economic point of view, increasing throughput for each storage aisle by the reduction of rack dimensions is not feasible.

Another approach to throughput improvement is to assign storage items to the closest-open-location, processing them on an FCFS policy. This is realistic for the storage process since for input and output most AS/R systems are interfaced with a conveyor loop, with little opportunity available for changing the sequence in which storage pallets are presented to the conveyor. For retrievals, requests are simple messages, particularly in a computer-controlled AS/R system. Thus, the retrieval requests could be arranged in any convenient sequence, and the travel time between storage and retrieval locations in a dual-command cycle could be reduced by adequate sequencing of the retrievals. Overall system performance improvement would be the result.

It is evident that the average travel time to move the crane from a storage location to a retrieval location can be reduced by optimizing the retrieval sequence. However, optimizing retrieval request sequencing is a complex problem. Given a retrieval list with a single open location for the initial storage operation, the sequencing problem is equivalent to the well-known traveling salesman problem in dynamic programming, which provides evidence that no efficient solution algorithm can be developed for optimization of the

sequencing problem. Moreover, the problem assumes even greater complexity when multiple open locations are available.

Another problem with the optimal retrieval sequencing is the dynamic nature of the retrieval list. New retrieval requests appear at times when prior requests are being performed, and in order to maintain an optimal retrieval sequence the list must be rearranged each time a new retrieval request arrives. The following alternatives may be considered to manage the dynamics of this situation:

1) Select a "block" of retrievals containing the first K entries in the retrieval queue. K retrievals are searched until the nearest-neighbor is found. The search is repeated each time the crane completes a storage assignment and once the block of retrievals has been completed, a second block of K retrieval requests is selected.

2) Maintain an optimal retrieval list by resequencing the list each time a new retrieval request arrives. To avoid the imposition of excessive waiting time when specific retrievals are at the farthest end of the aisle, a maximum waiting time limitation is employed.

In this study, the second alternative with varying maximum waiting time limits is examined as an alterna-

tive retrieval policy for throughput improvement in warehouse layouts. Results are compared with the FCFS retrieval policy for system performance evaluation.

In the prior research, it is generally assumed that the aisle I/O point was located at the lower left-corner of the storage rack. This means that for every crane cycle, points of origination and termination are bounded at the same location (Figure 2.3). With a dual-command crane traveling cycle (with interleaving allowed), the expected cycling time is the time the system requires for completion of one storage and one retrieval assignment. That is, assuming load and unload times to be negligible, the expected cycle travel time is the sum of travel time from the I/O point to the storage location, plus the travel time to retrieval point and the return time to the I/O point. These are common features of existing AS/R system layouts and are used as the basis for performance comparison of alternative layouts in this study.

To test the performance improvement of the AS/R system, the assumption of one I/O point for each aisle is relaxed. Rather, two I/O points for each aisle are considered, located at the left- and right-corners of the storage rack, and each I/O point is able to handle the transaction functions for incoming and outgoing pallets. In this configuration storage pallets and retrieval requests are initiated at either of the two

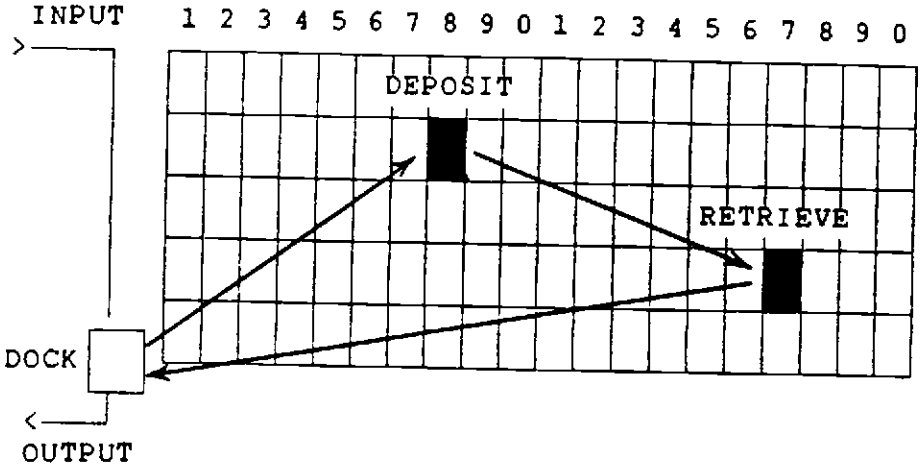


Figure 2.3 Dual Command Crane Cycle
With One I/O Point.

I/O points and in the dual-command crane cycle, the point of origination or termination can be either the same I/O point or the opposite I/O point (Figure 2.4). As previously noted in this section, a conveyor provides the link between the I/O points and the retrieval source or storage destination.

AS/R system experience has indicated that it generally has more than one storage source and retrieval destination available. In this situation, either of the two I/O points can be assigned to handle incoming storage pallets and retrieval requests which have an identical source and destination, while the other I/O point can be assigned to handle the remainder of the requests. Furthermore, at the time the crane initiates a dual-command travel cycle, it will have a broader selection of closest-open-locations for storage and a broader selection of the nearest-neighbor retrievals. Consequently, dual-command travel time may be reduced. An AS/R system with two I/O points in each aisle will be adopted as the alternative layout for this study and its performance will be compared with a layout with only one I/O point in each aisle.

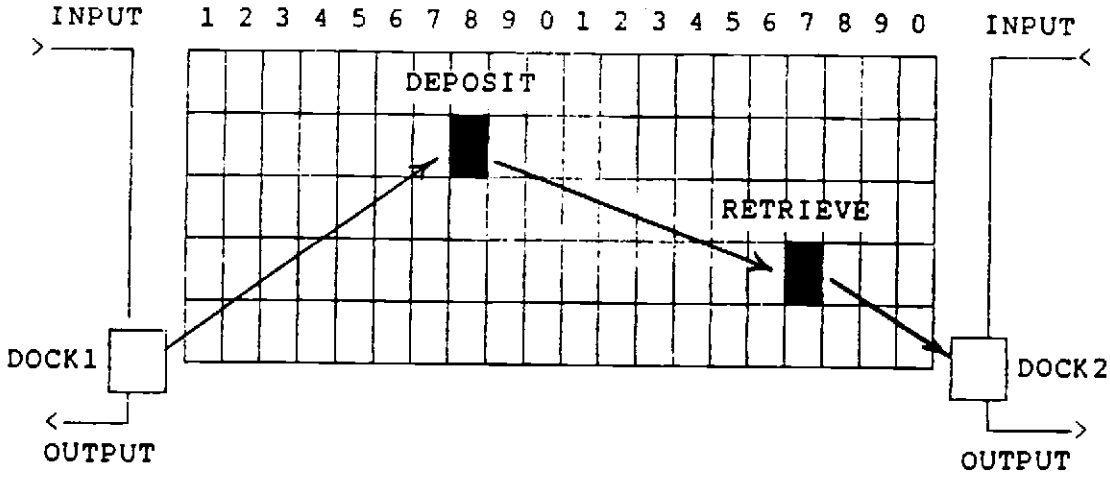


Figure 2.4 Dual Command Crane Cycle
With Two I/O Points.

Study Factors and Performance Measures

The number of factors which may be analyzed and compared in the study of AS/R systems are considerable. These include the dimensions of storage racks, crane speed, pallet assignment, maximum queue size, storage location assignment and retrieval/storage sequencing. The general assumptions necessary to reduce the magnitude of the problem and allow focus on the more important study factors are presented in the next chapter. The factors chosen for this study include: (1) scheduling rules, (2) the number of docks (I/O points), and (3) the relationship between storage source and retrieval destination.

The first factor, retrieval scheduling rules, involves examination of the effect of dual-command cycles on crane aisle travel time. Four different scheduling rules have been designed: (1) first-come-first serve, (2) nearest-neighbor without any maximum waiting time limit, (3) nearest-neighbor with a 30 minute maximum waiting time limit, and (4) nearest-neighbor with a 60 minute maximum waiting time limit. The expected decrease in dual-command travel time is due to the effect of optimum retrievals sequencing. The first scheduling rule was designed to create a baseline for the study. The second rule was directed at determining the maximum

possible throughput improvement; however, use of the nearest-neighbor rule may incur excessive waiting times. The third and fourth rules are directed at minimizing maximum waiting time.

The second factor is the number of docks used in the study. In previous AS/R systems that have been studied, the dock is usually located at either the lower left or the lower right corner. This research also examines the effect of a second dock. In a dual-command system with one dock, if the retrieval location is at the far end of the aisle, the crane visits that point and then must travel the entire aisle length to return to the dock. In a storage aisle with two docks, the docks are located at each end of the aisle. Thus, when retrieval is ordered by the dock which is located at the same end as the retrieval location, the crane has less distance to travel in order to return to its original dock. Thus, travel time is reduced and it is expected that inclusion of an additional dock will result in improved system throughput.

The third factor is the relationship between storage resources and retrieval destinations in conjunction with storage location assignment policy. Given that the general storage location assignment policy is the closest-open-location, in a two-dock layout the relationship between storage resources and retrieval destinations will affect the definition of the closest-open-

location. For a two-dock AS/R system with two different storage sources, A and B, and two different retrieval destinations, C and D, one dock can be assigned to handle storage pallets from source A and retrieval requests from destination C, while the other dock can handle storage pallets from source B and retrieval requests from destination D. Alternatively, the storage pallets from sources A and B can be retrieved by the crane assigned to either destination C or D. The definition of the closest-open-location for these two cases is different.

Objective

The primary objective of this study is to evaluate the effect of adding a second dock to each storage aisle on system throughput scheduling rules. The secondary performance measure used for this study is waiting time. For storage pallets, waiting time is the time interval between arrival of the storage pallet at the storage queue and deposit of the pallet into the storage location. For retrieval requests, waiting time is the time interval between the arrival of the retrieval request at the queue and the retrieval of the requested pallet and its arrival at the dock. It is expected that the system throughput will be significantly increased and the mean storage and retrieval waiting time will be reduced with the adoption of a

two-dock layout based upon the intelligent selection of a retrieval scheduling rule, though the maximum waiting time may be increased.

III. ASSUMPTIONS AND GENERAL APPROACH

The following assumptions are made in order to simplify the problem of assessing the effects of the dock, scheduling rules, and storage assignment policies on the AS/R system.

General Assumptions

AS/R System

The AS/R system considered in this study is a single crane serving a single aisle with storage racks placed on one side of the aisle. A conveyor provides the link between the system and the source or destination of pallets. The I/O exchange between crane and conveyor occurs at the dock (I/O point). The results of this study can be generalized to a multi-aisle system with identical assumptions for each aisle, that is, an N-aisle system with one crane per aisle may be viewed as an N-independent single-aisle systems.

Pallets

Each pallet contains only one part number or item type and the size of all pallets are equal. This as-

sumption removes pallet assignment as an independent variable in the study.

Storage Rack and Locations

A storage rack of 5 rows and 20 columns (100 storage locations) is located on one side of the aisle. Given that all storage location are identical in size, any pallet load may be stored in any storage location. This assumption removes the corresponding sizes of pallet and storage locations as an independent variable in the study.

Dock (I/O point)

The single dock layout where the exchange of pallets between the crane and I/O conveyor occurs is located at the left corner of the storage rack at an elevation equal to the lowest row of the rack. In the two-dock layout, dock 1 is located at the left corner of the rack and dock 2 is located at the right corner of the rack, and the elevations of these two docks are equal to the lowest row of the rack.

Crane Capacity and Velocity

The maximum capacity of the crane is one pallet and the crane is capable of simultaneous horizontal and vertical movement. In calculating the travel time, constant velocities are used for horizontal and

vertical travel, and acceleration is ignored. Given that the dock(s) are located at the lowest left or right corner of the rack at an elevation equal to that of the lowest row, horizontal and vertical crane velocities are such that the time for the crane to move to the farthest column equals the time for the crane to move to the farthest row. In this study, the crane horizontal velocity is 4 seconds/column and the crane vertical velocity is 20 seconds/row. Thus, the horizontal travel time required to go to the column farthest from the dock is

4 seconds/column × 20 columns = 80 seconds

and the vertical travel time required to go to the highest row, when the dock is located at an elevation equal to that of the lowest row, is

20 seconds/row × (5-1)rows = 80 seconds .

Since the time for the crane to move from the dock to the most distant column is equal to the time required to reach the most distant row, this assumption is referred to as the "square in time" storage rack design.

Basic Crane Operation

Crane operation is subject to a dual-command rule, allowing the completion of both a storage request and a retrieval request on a single cycle from the dock. The crane operates in accordance with this rule as long as both the storage and retrieval queues are non-empty.

When one of the queues is empty, the crane will perform requests from the other queue without interleaving. For each single trip-cycle in a one-dock layout:

1) Store one pallet + retrieve one pallet (if both of storage and retrieval queues are non-empty); or
2) Store one pallet (if retrieval queue is empty); or
3) Retrieve one pallet (if storage queue is empty).

The rules for the two-dock layout are complex and are briefly explained later in this chapter.

Dwell Point

When the crane is idle, its location is referred to as the dwell point, a status which affects system performance. In order to concentrate on factors chosen for evaluation in this study, the effects of dwell point variance are excluded from this study. The dwell point policy adopted for this study is as follows:

1) In a one-dock layout, the dwell point is at the dock; and
2) In a two-dock layout, the dwell point is at the midpoint of the lowest row in the storage rack.

Pick-Up and Deposit Time

The pick-up and deposit (P/D) time is generally independent of the crane travel velocity and the shape of the rack. Furthermore, given the crane load and unload characteristics, P/D time is usually deterministic. Hence, it is a useful to include P/D time only after average travel time is computed. Thus, the P/D time may be ignored. The neglect of P/D time may seem to be an overly simplistic approach. However, if the time to pick or deposit a pallet is constant, then the total P/D time is linear with respect to the number of pallet storage and retrieval cycles, and the omission of P/D time will not affect the relative performance of scheduling policies, which are consequently overstated due to the absence of this fixed-time.

Storage and Retrieval Assumptions

Job Definition

For storage and retrieval jobs, it is assumed that each storage or retrieval job incorporates a single pallet assignment. This assumption removes the dispatching problem related to crane capacity to be considered as an independent factor in this study.

Storage Sources and Retrieval Destinations

There are two sources for storage pallets from two different production departments, S1 and S2, respectively. Similarly, the two sources of retrieval requests are two different production areas, R1 and R2, respectively.

Arrival Rates and Queue Size Limit

In order to examine maximum throughput for different scheduling rules, it is necessary that all crane trips be maintained in the dual-command cycle during the simulation period. Thus, an exponential distribution with a mean time of 1 minute is designated as the interarrival time of storage pallets from S1 or S2 and retrieval requests from R1 or R2. However, in order to examine the effect of empty queues on waiting time as the crane travels between docks with a single command or even an empty load, a greater interarrival time must be designated. If the arrival rate is very high, the system will be completely operating on the dual-command cycle. By reducing the arrival rate, single- and dual-command cycles will both be raised for the system, thus providing a better basis for analyzing waiting time statistics. For such situations, waiting time analysis is based on an exponential distribution with a mean

time of 4 minutes for the storage and retrieval inter-arrival time. In addition, the maximum queue size for storage or retrieval is restricted to 10 for both maximum throughput and waiting time analysis.

Storage Location Assignment and Pallet Retrieval Selection

Each storage pallet will be stored into the closest-open-location. Every time the crane initiates a pallet storage cycle, a closest-open-location will be defined for pallet deposit. For retrievals, the pallet selected for a retrieval request is chosen randomly from all pallets currently stored in the rack. Therefore, each stored pallet in the rack has the same turn-over frequency. Furthermore, when a storage pallet arrives at a storage queue, and all of the storage locations have been occupied, the arriving pallet will be rejected from the storage queue. Similarly, when a retrieval request pallet arrives at a retrieval queue, and all the storage locations are empty, that retrieval request will be rejected.

Closest-Open-Location Selection

The closest-open-location is defined as an open storage location from among all available open locations which has the smallest sum of storage travel time from the dock to the open-location and expected

retrieval travel time from that open-location to the dock. For example, in a one-dock layout the dock is located at the left corner of the storage rack. Also, in the dual-command crane cycle the points of origin and termination are the same. Hence, storage travel time from the dock to a storage location and expected retrieval travel time from that storage location to the dock are equal. Therefore, in the one-dock layout the closest-open-location is the open-location to which the storage travel time from the dock is the least of all open-locations.

However, in the two-dock layout, docks are located at opposite corners of the rack and each dock is capable of handling transaction functions for input and output. In the dual-command cycle the points of origin or termination can be either the same dock or the opposite dock. Therefore, in the two-dock layout the storage travel time from the dock to a storage location is dependent on the dock from where the storage travel originated and the expected retrieval travel time from a storage location to the dock is dependent on the dock at which the retrieval travel is to terminate. Thus, when a storage pallet arrives at the dock in a two-dock layout selection of the closest-open-location depends on two factors:

1) The dock at which the storage pallet arrives, and

2) the dock from which the storage pallet will be requested.

There are two sources of storage pallets (S1, S2) and two retrieval destinations (R1, R2). The relationships among these sources and destinations and the dock assignments for arriving pallets and retrieval requests in a two-dock layout are as follows:

1) Storage pallets from S1 and S2 are assigned to dock 1 and dock 2, respectively;
2) Retrieval requests from R1 and R2 are assigned to dock 1 and dock 2, respectively;
3) Storage pallets from S1 and S2 may be retrieved by either R1 or R2; or
4) Storage pallets from S1 will be retrieved by R1 and storage pallets from S2 will be retrieved by R2.

Based on these assumptions, the closest-open-location in a one-dock or two-dock layout may be organized in the following three configurations:

Configuration 1: One-Dock, Layout 1 (Figure 3.1)

Since there is only one dock in this layout, storage pallets from S1 or S2 and retrieval requests from R1 or R2 are assigned to the same dock.

Let,

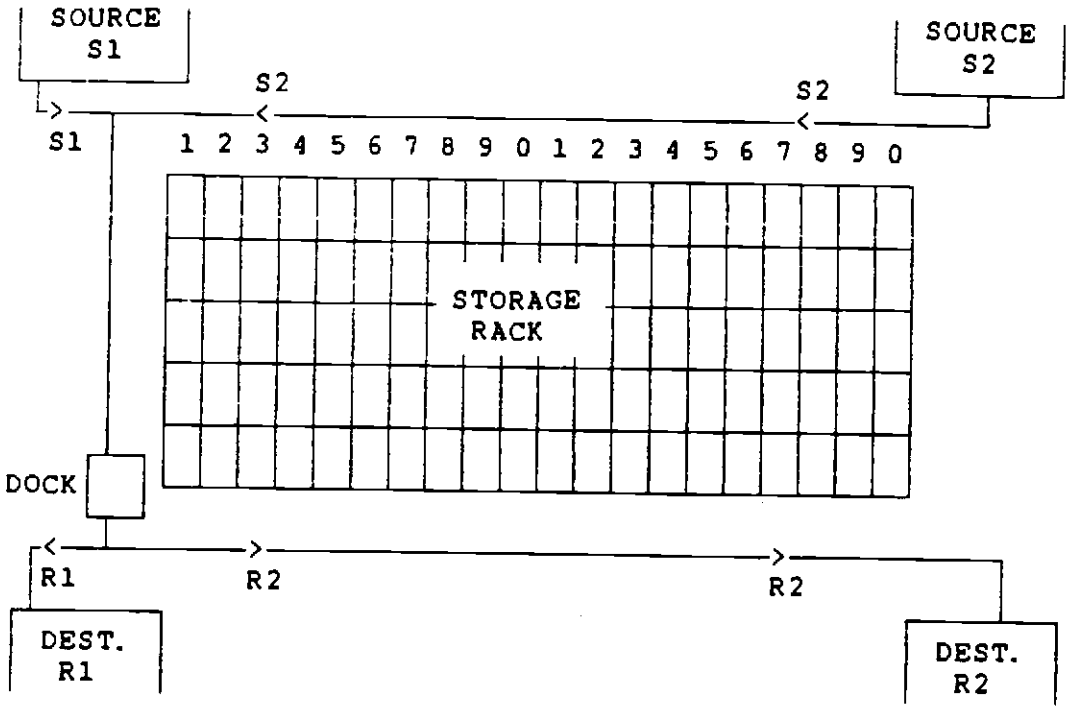


Figure 3.1 Layout 1 (one dock).

TS(i,j) = travel time from dock to storage location at row i and column j (i = 1 to 5, j = 1 to 20); and

TR(i,j) = travel time from storage location (i,j) to dock.

It follows, then, that for a storage pallet arriving at the dock, the sum of storage traveling time to storage location (i,j) and expected retrieval traveling time from that location to the dock is

$$T(i,j) = TS(i,j) + TR(i,j) \quad . \qquad (1)$$

Thus in Layout 1, when a storage pallet arrives at the dock, the closest-open-location is a location which minimizes T(i,j) in (1). For each storage location in Layout 1 the values of T(i,j) are shown in Figure 3.2. The lowest numbers have higher priority for closest-open-location selection. An example for computing the value of T(i,j) is as follows:

Let,

i = 3 (row 3),

j = 5 (column 5),

S(v) = vertical traveling speed
= 20 seconds/row,

S(h) = horizontal speed
= 5 seconds/column,

T(i) = time to reach row (i),

T(3) = 20 seconds *(3-1)
= 40 seconds,

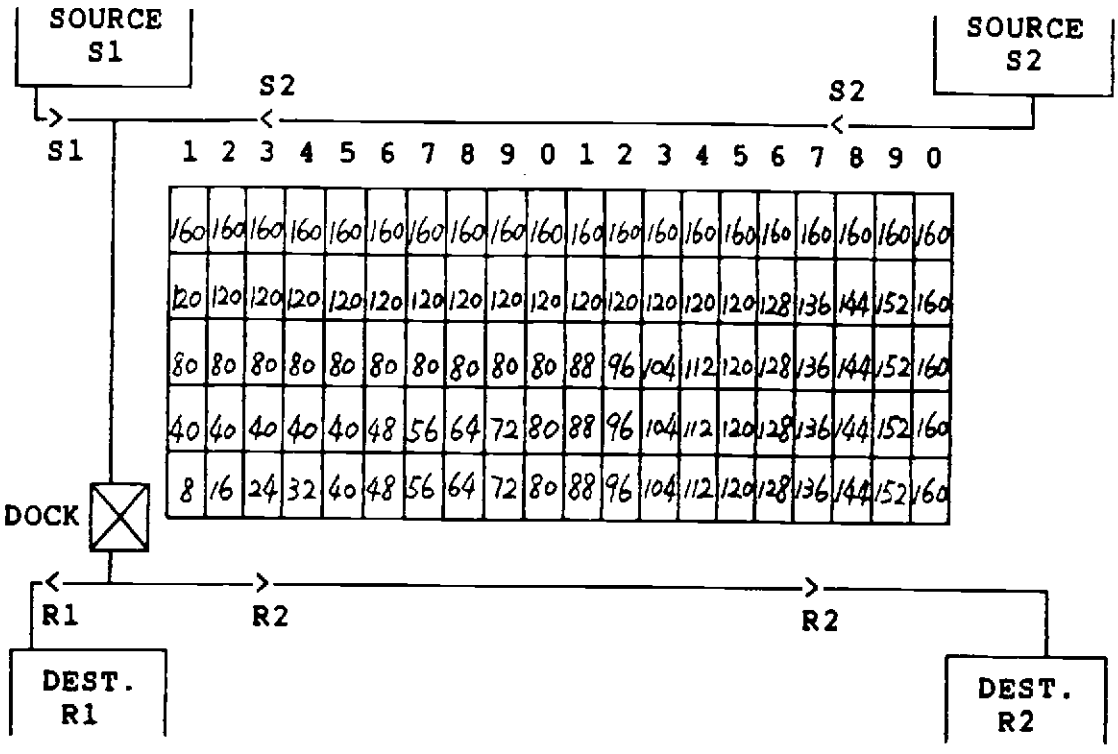


Figure 3.2 Layout 1, Closest-Open-Location.

T(j) = time to reach column (j),

T(s) = 4 seconds x 5

= 20 seconds,

TS(i,j) = time to reach row (i) and column (j) from dock

= max. {T(i), T(j)},

TS(3,5) = max. {40 seconds, 20 seconds}

= 40 seconds,

TR(i,j) = time to reach the dock from row (i) and column (j),

TR(i,j = TS(i,j), (one-dock layout)

TR(3,5) = TS(3,5)

= 40 seconds,

T(i,j) = TS(i,j) + TR(i,j),

T(3,5) = TS(3,5) + TR(3,5)

= 40 seconds + 40 seconds

= 40 seconds.

Configuration 2: Two-Docks, Layout 2

For the two-dock layout shown in Figure 3.3, let:

TS1(ij) = travel time from dock 1 to storage location at row i and column j (i = 1 to 5, j = 1 to 20);

TS2(ij) = travel time from dock 2 to storage location at row i and column j;

TR1(ij) = travel time from storage location at row i and column j to dock 1; and

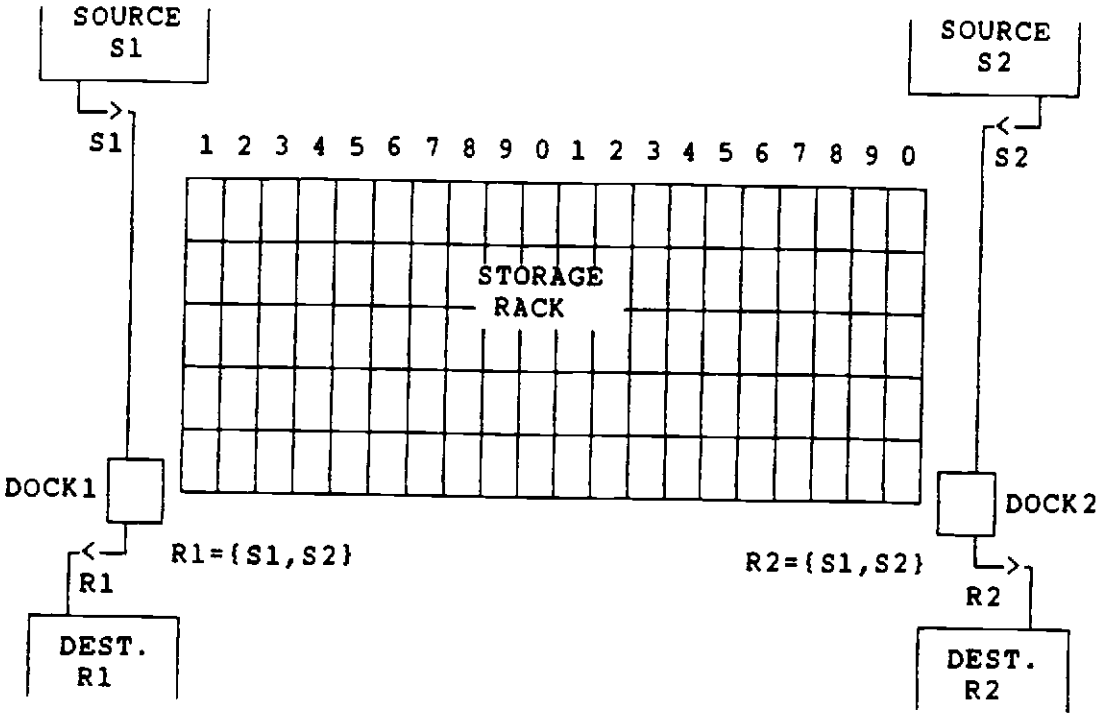


Figure 3.3 Layout 2 (two docks).

TR2(ij) travel time from storage location at (i,j) to the dock.

It follows, then, that when a storage pallet arrives at dock 1, the sum of storage travel time to location (i,j) and expected retrieval travel time to either dock 1 or dock 2 is:

$$T1(ij) = TS1(ij) + TR1(i,j) \times 0.5 + TR2(i,j) \times 0.5 \ . \quad (2)$$

Thus in Layout 2, the closest-open-location from dock 1 is an open-location at (i,j) which minimizes T1(i,j) in (2).

When a storage pallet arrives at dock 2, the sum of storage travel time to location (i,j) and expected retrieval travel time to the dock is:

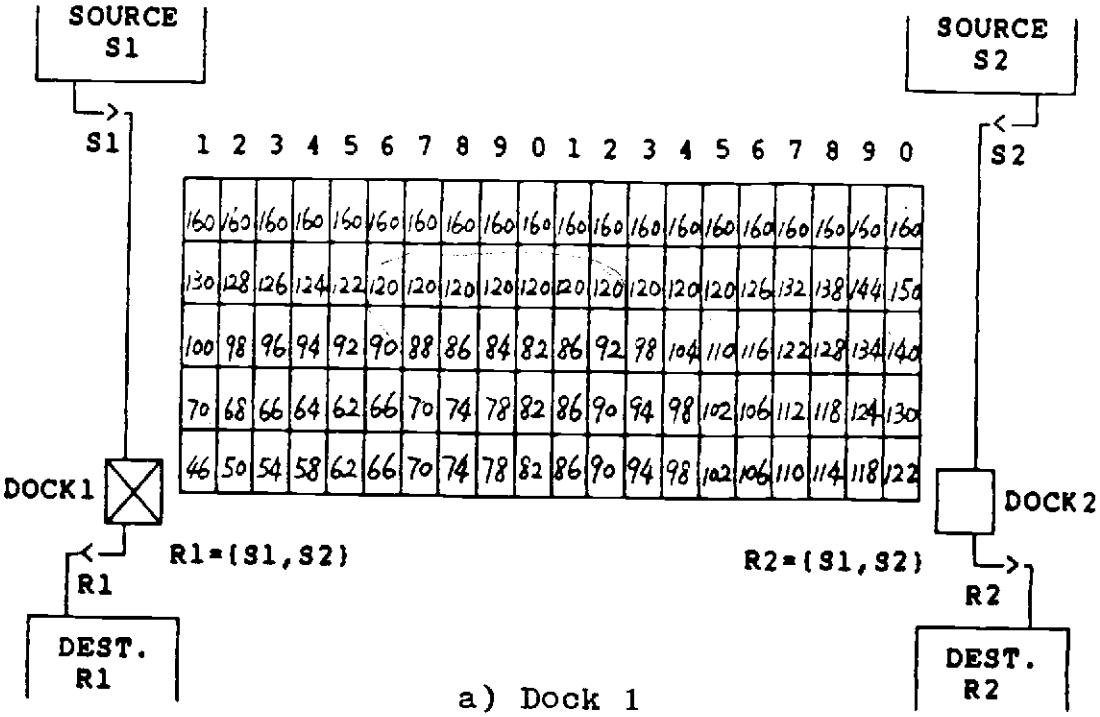
$$T2(i,j) = TS2(i,j) + TR2(i,j) \times 0.5 + TR2(i,j) \times 0.5 \ . \quad (3)$$

Thus, the closest-open-location from dock 2 is a open-location at (i,j) which minimizes T2(i,j) in (3).

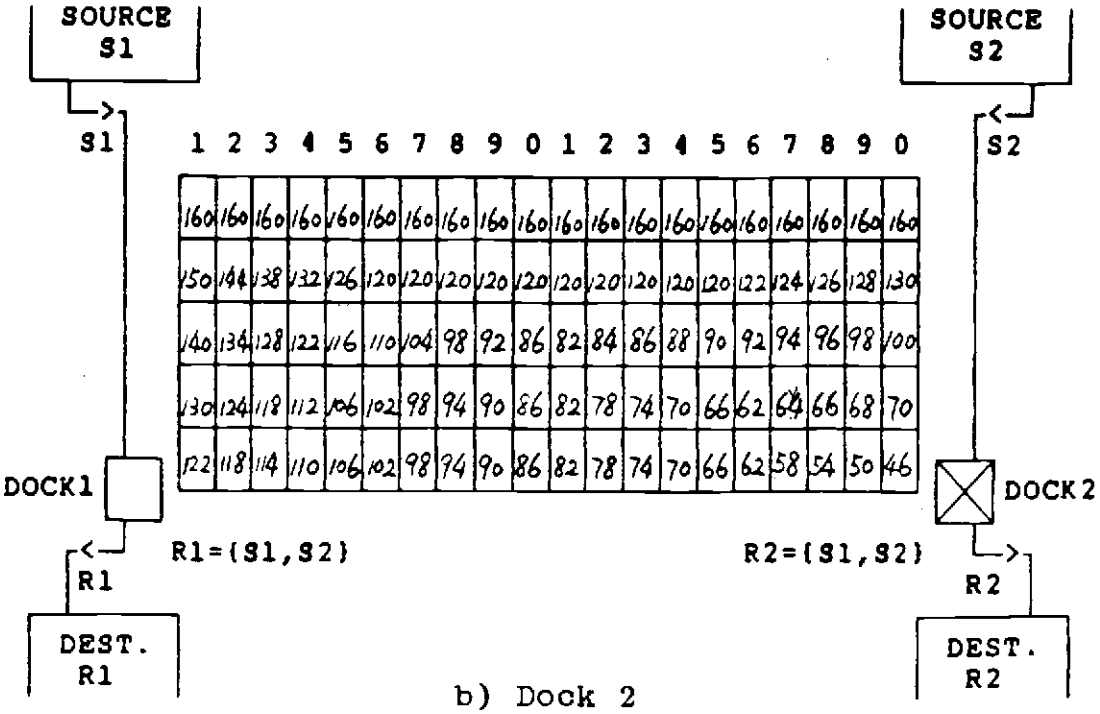
For each storage location in Layout 2, the sum of storage time and expected retrieval travel time of each dock is shown in Figure 3.4(a) and 3.4(b). An example of a storage pallet at dock 2 with three open-locations available is as follows:

1) The storage pallet arrives at dock 2.
2) The three open locations are (2,7), (4,5), and (3,6), respectively.

Let,



a) Dock 1



b) Dock 2

Figure 3.4 Layout 2, Closest-Open-Location.

TS2(i,j) = travel time from dock 2 to open-location (i,j) where TS2(2,7) = 56 seconds; TS2(4,5) = 64 seconds; and TS2(3,6) = 60 seconds;

TR1(i,j) = retrieval travel time from location (i,j) to dock 1 where TR1(2,7) = 28 seconds; TR1(4,5) = 60 seconds; and TR1(3,6) = 40 seconds;

TR2(i,j) = retrieval travel time from location (i,j) to dock 2 where TR2(2,7) = 56 seconds; TR2(4,5) = 64 seconds; and TR2(3,6) = 60 seconds;

T2(i,j) = the sum of storage travel time from dock 2 to location (i,j) and expected retrieval travel time from location (i,j) to the dock (the dock will be either dock 1 or dock 2);

$T2(i,j) = TS2(i,j) + TR1(i,j) \times 0.5 + TR2(i,j) \times 0.5$ where T2(2,7) = 98 seconds; T2(4,5) = 126 seconds; and T2(3,6) = 110 seconds.

The closest-open-location is an open-location which has the minimum value of T2. In this example, the closest-open-location is the open-location at (2,7).

Configuration 3: Two-Docks, Layout 3

For the two-dock layout shown in Figure 3.5, let

TS1(i,j) = travel time from dock 1 to storage location (i,j),

TS2(i,j) = travel time from dock 2 to storage location (i,j),

TR1(i,j) = travel time from storage location (i,j) to dock 1, and

TR2(i,j) = travel time from storage location (i,j) to dock 2.

Therefore, for a storage pallet arriving at dock 1, the sum of travel time to location (i,j) and expected retrieval travel time to dock 1 is:

$$T1(i,j) = TS1(i,j) + TR1(i,j) . \quad (4)$$

Thus, the closest-open-location from dock 1 in Layout 3 is a open-location which minimizes T1(i,j) in (4). Similarly, for a storage pallet arriving at dock 2, the sum of travel time to location (i,j) and expected retrieval travel time to dock 2 is:

$$T2(i,j) = TS2(i,j) + TR2(i,j) . \quad (5)$$

Therefore, the closest-open-location from dock 2 in Layout 3 is a location which minimizes T2(i,j) in (5).

For each storage location in Layout 3, the sum of storage travel time form dock 1 or dock 2 and the expected retrieval travel time to the dock are shown in

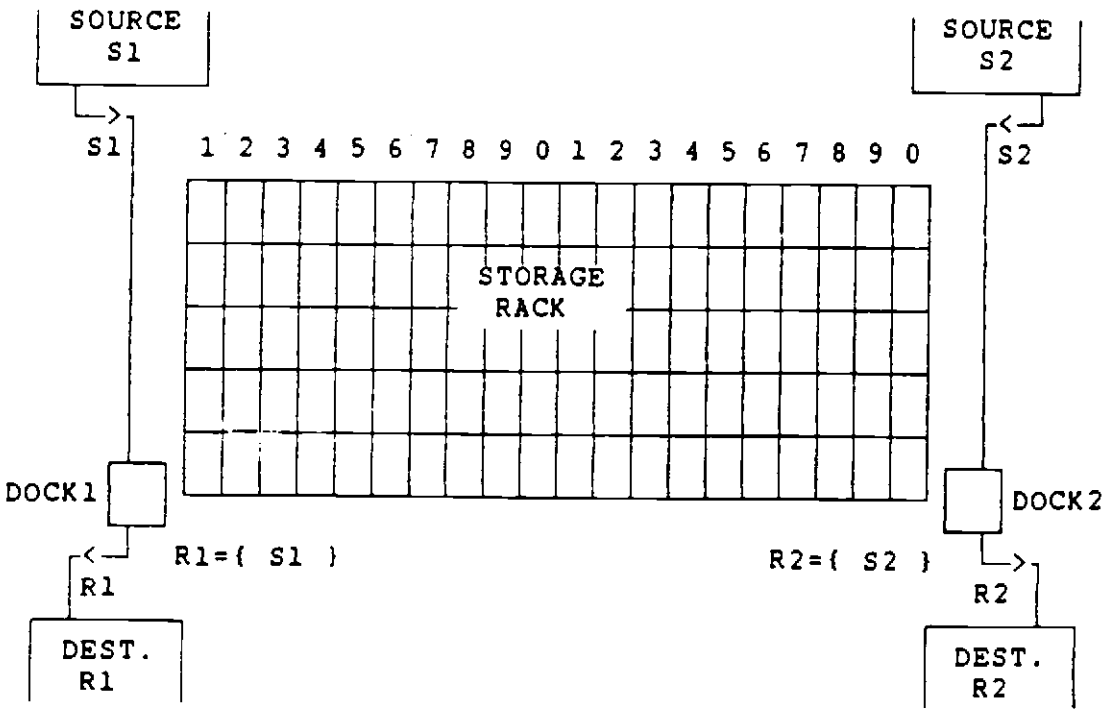


Figure 3.5 Layout 3 (two docks).

Figures 3.6(a) and 3.6(b). The lowest numbers have the highest priorities for closest-open-location selection.

Nearest-Neighbor and Scheduling Rules

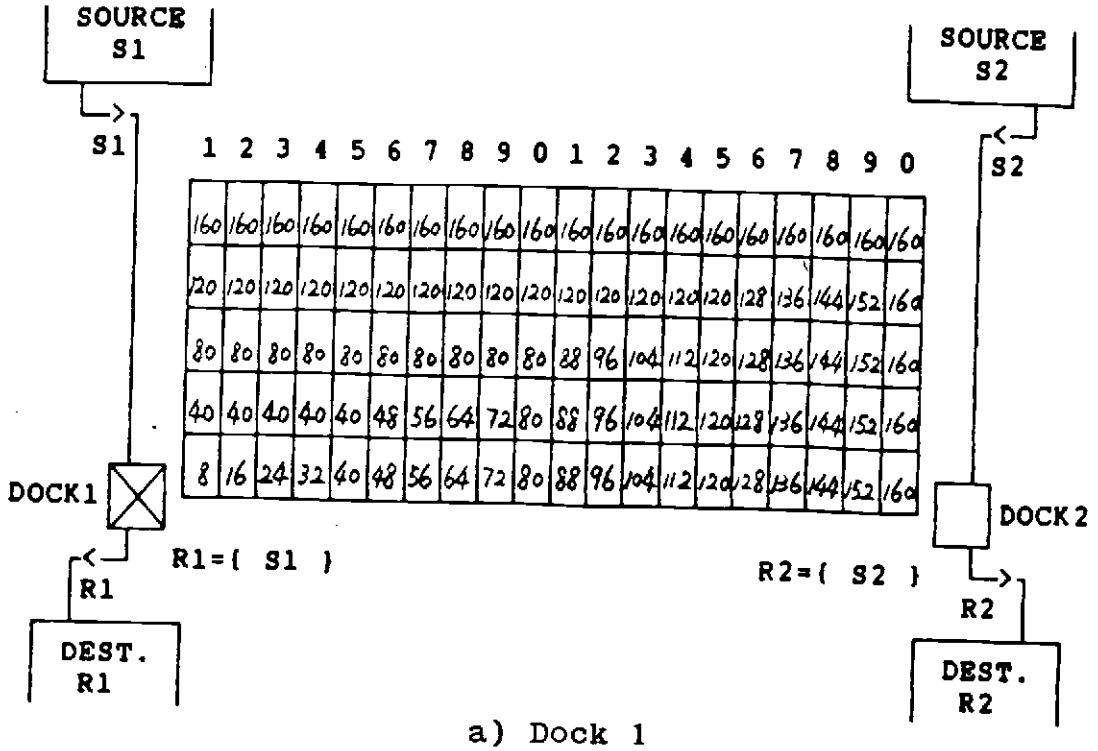
Since the conveyor provides the link between the source of storage and the dock, there is no capability for changing the sequence of storage pallets arriving on the conveyor. Therefore, the FCFS assumption is reasonable for storage. However, retrieval requests are simple messages which may be arranged in any convenient sequence. For this study, the nearest-neighbor rule is adopted as an alternative to the FCFS rule. The definition of the nearest-neighbor is as follows.

When the crane initiates travel for a retrieval cycle, a target retrieval location is selected from all the retrieval request queues. The nearest-neighbor retrieval location is determined when the sum of the travel time from the current crane location to the target location and from that location to the dock is the smallest of all options available from the retrieval queues.

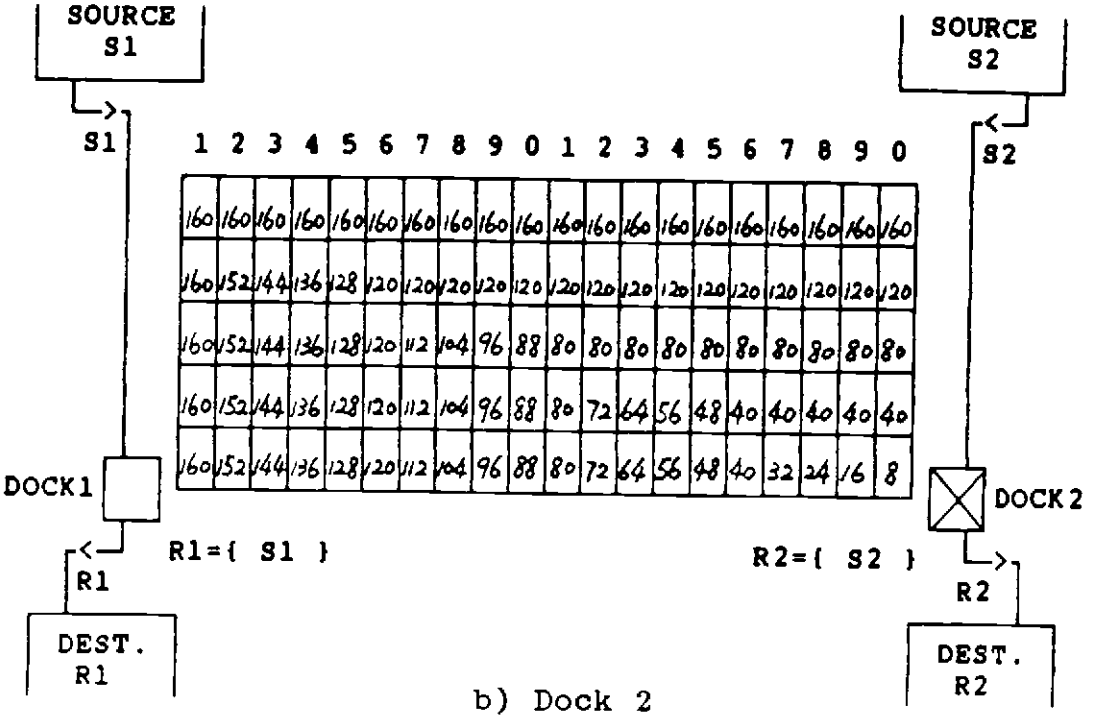
Let:

R(d,k) = retrieval request from dock d, d = 1 or 2, and k is the order of a retrieval request at dock d, k = 1 to 10 (queue limit size);

TCR(d,k) = travel time from current crane location to retrieval location, R(d,k);



a) Dock 1



b) Dock 2

Figure 3.6 Layout 3, Closest-Open-Location.

TDR(d,k) = travel time from retrieval location, R(d,k), to dock d;

T(d,k) = sum of travel time for the crane to travel from the current location to the retrieval point k and return to dock d; and

Thus,

$$T(d,k) = TCR(d,k) + TDR(d,k) \quad . \qquad (6)$$

Therefore, the nearest-neighbor is a retrieval request, R(d,k), by which T(d,k) is minimized in (6).

Based on the nearest-neighbor and closest-open-location definitions described above, four scheduling rules were designed to be evaluated in this study. These scheduling rules are shown in Table 3.1.

Table 3.1 Scheduling Rules.

Rule	Storage Policy	Retrieval Policy
1	First-Come-First-Served Closest-Open-Location	First-Come-First-Served
2	First-Come-First-Served Closest-Open-Location	Nearest-Neighbor
3	First-Come-First-Served Closest-Open-Location	Nearest-Neighbor, Max. Wait Time = 30 Min.
4	First-Come-First-Served Closest-Open-Location	Wait Time = 60 Min.

In order to assure that retrievals at the far end of the rack are not excessively delayed, a maximum waiting time limit has been adopted for rules 3 and 4.

Pre-Load Storage Rack

Since the general storage rule for input pallets is to store them in the closest-open-location, pre-loaded pallets are assigned to the set of closest-open-locations based on utilization levels and layouts.

Simulation Model

A simulation model, *DOCK*, was designed for the analysis of AS/R system operations. *DOCK* is a discrete model since time is the only independent system variable, and all other variables are dependent and change discretely at specified points in the simulation time. The simulation program was written in *PCmodel* and can be executed on standard IBM PC or compatible hardware.

Programming Language

The more commonly used languages for computer simulation studies are *GASP*, *SLAM*, and *SIMAN*. *PCModel*, a graphic simulation system, was used for this study for the following reasons:

1) A graphic simulation system allows for the visual analysis of arriving and retrieving pallet flow;
2) The behavior of the system resulting from the application of different scheduling rules and

dock layouts can be "physically" observed during simulation;

3) The pace of the simulation can be adjusted, thus allowing the user to observe the system operation in detail, particularly when the system behavior is not consistent with previous expectations;
4) Simulation can be paused at any time during execution, and other system functions from the menu screen may be utilized to examine system status data or other system parameters; and
5) The results of the simulation will be more readily apparent to the decision maker.

Program Structure

The simulation program, *DOCK*, consists of five main components:

1) System configuration and initialization.
2) Generating arrays for storage location and retrieval request references (Routing 1).
3) Paths of storage and retrieval pallets (Routings 2, 3, 4, and 5).
4) Crane operations (Routing 6).
5) Links.

The conceptual organization of *DOCK* is shown in Figure 3.7. A detailed model description is given in Appendix 1.

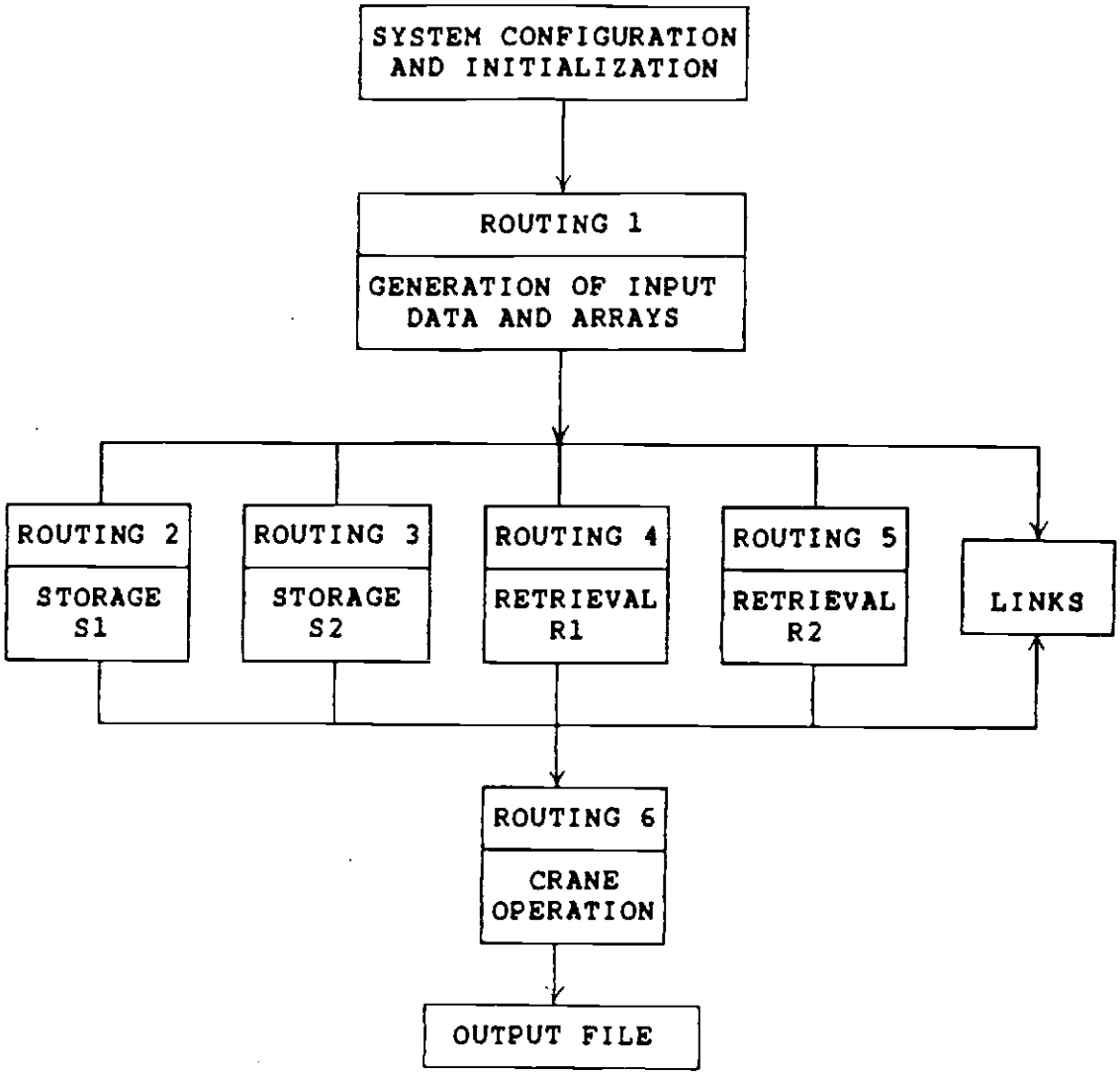


Figure 3.7 Conceptual Organization of Program *DOCK*.

Figure 3.8 shows the operational logic for the crane movements. In the two-dock layout, the possible origins and destination points for crane movement are: (1) dwell point, (2) dock1, (3) dock2, and (4) any storage location in the rack. These four points are referred to as the decision making points since, with the crane located at one of the four points, a decision must be made to provide the next movement of the crane. Each decision making point follows a unique decision making procedure. These decision making procedures for the two-dock layout are shown in Figures 3.9 through 3.12.

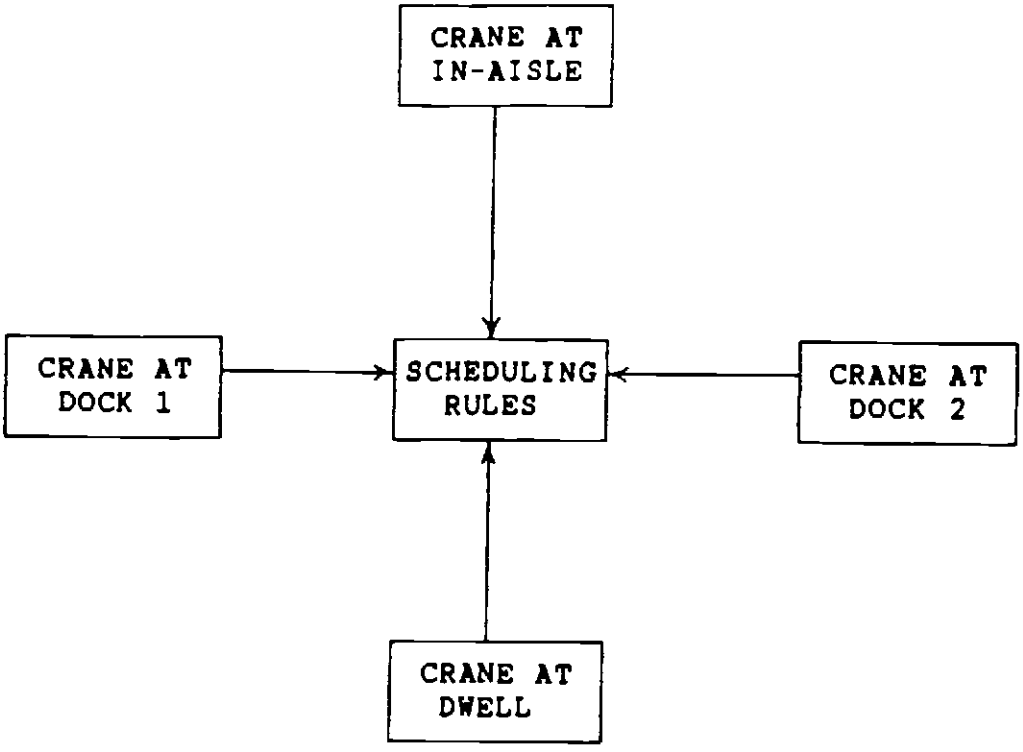


Figure 3.8 Conceptual Organization of Crane Operation.

CONDITIONS	S1	S2	R1	R2
1	0	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1

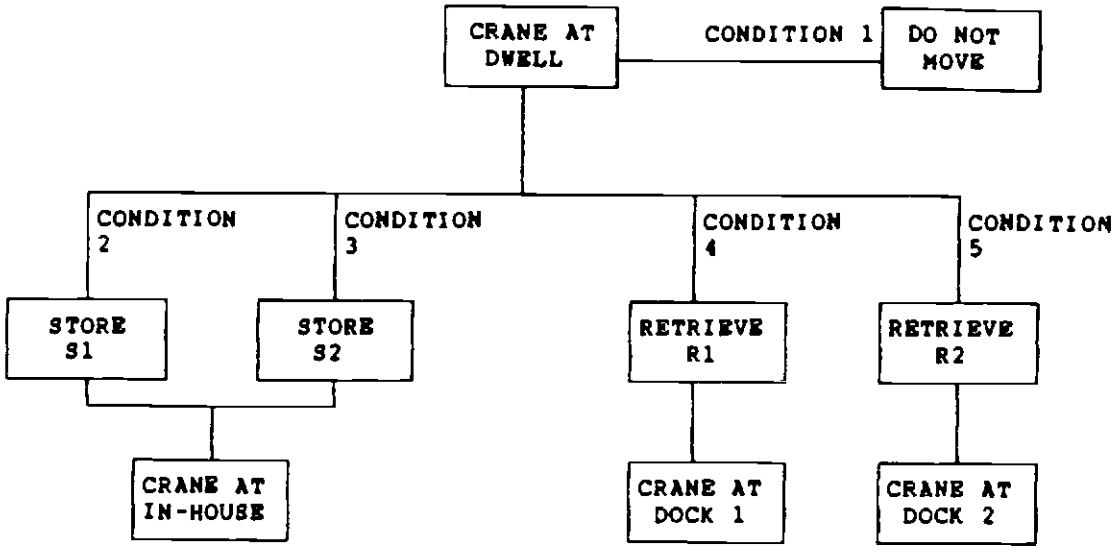


Figure 3.9 Decision-Making with Crane at Dwell Point.

CONDITIONS	S1	S2	R1	R2	CONDITIONS	S1	S2	R1	R2
1	0	0	0	0	9	1	0	0	1
2	1	0	0	0	10	0	1	1	0
3	0	1	0	0	11	0	0	1	1
4	0	0	1	0	12	1	1	1	0
5	0	0	0	1	13	1	1	0	1
6	1	1	0	0	14	1	0	1	1
7	1	0	1	0	15	0	1	1	1
8	0	1	0	1	16	1	1	1	1

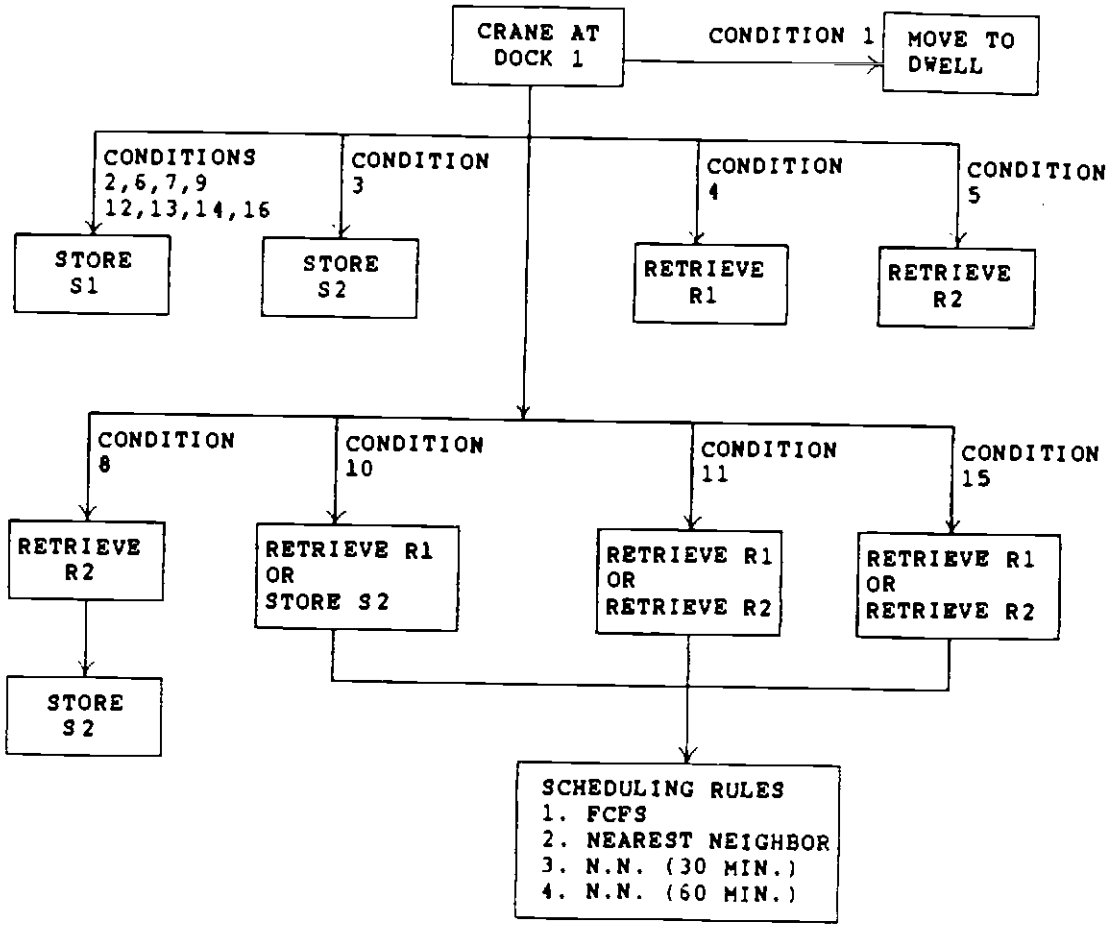


Figure 3.10 Decision-Making with Crane at Dock 1.

CONDITIONS	S1	S2	R1	R2	CONDITIONS	S1	S2	R1	R2
1	0	0	0	0	9	1	0	0	1
2	1	0	0	0	10	0	1	1	0
3	0	1	0	0	11	0	0	1	1
4	0	0	1	0	12	1	1	1	0
5	0	0	0	1	13	1	1	0	1
6	1	1	0	0	14	1	0	1	1
7	1	0	1	0	15	0	1	1	1
8	0	1	0	1	16	1	1	1	1

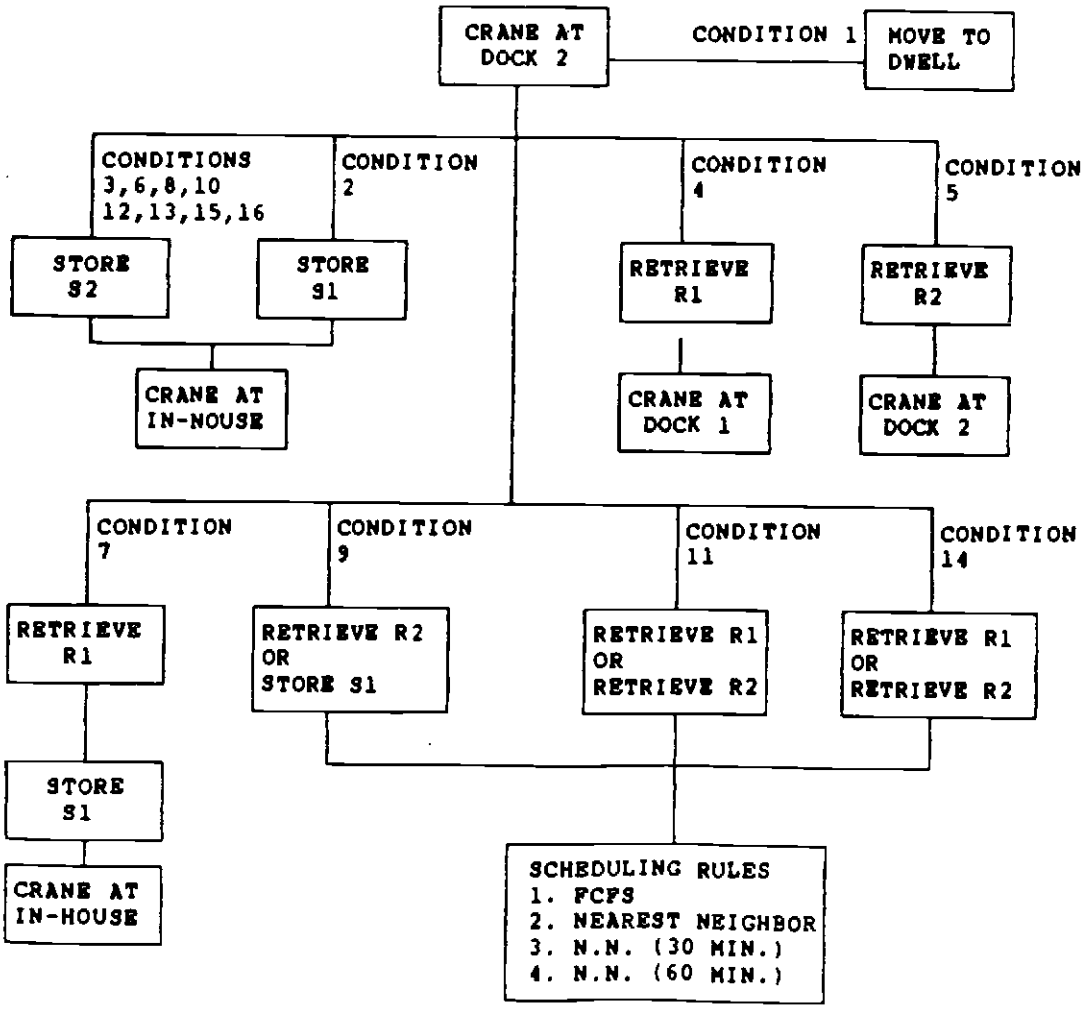


Figure 3.11 Decision-Making with Crane at Dock 2.

CONDITIONS	S1	S2	R1	R2	CONDITIONS	S1	S2	R1	R2
1	0	0	0	0	9	1	0	0	1
2	1	0	0	0	10	0	1	1	0
3	0	1	0	0	11	0	0	1	1
4	0	0	1	0	12	1	1	1	0
5	0	0	0	1	13	1	1	0	1
6	1	1	0	0	14	1	0	1	1
7	1	0	1	0	15	0	1	1	1
8	0	1	0	1	16	1	1	1	1

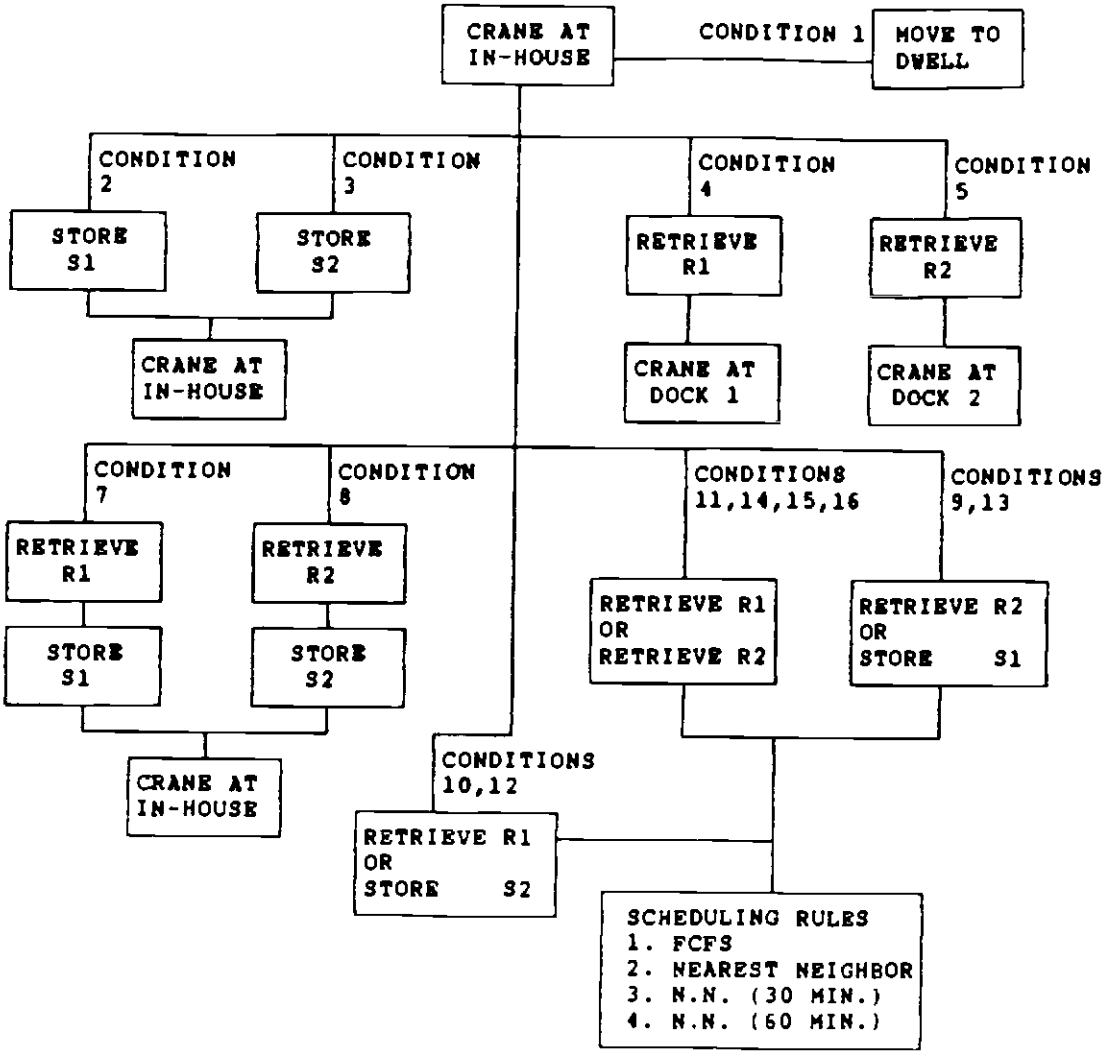


Figure 3.12 Decision-Making with Crane at In-House.

IV. RESULTS AND ANALYSIS OF RESULTS

Introduction

System throughput and job waiting time represent system performance measures for each of the alternative layouts and scheduling policies. A simulation-based computer model was developed to evaluate these measures for different layout strategies.

For throughput analysis, the simulation was initiated with storage rack utilization preloaded at 50 percent. This load was incremented by 5 percent for each run until a load factor of 98 percent was reached. Each of the simulated utilization levels were used for all of the layout and scheduling rule combinations. An initial simulated time of 10 hours was used to remove any start-up bias, followed by an additional 40 hours of simulated time for the determination of experimental results.

For waiting time analysis, storage rack utilization could not be fixed at a specific level over the length of the simulation in situations where the crane could not be placed in a 100 percent dual-command cycle due to the absence of a queue caused by a lower arrival rate. The target storage rack utilization level was

preloaded at 75 percent as the initial condition. This simulation was run for 100 hours to eliminate initial bias followed by an additional 500 hours of simulation runtime to determine the experimental results.

Throughput Analysis

The system throughput for each layout, simulated under four different scheduling rules, is presented in Figures 4.1 through 4.3. The plots indicate that the throughput for each of the layouts decreased with increased space utilization, and that the performance for schedule 2 (nearest-neighbor) was superior to those for the other schedules. These results revealed two important findings:

1) For each layout, the expected crane traveling time increased as space utilization increased, independent of the scheduling rule. This occurred because throughput is inversely related to crane traveling time.
2) The throughput can be increased for any layout by intelligent resequencing of retrieval requests, such as application of the nearest-neighbor rule.

Throughput comparison for each schedule across the layouts, at 50, 75, and 98 percent utilizations, is shown in Figure 4.4. The figure shows that the performance differences between layouts 1 and 2 become

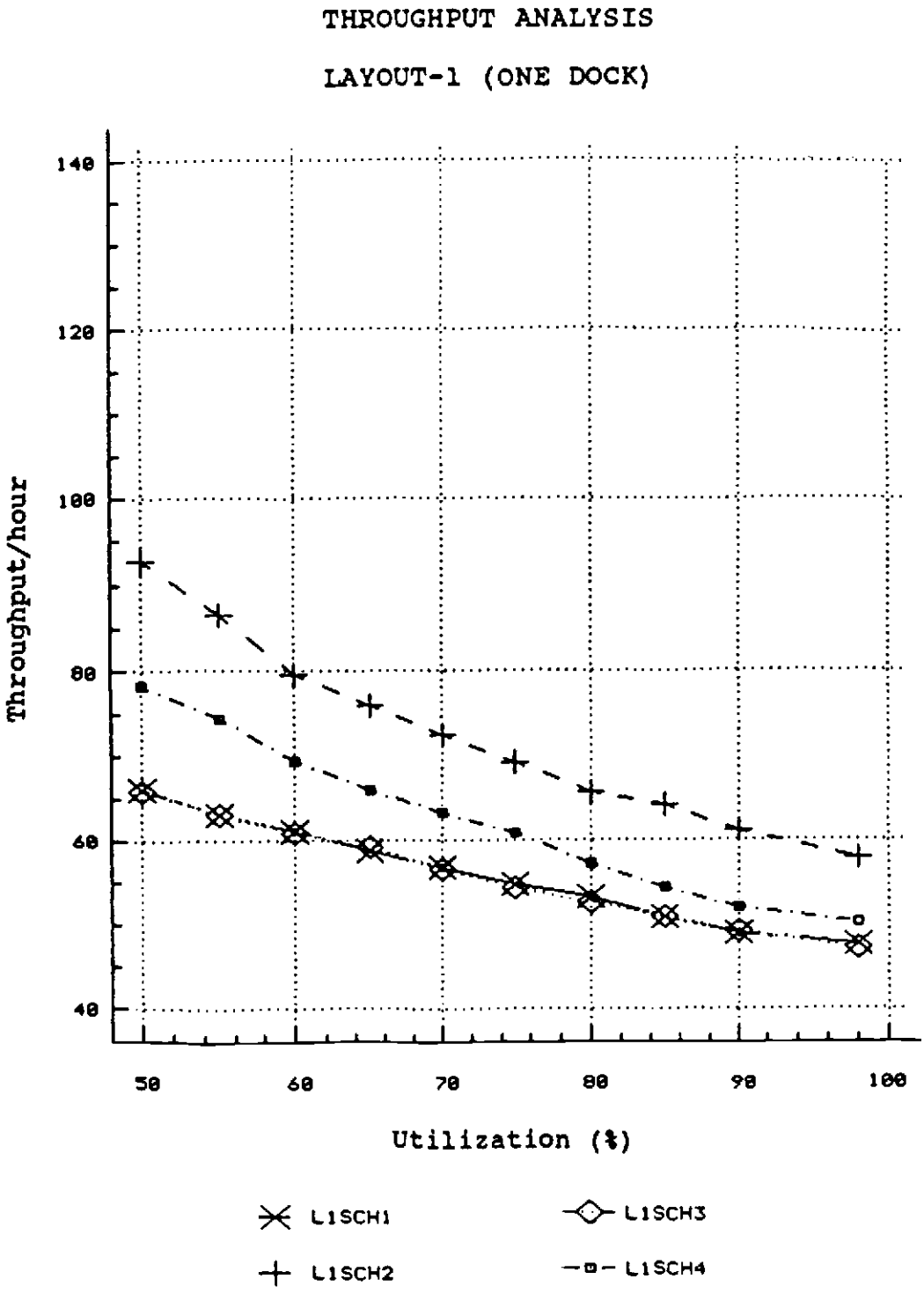


Figure 4.1 Layout 1, Throughput Analysis.

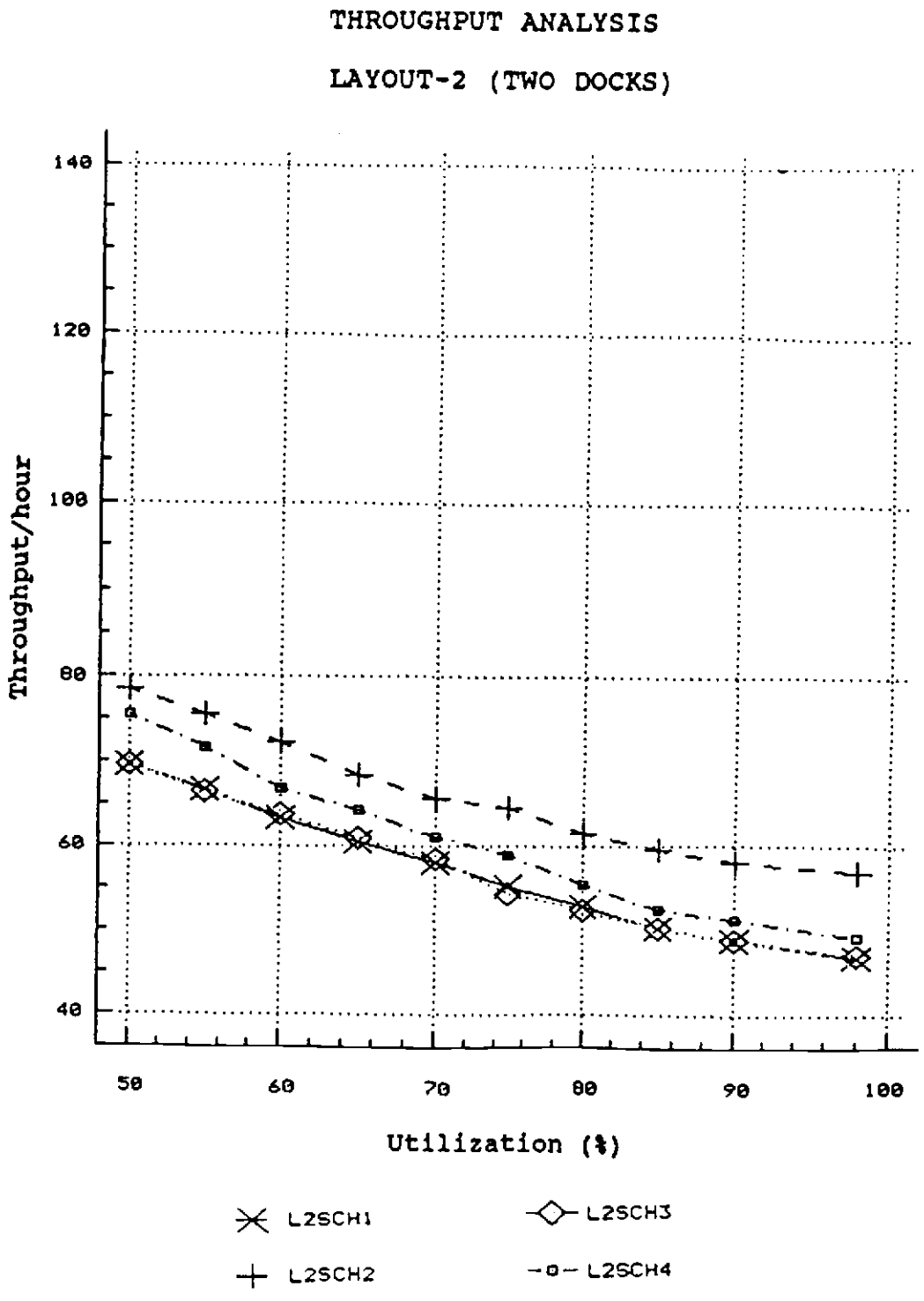


Figure 4.2 Layout 2, Throughput Analysis.

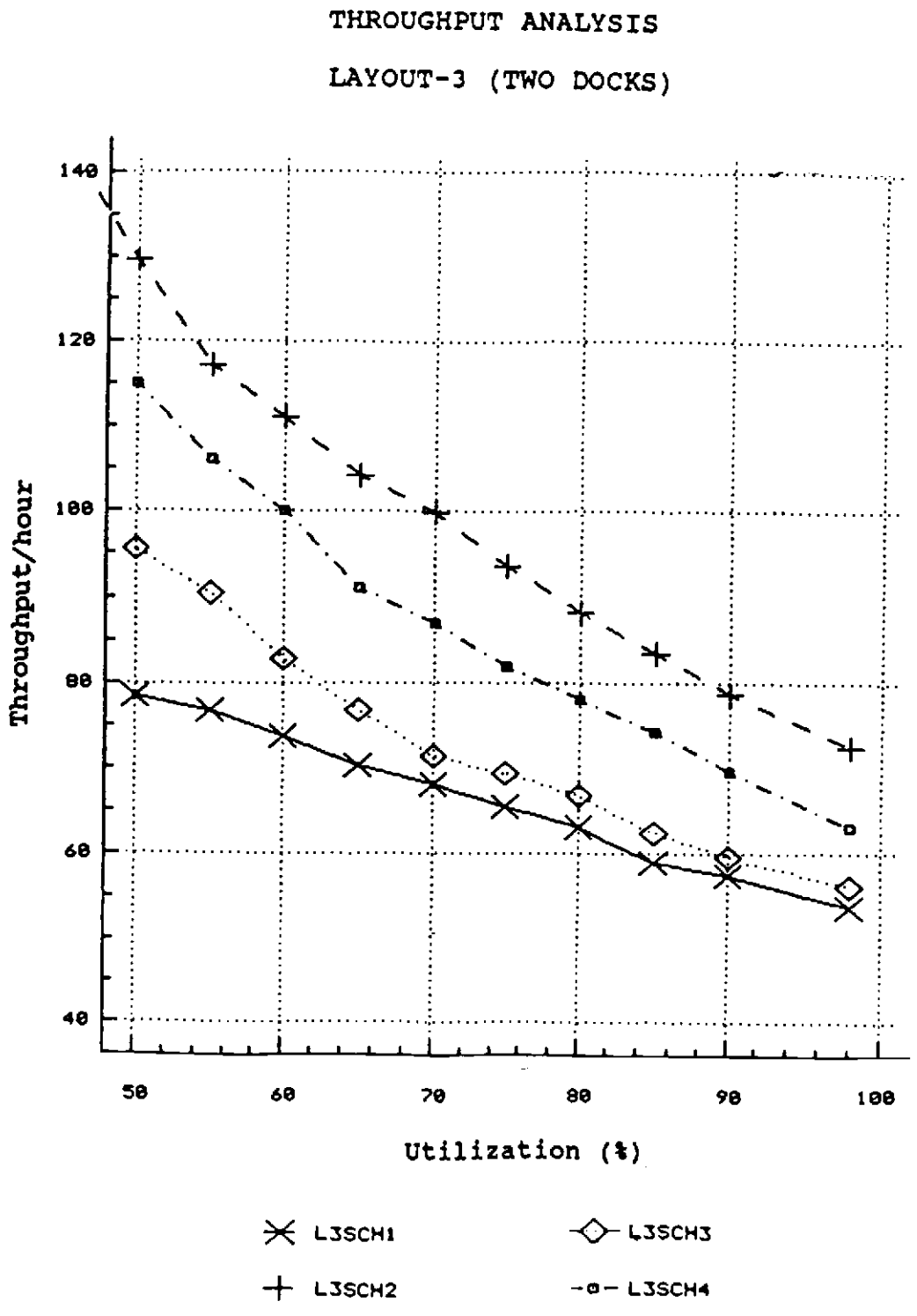


Figure 4.3 Layout 3, Throughput Analysis.

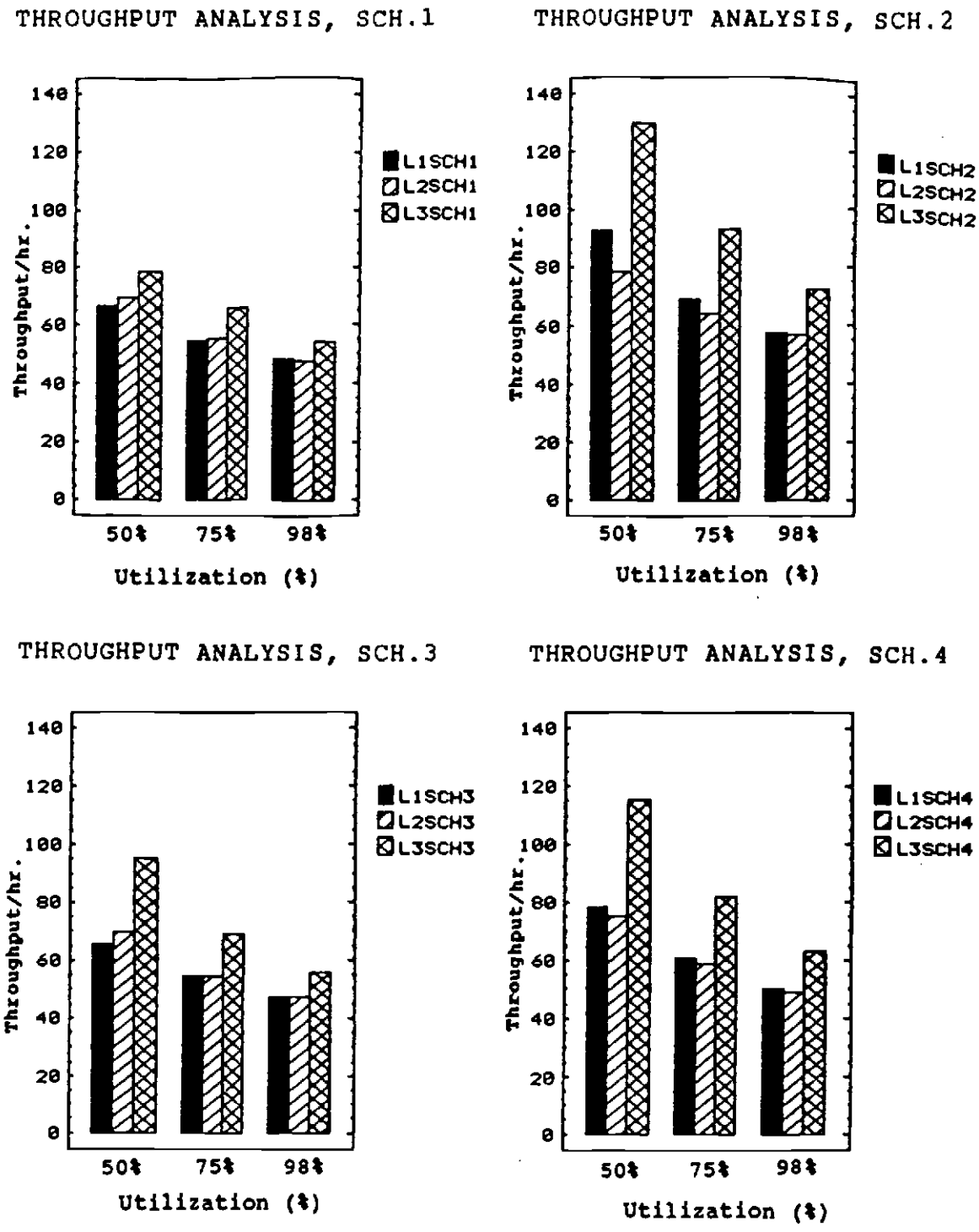


Figure 4.4 Throughput Comparison Across Layouts at 50%, 75%, and 98% utilization.

smaller as space utilization is increased. However, throughput performance for layout 3 was substantially better than either of the other layouts for all the schedules. With respect to layout characteristics, two important points may be drawn from these results:

1) When space utilization is high, and the relationship between storage sources and retrieval destinations is such that S1 and S2 pallets may be retrieved by either R1 or R2, there is no significant throughput difference between the one-dock (layout 1) and two-dock (layout 2) configurations under the same scheduling rule;
2) If S1 storage pallets are retrieved by R1 and S2 storage pallets are retrieved by R2, then the throughput for two-dock configuration (layout 3) is significantly better than the one-dock configuration (layout 1).

The effect of layout (factor A) and scheduling rules (factor B) on the throughput were further analyzed using a two-factor analysis of variance (ANOVA). The throughput from simulation runs at a 75 percent utilization rate with two different random number seeds and the ANOVA results are presented in Appendix 2. The ANOVA results indicate that the interaction of factors A and B is significant (F-ratio of 39). This necessitated additional analysis on the throughput.

Two approaches to throughput comparison were adopted. In the first approach, "comparison across schedules," changes in system throughput were measured as the scheduling policy was changed across identical layout configurations. In the second approach, "comparison across layouts," the maximum throughputs for various system layouts were compared under the best scheduling policy for each of the respective layouts. Schedule 1 (FCFS) was used as the base rule for the "comparison across schedules" and layout 1 was selected as the base layout for "comparison across layouts." The throughput comparisons concentrate on 75 percent space utilization. The effects on throughput at 50 and 98 percent space utilization are briefly discussed at the end of this section.

Approach 1: Comparison Across Schedules

The throughput comparison across schedules for all layouts is shown in Figure 4.5. For layout 1, schedule 2 (the nearest-neighbor rule) exceeded throughput for schedule 1 (FCFS) by 27 percent. This is because the sequence of retrieval requests in the nearest-neighbor rule is intelligently resequenced to reduce the dual-command crane cycle time. The throughput for schedule 3 (nearest-neighbor/30 minutes maximum wait time) was approximately equal to the performance of schedule 1 (FCFS). When the arrival rate is set at high levels

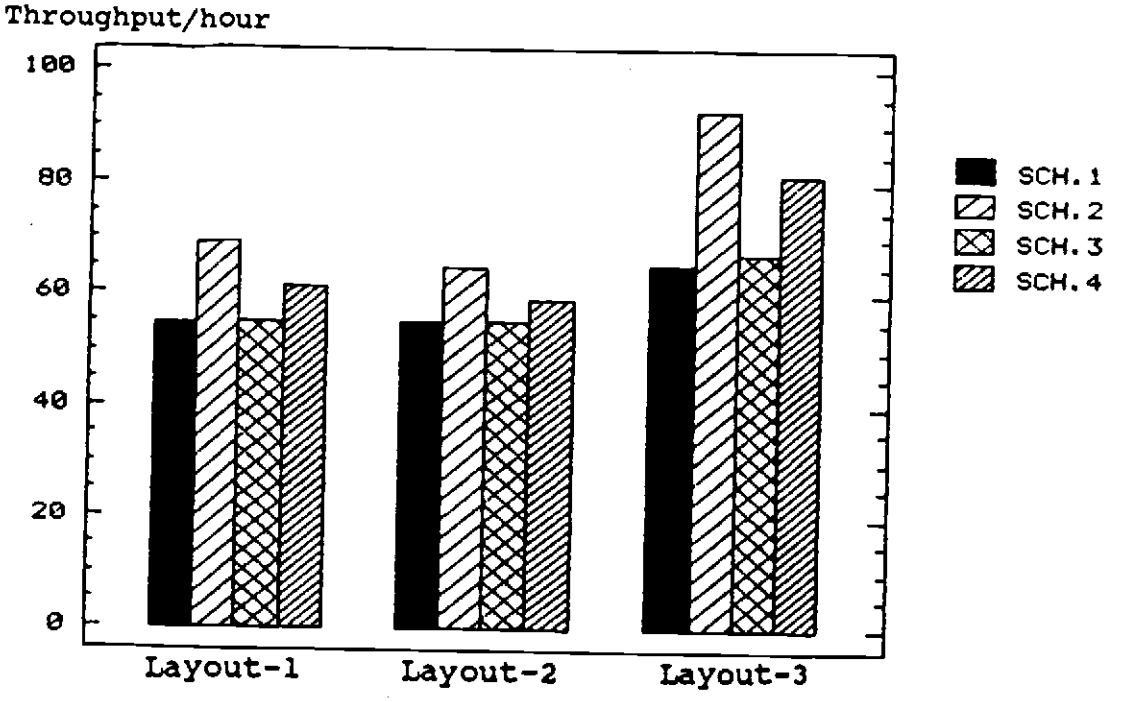


Figure 4.5 Throughput Comparison Across Schedules at 75% Utilization.

then under schedule rule 3 the maximum waiting time of retrieval is over 30 minutes and the system responds in an FCFS manner. Conversely, the application of schedule 4 (nearest-neighbor/60 minutes maximum wait time) resulted in throughput improvement of 11 percent. The results for schedules 3 and 4 indicate that the use of the nearest-neighbor rule with an inadequate maximum waiting time caused the system to respond in an FCFS manner.

The results obtained for Layout-2 were similar to those for layout 1. The throughput with schedule 2 (nearest-neighbor) was higher than that from schedule 1 (FCFS) by 17 percent, and that of schedule 4 (nearest-neighbor/60 minutes maximum wait time) exceeded the output from schedule 1 by 7 percent.

The results for layout 3 were similar to those of layouts 1 and 2; the performance of schedules 2 and 4 exceeded those of schedules 1 by 43 percent and 25 percent, respectively. However, schedule 3 performance was slightly better than that of schedule 1.

To summarize:

1) Throughput under schedule 2 (nearest-neighbor rule) is higher than the throughput for all other schedules, for all layouts. This is due to the reduction of dual-command crane cycle time by application of the nearest-neighbor rule for retrievals. After the crane deposits

a storage pallet, the retrieval pallet chosen from among the retrieval queues is the one for which the sum of travel time from the crane's current location to the retrieved pallet and back to the dock is the smallest. In comparison, in schedule 1 (FCFS) in the dual-command cycle, the retrieval pallet is the first retrieval request encountered in the retrieval queues, no matter how far this request is from the crane's current location.

2) For layouts 1 and 2, the difference in performance between schedules 1 and 3 are not statistically significant. However, schedule 3 performance in layout 3 is better than that for schedule 1. For layouts 1 and 2, the restriction of a 30-minute maximum waiting time with a 1 minute interarrival time for retrieval requests exceeds the normal capabilities of the system with respect to throughput. The maximum wait time limit of 30 minutes is set too low in relation to the service rate. Therefore, in the dual-command cycle, when the crane completes a storage assignment, the wait time of at least one request in the retrieval queues is in excess of 30 minutes. The system thus responds as if the FCFS rule were in effect. In the use of dual-command cycle for layout 3,

when the crane completes a storage assignment the wait times of all requests in the retrieval queue are often less than 30 minutes. Thus, the system can be operated under the nearest-neighbor rule whenever the 30-minute wait time limit is not exceeded by any request pallets in the retrieval queue.

3) Schedule 4 (nearest-neighbor/60 minutes maximum wait time) performance was superior to schedule 1 (FCFS) performance for all layouts, but the rate of schedule 4 improvement was lower than that for schedule 2 (nearest-neighbor rule). For all of the layouts, the 60-minute maximum wait time limit allows the system to operate with a mixed scheduling policy, that is, a combination of nearest-neighbor and FCFS. Therefore, schedule 4 performance is superior to that of schedule 1 (FCFS), but lower than that for schedule 2 (nearest-neighbor).

Approach 2: Comparison Across Layouts

Based upon the throughput analysis above, schedule 2 offers the best performance results. The performance of the layouts for schedule 2 is summarized in Figure 4.6. The figure indicates that:

1) The throughput for layout 3 (two-dock) was 35 percent and 45 percent higher than the through

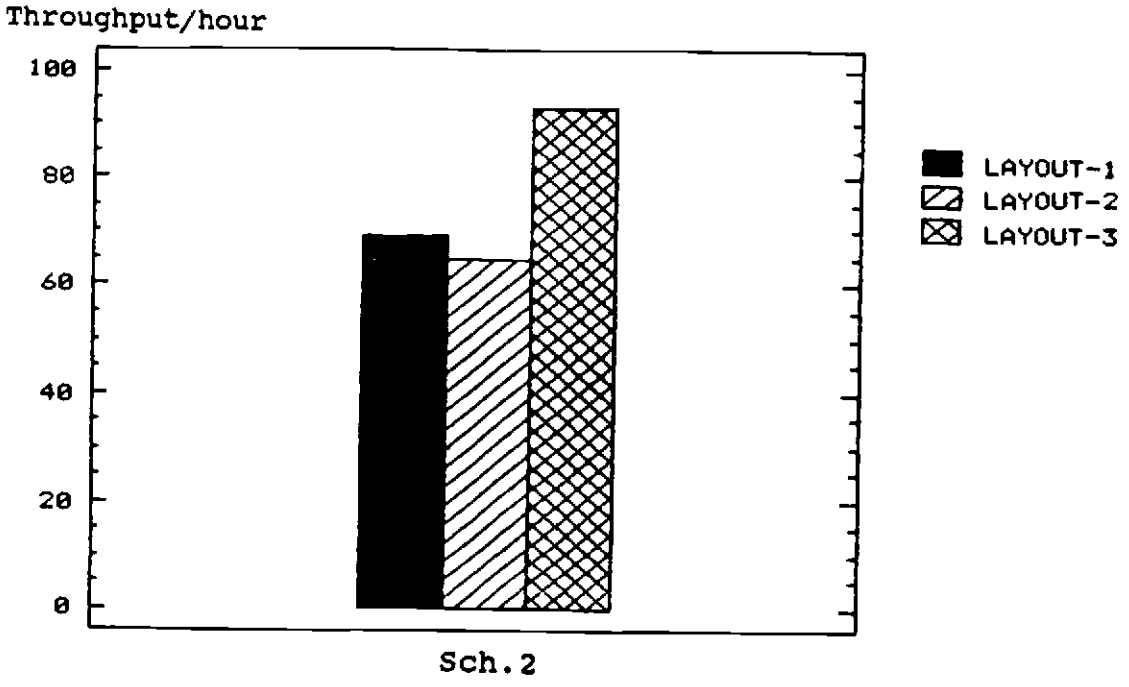


Figure 4.6 Throughput Comparison Across Layouts at 75% Utilization.

puts for layout 1 (one-dock) and layout 2 (two-dock), respectively.

2) The throughput performance for layout 2 (two-dock) was 7 percent less than that for layout 1 (one-dock).

In layouts 1 and 2, the storage pallets from S1 and S2 could be retrieved by either R1 or R2. In layout 3, the S1 storage pallets were retrieved only by R1, while S2 pallets were retrieved only by R2. Based upon identical space utilization (75 percent) and the closest-open-location storage rule, R1 or R2 retrieval requests in layouts 1 and 2 could be selected from any pallet in the storage rack. However, in layout 3, the R1 retrieval requests could be selected from only the one-half of the pallets in the storage rack which were located in the area closest to dock 1; the situation for R2 paralleled that for R1. Thus, the expected crane cycling time in layout 3 must be lower than for layouts 1 and 2. Consequently, the throughput for layout 3 must be higher than for layouts 1 and 2.

The second conclusion points to the importance of the effect of dock structure when the relationship between storage sources and retrieval destinations for one-dock (layout 1) and two-dock (layout 2) layouts are the same. When the space utilization for layouts 1 and 2 is equal (that is, 75 percent), pallets are located in a single area close to the dock for layout 1. How-

ever, for layout 2 the pallets are separated into two groups in the storage rack due to the different locations of the two docks and application of the closest-open-location storage policy: S1 pallets are grouped in the area closest to dock 1 and S2 pallets from are grouped in an area closest to dock 2. For layout 2, if either S1 or S2 storage pallets could be retrieved by either R1 or R2, the crane would occasionally travel the entire length of the storage rack to retrieve a pallet. However, for layout 1, the crane's trip is always restricted to and area within the 75 percent of the storage rack closest to the dock. Therefore, the expected dual-command crane travel time for layout 1 is lower than for layout 2 and the throughput of layout 1 is higher than for layout 2.

Performance Under Different Utilization Levels

Figure 4.4 indicates a fairly consistent trend at the three utilization levels (50, 75, and 90 percent) used in the study: layout 3 with each schedule performs better than the other layout-schedule combinations; and there is very little difference between layouts 1 and 2. However, the relative difference in magnitude of throughput decreases as utilization increases. The reason for this is that when space utilization is increased, the differences in expected crane travel time among layouts are decreased due to the

assumption of random select for retrieval and closest-open-location for storage.

Summary of Throughput Comparisons

1) The throughput improvement for each layout can be obtained by the application of the nearest-neighbor rule;
2) Layout 3 (two-dock) with schedule 2 (nearest-neighbor rule) offers the best combination of layout and scheduling policy for throughput improvement;
3) When space utilization is extremely high (such as 98 percent), there is no significant difference in throughput for layout 1 (one-dock) and layout 2 (two-docks) for any of the scheduling rules used in this study; at lower space utilizations (50 or 75 percent), the throughput for layout 1 exceeds that for layout 2.

Transition Analysis for System Throughput

As shown in Figures 4.4 and 4.5, layout 3 with schedule 2 produces the highest throughput. Further analysis of this combination was undertaken to examine the effects of transitions between docks and throughput.

Table 4.1 is the summary of results obtained from the simulation model for layout 3, schedule 2, showing

Table 4.1 Transition Analysis for Layout 3, Schedule 2.

Utilization (%)	Number of Trans. D-1	D-2	Dock 1 $\bar{X}_1$	S_1	Dock 2 $\bar{X}_2$	S_2	Mean Cycle Time
25	275	274	6.97	3.87	6.50	4.26	0.0108
50	22	23	59.32	49.27	52.37	35.09	0.0163
75	152	152	5.45	6.19	6.26	7.03	0.0219
98	199	199	3.54	3.88	3.68	4.45	0.0279

$\bar{X}_i$ = averge cycles per transition for Dock i.
S_i = standard deviation for average cycles per transition, Dock i.
D-1 = Dock 1.
D-2 = Dock 2.

the number of transitions between docks and the average number of cycles per transition. As space utilization increases, the number of transitions increase; correspondingly, the cycles per transition decrease. The mean cycle time, which increases with space utilization, indicating a decrease in throughput, is also shown in Table 4.1.

Recall that in layout 3, R1 is associated only with S1 and R2 is associated only with S2. Thus, when space utilization is around 50 percent, the utilized areas for the two docks lie close to the docks, are separated from one another, and the crane tends to focus upon only one area at a time. The crane moves to the second area only when the queue for the first area is exhausted, then works only in the second area for an extended period of time. As space utilization increases, these areas of utilization move closer to each other and overlap at very high utilization. Under these conditions transitions increase rapidly. The result is that the crane spends more time moving between docks than in productive time, resulting in loss of throughput efficiency. The decrease in throughput for the 50 to 98 percent utilization range in Table 4.1 is nonlinear with an increase in utilization level, being more rapid between 50 and 75 percent than it is over 75 percent. This pattern is a function of the arrival rate and queue size.

For the 25 percent utilization level, the number of transitions is very high, yet the mean cycle time is low, resulting in high throughput. The arrival rate used in the analysis (1 per minute) is low for the 25 percent level. The throughput is high because the effective areas for each dock are very small. However, the retrieval queues become empty relatively fast due to the low cycle time, resulting in a high transition rate between docks.

For comparison, a similar analysis was conducted for layout 2, schedule 2, as shown in Table 4.2. The trend for transitions and cycles per transition in Table 4.2 are reversed as compared to layout 3, schedule 2. For the layout 2, schedule 2 combination, R1 is associated with both S1 and S2; similarly, R2 is associated with both S1 and S2. Thus, with lower space utilization the utilization areas may be quite removed from each other, but unlike layout 3, as jobs for each dock are distributed between these two areas, the crane continuously moves between the two docks. As utilization increases, the areas tend to overlap, making it easier for the crane to retrieve a job from the "opposite" area which is row contiguous in nature. The net effect is a decrease in the number of transitions and an increase in the number of cycles per transition.

The results for the mean cycle time for layout 2, schedule 2 in Table 4.2 are interesting. They indicate

Table 4.2 Transition Analysis for Layout 2, Schedule 2.

Utilization (%)	Number of Trans. D-1	D-2	Dock 1 $\bar{X}_1$	S_1	Dock 2 $\bar{X}_2$	S_2	Mean Cycle Time
25	566	566	1.70	1.21	1.53	1.25	0.0219
50	387	387	1.87	1.67	2.11	1.92	0.0259
75	256	256	2.43	2.44	2.60	2.39	0.0310
98	165	165	3.16	2.20	3.61	3.12	0.0358

$\bar{X}_i$ = averge cycles per transition for Dock i.
S_i = standard deviation for average cycles per transition, Dock i.
D-1 = Dock 1.
D-2 = Dock 2.

that despite a decrease in the number of transitions, throughput decreases. The reason for this is that the dual command crane distance traveled from any given dock is high due to the distribution of jobs in both areas. Thus, any potential savings resulting from cutting down the number of transitions between docks is offset by the increased distance the crane has to travel in the dual command cycle.

Mean Waiting Time Analysis

Storage pallet waiting time is the time interval between the arrival of a pallet at a storage queue and the deposit of that pallet at a storage location; retrieval request waiting time is the time interval between the arrival of the request at a retrieval queue and the retrieval of the requested pallet to the dock. An objective of this study was to evaluate the effects of combinations of layouts and scheduling policies on mean storage waiting times and maximum waiting times.

The waiting times obtained from the simulation analysis at 75 percent space utilization and the associated ANOVA results are presented in Appendices 3, 4 and 5. A high degree of interaction between the two factors A (layouts) and B (schedules) led to the waiting time comparisons discussed in the following sections. The layouts and schedules combinations were compared using the same approach as for throughput

analysis: first, comparison across schedules, and second, comparison across layouts under the best schedule rule for each layout.

Approach 1: Comparison Across Schedules

The results for the mean storage waiting times and mean retrieval waiting times are shown in Figures 4.7 and 4.8, respectively. For layout 1, the mean storage waiting time performance of schedule 2 was superior to that for schedules 1, 3, and 4. Schedule 2 offered reductions in the mean storage waiting time of 79, 77, and 16 percent, and reductions in mean retrieval waiting time of 83, 82, and 18 percent, respectively, in comparison to comparable measurements for schedules 1, 3, and 4.

For layout 2, the results shown in Figures 4.7 and 4.8 indicated that the mean waiting time performance of schedule 2 was again superior to those of schedules 1, 3, and 4, resulting in reduction of mean storage waiting time of 77, 28, and 2 percent and reductions of mean retrieval waiting time of 86, 42, and 4 percent, respectively, when compared to comparable measurements for schedules 1, 3, and 4.

Based on the above comparisons, the nearest-neighbor rule (schedule 2) provided the lowest mean waiting time for layouts 1 and 2. Just as was the case

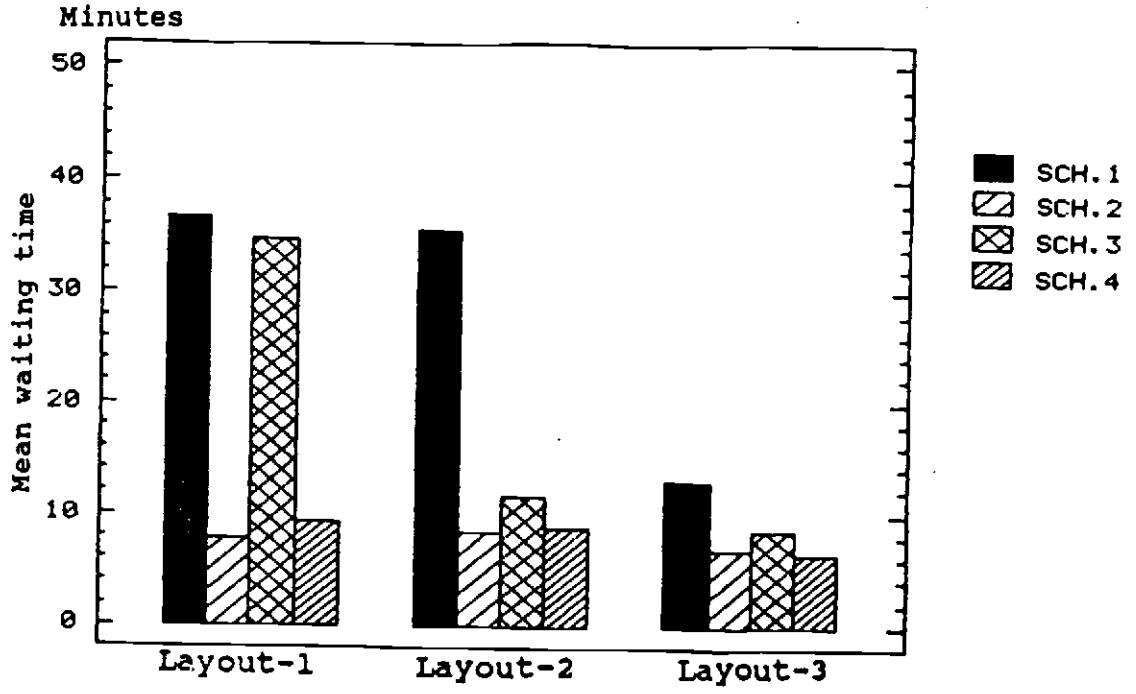


Figure 4.7 Mean Waiting Time, Comparison Across Schedules (Storage).

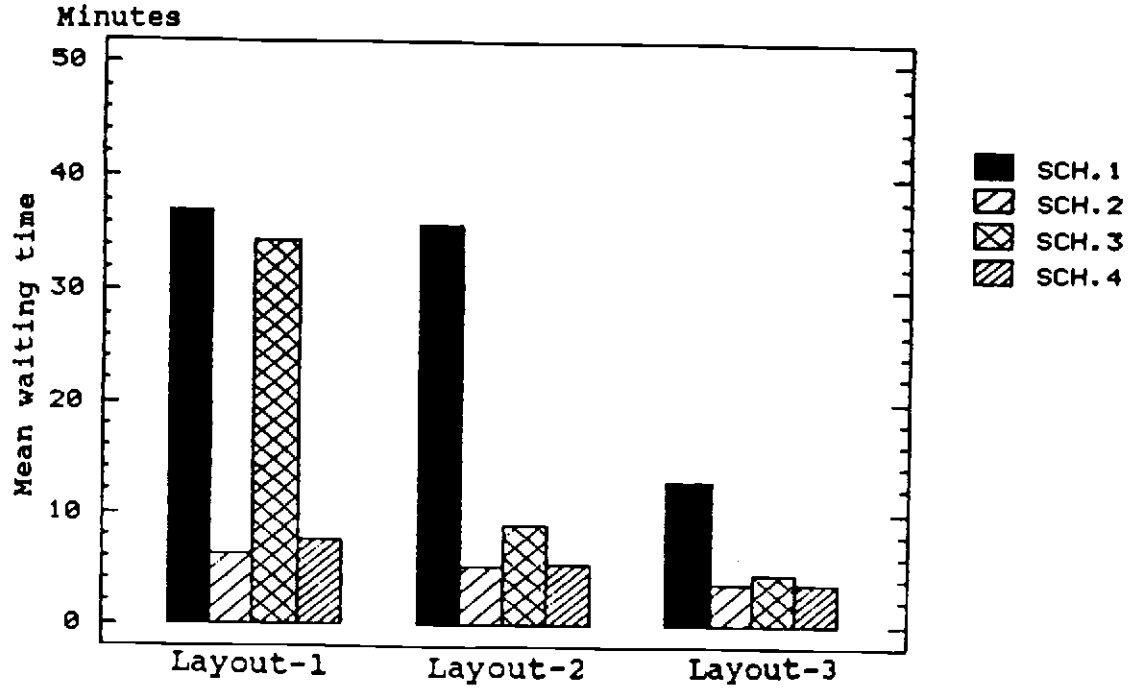


Figure 4.8 Mean Waiting Time, Comparison Across Schedules (Retrieval).

for throughput analysis, schedule 2 is superior to the other schedules with respect to mean waiting times (that is, both storage and retrieval).

For layout 3, the performance of schedule 2 is still better than schedules 1 and 3, but is slightly lower than that of schedule 4. The performance differences between schedules 2 and 4 in layout 3 can be explained as follows. With the closest-open-location storage policy, the S1 pallets are stored in the areas closest to dock 1, while S2 pallets are stored in areas closest to dock 2. In the dual-command crane cycle operating under the nearest-neighbor rule (schedule 2), if the crane travel for S1 is from dock 1, then following the deposit of S1 at the location closest to dock 1 (closest-open-location storage policy) the nearest-neighbor retrieval target for the crane will be chosen from the S1 group for R1. Therefore, the crane retrieves an S1 pallet for R1, then returns to dock 1. This behavior will be repeated until all the R1 queue has been retrieved, while all of the dock 2 storage pallets and retrieval requests are waiting at the dock. Only then can the crane move to dock 2 and repeat the pattern already established for dock 1. Thus, for layout 3, when a storage pallet or retrieval request arrives at a dock, storage or retrieval from that dock is delayed if the crane is currently occupied with orders from the opposite dock. However, for schedule 4 with a

60-minute maximum wait time, the maximum waiting time acts as a limit on the time the crane spends at a dock, resulting in a shorter waiting time at the "opposite" dock.

Approach 2: Comparison Across Layouts

The comparison results obtained for Approach 1 indicate that schedule 2 is the best policy for layouts 1 and 2 for minimizing mean waiting time, while schedule 4 is the best policy for minimizing waiting time for layout 3. Comparisons across layouts for the mean waiting times are shown in Figures 4.9 and 4.10. Based on the results shown in these figures, the performance of layout 3 with schedule 4 is the best combination with respect to mean waiting time. Since the throughput for layout 3 was higher as compared to layouts 1 and 2, the reduced mean waiting time of layout 3 as compared to layouts 1 and 2 was to be expected.

Comparisons of layouts 1 and 2 for schedule 2 (nearest-neighbor), on which these layouts perform best, indicates that the mean storage waiting time for layout 2 was 9 percent higher than for layout 1, while for mean retrieval waiting time, layout 2 was 16 percent lower than layout 1. Given that the throughput for layout 2 was lower than for layout 1, the higher mean storage waiting time of layout 2 was again an expected result. For retrievals, the one-dock layout

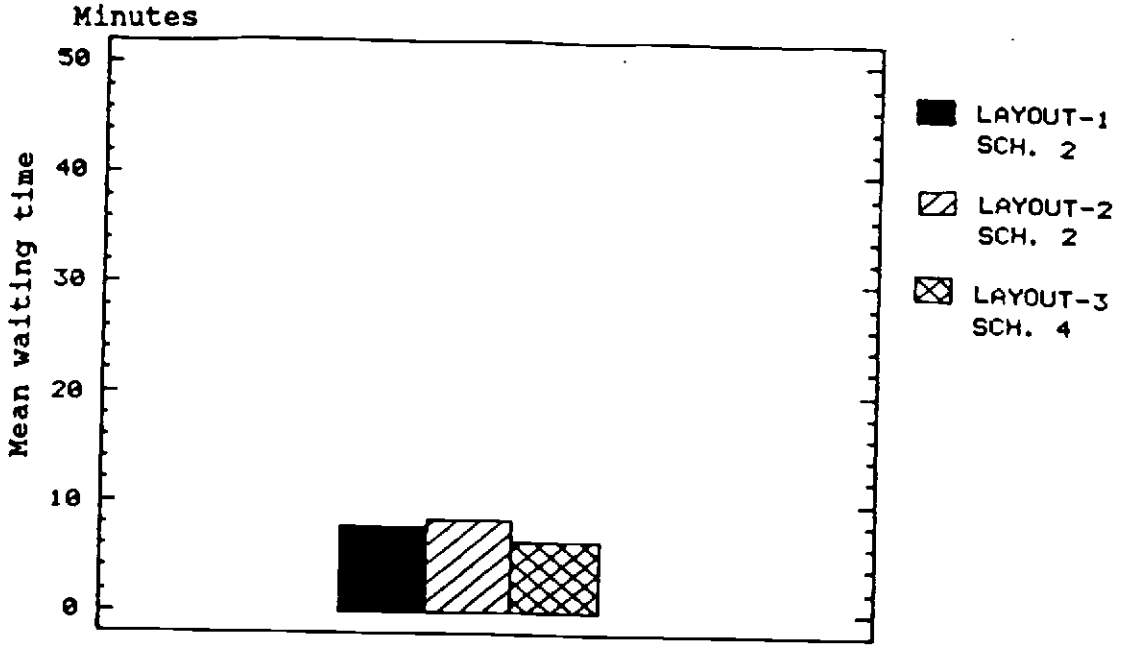


Figure 4.9 Mean Waiting Time, Comparison Across Layouts (Storage).

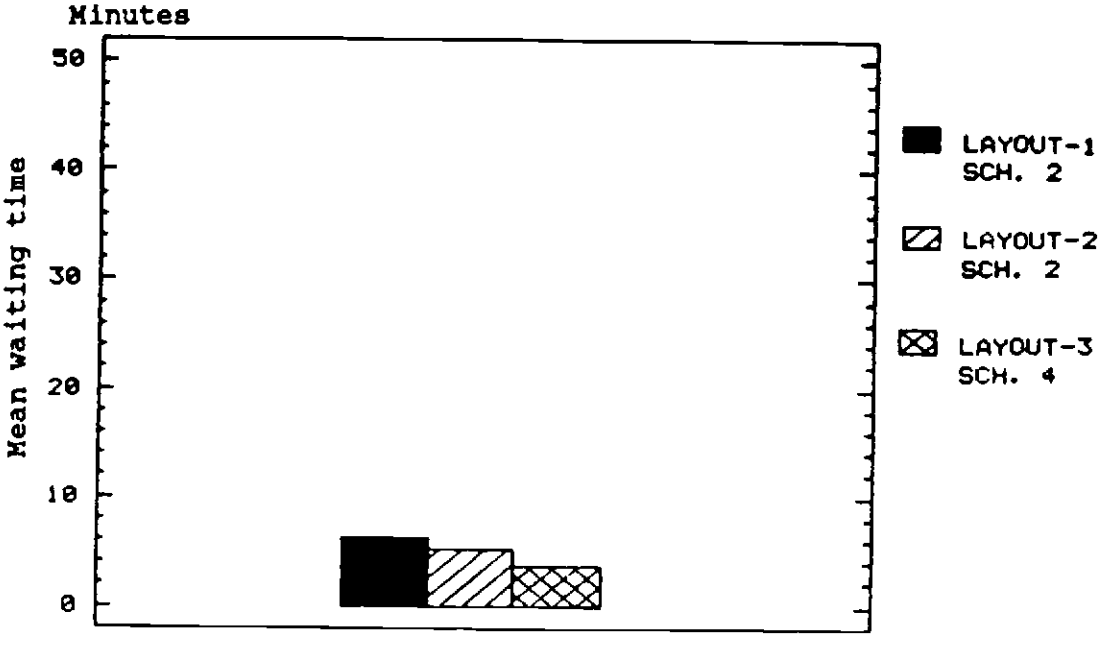


Figure 4.10 Mean Waiting Time, Comparison Across Layouts (Retrieval).

(layout 1) with schedule 2 (nearest-neighbor) could be subjected to excessive delays. Given the closest-open-location storage policy and nearest-neighbor storage retrieval rule, if the requested pallet is located at the far end of the storage rack it will wait in the retrieval queue until all other requests have been satisfied. In comparison, for the two-dock layout (layout 2), the crane travels between the two docks and R1 or R2 requests are made from either S1 or S2 sources; thus, there is a good probability that a pallet at a distant location will become the nearest-neighbor. Thus, the maximum retrieval waiting time for the two-dock layout is lower than for the one-dock layout, a hypothesis which is further confirmed in comparing maximum waiting times (as discussed in the next section). The lower maximum retrieval waiting for layout 2 causes the mean retrieval waiting time for layout 2 to be lower than for layout 1.

Summary of Mean Waiting Time Comparisons

1) Mean waiting time improvement for layouts 1 and 2 can be obtained by the application of nearest-neighbor rule;
2) Mean waiting time improvement for layout 3 can be obtained by the application of the nearest-neighbor rule with an adequate maximum waiting time limit;

3) Mean waiting time for layout 3 with schedule 4 (nearest-neighbor/60 minutes maximum wait time) is lower than layouts 1 and 2 with schedule 2 (nearest-neighbor rule); and
4) When using the nearest-neighbor rule with layouts 1 and 2, the storage and retrieval mean waiting times are such that the mean storage waiting time of layout 1 is lower than layout 2, while the mean retrieval waiting time of layout 1 is higher than layout 2.

Maximum Waiting Time Analysis

Due to the nature of the nearest-neighbor rule, retrieval requests could be excessively delayed if the requested pallet is located at the far end of the storage rack. The objective of maximum waiting time analysis is to concentrate on retrievals in order to analyze this problem.

Approach 1: Comparison Across Schedules

The maximum waiting time comparisons across schedules for each layout are shown in Figure 4.11. Figure 4.11 shows that for all three layouts the worst results are obtained for the nearest-neighbor scheduling rule with no time limit (schedule 2). This is because the requested pallet may be in a distant location in the storage rack, and can only be retrieved if the

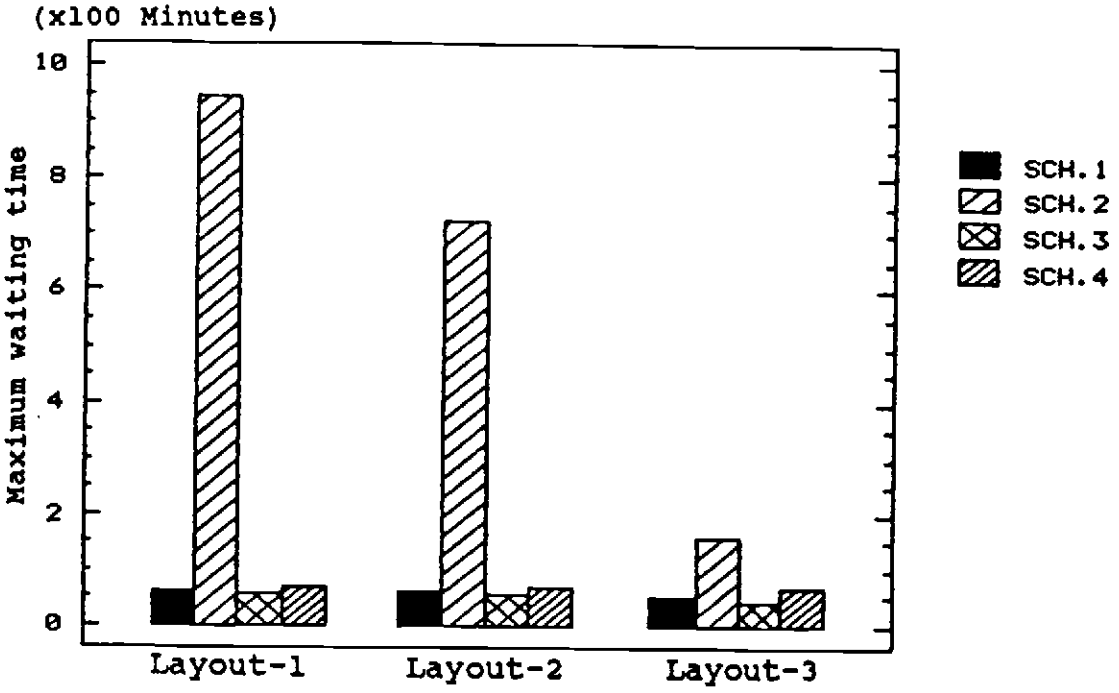


Figure 4.11 Maximum Waiting Time Comparison Across Schedules (Retrieval).

retrieval queue(s) are empty. Under this schedule, the maximum waiting time for layout 3 (dedicted storage-retrieval combination) is the smallest as the storage rack is split equally between the two docks and the maximum distance a unit can be stored is considerably less for the one-dock layout (layout 1) or the two-dock layout (layout 2) with mixed arrangement.

The use of schedule 4 (nearest-neighbor/60-minute maximum wait time limit) improved the performance considerably for all three layouts. However, the best results are obtained with schedule 3, which is due to the combination of the nearest-neighbor rule with a 30-minute maximum wait.

The problem of determining the "best" combination of layout and schedule is a multicriteria problem, with conflicts between the multiple criteria. Thus, schedule 2 results in maximum throughput for layouts, but at the expense of maximum waiting time. On the other hand, schedule 3 minimizes maximum waiting time, but also results in substantial reductions in system throughput.

Approach 2: Comparison Across Layouts

The results from Approach 1 indicate that schedule 3 (nearest-neighbor/30 minutes maximum wait time limit) minimizes the maximum waiting time for each layout. A comparison of the layouts under this scheduling policy

is shown in Figure 4.12 The maximum waiting times for layout 3 was 26 percent and 25 percent lower than those for layouts 1 and layout 2, respectively. The waiting time statistics for layouts 1 and 2 were approximately equal. As shown in Figure 4.5, for schedule 3, the throughput for layout 3 was higher than the throughput for layouts 1 and 2, while the throughput for layout 1 was approximately equal to that of layout 2. These results are consistent with the reasoning outlined in the previous section.

Summary of Maximum Waiting Time Comparisons

1) For each layout, the maximum waiting time is minimized by the use of the nearest-neighbor rule with a waiting time limit;
2) When such a policy is used, the maximum waiting time for layout 3 is lower than for layouts 1 and 2, while there is no significant difference between layouts 1 and 2; and
3) For each layout, the retrieval requests can be excessively delayed when using nearest-neighbor rule without a waiting time limit;

Summary of Results

The objective of this research was to analyze the effect of different scheduling priorities on three layouts. This chapter presented results for different

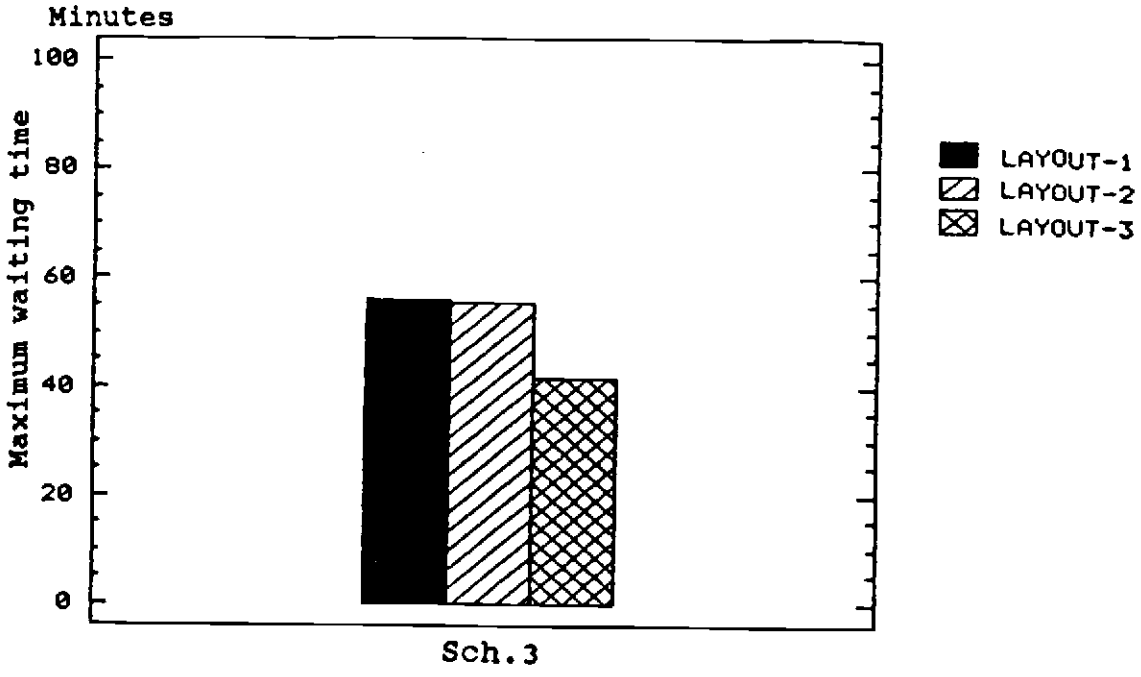


Figure 4.12 Maximum Waiting Time Comparison Across Layouts (Retrieval).

layout-schedule combinations obtained from simulation studies. The results were discussed on four performance measures: throughput, mean storage waiting times, mean retrieval waiting times, and maximum retrieval waiting times. These results are summarized in Tables 4.3 through 4.6. The tables show the best schedule for each layout under different performance measures. Also shown is the relative performance of each layout using layout 1 as the base layout.

A higher value of the relative performance is preferable for throughput, whereas lower values for the relative performance indicate better waiting times performance. As can be seen from these tables, no one combination results in a global optima; the optimum is a function of the performance measure used.

Table 4.3 Throughput Analysis (basis = layout 1).

Layouts	Best Schedule	Relative Performance
Layout 1	Sch. 2	1.00
Layout 2	Sch. 2	0.93
Layout 3	Sch. 2	1.35

Table 4.4 Mean Storage Waiting Times (basis = layout 1).

Layouts	Best Schedule	Relative Performance
Layout 1	Sch. 2	1.00
Layout 2	Sch. 2	1.09
Layout 3	Sch. 4	0.84

Table 4.5 Mean Retrieval Waiting Times (basis = layout 1).

Layouts	Best Schedule	Relative Performance
Layout 1	Sch. 2	1.00
Layout 2	Sch. 2	0.84
Layout 3	Sch. 4	0.60

Table 4.6 Maximum Retrieval Waiting Times (basis = layout 1).

Layouts	Best Schedule	Relative Performance
Layout 1	Sch. 3	1.00
Layout 2	Sch. 3	0.99
Layout 3	Sch. 3	0.74

V. CONCLUSIONS

General Conclusions

The primary objective of this research was to evaluate the effect of multiple dock placement in an AS/R system. The results obtained from this investigation show that the efficiency of the AS/R system can be improved by the introduction of two docks in each aisle when the input pallets are stored in the closest-open-location, and the input/output pallets for the two docks are independent of each other (that is, the output pallets processed through each dock consist only of the pallets which were input through that dock). A comparison of this type of two-dock layout (layout 3) with a one-dock (layout 1) showed that:

1) The two-dock layout with the nearest-neighbor rule resulted in a 35% throughput improvement over the one dock layout;
2) The two-dock layout with the nearest-neighbor rule and a 60-minute maximum waiting time limit resulted in 16% and 40% decreases in mean storage waiting time and mean retrieval waiting time over the one-dock layout; and

3) The two-dock layout operated with the nearest-neighbor rule and a 30-minute waiting time limit resulted in a 26% retrieval waiting time decrease over the one-dock layout.

Based on the results obtained from this investigation, each performance measure reflects an independent optimum solution for each layout, or combination of layout and scheduling policies. The use of one of these independent solutions can serve to optimize selected AS/R configurations with regard to the given performance measures, but at the same time other performance measures may be affected negatively. These tradeoff effects are summarized in Tables 5.1 through 5.3, which are based on 75% space utilization of the AS/R system. For example, Table 5.1 shows the performance measure tradeoffs for layout 1. Maximum throughput is achieved using schedule 2; this also results in optimizing the mean storage and the mean retrieval waiting times, but the maximum retrieval time is increased by almost 1,600 percent over the optimum that is obtained with schedule 3. Similarly, the maximum retrieval time is minimized with schedule 3, but throughput is reduced by 21 percent from its optimum value, and the mean storage and mean retrieval times are increased by 349 and 456 percent over their optimum values, respectively.

Table 5.1 Tradeoffs Among Performance Measures, Layout 1 (pallets from S1/S2 retrieved by R1/R2).

		Effect Upon:			
Objective	Best Schedule	Throughput	Mean Storage Wait Time	Mean Retrieve Wait Time	Maximum Retrieve Wait Time
Max Thruput	Sch. 2	optimum	optimum	optimum	+1589%
Mean Storage WT	Sch. 2	optimum	optimum	optimum	+1589%
Mean Retrieve WT	Sch. 2	optimum	optimum	optimum	+1589%
Max Retrieve WT	Sch. 3	-21%	+349%	+456%	optimum

Table 5.2 Tradeoffs Among Performance Measures, Layout 2 (pallets from S1/S2 retrieved by R1/R2).

		Effect Upon:			
Objective	Best Schedule	Throughput	Mean Storage Wait Time	Mean Retrieve Wait Time	Maximum Retrieve Wait Time
Max Thruput	Sch. 2	optimum	optimum	optimum	+1209%
Mean Storage WT	Sch. 2	optimum	optimum	optimum	+1209%
Mean Retrieve WT	Sch. 2	optimum	optimum	optimum	+1200%
Max Retrieve WT	Sch. 3	-16%	+38%	+73%	optimum

Table 5.3 Tradeoffs Among Performance Measures, Layout 3 (pallets from S1 retrieved by R1; pallets from S2 retrieved by R2).

Objective	Best Schedule	Effect Upon: Throughput	Mean Storage Wait Time	Mean Retrieve Wait Time	Maximum Retrieve Wait Time
Max Thruput	Sch. 2	optimum	+6%	optimum	+283%
Mean Storage WT	Sch. 4	-13%	optimum	optimum	+62%
Mean Retrieve WT	Sch. 4	-13%	optimum	optimum	+62%
Max Retrieve WT	Sch. 3	-26%	+31%	+19%	optimum

Recommendations for Further Research

There are a number of directions in which this research can be extended. Some of these are briefly outlined below.

1. Some of the assumptions used in this study can be relaxed, and additional factors may be considered for the two-dock layout. Examples of such factors include the crane capacity per trip, pallet turnover frequency and pallet storage location policies. In addition, situations where one crane is shared by two or multiple aisles may be of interest in certain environments.

2. The storage location assignment policies, other than those analyzed in this research, may be con-

sidered in the two-dock case. Graves and Hausman (1977) have studied the following three rules for the one-dock layout:

a) The random storage assignment rules, in which all the open locations have an equal chance to be selected as a storage location for pallets.

b) The class-based storage location assignment, in which storage pallets and storage locations are partitioned into a small numbers of classes (for example, 2 or 3), pairing the highest turnover class of pallets with the class of rack locations closest to the I/O (dock) point.

c) The full turnover-based storage location assignment, in which the pallet with the highest turnover frequency is assigned to the location closest to I/O (dock) point.

The results from Graves and Hausman (1977) indicated that rules (2) and (3) were capable of much higher throughput rates than rule (1). Extension of these rules to the AS/R system with dual-command crane travel would be interesting.

3. Another interesting area suggested for further study would be the effect of differentiated job assignments for each of the docks. When each dock has an independent source for both storage pallets and retrieval requests, given that a storage or retrieval queue is empty at one of the docks, the crane will not operate

on a 100% dual-command cycle or even transfer empty loads between the docks. Conversely, if the storage pallets and retrieval requests from all sources and destinations are given equal access to each of the two docks, the percentage of dual-command crane cycle operation can be substantially increased, resulting in the improvement of throughput.

4. The possibility of selecting open locations for input pallets and the nearest-neighbor in retrieval queues by the use of artificial intelligence should be considered. In ideal terms, this approach would include the means to predict future storage and retrieval requirements. Furthermore, using a look-ahead storage and retrieval policy in which a pallet in a storage queue could be issued a retrieval request from a retrieval queue at the time it received its storage assignment would result in that pallet bypassing the dock and directly accessding the destination.

5. Finally, some of the tools and techniques in decision analysis can be utilized to develop tradeoff functions for the conflicting set of objectives in the decision environment. Development of such tools may form the basis of decision support systems for selection and implementation of AS/R systems.

BIBLIOGRAPHY

Bozer, Yavuz A., & White, John A. "Traveling-Time Models For Automated Storage/Retrieval System." *AIIE Transactions*, December, 1984.

Francis, Richard L., & White, John A. *Facility Lay-Out and Location*, Englewood Cliffs, NJ: Prentice-Hall Inc., 1974.

Graves, Stephen C., & Hausman, Warren H. "Storage-Retrieval Interleaving in Automatic Warehousing Systems." *Management Science*, *23*(9), May, 1977.

Graves, Stephen C., & Hausman, Warren H. "Scheduling Policy for Automatic Warehousing Systems: Simulation Results." *AIIE Transaction*, *10*(3), September, 1978.

Han, Min-Hong, & McGinnis, Leon F. "On Sequencing Retrievals in an Automated Storage/Retrieval System." *AIIE Transactions*, March, 1987.

Handbook of Industrial Engineering. New York: John Wiley & Sons, Inc., 1982.

Hausman, Warren H., & Shwarz Leroy B. "Optimal Storage Assignment in Automatic Warehousing Systems." *Management Science*, *22*(6), February, 1976.

Hulett, Malcolm. *Unit Load Handling*. London: Gower Press Limited, 1970.

Khan, Mohammad Younas. *Increased Throughput by Aisle Zoning in End-of-Aisle Order Picking AS/R Systems*. Unpublished master's thesis, Oregon State University, Corvallis, OR, April, 1982.

White, David A. *Personal Computer Character Graphic Modeling System*. San Jose, CA, 1988.

APPENDICES

APPENDIX A

PCMODEL FUNCTIONS

The important functions of *PCModel* which have been included in *DOCK* include the following:

1) Overlay file. The primary purpose of an overlay is to visualize the model in order to examine activities as they are simulated. By constructing an overlay that is a reasonable approximation of the real system, the user will acquire a sense of the simulation dynamics as they relate to the real environment. The maximum overlay dimensions are 1,365 columns by 409 rows, with 4 background and 16 foreground colors.

2) *Username*.MDL file. This file contains the application routing definitions, the maximum work-in-process (WIP) value, symbolic values, job information, screen location definitions for statistics collection, and a screen of application descriptive text.

3) Routing. In *DOCK*, the path an object is to follow is called a "Rout." A routing defines routes for a group of objects following the

same path. Therefore, routing consists of a sequence of movement instructions used to define the various paths that objects of each job are to follow. The program *DOCK* contains five routings.

4) Link. "Link" is equivalent to subroutines in other computer programing languages. In *PCMOdel*, links are executed by several branching instructions and can be used for multiple routings.

5) Object system parameters:

   a. OBJ@ID is the object's ID character, which must be in the range 1 to 225;

   b. OBJ@SN is the object's serial number, which start at 1 and are incremented with each new release into the model; and

   c. OBJ%ST is set to the value of the system clock upon entry of the object into the model.

6) User Symbols. Users can use four types of symbols to store values: (1) constant value, "#", (2) 16-bit variable value, "@", (3) 32-bit clock value, "%", and (4) screen location offsets value, "*". Examples of these four types of values include:

a. #SEED1=(3578) has a constant value of 3578; this value cannot be changed during the simulation.

b. @WIP=(10) is set to a default of 10, but its value may be modified during simulation.

c. %ARRIV=(00:15:30) is a clock variable, set to 15 min/30 sec; it may be modified during simulation.

d. DOCK1=(XY(35,45)) designates that the location variable, *DOCK, refers to the screen locations, X=35 and Y=45, during the simulation.

7) Array. There are two types of two-dimensional array variables: (1) A 16-bit clock value starting with the set of character "@@" and (2) a 32-bit clock value starting with the set of character "%%". The largest index value or product of both indices is 32,735 for @@ arrays and 16,367 for %% arrays.

8) Posting and Clearing a Location Offset. This function may be used to post a screen location which is occupied by an object or to clear a screen location. It is useful for reserving a path, tool, or process. The posted screen location does not have to be in the path of

travel. Therefore, these functions can also be used as logic control flags. For example,

a. *CRANE=(XY(85,35)) defines a flag location for the crane;

b. JC(*CRANE,:DOCK1) signifies that if *CRANE is not posted, jump to the label :DOCK1; and

c. PO(*CRANE) signifies post the flag *CRANE when the crane is busy.

Program Structure

The simulation program, *DOCK*, consists of five main components:

1) System configuration and initialization.
2) Generating arrays for storage location and retrieval request references (Routing 1).
3) Path of storage and retrieval pallets (Routings 2, 3, 4, and 5).
4) Crane operations (Routing 6).
5) Links.

System Configuration and Initialization

The system configuration of the program includes the following specifications:

Overlay File Designation

The dimensions of the logical screen for *DOCK* are 80 columns by 22 rows. The logical screen overlay is stored in an overlay file, DOCK.OLY, which is called from the program file during the loading process. The dimensions of the storage rack are 40 columns by 10 rows, in which the crane speeds are 2 seconds/column and 10 seconds/row.

Job Descriptions

Job descriptions specify the manner in which groups of objects following the same rout are to be released during the simulation. The job description consists of seven parameter specifications with the following format:

J = (job,id,rout,cond,flag,priority,size)

Detailed definitions for each parameters can be found in the *PCModel User's Manual*. In the program *DOCK*, there are six job descriptions:

1) J=(1,%,1,0,0,0,1) for generation of storage location reference and initial condition of the storage rack;
2) J=(2,C,1,0,0,1,1) for crane operation;
3) J=(3,S,1,0,0,1,x) for storage of S1 pallets;
4) J=(4,P,1,0,0,1,x) for storage of S2 pallets;

5) J=(5,R,1,0,0,1,x) for retrieval requests R1; and

6) J=(6,G,1,0,0,1,x) for retrieval requests R2.

Location Offsets and Logic Control Flags

Two different location offsets are used in the program. The first is the location on the path of the reference for object movements. The three most important path location offsets are:

1) *DOCK1=(XY(19,17)) for dock1 location;

2) *DOCK2=(XY(61,17)) for dock2 location; and

3) *DWELL=(XY(39,17)) for the crane dwell point in the two-dock layout.

The second location offset is located outside the path as a flag for logic control indicators, e.g., *CRANE=(XY(1,1)) designates the busy or idle condition of the crane.

Definitions of Numeric Variables

The most important numeric variables include the following:

Name	Definition
@S1)	WIP of storage S1,
@S2)	WIP of storage S2,
@R1)	WIP of retrieval R1,
@R2)	WIP of retrieval R2,
@X)	current crane location (X-axis),
@Y)	current crane location (Y-axis) ,

@DX) destination of crane move (X axis), and

@DY) destination of crane move (Y-axis).

Random Number Seeds

Five random seeds are used in the program *DOCK*:

Seeds	Purposes
#SEEDS1)	interarrival time S1,
#SEEDS2)	interarrival time S2,
#SEEDR1)	interarrival time R1,
#SEEDR2)	interarrival time R2,
#SEEDR)	random select pallet for retrieval R1 and R2, and
#SEED)	others.

Generating Arrays for Storage Location and Retrieval Request References

For storage, it is necessary to create and store the utilization priority for each storage location in arrays. Based on the closest-open-location definitions for the three layouts, there are five arrays, each of which contain storage location utilization priorities for each layout. These arrays include:

@@CELL1) for layout 1,

@@CELL2) for layout 2, from dock1,

@@CELL3) for layout 2, from dock2,

@@CELL4) for layout 3, from dock1, and

@@CELL5) for layout 3, from dock2.

Each array contains 100 cells (20 columns × 5 rows), each of which represents a storage location in which the column and row number correspond to the indices of the cell. The value in each cell represents the utilization priority for that storage location with the lower value given the highest priority. Before the crane initiates a storage trip, the proper array is scanned to find the cell with the lowest value. The indices of this cell are then used as the column and row numbers of the closest-open-location in the storage rack. Conversely, once a storage location has been occupied, a larger value (999) will be stored into the corresponding indices of the cell. For the FCFS and nearest-neighbor retrieval discipline, 10 arrays are created and defined in this section of *DOCK*.

The following are the five arrays for the selection of retrieval pallets:

@@RETRV1) for layout 1,

@@RETRV2) for layout 2, from dock1,

@@RETRV3) for layout 2, from dock2,

@@RETRV4) for layout 3, from dock1, and

@@RETRV5) for layout 3, from dock2.

The information stored in these arrays is identical to the information stored in array @@CELL1-5. Once a retrieval request arrives, based on the layout in use, a cell with the value 999 will be randomly selected from @@RETRV array. The value in the selected cell will be

replaced by 91 or 92, depending on which dock of origin of the retrieval request.

The following are five clock value arrays for arrival time information for retrieval requests:

%%RETRV1) for layout 1,

%%RETRV2) for layout 2, from dock1,

%%RETRV3) for layout 2, from dock2,

%%RETRV4) for layout 3, from dock1, and

%%RETRV5) for layout 3, from dock2.

Once a pallet has been selected by a retrieval request, the arrival time of that request will be stored in the %%RETRV array cell. This information will be used to schedule the crane operations for different retrieval scheduling rules, e.g., FCFS or max. waiting time.

The relationships among arrays @@CELL, @@RETRV and %%RETRV are shown in Appendix 1.1

Path for Arrival and Retrieval Pallets

For the two sources of storage pallets, S1 and S2, and the two retrieval pallet destinations, R1 and R2, Routing 2 and 3 generate the interarrival time for arriving pallets and provide the path for S1 and S2 from their sources to the dock, while Routing 4 and 5 generate the interarrival time for pallets to be retrieved and provide the path for R1 and R2 from the dock to their destinations. In the one-dock layout, the terminus' of the S1 and S2 paths are merged at the

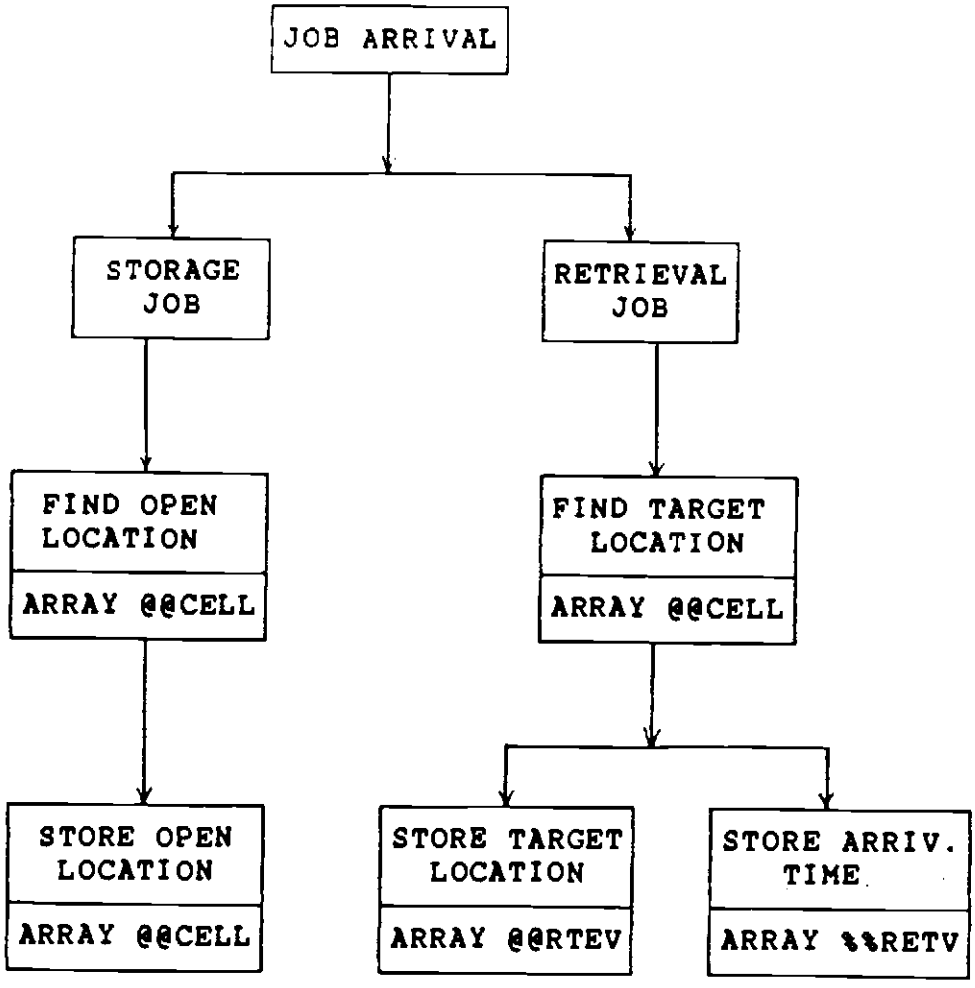


Figure A.1 Relationship of Array @@CELL, @@RETRV and %%RETRV.

dock and the origins of the R1 and R2 paths are separated at the dock. In the two-dock layout, the terminus of the S1 path is dock1 and the terminus of the S2 path is dock2, while the origin of the R1 path is for dock1 and the origin of the R2 path is for dock2.

Crane Movement Control

Routing 6 provides the operational logic, conceptualized in Appendix 1.2, for the crane movements. Crane movement is initiated from its current location, the coordinates of which are stored in the variables @X and @Y. The movement terminates at the destination, with the destination coordinates stored in the variables @DX, and @DY. Therefore, the movement distance in the X-axis is @X-@DX, denoted by @MX, and in the Y-axis is @Y-@DY, denoted by @MY. Furthermore, movement directions are controlled by the variables @DIC and @DIR. An example of crane movement control is shown in Fig A.2. The relationship among location variables, distance variables, and direction variables are as follows:

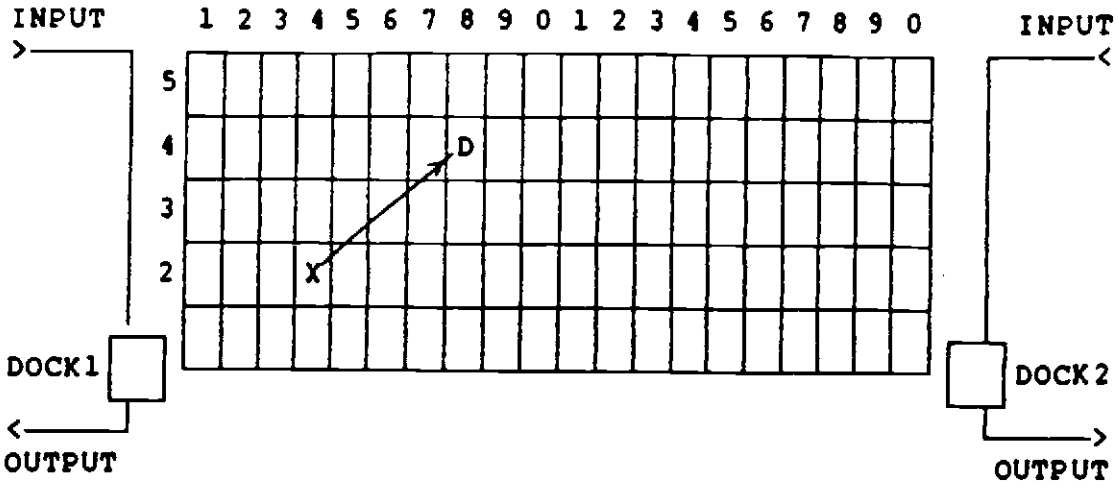
Movement Distance Control

Current Location - Destination = Distance, or

@X - @DX = @MX

and

@Y - @DY = @MY.



CURRENT LOCATION	DESTINATION	DISTANCE	DIRECTION
@X = 4	@DX = 8	@DX - @X =4 >0	--->@DIC=1 MOVE-RIGHT
@Y = 2	@DY = 4	@DY - @Y =2 >0	--->@DIR=1 MOVE-UP

Figure A.2 Crane Movement Control.

Movement Direction Control

Distance ⇒ Direction ⇒ Move Direction, or

@MX > 0 ⇒ @DIC = 1 ⇒ Move-up,

@MX < 0 ⇒ @DIC = 2 ⇒ Move-down,

@MX = 0 ⇒ @DIC = 0 ⇒ No-move

and

@MY > 0 ⇒ @DIR = 1 ⇒ Move-left,

@MY < 0 ⇒ @DIR = 2 ⇒ Move-right,

@MY = 0 ⇒ @DIR = 0 ⇒ No-move.

The values of these variables are calculated by LK!GOTO (a sub-routine), then executed by Routing 6 for crane movement control.

Links

Links are basically used to avoid the replication of the same program logic within the program, a function equivalent to the use of subrouting in *FORTRAN*. In *PCModel*, a link is defined by BL(!*linkname*), which ends with the suffix EL (i.e., BL implies Begin Link and EL implies End Link). Within routings, links are called by LK(!*linkname*). In the program *DOCK* there are seven defined links, which are described as follows:

1) LK(!FIND-S). This link provides the function of closest-open-location selection. Before the crane initiates a storage trip, LK(!FIND-S) scans the array @@CELL to determine the

closest-open-location for the storage pallet. The indices of the closest-open-location are then transferred into the variables @DX and @DY as the destination coordinates for crane movement.

2) LK(!FIND-R). When a retrieval request arrives at the retrieval queue, a pallet is randomly selected from the array @@RETRV. The arrival time of the retrieval request is then stored in the array %%RETRV.

3) LK(!SET-R). Prior to the initiation of a crane retrieval trip, the nearest-neighbor or FCFS retrieval request are selected from the arrays @@RETRV or %%RETRV, and the indices of the target pallet are transferred into destination variables @DX and @DY.

4) LK(!GOTO). The distance and direction of the crane movement are calculated by this link. The calculated distance is then stored in the variables @MX and @MY and the calculated direction is stored in the variables @DIC and @DIR.

5) LK(!SIMU). Since the crane is capable of simultaneous horizontal and vertical movements, this link calculates the traveling time of the crane movement from the variables @X and @Y to the variables @DX and @DY. This link is also

called by LK(!SET-R) for the calculation of the nearest-neighbor.

6) LK(!MOVE). Following the direction variables (@DIC, @DIR) and distance variables (@MX, @MY), this link moves the crane from the current location (@X, @Y) to the destination location (@DX, @DY). Following this movement, in order to update the current crane location, the values of the current location variables (@X, @Y) are replaced by the values of the destination variables (@DX, @DY).

7) LK(!MESAGE). For observation of program execution, messages must be displayed when a storage location has been occupied or a pallet in the rack has been posted for retrieval; this link provides for these functions.

APPENDIX B

ANOVA OF THROUGHPUT

Simulation results: Throughput
Arrival rate: 1 minutes

Throughput		Factor A (Layouts)			
		Layout-1	Layout-2	Layout-3	$\bar{B}$
Factor B(Scheds)	Sch-1	54.71,54.15	55.11,53.76	65.30,65.08	58.02
	Sch-2	69.16,68.64	64.42,64.84	93.55,92.16	75.46
	Sch-3	54.40,54.85	54.29,55.48	69.18,65.20	58.90
	Sch-4	60.82,61.24	58.80,59.36	81.85,81.04	67.19
	$\bar{A}$	59.75	58.26	76.67	64.89

Analysis of Variance for THRPUT

Source of variation	Sum of Squares	d.f.	Mean square	F-ratio	Sig. level
MAIN EFFECTS	2874.4953	5	574.89905	595.532	.0000
LAYOUT	1673.7329	2	836.86646	866.901	.0000
SCH	1200.7623	3	400.25412	414.619	.0000
2-FACTOR INTERACTIONS	226.46774	6	37.744624	39.099	.0000
LAYOUT SCH	226.46774	6	37.744624	39.099	.0000
RESIDUAL	11.584250	12	.9653542		
TOTAL (CORR.)	3112.5473	23			

0 missing values have been excluded.

APPENDIX C

ANOVA OF MEAN WAITING TIME (STORAGE)

Simulation results: Mean waiting time (Storage)
Arrival rate: 4 minutes

Mean Waiting Time		Factor A (Layouts)			
		Layout-1	Layout-2	Layout-3	$\bar{B}$
Factor B(Scheds)	Sch-1	36.55,36.82	35.94,35.27	12.78,12.93	28.39
	Sch-2	8.08, 7.46	8.64, 8.15	6.83, 6.97	7.82
	Sch-3	34.87,34.45	11.78,11.42	8.58, 8.49	16.85
	Sch-4	9.37, 9.09	8.80, 8.89	6.41, 6.55	7.16
	$\bar{A}$	22.19	16.12	6.86	15.06

Analysis of Variance for MSWT

Source of variation	Sum of Squares	d.f.	Mean square	F-ratio	Sig. level
MAIN EFFECTS	2540.7028	5	508.14056	1000.000	.0000
LAYOUT	764.5945	2	382.29727	871.407	.0000
SCH	1776.1082	3	592.03608	1000.000	.0000
2-FACTOR INTERACTIONS	800.64957	6	133.44159	304.166	.0000
LAYOUT SCH	800.64957	6	133.44159	304.166	.0000
RESIDUAL	5.2645500	12	.4387125		
TOTAL (CORR.)	3346.6169	23			

0 missing values have been excluded.

APPENDIX D

ANOVA OF MEAN WAITING TIME (RETRIEVAL)

Simulation results: Mean waiting time (Retrieval)
Arrival rate: 4 minutes

Mean Waiting Time		Factor A (Layouts)			
		Layout-1	Layout-2	Layout-3	$\bar{B}$
Factor B(Scheds)	Sch-1	37.08,37.21	36.33,35.25	12.73,12.79	28.57
	Sch-2	6.13, 6.26	5.22, 5.09	3.38, 3.71	4.97
	Sch-3	34.62,34.36	8.99, 8.67	4.23, 4.54	15.90
	Sch-4	7.63, 7.48	5.32,5.49	3.37, 3.95	5.54
	$\bar{A}$	21.35	13.80	6.09	13.75

Analysis of Variance for MRWT

Source of variation	Sum of Squares	d.f.	Mean square	F-ratio	Sig. level
MAIN EFFECTS	3143.5145	5	628.70289	1000.000	.0000
LAYOUT	931.3504	2	465.67518	1000.000	.0000
SCH	2212.1641	3	737.38804	1000.000	.0000
2-FACTOR INTERACTIONS	898.57267	6	149.76211	1000.000	.0000
LAYOUT SCH	898.57267	6	149.76211	1000.000	.0000
RESIDUAL	.9917500	12	.0826458		
TOTAL (CORR.)	4043.0789	23			

0 missing values have been excluded.

APPENDIX E

ANOVA OF MAXIMUM WAITING TIME (RETRIEVAL)

Simulation results: Maximum waiting time (Retrieval)
Arrival rate: 4 minutes

Maximum Waiting Time		Factor A (Layouts)			
		Layout-1	Layout-2	Layout-3	$\bar{B}$
Factor B(Scheds)	Sch-1	58.00,59.37	58.47,57.30	51.33,52.45	56.16
	Sch-2	950.7,977.7	727.9,717.6	159.1,158.2	609.2
	Sch-3	56.42,55.58	55.05,55.41	41.47,41.25	50.23
	Sch-4	70.40,68.94	69.71,68.18	67.18,67.30	68.82
	$\bar{A}$	282.6	225.8	79.8	196.1

Analysis of Variance for XRWT

Source of variation	Sum of Squares	d.f.	Mean square	F-ratio	Sig. level
MAIN EFFECTS	1576908.7	5	315381.73	1000.000	.0000
LAYOUT	181775.2	2	90887.62	1000.000	.0000
SCH	1395133.4	3	465044.47	1000.000	.0000
2-FACTOR INTERACTIONS	502174.60	6	83695.766	1000.000	.0000
LAYOUT SCH	502174.60	6	83695.766	1000.000	.0000
RESIDUAL	424.12495	12	35.343746		
TOTAL (CORR.)	2079507.4	23			

0 missing values have been excluded.

APPENDIX F

COMPUTER FLOWCHARTS

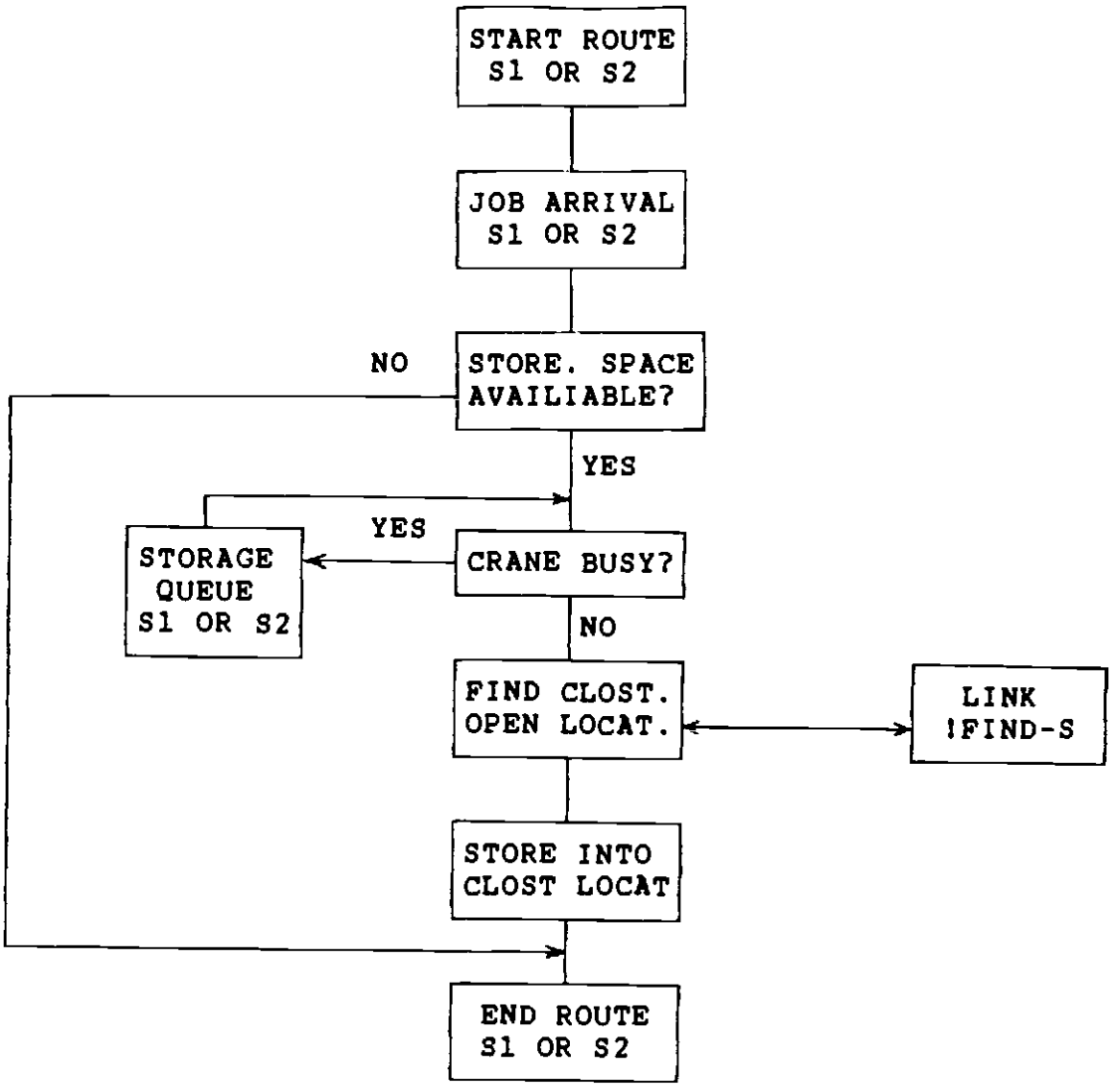


Figure F.1 Storage Route: S1, S2.

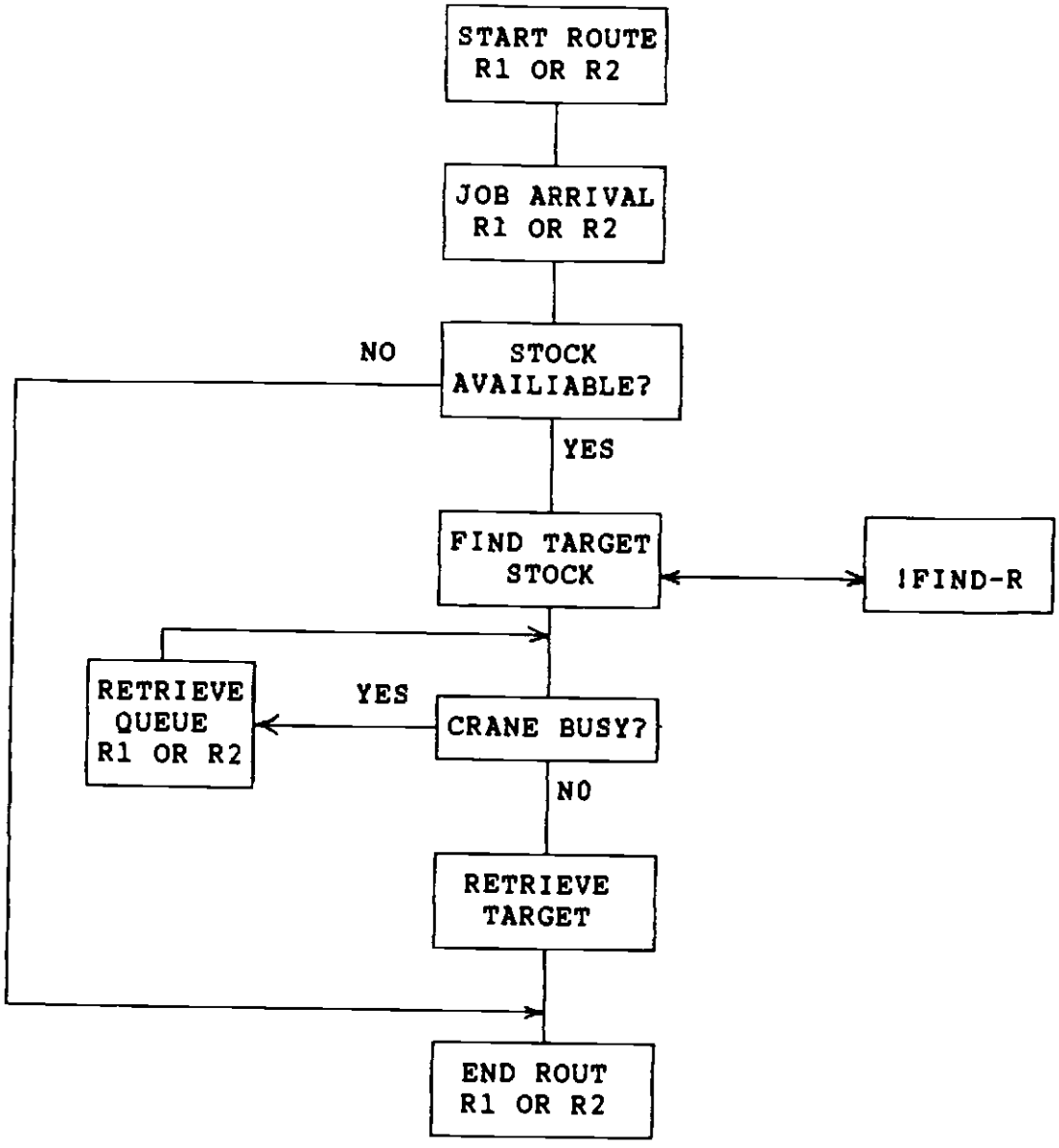


Figure F.2 Retrieval Route: R1, R2.

CONDITIONS	S1	S2	R1	R2
1	0	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1

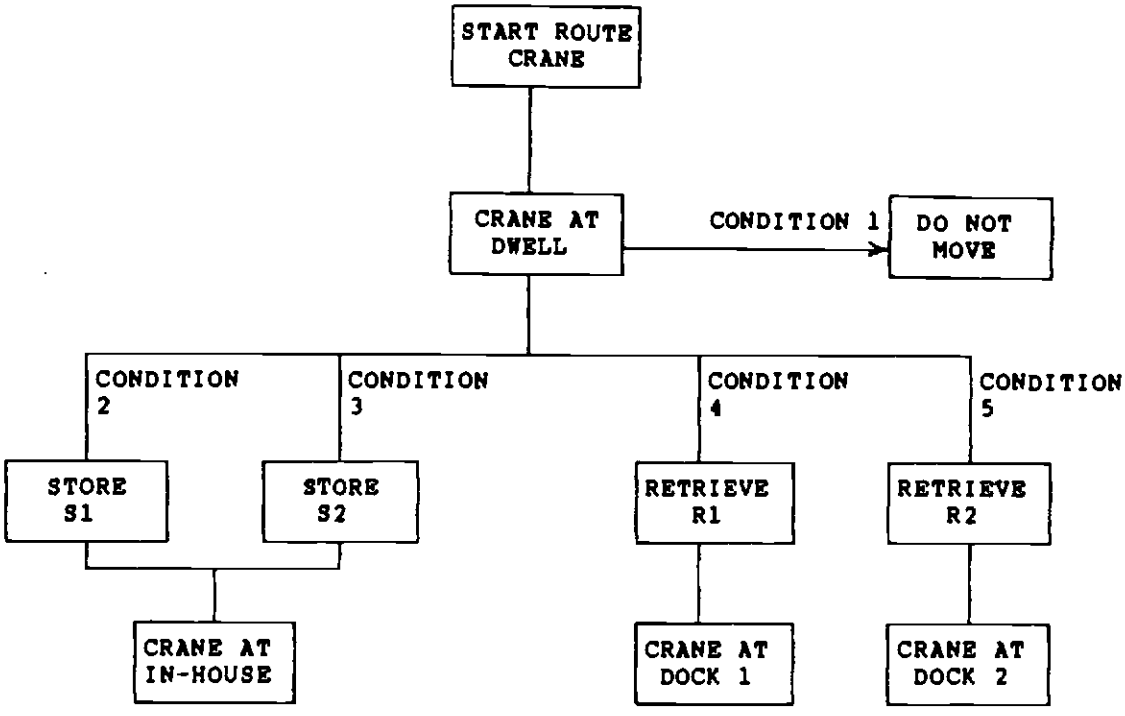


Figure F.3 Crane Route.

CONDITIONS	S1	S2	R1	R2	CONDITIONS	S1	S2	R1	R2
1	0	0	0	0	9	1	0	0	1
2	1	0	0	0	*10	0	1	1	0
3	0	1	0	0	*11	0	0	1	1
4	0	0	1	0	12	1	1	1	0
5	0	0	0	1	13	1	1	0	1
6	1	1	0	0	14	1	0	1	1
7	1	0	1	0	*15	0	1	1	1
8	0	1	0	1	16	1	1	1	1

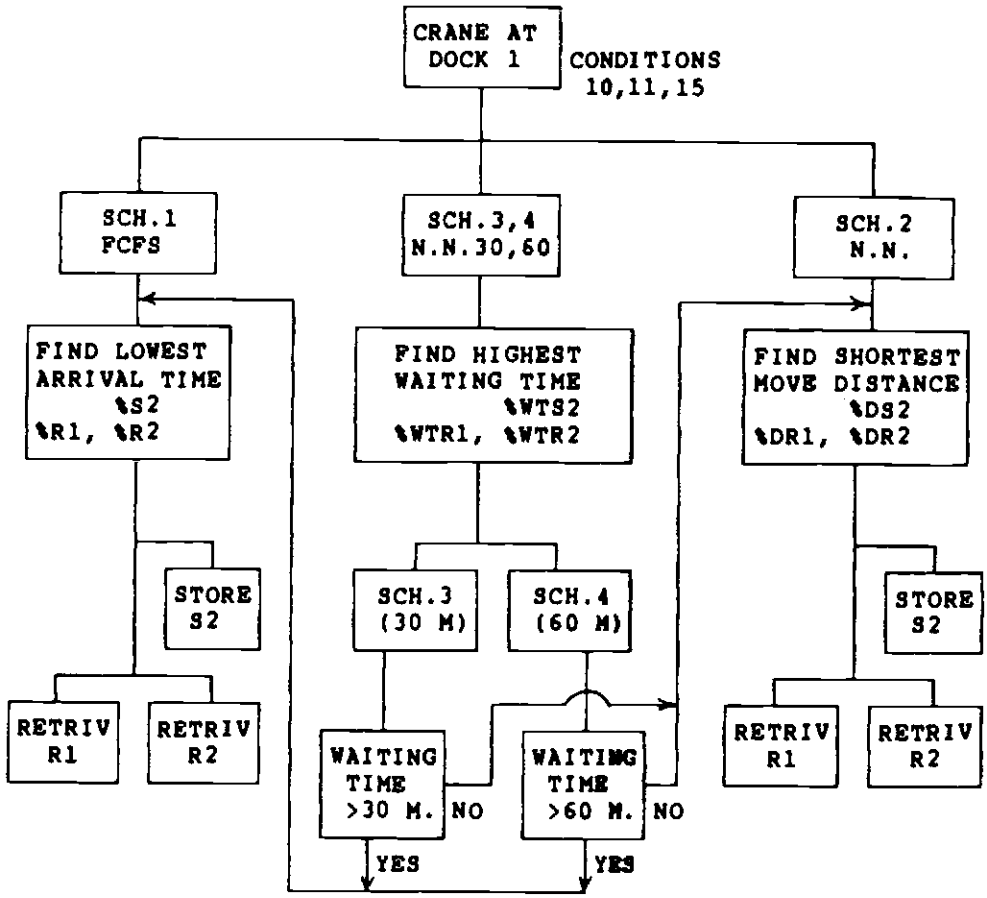


Figure F.4 Crane at Dock 1.

CONDITIONS	S1	S2	R1	R2	CONDITIONS	S1	S2	R1	R2
1	0	0	0	0	* 9	1	0	0	1
2	1	0	0	0	10	0	1	1	0
3	0	1	0	0	*11	0	0	1	1
4	0	0	1	0	12	1	1	1	0
5	0	0	0	1	13	1	1	0	1
6	1	1	0	0	*14	1	0	1	1
7	1	0	1	0	15	0	1	1	1
8	0	1	0	1	16	1	1	1	1

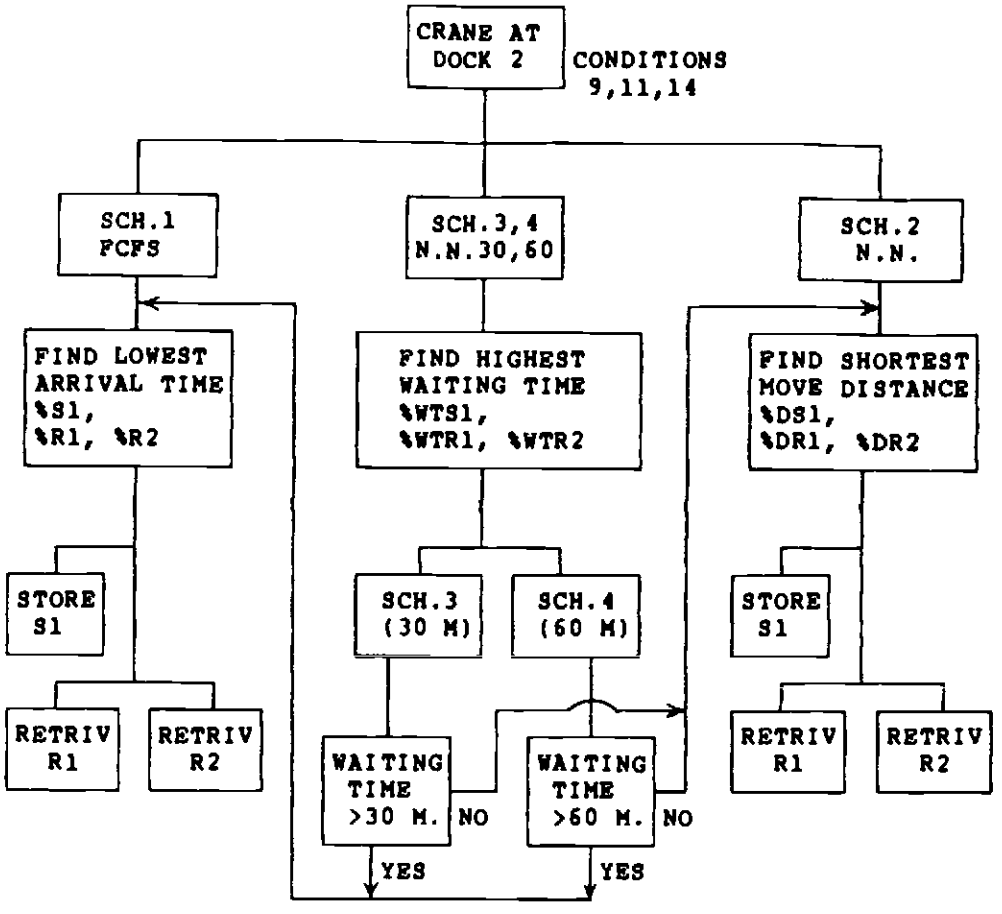


Figure F.5 Crane at Dock 2.

CONDITIONS	S1	S2	R1	R2	CONDITIONS	S1	S2	R1	R2
1	0	0	0	0	* 9	1	0	0	1
2	1	0	0	0	*10	0	1	1	0
3	0	1	0	0	*11	0	0	1	1
4	0	0	1	0	*12	1	1	1	0
5	0	0	0	1	*13	1	1	0	1
6	1	1	0	0	*14	1	0	1	1
7	1	0	1	0	*15	0	1	1	1
8	0	1	0	1	*16	1	1	1	1

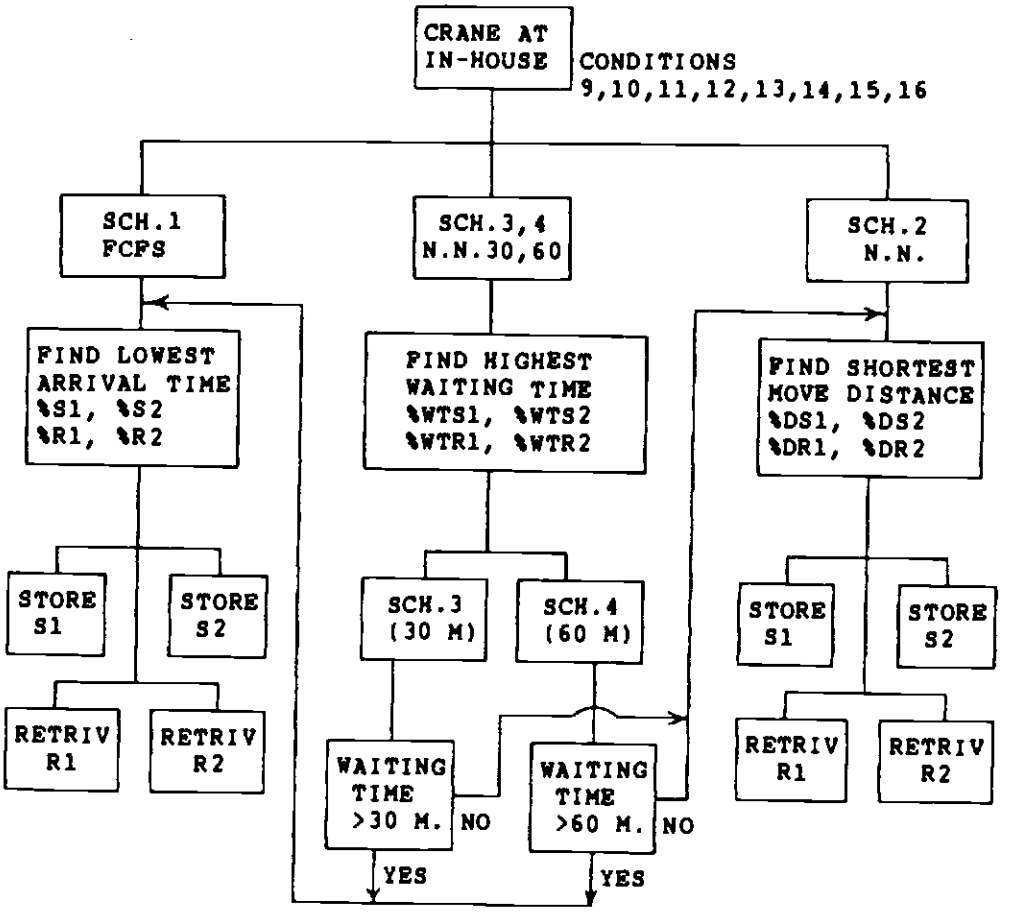


Figure F.6 Crane at In-House.

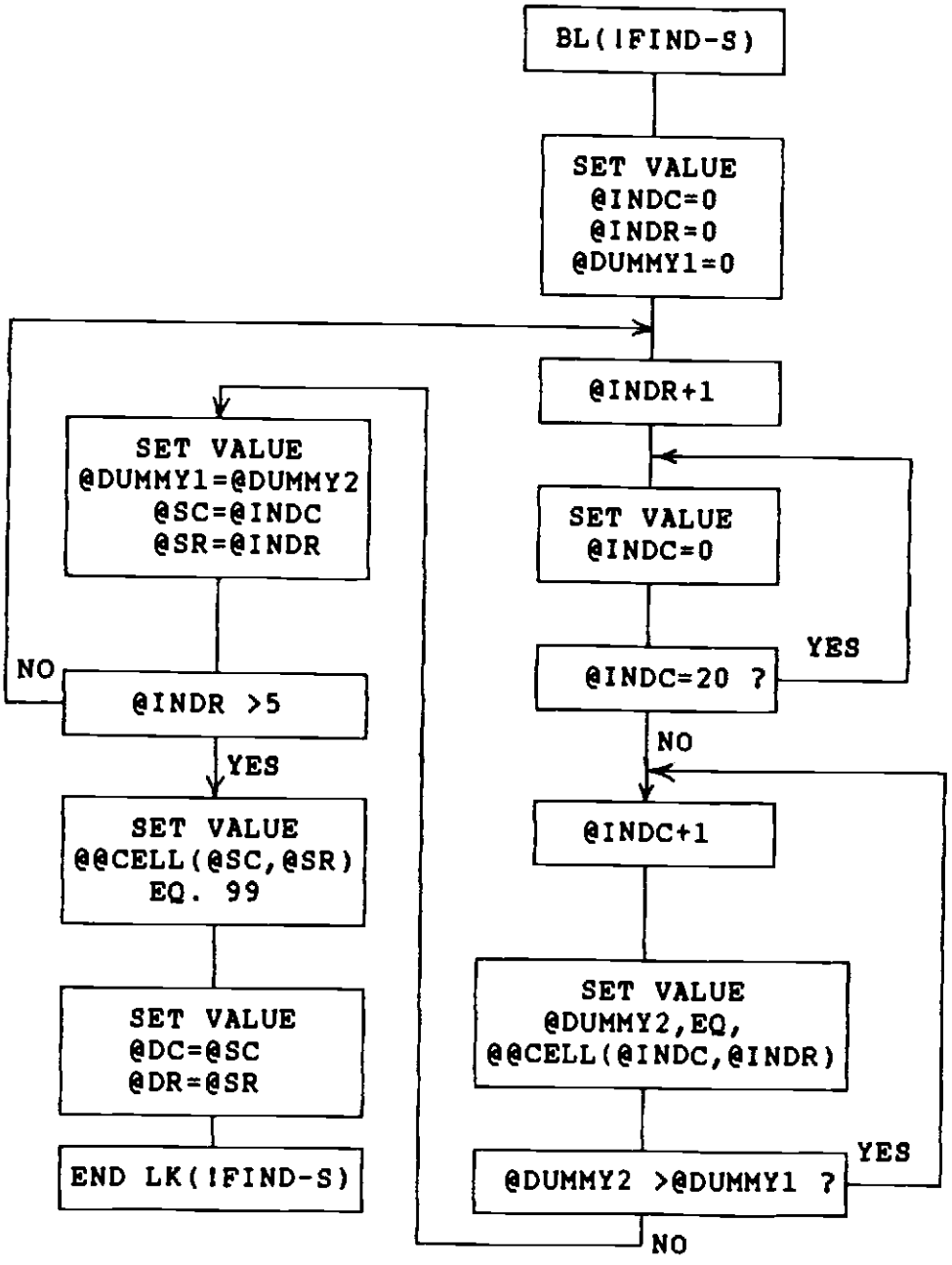


Figure F.7 Find Closest Open Location for Storage Pallets.

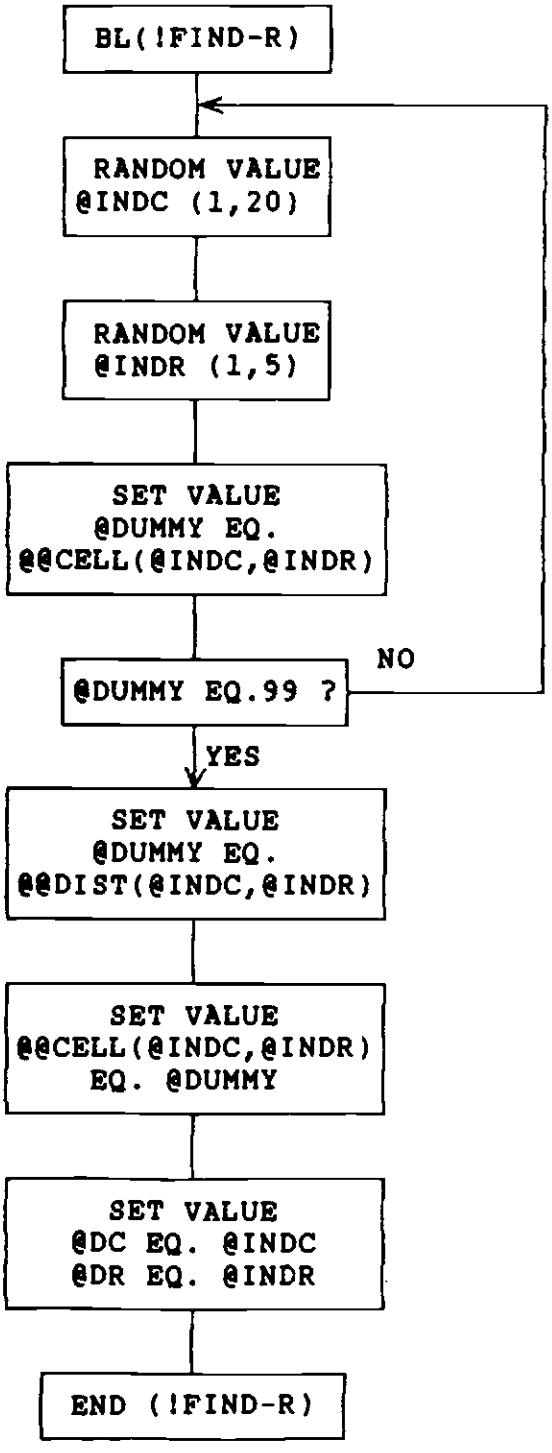


Figure F.8 Find Retrieval Pallets for Retrieval.

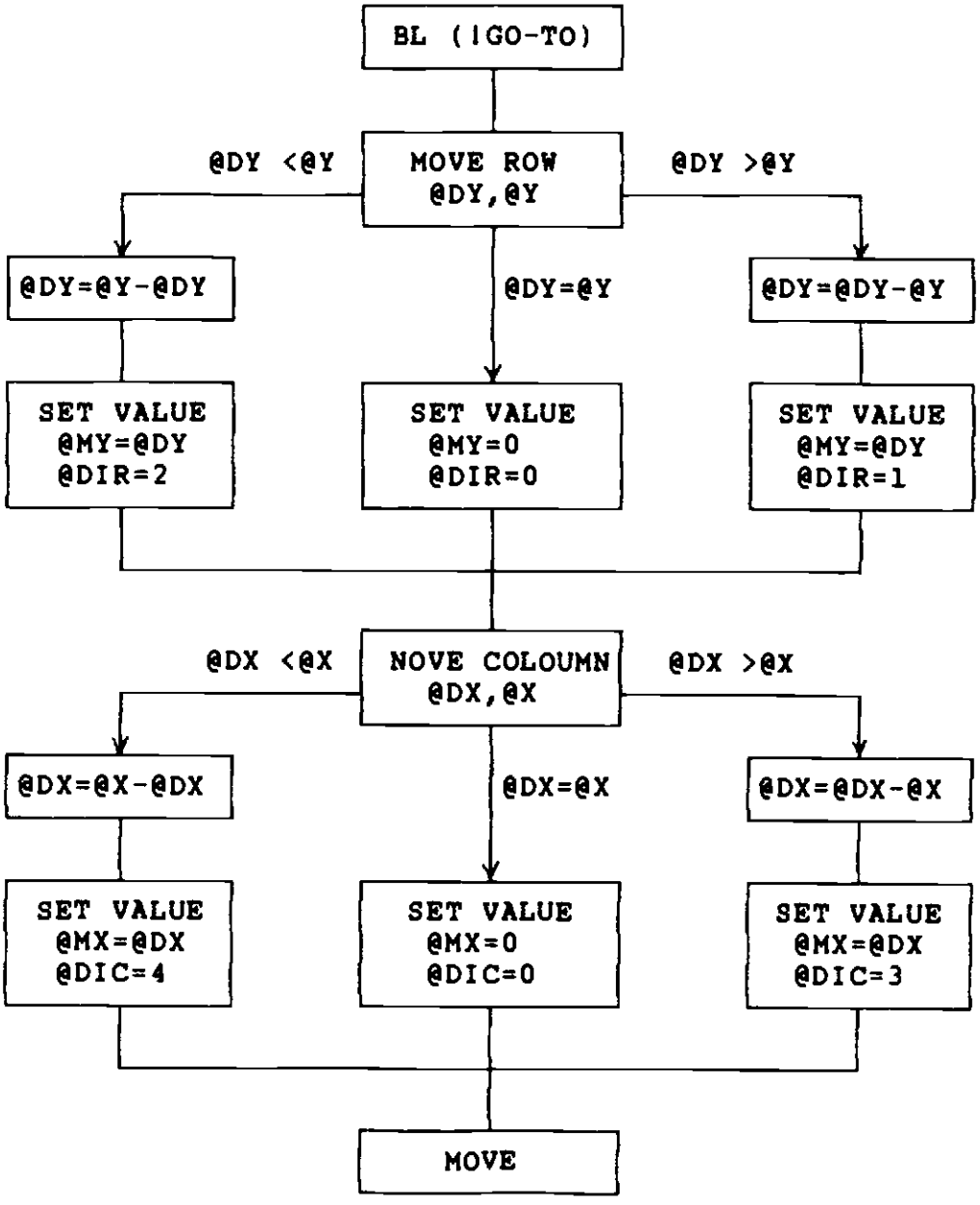


Figure F.9a Crane Movement.

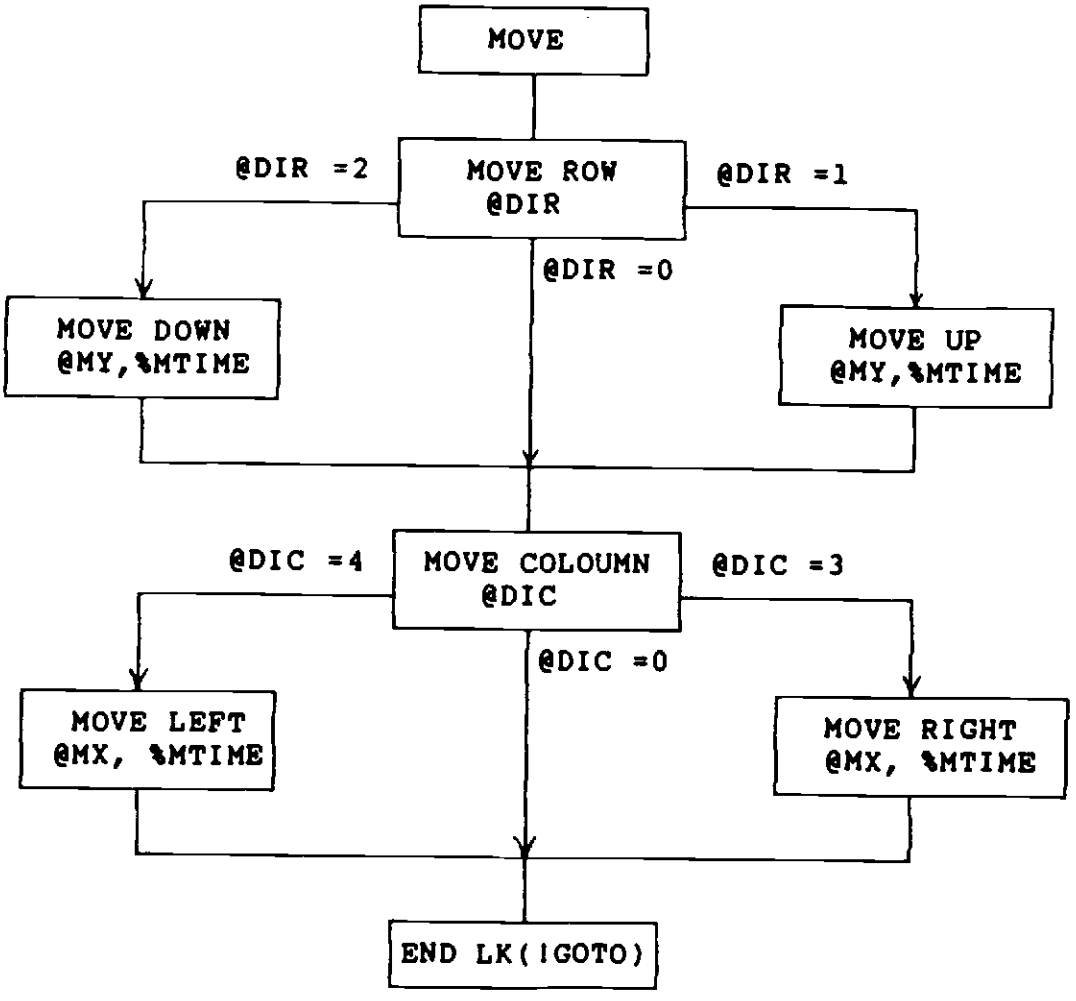


Figure F.9b Crane Movement (continued).

APPENDIX G

COMPUTER PROGRAM LISTINGS

System Initialization

```
;*********      MODEL:   DOCK
;*********               AUTOMATED STORAGE AND RETRIEVAL SYSTEM


;*********  VARIABLES AND ARRAY     ***************************
;*********  ARRIVAL RATE VARIABLE   ***************************
%OARRIVS1=(00:00.00)    ;ARRIVAL TIME OF STORAGE  S1
%OARRIVS2=(00:00.00)    ;ARRIVAL TIME OF STORAGE  S2
%OARRIVR1=(00:00.00)    ;ARRIVAL TIME OF RETRIVAL R1
%OARRIVR2=(00:00.00)    ;ARRIVAL TIME OF RETRIVAL R2

;*********  WAITING TIME VARIABLE  ***************************
%WTIME=(9999:00:00.00) ;MAXIMUM WAITING TIME
%STIME=(0000:00:00.00) ;SIMULATION TIME OF PREIOD

@PRE-UTI=(0)            ; PRE-SET UTILIZATION %

;**********  STATISTICS COLLECTION VARIABLE  ******************
@THROUGHPUTS1=(0)       ;THROUHHPUT S1
@THROUGHPUTS2=(0)       ;THROUHHPUT S2
@THROUGHPUTR1=(0)       ;THROUHHPUT R1
@THROUGHPUTR2=(0)       ;THROUHHPUT R2

%AVG-S1=(0)             ;MEAN WAITING TIME S1
%MAXS1=(0)              ;MAXIMUM WAITING TIME S1
%AVG-S2=(0)             ;MEAN WAITING TIME S2
%MAXS2=(0)              ;MAXIMUM WAITING TIME S2
%AVG-R1=(0)             ;MEAN WAITING TIME R1
%MAXR1=(0)              ;MAXIMUM WAITING TIME R1
%AVG-R2=(0)             ;MEAN WAITING TIME R2
%MAXR2=(0)              ;MAXIMUM WAITING TIME R2
@TTL1=(0)               ;TTL TRANSITIONS OF DOCK 1
@TTL2=(0)               ;TTL TRANSITIONS OF DOCK 2
@TTLHIT1=(0)            ;DUAL COMMAND CYCLE OF DOCK 1
@TTLHIT2=(0)            ;DUAL COMMAND CYCLE OF DOCK 2

;**********  SYSTEM VARIABLE  *********************************

O=(2DOCK)               ;OVERLAY OF LAYOUT-3
L=(50)
#SEEDS1=(151)           ;RANDOM NUMBER SEED 1
#SEEDS2=(152)           ;RANDOM NUMBER SEED 2
#SEEDR1=(351)           ;RANDOM NUMBER SEED 3
#SEEDR2=(452)           ;RANDOM NUMBER SEED 4
#SEEDR=(3578)           ;RANDOM NUMBER SEED 5
#SEED=(2578)            ;RANDOM NUMBER SEED 6

%ARRIVS1=(00:00.00)     ;ARRIVAL TIME OF STORAGE  S1
```

```
%ARRIVS2=(00:00.00)     ;ARRIVAL TIME OF STORAGE  S2
%ARRIVR1=(00:00.00)     ;ARRIVAL TIME OF RETRIVAL R1
%ARRIVR2=(00:00.00)     ;ARRIVAL TIME OF RETRIVAL R2
%MOVETH=(00:02.00)      ;CRANE HORRIZONTAL SPEED
%MOVETV=(00:10.00)      ;CRANE VERTICAL SPEED
%MOVET=(00:10.00)       ;CONVEYOR SPEED FOR R1, R2

;**********   SCHEDULE RULE VARIABLES  ************************
@HIT1=(0)               ;VISITS TO DOCK1
@HIT2=(0)               ;VISITS TO DOCK2
@CODE=(0)               ;VISIT TO DOCK1=1,VISIT TO DOCK2=2
%S1=(0)                 ;S1 STARTING TIME
%S2=(0)                 ;S2 STARTING TIME
%R1=(0)                 ;R1 STARTING TIME
%R2=(0)                 ;R2 STARTING TIME
%ST=(0)
;***********  JOB COUNTOR  ***************************************
@UTI=(0)                ;%UTILIZATION
@UTIS1=(0)
@UTIS2=(0)
@S1=(0)                 ;S1 WIP
@S2=(0)                 ;S2 WIP
@R1=(0)                 ;R1 WIP
@R2=(0)                 ;R2 WIP
@MESAGE=(0) ;MESAGE CODE; =0 PM" ", =1 PM"S", =2 PM"R",=3 PM"G"
@DOCK=(0)   ;@DOCK=1(DOCK1),=2(DOCK2),DOCK=3(IN-HOUSE),=0(DWELL)

;************  VARIABLES FOR SIMU MOVE SPEED ADJUST ************
%MTIME=(0:00.00)        ;ADJUSTED CRANE SPEED FOR SIMU. MOVE

;*************  FIND CELL FOR STORAGE SND RETRIEVAL ************
@INDEXC=(0)
@INDEXR=(0)
@SS=(0)
@SC=(0)
@SR=(0)
@RC=(0)
@RR=(0)

;*************  VARIABLE FOR CRANE MOVE CACULATION ***********
@MOVE=(0)               ;CRANE MOVEMENT INDICATOR
@C=(0)                  ;CRANE CURRENT LOCATION
@R=(0)                  ;CRANE CURRENT LOCATION
@DC=(0)                 ;CRANE DESTINATION LOCATION
@DR=(0)                 ;CRANE DESTINATION LOCATION
@X=(0)                  ;CRANE CURRENT LOCATION
@Y=(0)                  ;CRANE CURRENT LOCATION
@DX=(0)                 ;CRANE DESTINATION LOCATION
@DY=(0)                 ;CRANE DESTINATION LOCATION
@MX=(0)                 ;MOVE DISTANCE
@MY=(0)                 ;MOVE DISTANCE
@DIR=(0)                ;MOVE DIRECTION @DIC=1(UP),=2(DOWN)
@DIC=(0)                ;MOVE DIRECTION @DIC=3(LEFT),=4(RIGHT)
```

```
;************  UTILIZATION (%) PRE-SET VARIABLE **************
@AC=(0)
@BC=(0)
@DUMMYAC=(0)
@DUMMYR=(0)
%STD=(00:01.00)

;************  ARITHMETIC VARIABLE ****************************
%DUMMY=(0)
%DUMMY1=(0)
%DUMMY2=(0)
@DUMMY=(0)
@DUMMY1=(0)
@DUMMY2=(0)
@DUMMY3=(0)
@DUMMY4=(0)
@FIND-R=(0)
%TIMER=(0)
%TIMER-R1=(0)
%TIMER-R2=(0)
@TIMER-1C=(0)
@TIMER-1R=(0)
@TIMER-2C=(0)
@TIMER-2R=(0)
@DIST-1C=(0)
@DIST-1R=(0)
@DIST-2C=(0)
@DIST-2R=(0)
%DIST-R1=(0)
%DIST-R2=(0)
@S1S2=(0)
@R1R2=(0)
@SET1=(0)
@SET2=(0)
@SETR1C=(0)
@SETR1R=(0)
@SETR2C=(0)
@SETR2R=(0)
%TTL-S1=(0)
%TTL-S2=(0)
%TTL-R1=(0)
%TTL-R2=(0)

;*************  LOCATION VARIABLES ****************************
*CRANE=(XY(1,1))          ;INDICATOR OF CRANE
*MOVE=(XY(1,2))           ;INDICATOR OF CRANE MOVE
*S1=(XY(1,3))             ;INDICATOR OF STORAGE
*S2=(XY(1,4))             ;INDICATOR OF STORAGE
*R1=(XY(1,5))             ;INDICATOR OF R1
*R2=(XY(1,6))             ;INDICATOR OF R2
*DWELL=(XY(39,17))        ;CRANE DWELL POSITION
*DOCK1=(XY(19,17))        ;DOCK 1 LOCATION
```

```
*READY1=(XY(18,17))      ;STAND-BY POSITION FOR CRANE
*DOCK2=(XY(61,17))       ;DOCK 2 LOCATION
*READY2=(XY(62,17))      ;STAND-BY POSITION FOR CRANE
*MESAGE=(XY(19,16))      ;PRINT MASAGE

;************** JOB DESCRIPTION ****************************
J=(1,&,1,0,0,0,1)        ;INITIAL SYSTEM
J=(2,C,2,0,0,1,1)        ;CRANE
J=(3,S,3,0,0,1,25000)    ;STORAGE JOB  S1
J=(4,S,4,0,0,1,25000)    ;STORAGE JOB  S2
J=(5,R,5,0,0,1,25000)    ;RETRIEVAL JOB R1
J=(6,G,6,0,0,1,25000)    ;RETRIEVAL JOB R2

;***************  ARRAY  ***************************************
@@CELL=(20,5)            ;OPEN LOCATION
%%CELL=(20,5)            ;RETRIEVAL WAITING TIME
%%DISTA=(20,5)           ;DISTANCE FROM DOCK 1
%%DISTB=(20,5)           ;DISTANCE FROM DOCK 2
@@DA=(20,5)              ;DOCK 1 RETRIEVAL REFERENCE
@@DB=(20,5)              ;DOCK 2 RETRIEVAL REFERENCE
@@DAA=(20,5)             ;DOCK 1 RETRIEVAL MAP
@@DBB=(20,5)             ;DOCK 2 RETRIEVAL MAP
;*************** END OF VARIABLES AND ARRAY ******************
;*************** ROUT #1 ( SYSTEM INITIALIZATION ) ************
BR(1,XY(70,20),0)
GV(%OARRIVS1)
GV(%OARRIVR1)
GV(%OARRIVS2)
GV(%OARRIVR2)
GV(@PRE-UTI)
GV(%WTIME)
GV(%STIME)
SV(@TTLHIT1,0)
SV(@TTLHIT2,0)
SV(@TTL1,0)
SV(@TTL2,0)
SV(%TIMER,0)
SV(%DUMMY,0)
SV(%TIMER-R1,0)
SV(%TIMER-R2,0)
SV(@TIMER-1C,0)
SV(@TIMER-1R,0)
SV(@TIMER-2C,0)
SV(@TIMER-2R,0)
SV(@DIST-1C,0)
SV(@DIST-1R,0)
SV(@DIST-2C,0)
SV(@DIST-2R,0)
SV(@SET1,0)
SV(@SET2,0)
SV(@SETR1C,0)
SV(@SETR1R,0)
SV(@SETR2C,0)
```

```
SV(@SETR2R,0)
SV(@FIND-R,0)
SV(%DIST-R1,0)
SV(%DIST-R2,0)
SV(%STD,00:01.00)
;************* INITIAL ARRARY *********************************
IA(@@CELL)
IA(%%CELL)
IA(%%DISTA)
IA(%%DISTB)
IA(@@DA)
IA(@@DB)
IA(@@DAA)
IA(@@DBB)
;************* INITIAL DATA FILE ******************************
; OF(2DWT1S.RPT)
; OF(2DWT2S.RPT)
; OF(2DWT1R.RPT)
; OF(2DWT2R.RPT)
; OF(1DOCK.RPT)
; OF(2DOCK.RPT)
; CF(RPT)
;************* INITIAL VARIABLES ******************************
SV(%ARRIVS1,%OARRIVS1)
SV(%ARRIVS2,%OARRIVS2)
SV(%ARRIVR1,%OARRIVR1)
SV(%ARRIVR2,%OARRIVR2)
SV(%S1,0)
SV(%S2,0)
SV(%R1,0)
SV(%R2,0)
SV(%ST,0)
SV(@AC,0)
SV(@BC,0)
SV(@DUMMYAC,0)
SV(@DUMMYR,0)
SV(@UTI,0)
SV(@UTIS1,0)
SV(@UTIS2,0)
SV(@THROUGHPUTS1,0)
SV(@THROUGHPUTS2,0)
SV(@THROUGHPUTR1,0)
SV(@THROUGHPUTR2,0)
SV(@S1,0)
SV(@S2,0)
SV(@R1,0)
SV(@R2,0)
SV(@HIT1,0)
SV(@HIT2,0)
SV(@CODE,0)
SV(@INDEXR,0)
SV(@INDEXC,0)
SV(@SS,0)
```

```
SV(@SC,0)
SV(@SR,0)
SV(@RC,0)
SV(@RR,0)
SV(@R,0)
SV(@C,0)
SV(@DIC,0)
SV(@DIR,0)
SV(@X,0)
SV(@Y,0)
SV(@DX,0)
SV(@DY,0)
SV(@MX,0)
SV(@MY,0)
SV(@DUMMY1,0)
SV(@DUMMY2,0)
SV(@DUMMY3,0)
SV(@DUMMY4,0)
SV(@MOVE,0)
SV(@DOCK,0)
PO(*S1)
PO(*S2)
PO(*R1)
PO(*R2)
SV(%MTIME,0)
SV(%DUMMY1,0)
SV(%DUMMY2,0)
SV(%TTL-S1,0)
SV(%TTL-S2,0)
SV(%TTL-R1,0)
SV(%TTL-R2,0)
SV(%AVG-S1,0)
SV(%AVG-S2,0)
SV(%AVG-R1,0)
SV(%AVG-R2,0)
SV(%MAXS1,0)
SV(%MAXS2,0)
SV(%MAXR1,0)
SV(%MAXR2,0)
*************** END OF SYSTEM INITIALIZATION ****************
```

Layout 1

```
;*********      MODEL: AUTOMATED STOTAGE AND RETRIEVAL SYSTEM
;********* FILE NAME: LAYOUT1.MDL
;********* THIS PROGRAM IS FOR LAYOUT-1 (ONE DOCK)
;*********   STORAGE: FCFS, CLOSEST OPEN LOCATION
;*********  RETRIEVE: FCFS
;*********            NEAREST-NEIGHBOR

;----------- PRE-SET @@DIST FOR CLOSEST LOCATION REFERENCE ----
:SETDIST
        SV(%STD,00:04.00)
        SV(@DUMMY2,@INDEXR)
        IV(@INDEXC)
        SV(@DUMMY1,@INDEXC)
        AO(@DUMMY1,*,1)
        AO(@DUMMY2,*,5)
        IF(@DUMMY1,LE,@DUMMY2,:SETDUMMY2)
:SETDUMMY1
        IV(@INDEXR)
        SV(@@DIST(@INDEXC,@INDEXR),@DUMMY1)
        AO(%STD,*,@DUMMY1)
        SV(%%DIST(@INDEXC,@INDEXR),%STD)
        DV(@INDEXR)
        JP(:CHECK1)
:SETDUMMY2
        IV(@INDEXR)
        SV(@@DIST(@INDEXC,@INDEXR),@DUMMY2)
        AO(%STD,*,@DUMMY2)
        SV(%%DIST(@INDEXC,@INDEXR),%STD)
        DV(@INDEXR)
:CHECK1
        IF(@INDEXC,EQ,20,:CHECK2)
        JP(:SETDIST)
:CHECK2
        IF(@INDEXR,EQ,4,:ENDDIST)
        SV(@INDEXC,0)
        IV(@INDEXR)
        JP(:SETDIST)
:ENDDIST
        SV(@INDEXC,0)
        SV(@INDEXR,0)
;-------------------------------------------------------
;--- SET @@CELL ARRAY FOR UTILIZATION PRE-SET ---------

:SETCELL
        SV(@DUMMY2,@INDEXR)
        IV(@INDEXC)
        SV(@DUMMY1,@INDEXC)
        AO(@DUMMY1,*,1)
```

```
        AO(@DUMMY2,*,5)
        IF(@DUMMY1,LE,@DUMMY2,:SETCELL2)
:SETCELL1
        IV(@INDEXR)
        SV(@@CELL(@INDEXC,@INDEXR),@DUMMY1)
        DV(@INDEXR)
        JP(:CHECK11)
:SETCELL2
        IV(@INDEXR)
        SV(@@CELL(@INDEXC,@INDEXR),@DUMMY2)
        DV(@INDEXR)
:CHECK11
        IF(@INDEXC,EQ,20,:CHECK22)
        JP(:SETCELL)
:CHECK22
        IF(@INDEXR,EQ,4,:ENDCELL)
        SV(@INDEXC,0)
        IV(@INDEXR)
        JP(:SETCELL)
:ENDCELL
        SV(@INDEXC,0)
        SV(@INDEXR,0)
        SV(@DUMMY1,0)
        SV(@DUMMY2,0)

;-----------PRE-SET UTILIGATION (%)-----------------
:PRESET
 LK(!FIND-S)

        SV(@MESAGE,1)
 LK(!MESAGE)
        IV(@UTI)
        IV(@LKHUTI)
        IF(@UTI,LT,@PREUTI,:PRESET)
;-------------------------------------------------
RS(1,#SEEDS1)
RS(2,#SEEDS2)
RS(3,#SEEDR1)
RS(4,#SEEDR2)
RS(5,#SEEDR)
RS(9,#SEED)
WK(I)
      ER
;-------------------------------------------------
;-------- ROUT #3 (STORAGE S1)-------------------
      BR(3,XY(6,6),%ARRIVS1)

        RV(1,E,%ARRIVS1,%OARRIVS1)
        IF(@LKHUTI,GE,100,:END-S1)
        IF(@S1,GE,10,:END-S1)
        SV(OBJ%ST,CLOCK)
IV(@S1)
        MR(10,0)
```

```
          MD(10,0)
          SV(%S1,OBJ%ST)
          TP(*S1)
          WE
          PO(*S1)
          MA(*READY1,0)
          DV(@S1)
          IV(@UTI)
          IF(@MOVE,NE,999,WAIT)
          SV(@MOVE,0)
          SV(OBJ%2,CLOCK)
          AO(OBJ%2,-,OBJ%ST)
;         OF(1DWT1S.RPT[,A])
;         PV(F,OBJ%2)
;         CF(RPT)
          IV(@THROUGHPUTS1)
          AO(%TTL-S,+,OBJ%2)
          IF(OBJ%2,LT,%MAXS,:GO-S1)
          SV(%MAXS,OBJ%2)
:GO-S1
          IV(@LKHUTI)
:END-S1
       ER
;-----------------------------------------------
;----------- ROUT #4 (STORAGE S2)----------------
       BR(4,XY(7,4),%ARRIVS2)

          RV(2,E,%ARRIVS2,%OARRIVS2)
          IF(@LKHUTI,GE,100,:END-S2)
          IF(@S2,GE,10,:END-S2)
          SV(OBJ%ST,CLOCK)
          IV(@S2)
          MR(9,0)
          MD(12,0)
          SV(%S1,OBJ%ST)
          TP(*S1)
          WE
          PO(*S1)
          MA(*READY1,0)
          DV(@S2)
          IV(@UTI)
          IF(@MOVE,NE,999,WAIT)
          SV(@MOVE,0)
          SV(OBJ%2,CLOCK)
          AO(OBJ%2,-,OBJ%ST)
;         OF(1DWT1S.RPT[,A])
;         PV(F,OBJ%2)
;         CF(RPT)
          IV(@THROUGHPUTS2)
          AO(%TTL-S,+,OBJ%2)
          IF(OBJ%2,LT,%MAXS,:GO-S2)
          SV(%MAXS,OBJ%2)
:GO-S2
```

```
        IV(@LKHUTI)
:END-S2
      ER
;-------------------------------------------------------
;---------- ROUT# 5 (RETRIEVE R1) ----------------------
      BR(5,XY(42,18),%ARRIVR1)

        RV(3,E,%ARRIVR1,%OARRIVR1)
        IF(@LKHUTI,LE,0,:END-R1)
        IF(@R1,GE,10,:END-R1)
        SV(OBJ%ST,CLOCK)
 LK(!SET-R1)
        DV(@LKHUTI)
        IV(@R1)
        ML(15,0)
        SV(%R1,OBJ%ST)
        TP(*R1)
        WE
        PO(*R1)
        MA(*READY1,0)
        DV(@R1)
        DV(@UTI)
        SV(OBJ%1,%ST)
        IF(@R1,NE,0,:GO-ON-R1)
        SV(%R1,999:00:00.00)
:GO-ON-R1
        IF(@MOVE,NE,999,WAIT)
        SV(@MOVE,0)
        SV(OBJ%2,CLOCK)
        AO(OBJ%2,-,OBJ%1)
;       OF(1DWT1R.RPT[,A])
;       PV(F,OBJ%2)
;       CF(RPT)
        IV(@THROUGHPUTR1)
        AO(%TTL-R1,+,OBJ%2)
        IF(OBJ%2,LT,%MAXR1,:GO-R1)
        SV(%MAXR1,OBJ%2)
:GO-R1
;       MD(5,%MOVET)
;       ML(16,%MOVET)
:END-R1
ER
;-------------------------------------------------------
;----------- ROUT# 6 (RETRIEVE R2) ---------------------
      BR(6,XY(42,20),%ARRIVR2)

        RV(4,E,%ARRIVR2,%OARRIVR2)
        IF(@LKHUTI,LE,0,:END-R2)
        IF(@R2,GE,10,:END-R2)
        SV(OBJ%ST,CLOCK)
 LK(!SET-R2)
        DV(@LKHUTI)
        IV(@R2)
```

```
        ML(15,0)
        SV(%R2,OBJ%ST)
        TP(*R2)
        WE
        PO(*R2)
        MA(*READY1,0)
        DV(@R2)
        DV(@UTI)
        SV(OBJ%1,%ST)
        IF(@R2,NE,0,:GO-ON-R2)
        SV(%R2,999:00:00.00)
:GO-ON-R2
        IF(@MOVE,NE,999,WAIT)
        SV(@MOVE,0)
        SV(OBJ%2,CLOCK)
        AO(OBJ%2,-,OBJ%1)
;       OF(1DWT2R.RPT[,A])
;       PV(F,OBJ%2)
;       CF(RPT)
        IV(@THROUGHPUTR2)
        AO(%TTL-R2,+,OBJ%2)
        IF(OBJ%2,LT,%MAXR2,:GO-R2)
        SV(%MAXR2,OBJ%2)
:GO-R2
;       MD(5,%MOVET)
;       MR(49,%MOVET)
:END-R2
     ER
;-------------------------------------------------------
;---------- ROUT #2 (CRANE) ----------------------------
      BR(2,*DWELL,0)

:DOCK1
        WT(00:00.01)
        IF(CLOCK,GE,%STIMER,:END-CRANE)
        SV(@DOCK,1)
        SV(@C,0)
        SV(@R,0)
        IF(@UTI,GE,100,:DOCK1-UTI1)
        IF(@UTI,EQ,0,:DOCK1-UTI0)
        IF(@S1,NE,0,:DOCK1-S1)
        IF(@S2,NE,0,:DOCK1-S1)
        IF(@R1,NE,0,:DOCK1-R)
        IF(@R2,NE,0,:DOCK1-R)
        WE
        JP(:DOCK1)
:DOCK1-UTI1
        IF(@R1,NE,0,:DOCK1-R)
        IF(@R2,NE,0,:DOCK1-R)
        WE
        JP(:DOCK1-UTI1)
:DOCK1-UTI0
        IF(@S1,NE,0,:DOCK1-S1)
```

```
        IF(@S2,NE,0,:DOCK1-S1)
        WE
        JP(:DOCK1-UTI0)
:DOCK1-S1
        CL(*S1)
 LK(!FIND-S)
 LK(!GO-TO)
        SV(@MESAGE,1)
 LK(!MESAGE)
        SV(@MOVE,999)
        JP(:IN-HOUSE)
:DOCK1-R
 LK(!FIND-R)
        IF(@CODE,EQ,1,:DOCK1-R1)
        IF(@CODE,EQ,2,:DOCK1-R2)
:DOCK1-R1
        CL(*R1)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
 LK(!BACK-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
:DOCK1-R2
        CL(*R2)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
 LK(!BACK-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
;-------------------------------------------------
;------------ IN-HOUSE ---------------------------
:IN-HOUSE
        IF(CLOCK,GE,%STIMER,:END-CRANE)
        SV(@DOCK,3)
        SV(@C,@DC)
        SV(@R,@DR)

        IF(@UTI,GE,100,:DOCK3-UTI1)
        IF(@UTI,EQ,0,:DOCK3-UTI0)
:CHECK-DOCK3
        IF(@R1,NE,0,:DOCK3-R)
        IF(@R2,NE,0,:DOCK3-R)
        IF(@S1,NE,0,:DOCK3-DOCK1)
        IF(@S2,NE,0,:DOCK3-DOCK1)
        WE
        JP(:CHECK-DOCK3)
:DOCK3-UTI1
        IF(@R1,NE,0,:DOCK3-R)
        IF(@R2,NE,0,:DOCK3-R)
        WE
        JP(:DOCK3-UTI1)
```

```
:DOCK3-UTI0
        IF(@S1,NE,0,:DOCK3-DOCK1)
        IF(@S2,NE,0,:DOCK3-DOCK1)
        WE
        JP(:DOCK3-UTI0)
:DOCK3-R1R2
        IF(%R1,LT,%R2,:DOCK3-R1)
        IF(%R2,LT,%R1,:DOCK3-R2)
:DOCK3-R
 LK(!FIND-R)
        IF(@CODE,EQ,1,:DOCK3-R1)
        IF(@CODE,EQ,2,:DOCK3-R2)
:DOCK3-R1
        CL(*R1)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
 LK(!BACK-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
:DOCK3-R2
        CL(*R2)
        LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
 LK(!BACK-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
:DOCK3-DOCK1
        SV(@C,@DC)
        SV(@R,@DR)
 LK(!BACK-DOCK)
        JP(:DOCK1)
;-------------------------------------------------
:END-CRANE
;       OF(1DWT1S.RPT[,A])
;       PV(F,9999:00:00.00)
;       OF(1DWT1R.RPT[,A])
;       PV(F,9999:00:00.00)
;       OF(1DWT2R.RPT[,A])
;       PV(F,9999:00:00.00)
;       CF(RPT)
        SV(%AVG-S,%TTL-S)
        AO(%AVG-S,/,@THROUGHPUTS1)
        SV(%AVG-R1,%TTL-R1)
        AO(%AVG-R1,/,@THROUGHPUTR1)
        SV(%AVG-R2,%TTL-R2)
        AO(%AVG-R2,/,@THROUGHPUTR2)
      ER
;-------------------------------------------------
;--------  BL(!BACK-DOCK)  -----------------------
      BL(!BACK-DOCK)
```

```
:BACK-DOCK
        SV(@DOCK,1)
        SV(@C,@DC)
        SV(@R,@DR)
        SV(@DC,0)
        SV(@DR,0)
      LK(!GO-TO)
      EL
;-----------------------------------------------------
;-------- BL(!FIND-S) FIND CLOSET CELL FOR STORAGE -----
      BL(!FIND-S)
        SV(@INDEXC,0)
        SV(@INDEXR,0)
        SV(@DUMMY1,99)
:FIND-S
        IF(@INDEXR,EQ,5,:SETFIND-S)
        IV(@INDEXR)
        SV(@INDEXC,0)
:FIND-SS
        IF(@INDEXC,EQ,20,:FIND-S)
        IV(@INDEXC)
        SV(@DUMMY2,@@CELL(@INDEXC,@INDEXR))
        IF(@DUMMY2,GE,@DUMMY1,:FIND-SS)
:FIND-SSS
        SV(@DUMMY1,@DUMMY2)
        SV(@SC,@INDEXC)
        SV(@SR,@INDEXR)
        JP(:FIND-SS)
:SETFIND-S
        SV(@@CELL(@SC,@SR),99)
        AO(@SC,*,2)
        AO(@SR,*,2)
        SV(@DC,@SC)
        SV(@DR,@SR)
     EL
;-----------------------------------------------------
;------------ BL(!SET-R1) ----------------------------
     BL(!SET-R1)
:SET-R1
        RV(5,@SETR1C,1,20)
        RV(5,@SETR1R,1,5)
        SV(@SET1,@@CELL(@SETR1C,@SETR1R))
        IF(@SET1,NE,99,:SET-R1)
        SV(@@CELL(@SETR1C,@SETR1R),91)
        SV(%%CELL(@SETR1C,@SETR1R),OBJ%ST)
        SV(@RC,@SETR1C)
        SV(@RR,@SETR1R)
        AO(@RC,*,2)
        AO(@RR,*,2)
        SV(@MESAGE,2)
        LK(!MESAGE)
      EL
;-----------------------------------------------------
```

```
;------------ BL(!SET-R2) ------------------------------
     BL(!SET-R2)
:SET-R2
        RV(5,@SETR2C,1,20)
        RV(5,@SETR2R,1,5)
        SV(@SET2,@@CELL(@SETR2C,@SETR2R))
        IF(@SET2,NE,99,:SET-R2)
        SV(@@CELL(@SETR2C,@SETR2R),92)
        SV(%%CELL(@SETR2C,@SETR2R),OBJ%ST)
        SV(@RC,@SETR2C)
        SV(@RR,@SETR2R)
        AO(@RC,*,2)
        AO(@RR,*,2)
        SV(@MESAGE,3)
        LK(!MESAGE)
      EL
;-------------------------------------------------------
;------------ BL(!FIND-R) ------------------------------
      BL(!FIND-R)
        SV(@FIND-R,1)
        SV(%R-TIME,0)
        SV(%R-DIST,999:00:00.00)
        SV(@INDEXC,0)
        SV(@INDEXR,1)
:FIND-R
        SV(%TIME,CLOCK)
        IV(@INDEXC)
        IF(@INDEXC,GT,20,:CHECK-R)
        SV(@DUMMY,@@CELL(@INDEXC,@INDEXR))
        IF(@DUMMY,EQ,91,:R-TIME)
        IF(@DUMMY,EQ,92,:R-TIME)
        JP(:FIND-R)
:CHECK-R
        SV(@INDEXC,0)
        IV(@INDEXR)
        IF(@INDEXR,GT,5,:COMP)
        JP(:FIND-R)
:R-TIME
        SV(%DUMMY,%%CELL(@INDEXC,@INDEXR))
        AO(%TIME,-,%DUMMY)
        IF(%TIME,LT,%R-TIME,:R-DIST)
        IF(%TIME,GT,%R-TIME,:R-TIME1)
        RV(9,@R1R2,1,2)
        IF(@R1R2,EQ,1,:R-TIME1)
        IF(@R1R2,EQ,2,:R-DIST)
:R-TIME1
        SV(%R-TIME,%TIME)
        SV(@TIMEC,@INDEXC)
        SV(@TIMER,@INDEXR)
:R-DIST
        SV(@DC,@INDEXC)
        SV(@DR,@INDEXR)
        AO(@DC,*,2)
```

```
        AO(@DR,*,2)
      LK(!GO-TO)
        SV(%DUMMY1,%%DIST(@INDEXC,@INDEXR))
        AO(%DUMMY,+,%DUMMY1)
        IF(%DUMMY,GT,%R-DIST,:FIND-R)
        IF(%DUMMY,LT,%R-DIST,:R-DIST1)
        RV(9,@R1R2,1,2)
        IF(@R1R2,EQ,1,:FIND-R)
:R-DIST1
        SV(%R-DIST,%DUMMY)
        SV(@DISTC,@INDEXC)
        SV(@DISTR,@INDEXR)
        JP(:FIND-R)
:COMP
        IF(%R-TIME,LT,%WTIME,:DIST-R)
        SV(@DC,@TIMEC)
        SV(@DR,@TIMER)
        JP(:R1R2)
:DIST-R
        SV(@DC,@DISTC)
        SV(@DR,@DISTR)
        JP(:R1R2)
:R1R2
        SV(@DUMMY,@@CELL(@DC,@DR))
        IF(@DUMMY,EQ,91,:FIND-R1)
        IF(@DUMMY,EQ,92,:FIND-R2)
:FIND-R1
        SV(@CODE,1)
        JP(:END-FIND-R)
:FIND-R2
        SV(@CODE,2)
        JP(:END-FIND-R)
:END-FIND-R
        SV(@DUMMY,@@DIST(@DC,@DR))
        SV(@@CELL(@DC,@DR),@DUMMY)
        SV(%ST,%%CELL(@DC,@DR))
        SV(%%CELL(@DC,@DR),0)

        AO(@DC,*,2)
        AO(@DR,*,2)

        SV(@FIND-R,0)
     EL
;-------------------------------------------------
;-------  BL(!GO-TO) -----------------------------
    BL(!GO-TO)

        SV(@X,@C)
        SV(@Y,@R)
        SV(@DX,@DC)
        SV(@DY,@DR)
;-------- ROW ----------------
:ROW
```

```
        IF(@DY,GT,@Y,:ROW-UP)
        IF(@DY,EQ,@Y,:ROW-0)
        IF(@DY,LT,@Y,:ROW-DOWN)
:ROW-UP
         AO(@DY,-,@Y)
         SV(@MY,@DY)
         SV(@DIR,1)
         JP(:COLOUMN)
:ROW-0
         SV(@MY,0)
         SV(@DIR,0)
         JP(:COLOUMN)
:ROW-DOWN
         AO(@Y,-,@DY)
         SV(@MY,@Y)
         SV(@DIR,2)
         JP(:COLOUMN)
;-------- COLOUMN ----------
:COLOUMN
        IF(@DX,GT,@X,:COL-RIGHT)
        IF(@DX,EQ,@X,:COL-0)
        IF(@DX,LT,@X,:COL-LEFT)
:COL-RIGHT
         AO(@DX,-,@X)
         SV(@MX,@DX)
         SV(@DIC,4)
         JP(:END-ROWCOL)
:COL-0
         SV(@MX,0)
         SV(@DIC,0)
         JP(:END-ROWCOL)
:COL-LEFT
         AO(@X,-,@DX)
         SV(@MX,@X)
         SV(@DIC,3)
         JP(:END-ROWCOL)
:END-ROWCOL
 LK(!SIMU)
         IF(@FIND-R,EQ,1,:ENDLKGO-TO)
:MOVE-COL
         IF(@DIC,EQ,0,:MOVE-ROW)
         IF(@DIC,EQ,3,:MOVE-LEFT)
         IF(@DIC,EQ,4,:MOVE-RIGHT)
:MOVE-LEFT
         ML(@MX,%MTIME)
         JP(:MOVE-ROW)
:MOVE-RIGHT
         MR(@MX,%MTIME)
         JP(:MOVE-ROW)
:MOVE-ROW
         IF(@DIR,EQ,0,:ENDLKGO-TO)
         IF(@DIR,EQ,1,:MOVE-UP)
         IF(@DIR,EQ,2,:MOVE-DOWN)
```

```
:MOVE-UP
        MU(@MY,%MTIME)
        JP(:ENDLKGO-TO)
:MOVE-DOWN
        MD(@MY,%MTIME)
        JP(:ENDLKGO-TO)
:ENDLKGO-TO
      EL
;-------------------------------------------------
;--------------- BL(!SIMU) -----------------------
 BL(!SIMU)
        SV(@DUMMY1,@MX)
        SV(@DUMMY2,@MY)
        IF(@DOCK,EQ,3,:ID)
        AO(@DUMMY2,-,2)
        SV(%DUMMY1,%MOVETH)
        SV(%DUMMY2,%MOVETV)
        AO(%DUMMY1,*,@DUMMY1)
        AO(%DUMMY2,*,@DUMMY2)
        AO(@DUMMY1,+,@DUMMY2)
        AO(@DUMMY1,+,2)
        JP(:COMP-TIME)
:ID
        SV(%DUMMY1,%MOVETH)
        SV(%DUMMY2,%MOVETV)
        AO(%DUMMY1,*,@DUMMY1)
        AO(%DUMMY2,*,@DUMMY2)
        AO(@DUMMY1,+,@DUMMY2)

:COMP-TIME
        IF(@DUMMY1,EQ,0,:ENDLKSIMU)
        IF(%DUMMY1,LT,%DUMMY2,:Y_TIME)
        SV(%DUMMY,%DUMMY1)
        AO(%DUMMY1,/,@DUMMY1)
        SV(%MTIME,%DUMMY1)
        JP(:ENDLKSIMU)
:Y_TIME
        SV(%DUMMY,%DUMMY2)
        AO(%DUMMY2,/,@DUMMY1)
        SV(%MTIME,%DUMMY2)
:ENDLKSIMU
        EL
;-------------------------------------------------
;-------- BL(!MASAGE) PRINT MASAGE (S) -----------
      BL(!MESAGE)

        IF(@MESAGE,EQ,2,:PMRG)
        IF(@MESAGE,EQ,3,:PMRG)
        AO(@DC,*,2)
        AO(@DR,*,160)
        AO(*MESAGE,+,@DC)
        AO(*MESAGE,-,@DR)
        IF(@MESAGE,EQ,1,:PMS)
```

```
        PM(*MESAGE," ")
        JP(:RETURN1)
:PMS
        PM(*MESAGE,S)
        JP(:RETURN1)
:RETURN1
        AO(*MESAGE,-,@DC)
        AO(*MESAGE,+,@DR)
        AO(@DC,/,2)
        AO(@DR,/,160)
        JP(:ENDMESAGE)
:PMRG
        AO(@RC,*,2)
        AO(@RR,*,160)
        AO(*MESAGE,+,@RC)
        AO(*MESAGE,-,@RR)
    IF(@MESAGE,EQ,2,:PMR)
    IF(@MESAGE,EQ,3,:PMG)
:PMR
        PM(*MESAGE,R)
        JP(:RETURN2)
:PMG
        PM(*MESAGE,G)
:RETURN2
        AO(*MESAGE,-,@RC)
        AO(*MESAGE,+,@RR)
        AO(@RC,/,2)
        AO(@RR,/,160)
        JP(:ENDMESAGE)
:ENDMESAGE
        SV(@MESAGE,5)
      EL
;-------------------------------------------------------------
;********** END OF PROGRAM LAYOUT1.MDL*********************
```

Layout 2

```
;*********     MODEL: AUTOMATED STORAGE AND RETRIEVAL SYSTEM
;********* FILE NAME: LAYOUT2.MDL
;********* THIS PROGRAM IS FOR LAYOUT-2 (TWO DOCKS)
;*********   STORAGE: FCFS, CLOSEST OPEN LOCATION
;********* RETRIEVAL: FCFS
                     NEAREST-NEIGHBOR


;--------- PRE-SET ARRAY ---------------------------------
        SV(@AC,1)
        SV(@BC,20)
        SV(@R,0)
:SETARRAYAB
        SV(%STD,00:01.00)
        SV(@DUMMYAC,@AC)
        SV(@DUMMYR,@R)
        AO(@DUMMYAC,*,4)
        AO(@DUMMYR,*,20)
        IV(@R)
        IF(@DUMMYAC,LT,@DUMMYR,:SVDUMMYR)
        IF(@DUMMYAC,GE,@DUMMYR,:SVDUMMYAC)
:SVDUMMYAC
        AO(%STD,*,@DUMMYAC)
        SV(%%DISTA(@AC,@R),%STD)
        SV(%%DISTB(@BC,@R),%STD)
        SV(@@DA(@AC,@R),@DUMMYAC)
        SV(@@DB(@BC,@R),@DUMMYAC)
        JP(:CHECKC)
:SVDUMMYR
        AO(%STD,*,@DUMMYR)
        SV(%%DISTA(@AC,@R),%STD)
        SV(%%DISTB(@BC,@R),%STD)
        SV(@@DA(@AC,@R),@DUMMYR)
        SV(@@DB(@BC,@R),@DUMMYR)
:CHECKC
        DV(@R)
        IV(@AC)
        DV(@BC)
        IF(@AC,GT,20,:CHECKR)
        JP(:SETARRAYAB)
:CHECKR
        IF(@R,GE,4,:ENDARRAY)
        IV(@R)
        SV(@AC,1)
        SV(@BC,20)
        JP(:SETARRAYAB)
:ENDARRAY
        SV(@AC,1)
```

```
        SV(@BC,1)
        SV(@R,1)
:SETARRAYAABB
        SV(@DUMMY1,@@DA(@AC,@R))
        SV(@DUMMY2,@DUMMY1)
        SV(@DUMMY3,@@DB(@AC,@R))
        AO(@DUMMY2,+,@DUMMY3)
        AO(@DUMMY2,/,2)
        AO(@DUMMY1,+,@DUMMY2)
        SV(@@DAA(@AC,@R),@DUMMY1)
        SV(@@DA(@AC,@R),@DUMMY1)
        SV(@DUMMY1,@@DB(@BC,@R))
        AO(@DUMMY1,+,@DUMMY2)
        SV(@@DBB(@BC,@R),@DUMMY1)
        SV(@@DB(@BC,@R),@DUMMY1)
        IV(@AC)
        IV(@BC)
        IF(@AC,GT,20,:CHECKRR)
        JP(:SETARRAYAABB)
:CHECKRR
        IV(@R)
        IF(@R,GT,5,:ENDARRAYAABB)
        SV(@AC,1)
        SV(@BC,1)
        JP(:SETARRAYAABB)
:ENDARRAYAABB
;--------- PRE-SET UTILIGATION (%) ------------------------
:PRESET
        SV(@DOCK,1)
 LK(!FIND-S)
        SV(@MESAGE,1)
 LK(!MESAGE)
        IV(@UTI)
        IV(@LKHUTI)
        SV(@DOCK,2)
 LK(!FIND-S)
        SV(@MESAGE,1)
 LK(!MESAGE)
        IV(@UTI)
        IV(@LKHUTI)
        IF(@UTI,LT,@PRE-UTI,:PRESET)
;----------  RESET RANDOM SEEDS  ------------------------
RS(1,#SEEDS1)
RS(2,#SEEDS2)
RS(3,#SEEDR1)
RS(4,#SEEDR2)
RS(5,#SEEDR)
RS(9,#SEED)
WK
ER
;-------------------------------------------------------
;------------ ROUT #3 (STORAGE S1) ---------------------
      BR(3,XY(16,1),%ARRIVS1)
```

```
        RV(1,E,%ARRIVS1,%OARRIVS1)
        IF(@LKHUTI,GE,100,:END-S1)
        IF(@S1,GE,10,:END-S1)
        SV(OBJ%ST,CLOCK)
        IV(@S1)
        MD(15,0)
        SV(%S1,OBJ%ST)
        TP(*S1)
        WE
        PO(*S1)
        MA(*READY1,0)
        IV(@UTI)
        DV(@S1)
        IF(@MOVE,NE,999,WAIT)
        SV(@MOVE,0)
        IV(@THROUGHPUTS1)
        SV(OBJ%2,CLOCK)
        AO(OBJ%2,-,OBJ%ST)
;       OF(2DWT1S.RPT[,A])
;       PV(F,OBJ%2)
;       CF(RPT)
        AO(%TTL-S1,+,OBJ%2)
        IF(OBJ%2,LT,%MAXS1,:GO-ON-S1)
        SV(%MAXS1,OBJ%2)
:GO-ON-S1
        IV(@LKHUTI)
:END-S1
      ER
;-----------------------------------------------------------
;--------------- ROUT #4 (RETRIEVE S2) --------------------
      BR(4,XY(64,1),%ARRIVS2)

        RV(2,E,%ARRIVS2,%OARRIVS2)
        IF(@LKHUTI,GE,100,:END-S2)
        IF(@S2,GE,10,:END-S2)
        SV(OBJ%ST,CLOCK)
        IV(@S2)
        MD(15,0)
        SV(%S2,OBJ%ST)
        TP(*S2)
        WE
        PO(*S2)
        MA(*READY2,0)
        IV(@UTI)
        DV(@S2)
        IF(@MOVE,NE,999,WAIT)
        SV(@MOVE,0)
        SV(OBJ%2,CLOCK)
        AO(OBJ%2,-,OBJ%ST)
;       OF(2DWT2S.RPT[,A])
;       PV(F,OBJ%2)
;       CF(RPT)
```

```
        IV(@THROUGHPUTS2)
        AO(%TTL-S2,+,OBJ%2)
        IF(OBJ%2,LT,%MAXS2,:GO-ON-S2)
        SV(%MAXS2,OBJ%2)
:GO-ON-S2
        IV(@LKHUTI)
:END-S2
       ER
;-------------------------------------------------------------
;------------- ROUT# 5 (RETRIEVE R1) ----------------------
      BR(5,XY(38,20),%ARRIVR1)

        RV(3,E,%ARRIVR1,%OARRIVR1)
        IF(@LKHUTI,LE,0,:END-R1)
        IF(@R1,GE,10,:END-R1)
        SV(OBJ%ST,CLOCK)
        DV(@LKHUTI)
        IV(@R1)
 LK(!SET-R1)
        ML(15,0)
        SV(%R1,OBJ%ST)
        TP(*R1)
        WE
        PO(*R1)
        MA(*READY1,0)
        DV(@UTI)
        DV(@R1)
        SV(OBJ%1,%ST)
        IF(@MOVE,NE,999,WAIT)
        SV(@MOVE,0)
        SV(OBJ%2,CLOCK)
        AO(OBJ%2,-,OBJ%1)
;       OF(2DWT1R.RPT[,A])
;       PV(F,OBJ%2)
;       CF(RPT)
        IV(@THROUGHPUTR1)
        AO(%TTL-R1,+,OBJ%2)
        IF(OBJ%2,LT,%MAXR1,:GO-ON-R1)
        SV(%MAXR1,OBJ%2)
:GO-ON-R1
;       MD(5,%MOVET)
;       ML(15,%MOVET)
:END-R1
       ER
;-------------------------------------------------------------
;------------ ROUT# 6 (RETRIEVE R2) ----------------------
      BR(6,XY(42,20),%ARRIVR2)

        RV(4,E,%ARRIVR2,%OARRIVR2)
        IF(@LKHUTI,LE,0,:END-R2)
        IF(@R2,GE,10,:END-R2)
        SV(OBJ%ST,CLOCK)
        DV(@LKHUTI)
```

```
        IV(@R2)
 LK(!SET-R2)
        MR(15,0)
        SV(%R2,OBJ%ST)
        TP(*R2)
        WE
        PO(*R2)
        MA(*READY2,0)
        DV(@UTI)
        DV(@R2)
        SV(OBJ%1,%ST)
        IF(@MOVE,NE,999,WAIT)
        SV(@MOVE,0)
        SV(OBJ%2,CLOCK)
        AO(OBJ%2,-,OBJ%1)
;       OF(2DWT2R.RPT[,A])
;       PV(F,OBJ%2)
;       CF(RPT)
        IV(@THROUGHPUTR2)
        AO(%TTL-R2,+,OBJ%2)
        IF(OBJ%2,LT,%MAXR2,:GO-ON-R2)
        SV(%MAXR2,OBJ%2)
:GO-ON-R2
;       MD(5,%MOVET)
;       MR(15,%MOVET)
:END-R2
       ER
;-------------------------------------------------
;---------------- ROUT #2 (CRANE) ----------------
      BR(2,*DWELL,0)
;------------CRANE AT DWELL POINT-----------------
:DWELL
        SV(@CODE,0)
        IF(CLOCK,GE,%STIMER,:END-CRANE)
        SV(@DOCK,0)
        SV(@C,20)
        SV(@R,0)
        IF(@UTI,GE,100,:DW-UTI1)
        IF(@UTI,EQ,0,:DW-UTI0)
        IF(@S1,NE,0,:DWELL-DOCK12)
        IF(@S2,NE,0,:DWELL-DOCK2)
        IF(@R1,NE,0,:DWELL-R)
        IF(@R2,NE,0,:DWELL-R)
        WE
        JP(:DWELL)
:DW-UTI1
        IF(@R1,NE,0,:DWELL-R)
        IF(@R2,NE,0,:DWELL-R)
        WE
        JP(:DW-UTI1)
:DW-UTI0
        IF(@S1,NE,0,:DWELL-DOCK12)
        IF(@S2,NE,0,:DWELL-DOCK2)
```

```
        WE
        JP(:DW-UTI0)
:DWELL-DOCK12
        IF(@S2,EQ,0,:DWELL-DOCK1)
        IF(%S1,LT,%S2,:DWELL-DOCK1)
        IF(%S2,LT,%S1,:DWELL-DOCK2)
:DWELL-DOCK1
        ML(20,%MOVETH)
        JP(:DOCK1)
:DWELL-DOCK2
        MR(22,%MOVETH)
        JP(:DOCK2)
:DWELL-R
 LK(!FIND-R)
        IF(@DOCK,EQ,1,:DW-R1)
        IF(@DOCK,EQ,2,:DW-R2)
:DW-R1
        CL(*R1)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
:DW-R2
        CL(*R2)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK2)
;-------------CRANE AT DOCK 1---------------
:DOCK1
        IF(CLOCK,GE,%STIMER,:END-CRANE)
        IF(@CODE,EQ,1,:GO-ON-1)
        AO(@TTLHIT2,+,@HIT2)
        AO(@TTL2,+,1)
;       OF(2DOCK.RPT[,A])
;       PV(F,@HIT2)
;       SV(@HIT2,0)
;       CF(RPT)
:GO-ON-1
        AO(@HIT1,+,1)
        SV(@CODE,1)
        SV(@DOCK,1)
        SV(@C,0)
        SV(@R,0)
        IF(@UTI,GE,100,:DOCK1-UTI1)
```

```
        IF(@UTI,EQ,0,:DOCK1-UTI0)
        IF(@S1,NE,0,:DOCK1-S1)
        IF(@R1,NE,0,:DOCK1-R)
        IF(@R2,NE,0,:DOCK1-R)
        IF(@S2,NE,0,:DOCK1-DOCK2)
        JP(:DOCK1-DWELL)
:DOCK1-UTI1
        IF(@R1,NE,0,:DOCK1-R)
        IF(@R2,NE,0,:DOCK1-R)
        WE
        JP(:DOCK1-UTI1)
:DOCK1-UTI0
        IF(@S1,NE,0,:DOCK1-S12)
        IF(@S2,NE,0,:DOCK1-DOCK2)
        WE
        JP(:DOCK1-UTI0)
:DOCK1-S12
        IF(@S2,EQ,0,:DOCK1-S1)
        IF(%S1,LT,%S2,:DOCK1-S1)
        IF(%S2,LT,%S1,:DOCK1-DOCK2)
:DOCK1-S1
        CL(*S1)
 LK(!FIND-S)
        LK(!GO-TO)
        SV(@MESAGE,1)
 LK(!MESAGE)
        SV(@MOVE,999)
        JP(:IN-HOUSE)
:DOCK1-R
 LK(!FIND-R)
        IF(@DOCK,EQ,1,:DOCK1-R1)
        IF(@DOCK,EQ,2,:DOCK1-R2)
:DOCK1-R1
        CL(*R1)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
:DOCK1-R2
        CL(*R2)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
        SV(@DOCK,2)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK2)
```

```
:DOCK1-DOCK2
        MR(42,%MOVETH)
        JP(:DOCK2)
:DOCK1-DWELL
        MR(20,%MOVETH)
        JP(:DWELL)
;------------ CRANE AT DOCK 2-----------------------
:DOCK2
        IF(CLOCK,GE,%STIMER,:END-CRANE)
        IF(@CODE,EQ,2,:GO-ON-2)
        AO(@TTLHIT1,+,@HIT1)
        AO(@TTL1,+,1)
;       OF(1DOCK.RPT[,A])
;       PV(F,@HIT1)
;       SV(@HIT1,0)
;       CF(RPT)
:GO-ON-2
        AO(@HIT2,+,1)
        SV(@CODE,2)
        SV(@DOCK,2)
        SV(@C,42)
        SV(@R,0)
        IF(@UTI,GE,100,:DOCK2-UTI1)
        IF(@UTI,EQ,0,:DOCK2-UTI0)
        IF(@S2,NE,0,:DOCK2-S2)
        IF(@R1,NE,0,:DOCK2-R)
        IF(@R2,NE,0,:DOCK2-R)
        IF(@S1,NE,0,:DOCK2-DOCK1)
        JP(:DOCK2-DWELL)
:DOCK2-UTI1
        IF(@R1,NE,0,:DOCK1-R)
        IF(@R2,NE,0,:DOCK1-R)
        WE
        JP(:DOCK2-UTI1)
:DOCK2-UTI0
        IF(@S2,NE,0,:DOCK2-S12)
        IF(@S1,NE,0,:DOCK2-DOCK1)
        WE
        JP(:DOCK2-UTI0)
:DOCK2-S12
        IF(@S1,EQ,0,:DOCK2-S2)
        IF(%S1,LT,%S2,:DOCK2-DOCK1)
        IF(%S2,LT,%S1,:DOCK2-S2)
:DOCK2-S2
        CL(*S2)
 LK(!FIND-S)
 LK(!GO-TO)
        SV(@MESAGE,1)
 LK(!MESAGE)
        SV(@MOVE,999)
        JP(:IN-HOUSE)
:DOCK2-R
 LK(!FIND-R)
```

```
        IF(@DOCK,EQ,1,:DOCK2-R1)
        IF(@DOCK,EQ,2,:DOCK2-R2)
:DOCK2-R1
        CL(*R1)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
        SV(@DOCK,1)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
:DOCK2-R2
        CL(*R2)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK2)
:DOCK2-DOCK1
        ML(42,%MOVETH)
        JP(:DOCK1)
:DOCK2-DWELL
        ML(22,%MOVETH)
        JP(:DWELL)
;-------------------------------------------------
;-------------- CRANE AT IN HOUSE--------------------
:IN-HOUSE
        IF(CLOCK,GE,%STIMER,:END-CRANE)
        SV(@DOCK,3)
        SV(@C,@DC)
        SV(@R,@DR)
        IF(@UTI,GE,100,:HOUSE-UTI1)
        IF(@UTI,EQ,0,:HOUSE-UTI0)
        IF(@R1,NE,0,:HOUSE-R)
        IF(@R2,NE,0,:HOUSE-R)
        IF(@S1,NE,0,:HOUSE-S12)
        IF(@S2,NE,0,:HOUSE-S2)
        JP(:HOUSE-DW)
:HOUSE-S12
        IF(@S2,EQ,0,:HOUSE-S1)
        IF(%S1,LT,%S2,:HOUSE-S1)
        IF(%S2,LT,%S1,:HOUSE-S2)
:HOUSE-UTI1
        IF(@R1,NE,0,:HOUSE-R)
        IF(@R2,NE,0,:HOUSE-R)
        WE
        JP(:HOUSE-UTI1)
:HOUSE-UTI0
```

```
        IF(@S1,NE,0,:HOUSE-S12)
        IF(@S2,NE,0,:HOUSE-S2)
        WE
        JP(:HOUSE-UTIO)
:HOUSE-S1
        SV(@DOCK,1)
 LK(!GO-DOCK)
        JP(:DOCK1)
:HOUSE-S2
        SV(@DOCK,2)
 LK(!GO-DOCK)
        JP(:DOCK2)
:HOUSE-R
 LK(!FIND-R)
        IF(@DOCK,EQ,1,:HOUSE-R1)
        IF(@DOCK,EQ,2,:HOUSE-R2)
:HOUSE-R1
        SV(@DOCK,3)
        CL(*R1)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
        SV(@DOCK,1)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
:HOUSE-R2
        CL(*R2)
        SV(@DOCK,3)
 LK(!GO-TO)
        SV(@MESAGE,4)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
        SV(@DOCK,2)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK2)
:HOUSE-DW
        SV(@DC,20)
        SV(@DR,0)
 LK(!GO-TO)
        JP(:DWELL)
;-------------------------------------------------
:END-CRANE
;  OF(2DWT1S.RPT[,A])
;  PV(F,9999:00:00.00)
;  OF(2DWT2S.RPT[,A])
;  PV(F,9999:00:00.00)
;  OF(2DWT1R.RPT[,A])
;  PV(F,9999:00:00.00)
```

```
;  OF(2DWT2R.RPT[,A])
;  PV(F,9999:00:00.00)
;  OF(1DOCK.RPT[,A])
;  PV(F,@HIT1)
;  PV(F,9999)
;  OF(2DOCK.RPT[,A])
;  PV(F,@HIT2)
;  PV(F,9999)
;  CF(RPT)
   SV(%AVG-S1,%TTL-S1)
   AO(%AVG-S1,/,@THROUGHPUTS1)
   SV(%AVG-S2,%TTL-S2)
   AO(%AVG-S2,/,@THROUGHPUTS2)
   SV(%AVG-R1,%TTL-R1)
   AO(%AVG-R1,/,@THROUGHPUTR1)
   SV(%AVG-R2,%TTL-R2)
   AO(%AVG-R2,/,@THROUGHPUTR2)
  ER
;-------------------------------------------------
;------------ BL(!GO-DOCK) ------------------------
   BL(!GO-DOCK)
        IF(@DOCK,EQ,1,:GO-DOCK1)
        IF(@DOCK,EQ,2,:GO-DOCK2)
:GO-DOCK1
        SV(@DC,0)
        SV(@DR,0)
 LK(!GO-TO)
        JP(:ENDGO-DOCK)
:GO-DOCK2
        SV(@DC,42)
        SV(@DR,0)
 LK(!GO-TO)
:ENDGO-DOCK
     EL
;-------------------------------------------------
;--------- BL(!FINDS) FIND CLOSET CELL FOR STORAGE --
      BL(!FIND-S)
:FIND-S
        IF(@DOCK,EQ,1,:FIND-SA)
        IF(@DOCK,EQ,2,:FIND-SB)
:FIND-SA
        SV(@DUMMY1,990)
        SV(@INDEXR,0)
        SV(@INDEXC,0)
:FIND-SAA
        IF(@INDEXR,EQ,5,:RESET-S)
        IV(@INDEXR)
        SV(@INDEXC,0)
:FIND-SAAA
        IF(@INDEXC,EQ,20,:FIND-SAA)
        IV(@INDEXC)
        SV(@DUMMY,@@DAA(@INDEXC,@INDEXR))
        IF(@DUMMY,GE,@DUMMY1,:FIND-SAAA)
```

```
        SV(@DUMMY1,@DUMMY)
        SV(@SC,@INDEXC)
        SV(@SR,@INDEXR)
        JP(:FIND-SAAA)
:FIND-SB
        SV(@DUMMY,0)
        SV(@DUMMY1,990)
        SV(@INDEXR,0)
        SV(@INDEXC,21)
:FIND-SBB
        IF(@INDEXR,EQ,5,:RESET-S)
        IV(@INDEXR)
        SV(@INDEXC,21)
:FIND-SBBB
        IF(@INDEXC,EQ,1,:FIND-SBB)
        DV(@INDEXC)
        SV(@DUMMY,@@DBB(@INDEXC,@INDEXR))
        IF(@DUMMY,GE,@DUMMY1,:FIND-SBBB)
        SV(@DUMMY1,@DUMMY)
        SV(@SC,@INDEXC)
        SV(@SR,@INDEXR)
        JP(:FIND-SBBB)
:RESET-S
        SV(@@CELL(@SC,@SR),999)
        SV(@@DAA(@SC,@SR),999)
        SV(@@DBB(@SC,@SR),999)
:ENDFIND-S
        AO(@SC,*,2)
        AO(@SR,*,2)
        SV(@DC,@SC)
        SV(@DR,@SR)
     EL
;-------------------------------------------------
;-------BL(!SET-R1) FIND WILL RETRIEVED CELL R2 ------
     BL(!SET-R1)
:SET-R1
        RV(5,@SETR1C,1,20)
        RV(5,@SETR1R,1,5)
        SV(@SET1,@@CELL(@SETR1C,@SETR1R))
        IF(@SET1,NE,999,:SET-R1)
        SV(@@CELL(@SETR1C,@SETR1R),991)
        SV(%%CELL(@SETR1C,@SETR1R),OBJ%ST)

        SV(@RC,@SETR1C)
        SV(@RR,@SETR1R)
        AO(@RC,*,2)
        AO(@RR,*,2)
        SV(@MESAGE,2)
 LK(!MESAGE)
     EL
;-------------------------------------------------
;------BL(!SET-R2) FIND WILL RETRIEVED CELL R2 -------
     BL(!SET-R2)
```

```
:SET-R2
        RV(5,@SETR2C,1,20)
        RV(5,@SETR2R,1,5)
        SV(@SET2,@@CELL(@SETR2C,@SETR2R))
        IF(@SET2,NE,999,:SET-R2)
        SV(@@CELL(@SETR2C,@SETR2R),992)
        SV(%%CELL(@SETR2C,@SETR2R),OBJ%ST)
        SV(@RC,@SETR2C)
        SV(@RR,@SETR2R)
        AO(@RC,*,2)
        AO(@RR,*,2)
        SV(@MESAGE,3)
 LK(!MESAGE)
     EL
;-------------------------------------------------------
;------BL(!FIND-R) FIND RETRIEVE CELL ----------------
     BL(!FIND-R)
        SV(@FIND-R,1)
        SV(%TIMER-R1,0)
        SV(%DIST-R1,999:00:00.00)
        SV(%TIMER-R2,0)
        SV(%DIST-R2,999:00:00.00)
        SV(@INDEXC,0)
        SV(@INDEXR,1)
:FIND-R
        SV(%TIMER,CLOCK)
        IV(@INDEXC)
        IF(@INDEXC,GT,20,:CHECK-ROW)
        SV(@DUMMY,@@CELL(@INDEXC,@INDEXR))
        IF(@DUMMY,EQ,991,:FIND-R1)
        IF(@DUMMY,EQ,992,:FIND-R2)
        JP(:FIND-R)
:FIND-R1
        SV(%DUMMY,%%CELL(@INDEXC,@INDEXR))
        AO(%TIMER,-,%DUMMY)
        IF(%TIMER,LT,%TIMER-R1,:DIST-R1)
        SV(%TIMER-R1,%TIMER)
        SV(@TIMER-1C,@INDEXC)
        SV(@TIMER-1R,@INDEXR)
:DIST-R1
        SV(@DC,@INDEXC)
        SV(@DR,@INDEXR)
        AO(@DC,*,2)
        AO(@DR,*,2)
 LK(!GO-TO)
        SV(%DUMMY1,%%DISTA(@INDEXC,@INDEXR))
        AO(%DUMMY,+,%DUMMY1)
        IF(%DUMMY,GT,%DIST-R1,:FIND-R)
        IF(%DUMMY,LT,%DIST-R1,:FIND-R11)
        RV(9,@R1R2,1,2)
        IF(@R1R2,EQ,1,:FIND-R)
:FIND-R11
        SV(%DIST-R1,%DUMMY)
```

```
        SV(@DIST-1C,@INDEXC)
        SV(@DIST-1R,@INDEXR)
        JP(:FIND-R)
:FIND-R2
        SV(%DUMMY,%%CELL(@INDEXC,@INDEXR))
        AO(%TIMER,-,%DUMMY)
        IF(%TIMER,LT,%TIMER-R2,:DIST-R2)
        SV(%TIMER-R2,%TIMER)
        SV(@TIMER-2C,@INDEXC)
        SV(@TIMER-2R,@INDEXR)
:DIST-R2
        SV(@DC,@INDEXC)
        SV(@DR,@INDEXR)
        AO(@DC,*,2)
        AO(@DR,*,2)
   LK(!GO-TO)
        SV(%DUMMY1,%%DISTB(@INDEXC,@INDEXR))
        AO(%DUMMY,+,%DUMMY1)
        IF(%DUMMY,GT,%DIST-R2,:FIND-R)
        IF(%DUMMY,LT,%DIST-R2,:FIND-R22)
        RV(9,@R1R2,1,2)
        IF(@R1R2,EQ,1,:FIND-R)
:FIND-R22
        SV(%DIST-R2,%DUMMY)
        SV(@DIST-2C,@INDEXC)
        SV(@DIST-2R,@INDEXR)
        JP(:FIND-R)
:CHECK-ROW
        SV(@INDEXC,0)
        IV(@INDEXR)
        IF(@INDEXR,GT,5,:SELECT-R)
        JP(:FIND-R)
:SELECT-R
        IF(%TIMER-R1,GT,%WTIME,:HOT-R)
        IF(%TIMER-R2,GT,%WTIME,:HOT-R)
        IF(%DIST-R1,LT,%DIST-R2,:SET-DISTR1)
        IF(%DIST-R2,LT,%DIST-R1,:SET-DISTR2)
        RV(9,@DUMMY,1,2)
        IF(@DUMMY,EQ,1,:SET-DISTR1)
        IF(@DUMMY,EQ,2,:SET-DISTR2)
:HOT-R
        IF(%TIMER-R1,GT,%TIMER-R2,:SET-HOT-R1)
        IF(%TIMER-R2,GT,%TIMER-R1,:SET-HOT-R2)
:SET-HOT-R1
        SV(@DOCK,1)
        SV(@DC,@TIMER-1C)
        SV(@DR,@TIMER-1R)
        JP(:ENDFIND-R)
:SET-HOT-R2
        SV(@DOCK,2)
        SV(@DC,@TIMER-2C)
        SV(@DR,@TIMER-2R)
        JP(:ENDFIND-R)
```

```
:SET-DISTR1
        SV(@DOCK,1)
        SV(@DC,@DIST-1C)
        SV(@DR,@DIST-1R)
        JP(:ENDFIND-R)
:SET-DISTR2
        SV(@DOCK,2)
        SV(@DC,@DIST-2C)
        SV(@DR,@DIST-2R)
:ENDFIND-R
        SV(@@CELL(@DC,@DR),0)
        SV(%ST,%%CELL(@DC,@DR))
        SV(%%CELL(@DC,@DR),0)
        SV(@DUMMY,@@DA(@DC,@DR))
        SV(@@DAA(@DC,@DR),@DUMMY)
        SV(@DUMMY,@@DB(@DC,@DR))
        SV(@@DBB(@DC,@DR),@DUMMY)
        AO(@DC,*,2)
        AO(@DR,*,2)
        SV(@FIND-R,0)
     EL
;-----------------------------------------------
;-------  BL(!GOTO) ----------------------------
    BL(!GO-TO)
        SV(@X,@C)
        SV(@Y,@R)
        SV(@DX,@DC)
        SV(@DY,@DR)
;-------- ROW -----------------
        IF(@DY,GT,@Y,:ROW-UP)
        IF(@DY,EQ,@Y,:ROW-0)
        IF(@DY,LT,@Y,:ROW-DOWN)
:ROW-UP
        AO(@DY,-,@Y)
        SV(@MY,@DY)
        SV(@DIR,1)
        JP(:COLOUMN)
:ROW-0
        SV(@MY,0)
        SV(@DIR,0)
        JP(:COLOUMN)
:ROW-DOWN
        AO(@Y,-,@DY)
        SV(@MY,@Y)
        SV(@DIR,2)
;--------  COLOUMN ----------
:COLOUMN
        IF(@DX,GT,@X,:COL-RIGHT)
        IF(@DX,EQ,@X,:COL-0)
        IF(@DX,LT,@X,:COL-LEFT)
:COL-RIGHT
        AO(@DX,-,@X)
        SV(@MX,@DX)
```

```
        SV(@DIC,4)
        JP(:ENDROW-COL)
:COL-0
        SV(@MX,0)
        SV(@DIC,0)
        JP(:ENDROW-COL)
:COL-LEFT
        AO(@X,-,@DX)
        SV(@MX,@X)
        SV(@DIC,3)
:ENDROW-COL
 LK(!SIMU)
IF(@FIND-R,EQ,1,:END-MOVE)
;--------- MOVE-COL -----------
        IF(@DIC,EQ,0,:MOVE-ROW)
        IF(@DIC,EQ,3,:MOVE-LEFT)
        IF(@DIC,EQ,4,:MOVE-RIGHT)
:MOVE-LEFT
        ML(@MX,%MTIME)
        JP(:MOVE-ROW)
:MOVE-RIGHT
        MR(@MX,%MTIME)
;--------- MOVE-ROW ------------
:MOVE-ROW
        IF(@DIR,EQ,0,:END-MOVE)
        IF(@DIR,EQ,1,:MOVE-UP)
        MD(@MY,%MTIME)
        JP(:END-MOVE)
:MOVE-UP
        MU(@MY,%MTIME)
:END-MOVE
      EL
;-------------------------------------------------
;--------------- BL(!SIMU) -----------------------
      BL(!SIMU)
        SV(@DUMMY1,@MX)
        SV(@DUMMY2,@MY)
        IF(@DOCK,EQ,3,:ID1)
        AO(@DUMMY2,-,2)
:ID1
        SV(%DUMMY1,%MOVETH)
        SV(%DUMMY2,%MOVETV)
        AO(%DUMMY1,*,@DUMMY1)
        AO(%DUMMY2,*,@DUMMY2)
        AO(@DUMMY1,+,@DUMMY2)
        IF(@DUMMY1,EQ,0,:ENDLKSIMU)
        IF(@DOCK,EQ,3,:ID2)
        AO(@DUMMY1,+,2)
:ID2
        IF(%DUMMY1,LT,%DUMMY2,:Y-TIME)
:X-TIME
        SV(%DUMMY,%DUMMY1)          ;FOR NEAREST NEIGHBOR
        AO(%DUMMY1,/,@DUMMY1)
```

```
        SV(%MTIME,%DUMMY1)
        JP(:ENDLKSIMU)
:Y-TIME
        SV(%DUMMY,%DUMMY2)          ;FOR NEAREST NEIGHBOR
        AO(%DUMMY2,/,@DUMMY1)
        SV(%MTIME,%DUMMY2)
:ENDLKSIMU
        EL
;-------------------------------------------------
;---------- BL(!MASAGE) PRINT MASAGE  -------------
      BL(!MESAGE)
        IF(@MESAGE,EQ,2,:PMRG)
        IF(@MESAGE,EQ,3,:PMRG)
        AO(@DC,*,2)
        AO(@DR,*,160)
        AO(*MESAGE,+,@DC)
        AO(*MESAGE,-,@DR)
        IF(@MESAGE,EQ,4,:PM"")
:PMS
        PM(*MESAGE,S)
        JP(:RETURN1)
:PM""
        PM(*MESAGE," ")
:RETURN1
        AO(*MESAGE,-,@DC)
        AO(*MESAGE,+,@DR)
        AO(@DC,/,2)
        AO(@DR,/,160)
        JP(:ENDMESAGE)
:PMRG
        AO(@RC,*,2)
        AO(@RR,*,160)
        AO(*MESAGE,+,@RC)
        AO(*MESAGE,-,@RR)
        IF(@MESAGE,EQ,3,:PMG)
:PMR
        PM(*MESAGE,R)
        JP(:RETURN2)
:PMG
        PM(*MESAGE,G)
:RETURN2
        AO(*MESAGE,-,@RC)
        AO(*MESAGE,+,@RR)
        AO(@RC,/,2)
        AO(@RR,/,160)
:ENDMESAGE
        SV(@MESAGE,5)
      EL
;-------------------------------------------------------
;******** END OF PROGRAM LAYOUT2.MDL **************************
```

Layout 3

```
;*********      MODEL:   AUTOMATED STORAGE AND RETRIEVAL SYSTEM
;********* FILE NAME:   LAYOUT3.MDL
;********* THIS PROGRAM IS FOR LAYOUT-3
;*********   STORAGE:   FCFS, CLOSEST OPEN LOCATION
;********* RETRIEVAL:   FCFS
;*********              NEAREST-NEIGHBOR

;------------ PRE-SET ARRAY -------------------------
        SV(@AC,1)
        SV(@BC,20)
        SV(@R,0)
:SETARRAYAB
        SV(%STD,00:01.00)
        SV(@DUMMYAC,@AC)
        SV(@DUMMYR,@R)
        AO(@DUMMYAC,*,4)
        AO(@DUMMYR,*,20)
        IV(@R)
        IF(@DUMMYAC,LT,@DUMMYR,:SVDUMMYR)
        IF(@DUMMYAC,GE,@DUMMYR,:SVDUMMYAC)
:SVDUMMYAC
        SV(@DUMMY,@DUMMYAC)
        JP(:PRE-SET)
:SVDUMMYR
        SV(@DUMMY,@DUMMYR)
:PRE-SET
        AO(%STD,*,@DUMMY)
        SV(%%DISTA(@AC,@R),%STD)
        SV(%%DISTB(@BC,@R),%STD)
        SV(@@DA(@AC,@R),@DUMMY)
        SV(@@DAA(@AC,@R),@DUMMY)
        SV(@@DB(@BC,@R),@DUMMY)
        SV(@@DBB(@BC,@R),@DUMMY)
        DV(@R)
        IV(@AC)
        DV(@BC)
        IF(@AC,GT,20,:CHECKR)
        JP(:SETARRAYAB)
:CHECKR
        IF(@R,GE,4,:ENDARRAY)
        IV(@R)
        SV(@AC,1)
        SV(@BC,20)
        JP(:SETARRAYAB)
:ENDARRAY
;-----------------------------------------------------
;----------PRE-SET UTILIGATION (%)-------------------
:PRESET-S2
```

```
 LK(!FIND-S2)
        SV(@MESAGE,2)
 LK(!MESAGE)
        IV(@UTI)
        IV(@UTIS2)
        IF(@UTI,GE,@PRE-UTI,:END-PRESET)
:PRESET-S1
 LK(!FIND-S1)
        SV(@MESAGE,1)
 LK(!MESAGE)
        IV(@UTI)
        IV(@UTIS1)
        IF(@UTI,LT,@PRE-UTI,:PRESET-S2)
:END-PRESET
;---------- PRE-SET RANDOM VARIABLE SEEDS -------------
RS(1,#SEEDS1)
RS(2,#SEEDS2)
RS(3,#SEEDR1)
RS(4,#SEEDR2)
RS(5,#SEEDR)
RS(9,#SEED)
WK
ER
;-----------------------------------------------------
;------------- ROUT #3 (STORAGE S1) -------------------
      BR(3,XY(16,2),%ARRIVS1)

        RV(1,E,%ARRIVS1,%OARRIVS1)
        IF(@UTI,GE,100,:END-S1)
        IF(@S1,GE,10,:END-S1)
        SV(OBJ%ST,CLOCK)
        IV(@S1)
        MD(14,0)
        SV(%S1,OBJ%ST)
        TP(*S1)
        WE
        PO(*S1)
        MA(*READY1,0)
        IV(@UTI)
        DV(@S1)
        IF(@MOVE,NE,999,WAIT)
        SV(@MOVE,0)
        SV(OBJ%2,CLOCK)
        AO(OBJ%2,-,OBJ%ST)
;       OF(2DWT1S.RPT[,A])
;       PV(F,OBJ%2)
;       CF(RPT)
        AO(%TTL-S1,+,OBJ%2)
        IF(OBJ%2,LT,%MAXS1,:GO-ON-S1)
        SV(%MAXS1,OBJ%2)
:GO-ON-S1
        IV(@UTIS1)
        IV(@THROUGHPUTS1)
```

```
:END-S1
        ER
;-------------------------------------------------------
;---------------- ROUT #4 (RETRIEVE S2) ----------------
      BR(4,XY(64,2),%ARRIVS2)

        RV(2,E,%ARRIVS2,%OARRIVS2)
        IF(@UTI,GE,100,:END-S2)
        IF(@S2,GE,10,:END-S2)
        SV(OBJ%ST,CLOCK)
        IV(@S2)
        RV(2,E,%ARRIVS2,%OARRIVS2)
        MD(14,0)
        SV(%S2,OBJ%ST)
        TP(*S2)
        WE
        PO(*S2)
        MA(*READY2,0)
        IV(@UTI)
        DV(@S2)
        IF(@MOVE,NE,999,WAIT)
        SV(@MOVE,0)
        SV(OBJ%2,CLOCK)
        AO(OBJ%2,-,OBJ%ST)
;       OF(2DWT2S.RPT[,A])
;       PV(F,OBJ%2)
;       CF(RPT)
        AO(%TTL-S2,+,OBJ%2)
        IF(OBJ%2,LT,%MAXS2,:GO-ON-S2)
        SV(%MAXS2,OBJ%2)
:GO-ON-S2
        IV(@UTIS2)
        IV(@THROUGHPUTS2)
:END-S2
        ER
;-------------------------------------------------------
;------------------ ROUT# 5 (RETRIEVE R1) --------------
      BR(5,XY(37,20),%ARRIVR1)

        RV(3,E,%ARRIVR1,%OARRIVR1)
        IF(@UTIS1,LE,0,:END-R1)
        IF(@R1,GE,10,:END-R1)
        SV(OBJ%ST,CLOCK)
 LK(!SET-R1)
        DV(@UTIS1)
        IV(@R1)
        ML(14,0)
        SV(%R1,OBJ%ST)
        TP(*R1)
        WE
        PO(*R1)
        MA(*READY1,0)
        DV(@UTI)
```

```
        DV(@R1)
        SV(OBJ%1,%ST)
        IF(@MOVE,NE,999,WAIT)
        SV(@MOVE,0)
        SV(OBJ%2,CLOCK)
        AO(OBJ%2,-,OBJ%1)
;       OF(2DWT1R.RPT[,A])
;       PV(F,OBJ%2)
;       CF(RPT)
        AO(%TTL-R1,+,OBJ%2)
        IF(OBJ%2,LT,%MAXR1,:GO-ON-R1)
        SV(%MAXR1,OBJ%2)
:GO-ON-R1
;       MD(5,%MOVET)
;       ML(15,%MOVET)
        IV(@THROUGHPUTR1)
:END-R1
        ER
;-------------------------------------------------------
;------------------- ROUT# 6 (RETRIEVE R2) -------------
      BR(6,XY(43,20),%ARRIVR2)

        RV(4,E,%ARRIVR2,%OARRIVR2)
        IF(@UTIS2,LE,0,:END-R2)
        IF(@R2,GE,10,:END-R2)
        SV(OBJ%ST,CLOCK)
        DV(@UTIS2)
        IV(@R2)
 LK(!SET-R2)
        MR(14,0)
        SV(%R2,OBJ%ST)
        TP(*R2)
        WE
        PO(*R2)
        MA(*READY2,0)
        DV(@UTI)
        DV(@R2)
        SV(OBJ%1,%ST)
        IF(@MOVE,NE,999,WAIT)
        SV(@MOVE,0)
        SV(OBJ%2,CLOCK)
        AO(OBJ%2,-,OBJ%1)
;       OF(2DWT2R.RPT[,A])
;       PV(F,OBJ%2)
;       CF(RPT)
        AO(%TTL-R2,+,OBJ%2)
        IF(OBJ%2,LT,%MAXR2,:GO-ON-R2)
        SV(%MAXR2,OBJ%2)
:GO-ON-R2
;       MD(5,%MOVET)
;       MR(15,%MOVET)
        IV(@THROUGHPUTR2)
:END-R2
```

```
        ER
;-----------------------------------------------------
;---------------- ROUT #2 (CRANE) --------------------
      BR(2,*DWELL,0)
;---------------- CRANE AT DWELL POINT-----------------
:DWELL
        IF(CLOCK,GE,%STIME,:END-CRANE)
        SV(@CODE,0)
        SV(@DOCK,0)
        SV(@C,20)
        SV(@R,0)
        IF(@UTI,GE,100,:DW-UTI1)
        IF(@UTI,EQ,0,:DW-UTI0)
        IF(@S1,NE,0,:DWELL-DOCK12)
        IF(@S2,NE,0,:DWELL-DOCK2)
        IF(@R1,NE,0,:DWELL-R)
        IF(@R2,NE,0,:DWELL-R)
        WE
        JP(:DWELL)
:DW-UTI1
        IF(@R1,NE,0,:DWELL-R)
        IF(@R2,NE,0,:DWELL-R)
        WE
        JP(:DW-UTI1)
:DW-UTI0
        IF(@S1,NE,0,:DWELL-DOCK12)
        IF(@S2,NE,0,:DWELL-DOCK2)
        WE
        JP(:DW-UTI0)
:DWELL-DOCK12
        IF(@S2,EQ,0,:DWELL-DOCK1)
        IF(%S1,LT,%S2,:DWELL-DOCK1)
        IF(%S2,LT,%S1,:DWELL-DOCK2)
:DWELL-DOCK1
        ML(20,%MOVETH)
        SV(@CODE,1)
        JP(:DOCK1)
:DWELL-DOCK2
        MR(22,%MOVETH)
        SV(@CODE,2)
        JP(:DOCK2)
:DWELL-R
 LK(!FIND-R)
        IF(@DOCK,EQ,1,:DW-R1)
        IF(@DOCK,EQ,2,:DW-R2)
:DW-R1
        CL(*R1)
 LK(!GO-TO)
        SV(@MESAGE,5)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
 LK(!GO-DOCK)
```

```
        SV(@MOVE,999)
        JP(:DOCK1)
:DW-R2
        CL(*R2)
 LK(!GO-TO)
        SV(@MESAGE,5)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK2)
;---------------  CARNE AT DOCK 1 --------------------
:DOCK1
        IF(CLOCK,GE,%STIME,:END-CRANE)
;       IF(@CODE,EQ,1,:GO-ON-1)
;       AO(@TTLHIT2,+,@HIT2)
;       AO(@TTL2,+,1)
;       OF(2DOCK.RPT[,A])
;       PV(F,@HIT2)
;       SV(@HIT2,0)
;       CF(RPT)
;:GO-ON-1
;       AO(@HIT1,+,1)
;       SV(@CODE,1)
        SV(@DOCK,1)
        SV(@C,0)
        SV(@R,0)
        IF(@UTI,GE,100,:DOCK1-UTI1)
        IF(@UTI,EQ,0,:DOCK1-UTI0)

        IF(@S1,NE,0,:DOCK1-S1)
        IF(@R1,NE,0,:DOCK1-R)
        IF(@R2,NE,0,:DOCK1-R)
        IF(@S2,NE,0,:DOCK1-DOCK2)
        JP(:DOCK1-DWELL)
:DOCK1-UTI1
        IF(@R1,NE,0,:DOCK1-R)
        IF(@R2,NE,0,:DOCK1-R)
        WE
        JP(:DOCK1-UTI1)
:DOCK1-UTI0
        IF(@S1,NE,0,:DOCK1-S12)
        IF(@S2,NE,0,:DOCK1-DOCK2)
        WE
        JP(:DOCK1-UTI0)
:DOCK1-S12
        IF(@S2,EQ,0,:DOCK1-S1)
        IF(%S1,LT,%S2,:DOCK1-S1)
        IF(%S2,LT,%S1,:DOCK1-DOCK2)
:DOCK1-S1
        CL(*S1)
 LK(!FIND-S1)
```

```
 LK(!GO-TO)
        SV(@MESAGE,1)
 LK(!MESAGE)
        SV(@MOVE,999)
        JP(:IN-HOUSE)
:DOCK1-R
 LK(!FIND-R)
        IF(@DOCK,EQ,1,:DOCK1-R1)
        IF(@DOCK,EQ,2,:DOCK1-R2)
:DOCK1-R1
        CL(*R1)
 LK(!GO-TO)
        SV(@MESAGE,5)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
:DOCK1-R2
        CL(*R2)
 LK(!GO-TO)
        SV(@MESAGE,5)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
        SV(@DOCK,2)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK2)
:DOCK1-DOCK2
        MR(42,%MOVETH)
        JP(:DOCK2)
:DOCK1-DWELL
        MR(20,%MOVETH)
        JP(:DWELL)
;----------------  CRANE AT DOCK 2 --------------------
:DOCK2
        IF(CLOCK,GE,%STIME,:END-CRANE)
;       IF(@CODE,EQ,2,:GO-ON-2)
;       AO(@TTLHIT1,+,@HIT1)
;       AO(@TTL1,+,1)
;       OF(1DOCK.RPT[,A])
;       PV(F,@HIT1)
;       SV(@HIT1,0)
;       CF(RPT)
;:GO-ON-2
;       AO(@HIT2,+,1)
;       SV(@CODE,2)
        SV(@DOCK,2)
        SV(@C,42)
        SV(@R,0)
        IF(@UTI,GE,100,:DOCK2-UTI1)
```

```
        IF(@UTI,EQ,0,:DOCK2-UTI0)
        IF(@S2,NE,0,:DOCK2-S2)
        IF(@R1,NE,0,:DOCK2-R)
        IF(@R2,NE,0,:DOCK2-R)
        IF(@S1,NE,0,:DOCK2-DOCK1)
        JP(:DOCK2-DWELL)
:DOCK2-UTI1
        IF(@R1,NE,0,:DOCK1-R)
        IF(@R2,NE,0,:DOCK1-R)
        WE
        JP(:DOCK2-UTI1)
:DOCK2-UTI0
        IF(@S2,NE,0,:DOCK2-S12)
        IF(@S1,NE,0,:DOCK2-DOCK1)
        WE
        JP(:DOCK2-UTI0)
:DOCK2-S12
        IF(@S1,EQ,0,:DOCK2-S2)
        IF(%S1,LT,%S2,:DOCK2-DOCK1)
        IF(%S2,LT,%S1,:DOCK2-S2)
:DOCK2-S2
        CL(*S2)
 LK(!FIND-S2)
 LK(!GO-TO)
        SV(@MESAGE,2)
 LK(!MESAGE)
        SV(@MOVE,999)
        JP(:IN-HOUSE)
:DOCK2-R
 LK(!FIND-R)
        IF(@DOCK,EQ,1,:DOCK2-R1)
        IF(@DOCK,EQ,2,:DOCK2-R2)
:DOCK2-R1
        CL(*R1)
 LK(!GO-TO)
        SV(@MESAGE,5)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
        SV(@DOCK,1)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
:DOCK2-R2
        CL(*R2)
 LK(!GO-TO)
        SV(@MESAGE,5)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK2)
```

```
:DOCK2-DOCK1
        ML(42,%MOVETH)
        JP(:DOCK1)
:DOCK2-DWELL
        ML(22,%MOVETH)
        JP(:DWELL)
;-----------------------------------------------------
;-------------  CRANE AT IN-HOUSE --------------------
:IN-HOUSE
        IF(CLOCK,GE,%STIME,:END-CRANE)
        SV(@DOCK,3)
        SV(@C,@DC)
        SV(@R,@DR)
        IF(@UTI,GE,100,:HOUSE-UTI1)
        IF(@UTI,EQ,0,:HOUSE-UTI0)
        IF(@R1,NE,0,:HOUSE-R)
        IF(@R2,NE,0,:HOUSE-R)
        IF(@S1,NE,0,:HOUSE-S12)
        IF(@S2,NE,0,:HOUSE-S2)
        JP(:HOUSE-DW)
:HOUSE-S12
        IF(@S2,EQ,0,:HOUSE-S1)
        IF(%S1,LT,%S2,:HOUSE-S1)
        IF(%S2,LT,%S1,:HOUSE-S2)
:HOUSE-UTI1
        IF(@R1,NE,0,:HOUSE-R)
        IF(@R2,NE,0,:HOUSE-R)
        WE
        JP(:HOUSE-UTI1)
:HOUSE-UTI0
        IF(@S1,NE,0,:HOUSE-S12)
        IF(@S2,NE,0,:HOUSE-S2)
        WE
        JP(:HOUSE-UTI0)
:HOUSE-S1
        SV(@DOCK,1)
 LK(!GO-DOCK)
        JP(:DOCK1)
:HOUSE-S2
        SV(@DOCK,2)
 LK(!GO-DOCK)
        JP(:DOCK2)
:HOUSE-R
 LK(!FIND-R)
        IF(@DOCK,EQ,1,:HOUSE-R1)
        IF(@DOCK,EQ,2,:HOUSE-R2)
:HOUSE-R1
        SV(@DOCK,3)
        CL(*R1)
 LK(!GO-TO)
        SV(@MESAGE,5)
 LK(!MESAGE)
        SV(@C,@DC)
```

```
        SV(@R,@DR)
        SV(@DOCK,1)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK1)
:HOUSE-R2
        CL(*R2)
        SV(@DOCK,3)
 LK(!GO-TO)
        SV(@MESAGE,5)
 LK(!MESAGE)
        SV(@C,@DC)
        SV(@R,@DR)
        SV(@DOCK,2)
 LK(!GO-DOCK)
        SV(@MOVE,999)
        JP(:DOCK2)
:HOUSE-DW
        SV(@DC,20)
        SV(@DR,0)
 LK(!GO-TO)
        JP(:DWELL)
;-------------------------------------------------------
:END-CRANE
;  OF(2DWT1S.RPT[,A])
;  PV(F,9999:00:00.00)
;  OF(2DWT2S.RPT[,A])
;  PV(F,9999:00:00.00)
;  OF(2DWT1R.RPT[,A])
;  PV(F,9999:00:00.00)
;  OF(2DWT2R.RPT[,A])
;  PV(F,9999:00:00.00)
;  OF(1DOCK.RPT[,A])
;  PV(F,@HIT1)
;  PV(F,9999)
;  OF(2DOCK.RPT[,A])
;  PV(F,@HIT2)
;  PV(F,9999)
;  CF(RPT)
   SV(%AVG-S1,%TTL-S1)
   AO(%AVG-S1,/,@THROUGHPUTS1)
   SV(%AVG-S2,%TTL-S2)
   AO(%AVG-S2,/,@THROUGHPUTS2)
   SV(%AVG-R1,%TTL-R1)
   AO(%AVG-R1,/,@THROUGHPUTR1)
   SV(%AVG-R2,%TTL-R2)
   AO(%AVG-R2,/,@THROUGHPUTR2)
ER
;-------------------------------------------------------
;------------ BL(!GO-DOCK) ----------------------------
   BL(!GO-DOCK)
        IF(@DOCK,EQ,1,:GO-DOCK1)
        IF(@DOCK,EQ,2,:GO-DOCK2)
```

```
:GO-DOCK1
        SV(@DC,0)
        SV(@DR,0)
 LK(!GO-TO)
        JP(:ENDGO-DOCK)
:GO-DOCK2
        SV(@DC,42)
        SV(@DR,0)
 LK(!GO-TO)
:ENDGO-DOCK
     EL
;-------------------------------------------------------
;------ BL(!FIND-S1) FIND CLOSET CELL FOR STORAGE ------
      BL(!FIND-S1)
        SV(@DUMMY1,990)
        SV(@INDEXR,0)
        SV(@INDEXC,0)
:FIND-S1
        IF(@INDEXR,EQ,5,:RESET-S1)
        IV(@INDEXR)
        SV(@INDEXC,0)
:FIND-SA
        IF(@INDEXC,EQ,20,:FIND-S1)
        IV(@INDEXC)
        SV(@DUMMY,@@DAA(@INDEXC,@INDEXR))
        IF(@DUMMY,GE,@DUMMY1,:FIND-SA)
:FIND-SAA
        SV(@DUMMY1,@DUMMY)
        SV(@SC,@INDEXC)
        SV(@SR,@INDEXR)
        JP(:FIND-SA)
:RESET-S1
        SV(@@CELL(@SC,@SR),991)
        SV(@@DAA(@SC,@SR),991)
        SV(@@DBB(@SC,@SR),991)
:ENDFIND-S1
        AO(@SC,*,2)
        AO(@SR,*,2)
        SV(@DC,@SC)
        SV(@DR,@SR)
     EL
;-------------------------------------------------------
;-- BL(!FIND-S2) FIND CLOSET CELL FOR STORAGE ----------
      BL(!FIND-S2)
        SV(@DUMMY1,990)
        SV(@INDEXR,0)
:FIND-S2
        IF(@INDEXR,EQ,5,:RESET-S2)
        IV(@INDEXR)
        SV(@INDEXC,21)
:FIND-SB
        IF(@INDEXC,EQ,1,:FIND-S2)
        DV(@INDEXC)
```

```
        SV(@DUMMY,@@DBB(@INDEXC,@INDEXR))
        IF(@DUMMY,GE,@DUMMY1,:FIND-SB)
:FIND-SBB
        SV(@DUMMY1,@DUMMY)
        SV(@SC,@INDEXC)
        SV(@SR,@INDEXR)
        JP(:FIND-SB)
:RESET-S2
        SV(@@CELL(@SC,@SR),992)
        SV(@@DAA(@SC,@SR),992)
        SV(@@DBB(@SC,@SR),992)
:ENDFIND-S2
        AO(@SC,*,2)
        AO(@SR,*,2)
        SV(@DC,@SC)
        SV(@DR,@SR)
     EL
;-------------------------------------------------------
;---------  BL(!SET-R1) FIND WILL RETRIEVED CELL R2 ----
     BL(!SET-R1)
:SET-R1
        RV(5,@SETR1C,1,20)
        RV(5,@SETR1R,1,5)
        SV(@SET1,@@CELL(@SETR1C,@SETR1R))
        IF(@SET1,NE,991,:SET-R1)
        SV(@@CELL(@SETR1C,@SETR1R),993)
        SV(%%CELL(@SETR1C,@SETR1R),OBJ%ST)
        SV(@RC,@SETR1C)
        SV(@RR,@SETR1R)
        AO(@RC,*,2)
        AO(@RR,*,2)
        SV(@MESAGE,3)
 LK(!MESAGE)
     EL
;-------------------------------------------------------
;--------  BL(!SET-R2) FIND WILL RETRIEVED CELL R2 -----
     BL(!SET-R2)
:SET-R2
        RV(5,@SETR2C,1,20)
        RV(5,@SETR2R,1,5)
        SV(@SET2,@@CELL(@SETR2C,@SETR2R))
        IF(@SET2,NE,992,:SET-R2)
        SV(@@CELL(@SETR2C,@SETR2R),994)
        SV(%%CELL(@SETR2C,@SETR2R),OBJ%ST)
        SV(@RC,@SETR2C)
        SV(@RR,@SETR2R)
        AO(@RC,*,2)
        AO(@RR,*,2)
        SV(@MESAGE,4)
 LK(!MESAGE)
     EL
;-------------------------------------------------------
;-------  BL(!FIND-R) FIND RETRIEVE CELL ---------------
```

```
     BL(!FIND-R)
        SV(@FIND-R,1)
        SV(%TIMER-R1,0)
        SV(%DIST-R1,999:00:00.00)
        SV(%TIMER-R2,0)
        SV(%DIST-R2,999:00:00.00)
        SV(@INDEXC,0)
        SV(@INDEXR,1)
:FIND-R
        SV(%TIMER,CLOCK)
        IV(@INDEXC)
        IF(@INDEXC,GT,20,:CHECK-ROW)
        SV(@DUMMY,@@CELL(@INDEXC,@INDEXR))
        IF(@DUMMY,EQ,993,:FIND-R1)
        IF(@DUMMY,EQ,994,:FIND-R2)
        JP(:FIND-R)
:FIND-R1
        SV(%DUMMY,%%CELL(@INDEXC,@INDEXR))
        AO(%TIMER,-,%DUMMY)
        IF(%TIMER,LT,%TIMER-R1,:DIST-R1)
        SV(%TIMER-R1,%TIMER)
        SV(@TIMER-1C,@INDEXC)
        SV(@TIMER-1R,@INDEXR)
:DIST-R1
        SV(@DC,@INDEXC)
        SV(@DR,@INDEXR)
        AO(@DC,*,2)
        AO(@DR,*,2)
 LK(!GO-TO)
        SV(%DUMMY1,%%DISTA(@INDEXC,@INDEXR))
        AO(%DUMMY,+,%DUMMY1)
        IF(%DUMMY,GT,%DIST-R1,:FIND-R)
        IF(%DUMMY,LT,%DIST-R1,:DIST-R11)
        RV(9,@R1R2,1,2)
        IF(@R1R2,EQ,1,:FIND-R)
:DIST-R11
        SV(%DIST-R1,%DUMMY)
        SV(@DIST-1C,@INDEXC)
        SV(@DIST-1R,@INDEXR)
        JP(:FIND-R)
:FIND-R2
        SV(%DUMMY,%%CELL(@INDEXC,@INDEXR))
        AO(%TIMER,-,%DUMMY)
        IF(%TIMER,LT,%TIMER-R2,:DIST-R2)
        SV(%TIMER-R2,%TIMER)
        SV(@TIMER-2C,@INDEXC)
        SV(@TIMER-2R,@INDEXR)
:DIST-R2
        SV(@DC,@INDEXC)
        SV(@DR,@INDEXR)
        AO(@DC,*,2)
        AO(@DR,*,2)
 LK(!GO-TO)
```

```
        SV(%DUMMY1,%%DISTB(@INDEXC,@INDEXR))
        AO(%DUMMY,+,%DUMMY1)
        IF(%DUMMY,GT,%DIST-R2,:FIND-R)
        IF(%DUMMY,LT,%DIST-R2,:DIST-R22)
        RV(9,@R1R2,1,2)
        IF(@R1R2,EQ,1,:FIND-R)
:DIST-R22
        SV(%DIST-R2,%DUMMY)
        SV(@DIST-2C,@INDEXC)
        SV(@DIST-2R,@INDEXR)
        JP(:FIND-R)
:CHECK-ROW
        SV(@INDEXC,0)
        IV(@INDEXR)
        IF(@INDEXR,GT,5,:SELECT-R)
        JP(:FIND-R)
:SELECT-R
        IF(%TIMER-R1,GT,%WTIME,:HOT-R)
        IF(%TIMER-R2,GT,%WTIME,:HOT-R)
        IF(%DIST-R1,LT,%DIST-R2,:SET-DISTR1)
        IF(%DIST-R2,LT,%DIST-R1,:SET-DISTR2)
        RV(9,@DUMMY,1,2)
        IF(@DUMMY,EQ,1,:SET-DISTR1)
        IF(@DUMMY,EQ,2,:SET-DISTR2)
:HOT-R
        IF(%TIMER-R1,GT,%TIMER-R2,:SET-HOT-R1)
        IF(%TIMER-R2,GT,%TIMER-R1,:SET-HOT-R2)
:SET-HOT-R1
        SV(@DOCK,1)
        SV(@DC,@TIMER-1C)
        SV(@DR,@TIMER-1R)
        JP(:ENDFIND-R)
:SET-HOT-R2
        SV(@DOCK,2)
        SV(@DC,@TIMER-2C)
        SV(@DR,@TIMER-2R)
        JP(:ENDFIND-R)
:SET-DISTR1
        SV(@DOCK,1)
        SV(@DC,@DIST-1C)
        SV(@DR,@DIST-1R)
        JP(:ENDFIND-R)
:SET-DISTR2
        SV(@DOCK,2)
        SV(@DC,@DIST-2C)
        SV(@DR,@DIST-2R)
:ENDFIND-R
        SV(@@CELL(@DC,@DR),0)
        SV(%ST,%%CELL(@DC,@DR))
        SV(%%CELL(@DC,@DR),0)
        SV(@DUMMY,@@DA(@DC,@DR))
        SV(@@DAA(@DC,@DR),@DUMMY)
        SV(@DUMMY,@@DB(@DC,@DR))
```

```
        SV(@@DBB(@DC,@DR),@DUMMY)
        AO(@DC,*,2)
        AO(@DR,*,2)
        SV(@FIND-R,0)
     EL
;-------------------------------------------------
;---------------  BL(!GOTO) ----------------------
    BL(!GO-TO)
        SV(@X,@C)
        SV(@Y,@R)
        SV(@DX,@DC)
        SV(@DY,@DR)
;-------- ROW -----------------
        IF(@DY,GT,@Y,:ROW-UP)
        IF(@DY,EQ,@Y,:ROW-0)
        IF(@DY,LT,@Y,:ROW-DOWN)

:ROW-UP
        AO(@DY,-,@Y)
        SV(@MY,@DY)
        SV(@DIR,1)
        JP(:COLOUMN)
:ROW-0
        SV(@MY,0)
        SV(@DIR,0)
        JP(:COLOUMN)
:ROW-DOWN
        AO(@Y,-,@DY)
        SV(@MY,@Y)
        SV(@DIR,2)
;--------  COLOUMN ----------
:COLOUMN
        IF(@DX,GT,@X,:COL-RIGHT)
        IF(@DX,EQ,@X,:COL-0)
        IF(@DX,LT,@X,:COL-LEFT)

:COL-RIGHT
        AO(@DX,-,@X)
        SV(@MX,@DX)
        SV(@DIC,4)
        JP(:ENDROW-COL)
:COL-0
        SV(@MX,0)
        SV(@DIC,0)
        JP(:ENDROW-COL)
:COL-LEFT
        AO(@X,-,@DX)
        SV(@MX,@X)
        SV(@DIC,3)
:ENDROW-COL
 LK(!SIMU)
        IF(@FIND-R,EQ,1,:END-MOVE)
;--------- MOVE-COL -----------
```

```
        IF(@DIC,EQ,0,:MOVE-ROW)
        IF(@DIC,EQ,3,:MOVE-LEFT)
        IF(@DIC,EQ,4,:MOVE-RIGHT)
:MOVE-LEFT
        ML(@MX,%MTIME)
        JP(:MOVE-ROW)
:MOVE-RIGHT
        MR(@MX,%MTIME)
;--------- MOVE-ROW ----------
:MOVE-ROW
        IF(@DIR,EQ,0,:END-MOVE)
        IF(@DIR,EQ,1,:MOVE-UP)

        MD(@MY,%MTIME)
        JP(:END-MOVE)
:MOVE-UP
        MU(@MY,%MTIME)
:END-MOVE
        EL
;-----------------------------------------------------
;--------------- BL(!SIMU) ----------------------
      BL(!SIMU)
        SV(@DUMMY1,@MX)
        SV(@DUMMY2,@MY)
        IF(@DOCK,EQ,3,:ID1)
        AO(@DUMMY2,-,2)
:ID1
        SV(%DUMMY1,%MOVETH)
        SV(%DUMMY2,%MOVETV)
        AO(%DUMMY1,*,@DUMMY1)
        AO(%DUMMY2,*,@DUMMY2)
        AO(@DUMMY1,+,@DUMMY2)

        IF(@DUMMY1,EQ,0,:ENDLKSIMU)
        IF(@DOCK,EQ,3,:ID2)
        AO(@DUMMY1,+,2)
:ID2
        IF(%DUMMY1,LT,%DUMMY2,:Y-TIME)
:X-TIME
        SV(%DUMMY,%DUMMY1)
        AO(%DUMMY1,/,@DUMMY1)
        SV(%MTIME,%DUMMY1)
        JP(:ENDLKSIMU)
:Y-TIME
        SV(%DUMMY,%DUMMY2)
        AO(%DUMMY2,/,@DUMMY1)
        SV(%MTIME,%DUMMY2)
:ENDLKSIMU
        EL
;-----------------------------------------------------
;---------------- BL(!MASAGE) PRINT MASAGE ----------
      BL(!MESAGE)
        IF(@MESAGE,EQ,3,:PMRG)
```

```
        IF(@MESAGE,EQ,4,:PMRG)
        IF(@MESAGE,EQ,6,:ENDMESAGE)

        AO(@DC,*,2)
        AO(@DR,*,160)
        AO(*MESAGE,+,@DC)
        AO(*MESAGE,-,@DR)
        IF(@MESAGE,EQ,1,:PMS)
        IF(@MESAGE,EQ,2,:PMF)
        IF(@MESAGE,EQ,5,:PM"")
:PMS
        PM(*MESAGE,S)
        JP(:RETURN1)
:PMF
        PM(*MESAGE,F)
        JP(:RETURN1)
:PM""
        PM(*MESAGE," ")
:RETURN1
        AO(*MESAGE,-,@DC)
        AO(*MESAGE,+,@DR)
        AO(@DC,/,2)
        AO(@DR,/,160)
        JP(:ENDMESAGE)
:PMRG
        AO(@RC,*,2)
        AO(@RR,*,160)
        AO(*MESAGE,+,@RC)
        AO(*MESAGE,-,@RR)
        IF(@MESAGE,EQ,4,:PMG)
:PMR
        PM(*MESAGE,R)
        JP(:RETURN2)
:PMG
        PM(*MESAGE,G)
:RETURN2
        AO(*MESAGE,-,@RC)
        AO(*MESAGE,+,@RR)
        AO(@RC,/,2)
        AO(@RR,/,160)
:ENDMESAGE
        SV(@MESAGE,6)
      EL
;-------------------------------------------------------
;***********  END OF LAYOUT-3 **********************
→
```