AN EXPERT SYSTEM FOR ADAPTIVE PART ROUTING
IN COMPUTER INTEGRATED MANUFACTURING

A Thesis Presented to
The Faculty of the College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Fook Cheon Khaw

March, 1987
This thesis is dedicated to my parents in gratitude for their love, caring and encouragement.
ACKNOWLEDGEMENTS

I wish to express deepest gratitude to my thesis director, Dr. Robert Terry, for his dedication throughout this work. His constant availability for help, guidance and resources in the most friendly atmosphere is my most delightful experience. I would also like to thank him for introducing me to this exciting research topic.

I also wish to thank the other members of my committee: Drs. Donald Scheck, Manjulika Koshal and Kenneth Cutright, for their invaluable criticism.

To Dr. David Ben-Arie of AT&T Bell Laboratories, Columbus, Ohio, I express my sincere appreciation. Without his help in knowledge based simulation concepts, this work could not be realized today.

This is also the place to acknowledge Dr. Joseph Elgomyel, Professor of Industrial Engineering at Purdue University, for his involvement and help in many specific issues of computer integrated manufacturing systems.

Dr. Charles Overby provided me his IBM PC-AT and his excellent office where most of this work was being done. For this, I am much indebted.

The friendship and companionship of my fellow graduate students and of my friends made my stay in the United States of America a very happy one. I thank them all.

Fook C. Khaw
December, 1986
<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>The Architecture of a Knowledge Based Expert System.</td>
</tr>
<tr>
<td>3.2</td>
<td>An Example of a Semantic Network.</td>
</tr>
<tr>
<td>3.3</td>
<td>A Frame Representation of Milling Operation.</td>
</tr>
<tr>
<td>4.1</td>
<td>Structure of the Expert System.</td>
</tr>
<tr>
<td>4.2</td>
<td>State space for the Routing of a Part.</td>
</tr>
<tr>
<td>4.3</td>
<td>The Cost of the Cheapest Path from s to t via n for a single node tree (a leaf).</td>
</tr>
<tr>
<td>4.4</td>
<td>The Relation of Expanding Tree from n until the f-value exceeds Bound results in Tree 1.</td>
</tr>
<tr>
<td>4.5</td>
<td>State Space for the Example.</td>
</tr>
<tr>
<td>4.6</td>
<td>The Hypothetical Computer Integrated Manufacturing System (modified from the automated research facility of the NBS).</td>
</tr>
<tr>
<td>5.1</td>
<td>Numbers of Parts Produced number Machine Failures and No Failures Conditions (Original Process Plan).</td>
</tr>
<tr>
<td>5.2</td>
<td>Average Queue Length for Original Process Plan with Failure Rate of Expon (60).</td>
</tr>
<tr>
<td>5.3</td>
<td>Average Queue Length for Original Process Plan under No Machine Failures Condition.</td>
</tr>
<tr>
<td>5.4</td>
<td>Average Queue Length for Original Process Plan with Failure Rate of Expon 120.</td>
</tr>
<tr>
<td>5.5</td>
<td>Average Queue Length with Different Failure Rates for Routing Expert System.</td>
</tr>
<tr>
<td>5.6</td>
<td>Average Queue Length for Different Process Plans for Routing Expert System with Machine Failure Rate of Expon 120.</td>
</tr>
<tr>
<td>5.7</td>
<td>Average Queue Length for Process Plan 1 with Machine Failure Rate of Expon 120.</td>
</tr>
<tr>
<td>5.8</td>
<td>Average Queue Length for Process Plan 2 with Machine Failure Rate of Expon 120.</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Number of Parts Produced under Different Machine Failure Conditions for the Original Process Plan.</td>
<td>96</td>
</tr>
<tr>
<td>5.2</td>
<td>Average Time Spent in System per Part under Different Machine Failure Conditions for the Original Process Plan (with standard deviation).</td>
<td>96</td>
</tr>
<tr>
<td>5.3</td>
<td>Number of Parts Produced under Different Machine Failure Conditions for Process Plan 1.</td>
<td>99</td>
</tr>
<tr>
<td>5.4</td>
<td>Average Time Spent in System per Part under Different Machine Failure Conditions for Process Plan 1.</td>
<td>99</td>
</tr>
<tr>
<td>5.5</td>
<td>Number of Parts Produced under Different Machine Failure Conditions for Process Plan 2.</td>
<td>100</td>
</tr>
<tr>
<td>5.6</td>
<td>Average Time Spent in System per Part under Different Machine Failure Conditions for Process Plan 2.</td>
<td>100</td>
</tr>
<tr>
<td>5.7</td>
<td>Maximum and Minimum Queue Length at the Production Area for the Original Process Plan.</td>
<td>108</td>
</tr>
<tr>
<td>5.8</td>
<td>Maximum and Minimum Queue Length at the Production Area for Process Plan 1.</td>
<td>108</td>
</tr>
<tr>
<td>5.9</td>
<td>Maximum and minimum Queue Length at the Production Area for Process Plan 2.</td>
<td>109</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

1.1 OVERVIEW

The manufacturing environment is extremely competitive these days compared with even a few years ago. International competition and advances in technology have accelerated the move towards the so-called computer integrated manufacturing systems (CIMS) and lightless plants\(^1\). The main concept behind these modern manufacturing facilities is flexibility and productivity. The need for flexibility and productivity is becoming more crucial in order to cope with customer's demand for increased product variety, improved quality, and shorter delivery lead times.

The flexibility of a CIMS is due to (i) the ability to perform a given set of operations on a part by using alternate routings through the machines, and (ii) the ability to simultaneously work on several types of parts [Sarin and Dar-El 1985]. Both (i) and (ii) are achieved due to the use of multiple tool carrying numerically controlled machines. The impact of these flexibilities is in reducing work-in-process inventory and setup time on the machines.

Due to these flexibilities, the control of a CIMS can be very complex. An efficient control system for a CIMS must consider and

\(^1\)Lightless plant is the term coined to depict computer-controlled, flexible manufacturing facilities, specifically arranged to operate completely unmanned for at least the night shift (and so the name "lightless" because no light is needed for the manufacturing
utilize all the "knowledge" and information available in the system in order to make intelligent decisions for the best performance of the system. The productivity of a CIMS is largely based on how well these decisions are made under various constraints and flexibilities in the system. The constraints and flexibilities in a CIMS can be the managerial goals (such as due date requirements), physical limitations and flexibilities (such as machine capabilities, etc.), and other related ones. An important decision making level in a CIMS is the routing of parts. This decision is concerned with the problem of which machine to choose for a process, given the flexibility of the various machines and the constraints in the CIMS.

1.2 STATEMENT OF THE RESEARCH PROBLEM

This research is directed at the intelligent decision making and real time control of a computer integrated manufacturing system. The specific issue of interest of the control system concerns with the alternate routing of parts in the CIMS. Since the various machines in a CIMS are multi-functional, they can perform a variety of operations. The processing of parts can be done on more than one machine (alternate routing), thus giving rise to the flexibility of the CIMS. The flexibility of the CIMS is further complicated when different machines require different processing time for the same operation. For example, drilling operation can be done on vertical milling operations to carry on during the night shift).
machine, horizontal milling machine as well as turning machine. However, the efficiency of performing the drilling operation on the machines can be different.

The decision of which machine to choose for the operation must also be based on the current state of the machines. A machine might have very short processing time for the operation but if the current queue of the machine is long, the best decision might be to route the part to an alternative machine. In addition the control systems must also be able to cope with many unpredictable situations such as machine failure, tool breakage, robot malfunction or AGVS breakdown, that might constitute the major sources of disruption for the CIMS. A good control system for routing of parts in a CIMS must, therefore, be dynamic and adaptive. Real time intelligent decision making, taking account of the unpredictable situations and the dynamic environment of the system are desired. Coupled with these constraints the control system should still be able to perform well and increase productivity which are necessary to justify the implementation of the rather costly computer integrated manufacturing system.

1.3 PROPOSED METHODOLOGY

This research explores the use of artificial intelligence approach by means of developing an expert system for controlling the routing of parts through a computer integrated manufacturing system. The expert system (sometimes it is also called knowledge-based system)
developed is called the "Routing Expert System" (RES) and is written in PROLOG language. The RES consists of three parts: a knowledge base which contains all the static and dynamic information (such as part types, process plans, machine status, queue length, etc.) of the CIMS, a heuristic knowledge part which provides immediate or real time decisions regarding the best routes for a part based on the state of the system, and a meta knowledge part which assesses the decisions and performance made at the first level and makes intelligent overriding decisions if necessary.

In order to evaluate this expert system, a knowledge based simulation model for a CIMS is developed. The knowledge based simulation models developed are based entirely on rules that describe the behavior of the CIMS. They are used for comparing the performance of alternative RES designs. In particular, the RES itself is also compared with the RES without the meta knowledge and random routing policy with the added meta knowledge, to further evaluate the performance of the meta knowledge. A random routing policy is used as a baseline in making these comparisons.

1.4 THESIS ORGANIZATION

Chapter II surveys the literature related to routing in computer integrated manufacturing systems, human decision making in CIMS and concludes with a statement of the objectives of the research. Chapter III gives an introduction to expert systems, PROLOG language,
and knowledge based simulation. Chapter IV describes the basic structure and knowledge representation approach for "Routing Expert System" (RES). This chapter also describes the knowledge based simulation model for the computer integrated manufacturing system. Chapter V discusses the validation of the knowledge based simulation model, and the design and results of the experiments conducted to evaluate alternative routing policies. Chapter VI concludes this thesis with a summary and contribution for this research, and potential topics for future research in this area.
CHAPTER II

BACKGROUND AND OBJECTIVES

2.1 ROUTING PROBLEMS IN CIMS

The routing problem in CIMS is known to be combinatorial complex especially when the system is large, as will be discussed in Chapter IV. The situation is further complicated due to the dynamic flow of the manufacturing environment with uncertainties such as machine failures, failure of material handling system (e.g. AGVS), unavailability of raw material, and unacceptable workpiece. Some algorithms have been offered for the problem [Blair and Vasquez 1984] that consider only the static routing of parts with a fixed number of routes and Yao [1984] who developed a routing scheme that routes parts to the shortest queue with the highest probability. Ben-Arieh [1985] listed six static routing policies for comparison of his knowledge based routing system. Some of the static routing policies are: choose the fastest machine for part routing and the machines give higher queue priority to parts with fewer operations left, mixed routing policy that routes parts to the slowest machine for the first two operations and the remaining operations are routed to the fastest machine, heuristic algorithm that is based on transportation problem that minimizes mean finishing time, etc. Clearly, these static routing policies are not adequate for the dynamic routing problem in CIMS.
In another study by Yao [1985], the author emphasizes the information flow in CIMS as the basis for the routing flexibility of a part. The objective is to control and utilize routing flexibility in CIMS. The information required for achieving this flexibility has been identified (predetermined) as the sequence of operations and the alternative machines of the parts. This information is carried in the "Next Operation Lists" which also play the role of interfaces between the material (e.g. parts, machines and the material handling system) and the information modules. An information-theoretic quantity, routing entropy, has been developed to measure the variable amount of information, and hence the routing flexibility. The routing entropy is then used to formulate the "Least Reduction in Entropy" (LRE) principle for part routing. The approach developed is the first quantitative model for the modeling of the information flow in CIMS. However, the paper stops short of showing any numerical results of improvement in performance to justify this method.

In another study, Maimon and Chong [1986] propose a dynamic routing strategy for real time operational control of CIMS. A mathematical model of work-in-process flow through a pod\(^2\) and the existence of a stationary policy for dynamic routing are presented. The goal of dynamic routing in the research is to achieve a better distribution of temporary workload unbalances across the pods at all times, and thus improve manufacturing throughput and production smoothness.

\(^2\)A pod is an independent entity that consists of groups of workstations/cells which together can process raw materials into
2.2 HUMAN DECISION MAKING FOR ROUTING IN CIMS

When a CIMS is operating reliably, the number of people required to maintain its operation can be very few. Due to the combinatorial nature of a computer integrated manufacturing system, however, unpredictable events such as machine breakdowns, changes in priorities, etc., create system states that might be best handled by a human rather than a computer with predetermined logic structure in the software. A human employs rules of thumb and experience that is hard for a computer to capture despite progress in the area of artificial intelligence and expert systems.

Decision making for routing is concerned with detailed decision making for real time operation of the CIMS. The time horizon is typically a few minutes or hours, and the decisions involved are [Suri and Whitney 1984]:

(1) Work order scheduling and dispatching (which part to introduce next into the CIMS and when).
(2) Movement of parts and material handling system (which machine to send this part to next, which AGVS to send to pick up this part, etc.).
(3) Tool management.
(4) System monitoring and diagnostics.

finished products.
(5) Reacting to disruptions (failures of one or more system components, etc.).

In most cases, decision support systems (DDS) are used to aid and support in making these decisions. However, when a non-routine event occurs, such as failure of a machine, the CIMS supervisor will usually take charge of the decision making. If a machine is going to take a long time to repair, the supervisor may, for example, decide to reallocate (reroute) its production to other machines. Rerouting involves a complex sequence of trade-offs between part production rates and machine capacities. The supervisor may sometimes have to make many interactive attempts based on his experience before a satisfactory decision is found. Corrective action is then taken to overcome the disruptions.

2.3 ARTIFICIAL INTELLIGENCE IN MANUFACTURING

Artificial intelligence is a branch of computer science which attempts to make the computer think like human beings. Human intelligence has the ability to make decisions when encountered with new situations and to interrelate facts in order to take a rational action towards a desired goal. Artificial intelligence tends to capture these aspects of human intelligence. In a computer integrated manufacturing system, the computer controls the entire manufacturing operations, but for greater effectiveness human decisions are still needed. Artificial intelligence can be used instead to automate the
entire decision process in the operation of computer integrated manufacturing system.

In manufacturing domain, Iskanker [1975] is perhaps the first person who actually investigated the use of artificial intelligence in job shop scheduling. In the research, a special technique called the "Trainable Heuristic Procedure" was used as one of the methods of combining heuristic rules in solving the dynamic job shop sequencing problem. The "Trainable Heuristic Procedure" can "learn" from past experience. The approach to train the procedure is to first use the procedure to solve problems with known solutions (which was formulated as a nonlinear programming problem), and then use it to solve new problems. The results indicated that when the "Trainable Heuristic Procedure" is used with a sufficient number of rules such that most of the characteristics of the shop are covered, it produces better results than all other methods of combination of rules and individual rules. This shows that the more knowledge and information about the shop is acquired, the better the decisions can be made.

In a more recent study, Bullers et. al. [1980] described how artificial intelligence methods can be applied for manufacturing planning and control problems. The approach is based on predicate logic and theorem proving techniques using resolution (explained in Chapter III) in a manufacturing environment. Predicate logic (similar to PROLOG) is used to describe and represent knowledge related to the operation of the facility. It also allows both static and dynamically
changing knowledge to be modeled. Theorem proving techniques are used to deduce "new" knowledge from the current knowledge base. The knowledge in the knowledge base is represented as assertions of facts and axioms. Even though the approach is viable for question answering, the complexity of the job shop is ignored. Therefore, the approach is sufficient for small problems only.

A major development of applying artificial intelligence techniques for large scale manufacturing control system is the Intelligent Scheduling and Information System (ISIS) [Fox 1983]. ISIS is an expert system that uses constraint-directed search paradigm to solve scheduling problems. Constraint-directed search algorithm is an artificial intelligence search technique that considers constraints as the major knowledge for the search to generate the best solutions. The knowledge representation strategy used in the expert system is based on frames and it is written in Schema Representation Language, a LISP based knowledge representation language. The objective of ISIS is to meet due dates while satisfying the constraints in the plant. The constraints are divided into five main groups: the first constraint represents the organization goals such as due date requirements, cost restrictions, machine utilization goals, and work-in-process time requirements. The second constraint is the physical limitations. Examples are machine capabilities, product size, and quality limitations. The third constraint specifies the causal requirements to perform an operation. This includes the precedence of operations, and
resource requirements to perform an operation. The fourth constraint deals with the availability of resources. Finally, the fifth constraint considers the qualitative preferences for operations, machines, and other resources.

ISIS is hierarchical and the inputs are the bounds for solution space, selection of context sensitive constraints, and a weighted interpretation of constraints. When the search encounters conflict, ISIS relaxes the appropriate constraints. As the search works through the search tree (state space), ISIS decides which paths to pursue by keeping a set of the best rated partial schedules. ISIS has been successfully tested in Westinghouse Turbine Component Plant and is rated highly by experts. The latest version of ISIS is now commercially available.

In another study by Chang [1985], a knowledge based real time decision support system was developed for job shop scheduling at the shop floor level. The knowledge based system consists of 17 rules which are based on the experience of a scheduler on scheduling a simulated job shop. The rules are of the form of "IF-THEN" production rules and they can be classified into three categories: (A) State Identification Rules, (B) Mode Selection Rules, and (C) Execution Rules. State identification rules identify the status of the shop, the machines, and the jobs in the queue of the machine. If the preconditions for these rules are met, then the current status of the shop, the machines or the jobs in the queue is determined. Mode
selection rules select the appropriate objectives (or goals) based on the current status of the shop, the machines, and the jobs. The objectives of the rules might be to maximize machine utilization, to minimize manufacturing lead time, or to minimize the mean tardiness per job. The execution rules choose the specific job to be processed next by the machine, based on the objective decided by the mode selection rules.

The inference procedure used by Chang's knowledge based system is data driven search (or forward chaining). When a scheduling decision needs to be made, the following steps are taken in sequence: (1) state identification rules are applied to determine the current status of the shop, the machine, and the jobs in the queue, (2) based on state identified, one of the mode selection rules is used to select the appropriate goal, and (3) based on the goal selected, one or more execution rules are applied to choose a job to be processed next. In a more compact representation of the knowledge based system, the mode selection rules are really unnecessary. In this case, the execution rules are executed directly when the preconditions of the state identification rules are satisfied.

The execution rules that Chang used in his knowledge based system consist of three dispatching (heuristic) rules. They are the SPT (shortest processing time) rules, LSTR (least slack time remaining) rules, and COVERT (a heuristic procedure that considers job lateness and job processing time) rule. The idea of the knowledge
A knowledge based system is that since the best dispatching rules for scheduling a
task shop depends on the performance measures and the current state of
the system, a knowledge based approach that determines the most
appropriate rule to apply based on the two criteria will provide
better results than one which does not. For example, one of Chang's
execution rules says if a machine is overloaded, then SPT should be
applied to minimize manufacturing lead time. This is because SPT
gives higher priority to the job with the shortest imminent processing
time. For comparison of results, the three individual dispatching
rules were used to schedule the job shop simulation model with regard
to the knowledge based system. Their performances were compared in
terms of mean tardiness per job. The results showed that the
knowledge based system performed considerably better than all the
other dispatching rules.

According to Chang, the advantages of using the knowledge
based approach for job shop scheduling are:

(i) Knowledge based systems can organize the information
available in the shop and suggest specific actions.
(ii) Knowledge based systems are adaptive, and goals are
changed based on the current state of the system.
(iii) Knowledge based systems can easily be modified by adding
or changing the rules without affecting the other modules
of the system.
(iv) Knowledge based system can be implemented on real time
basis since computational requirements of a knowledge based system are not as complicated as analytical approaches and simulation approaches.

However, it is conceivable that the knowledge based system is highly system dependent. For example, the state identification rules of the knowledge based system are specially designed for the specific job shop the author was using. So if the knowledge based system is to be applied on other job shops, the rules should be properly modified.

In another similar research, Ben-Arieh [1985, 1986] developed a knowledge based routing system (KBRS) for the control of production and assembly operations of a computer integrated manufacturing system. The research was concerned with the dynamic routing problem of the CIMS. Routing in a CIMS is considered a combinatorial problem due to the flexibility of machines in the CIMS. The machines are flexible because the parts can be operated on these machines with different efficiencies. In addition to the flexibility of machines, the possible routings of parts also depend upon the queues that each machine has, the availability of the machines and other criteria. In the research, the KBRS is used to control the routing of parts through the system.

The KBRS is an expert system that consists of static and dynamic databases, a behavioral knowledge part, an algorithmic knowledge part and a simulation driver. The static database contains infor-
mation about the processes required, the structure of the part, the available machines and their capabilities, and other sources of information that do not change with time. The dynamic database stores the data that represent the state of the system. It contains the queue sizes and contents, the status of the machines, the current processes of the parts, the time a part is required, and other time dependent data. The behavioral knowledge part of the KBRS describes the behavior of the CIMS in terms of actions that cause other actions to take place. It is also responsible for the generation of events in the CIMS and performing the events in the event list. This part of the expert system is in fact the simulated environment of the CIMS. It basically consists of five primary events: part arrival, process finishing, assembly event, machine failures, and repair event. The algorithmic knowledge part is the main component of the KBRS. It solves the routing decisions in real time. The inputs of this part of the knowledge are two algorithms which are dynamic and dependent upon the system states as reflected in the dynamic database. The algorithms search for the best routes for incoming parts. They are defined as follows:

1. **The initialization algorithm.** This algorithm is used to determine the expected time for the parts to reach the assembly station for assembly and sub-assembly operations. It considers the time for assembly of each sub-assembly of the product tree that is available in the static database.
This algorithm is required by the search algorithm as a part of the knowledge the KBRS uses. The expected time a part is needed for assembly is calculated by the algorithm and is used in the evaluation function of the search algorithm for generating a feasible route for the part.

2. **The search algorithm.** Basically, this algorithm is a modified version of the algorithm A\* (also known as best-first search in the artificial intelligence domain). However, Ben-Arieh did not consider goal node of the search tree. Only the initial state is known. This implies the heuristic estimate for the algorithm A\* cannot be applied, and therefore optimality cannot be guaranteed. Heuristic estimate is required by algorithm A\* as the guide for the search algorithm to reach the best goal node. Since goal node is not assumed in the research, algorithm A\* cannot be used as it is and some modifications must be made. The modified search algorithm considers the expected time (from the initialization algorithm) the component is required for the assembly and all the future possible routes for that component. The evaluation function of a route for part x is:

\[
f(x) = M^*I_a + \sum_j W_j = M^* \max \{0, (\sum_j A_j - Et_x)\} + \sum_j W_j
\]

where:
Iₐ is the lateness of part to the assembly station

Wⱼ is the waiting time at station j

Aⱼ is the total (waiting + processing) time spent in station j

M is an appropriate constant

Etₓ is the expected time part x reaches the assembly station

The two algorithms just described were written in PASCAL. They are used to compute the cost of the various candidate solutions (routes). They are considered by the KBRS whenever a new part is introduced or a part finishes an operation.

The simulation driver of the KBRS is composed of the simulation clock and the event file. The event file contains the events generated by the behavioral knowledge part of the KBRS. The experimental manufacturing environment used to evaluate the KBRS consists of a five-machine CIMS and automated assembly station. The input to the CIMS are nine components that are assembled into an end product. The performance of the KBRS was evaluated with seven other routing policies which include the human schedulers. The results show that the KBRS produces the best results in terms of machine utilization, average queue length, production rate of each machine, and production rate of each part type.

The KBRS is adaptive and generative. Generative means a decision will be made every time a part arrives (or when a part finishes a process). The KBRS does not have overriding capability that may
change the decisions already made (that is the decisions that have earlier been made regarding the best routes a part should follow). This aspect of adaptability can be critical if unexpected events occur such as when a machine fails for a long period of time. In that case, the parts remain idle and wait passively until the machine is back to operational. Another assumption made in the research is that there is an unlimited buffer storage for each machine. The decisions made by the KBRS are based on the processing time and waiting time of candidate machines, without considering the capacity of the buffer size of the machines.

The KBRS was written in PROLOG, an artificial intelligence language that is mainly used in building expert systems. The two algorithms in the algorithmic knowledge part of the KBRS were written in PASCAL which interfaces with the databases and the behavioral knowledge of the KBRS. The behavioral knowledge of the KBRS can be considered as the simulation model of the CIMS. The KBRS has a very small second level knowledge whose main function is to assign the appropriate constant to the search algorithm in the algorithmic knowledge.

2.4 RESEARCH OBJECTIVES

In this research, a hybrid approach is used for developing a Routing Expert System (RES) for solving routing problems in CIMS. The Routing Expert System developed has the foundation from Ben-Arieh's
work [1985] and Chang's work [1985]. The RES integrates Ben-Arieh's approach which is generative in nature with Chang's approach which is variant in nature. Variant means decisions (rules and knowledge) are prepared in advance and a decision is retrieved when the current state of the system matches the conditions of the decision. The objective of using the hybrid approach is to remedy Ben-Arieh's expert system that lacks the capability of dealing with the routing parts that have already been decided. This overriding capability is important when dealing with random disturbance (such as machine failures, etc.) in a CIMS. In practice, overriding decisions are made by human supervisors of the CIMS. When the CIMS is not running smoothly due to some problems such as tool breakage, machine failures, etc., the human supervisor usually interferes with the control system and decides the best approach to deal with the problem based on his/her experience. The decisions made might be to reroute parts from the problematic machines to other machines or reroute parts to other machines if the buffer size of the machine is already full and other decisions. This type of overriding decisions is important in terms of increasing the productivity of the system in a disturbed environment. In this research, the task of making overriding decisions is implemented by the RES automatically. This is done by encoding human expert's knowledge in the form of pattern-directed modules (which consist of a set of IF-THEN rules) prepared in advance. When the conditions of the system matches the preconditions of the pattern-directed modules, actions are then taken. In this way, a more efficient expert system for controlling the routing of parts in CIMS can be achieved.
CHAPTER III

EXPERT SYSTEMS AND PROLOG

3.1 EXPERT SYSTEMS: A REVIEW

An expert system is defined by Professor Edward Feigenbaum of Stanford University, one of the leading researchers in expert systems, as:

...an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. Knowledge necessary to perform at such level, plus the inference procedures used, can be thought of as a model of the expertise of the best practitioners of the field. The knowledge of an expert system consists of facts and heuristics. The facts constitute a body of information that is widely shared, publicly available, and generally agreed upon by experts in a field. The heuristics are mostly private, little discussed rules of good judgement (rules of good guessing) that characterize expert level decision making in the field. The performance level of an expert system is primarily a function of the size and the quality of a knowledge base it possesses.

Basically, an expert system utilizes heuristics\(^3\) or rules of thumb to focus on the key aspects of a problem and to manipulate sym-

---

\(^3\)Heuristics are criteria, methods, or principles for deciding which
bolic descriptions in order to make reasoning about the knowledge given. A good expert system can solve difficult problems within a very narrow domain, as well as or better than a human expert can. Additionally, an expert system is more consistent, easier to transfer and document than a human expert is.

An expert system is basically composed of a knowledge base and an inference engine. The knowledge base contains a collection of domain knowledge of facts and rules that use those facts as the basis for decision making. The inference engine contains an interpreter that decides how to apply the rules to infer new knowledge and decides the order in which the rules should be applied. The structure of an expert system is illustrated in Figure 3.1.

3.2 STRATEGIES FOR REPRESENTING KNOWLEDGE: THE KNOWLEDGE BASE

The knowledge base of an expert system can be represented in one of the following four methods:

1. Rules. This is the most popular knowledge representation scheme used. Rules provide a formal way of expressing recommendations, directives, or strategies. They are expressed as IF-THEN statements as shown below:

among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises between two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices [Pearl 1984].
Figure 3.1 The architecture of a knowledge-based expert system
IF a machine is down,
THEN check for its possible causes of failure

IF a part arrives,
THEN route it to a machine for processing

2. **Semantic Networks.** This method represents knowledge in a network link or a collection of objects called nodes, connected by arcs that describe the relations between nodes. The nodes in a semantic network stand for objects, concepts or events. The arcs relate objects and descriptions. Common arcs used for representing hierarchies include "isa" and "has-part." A simple semantic network is shown in Figure 3.2.

3. **Frames.** A frame is a description of an object that contains slots for all information associated with the object. Slots may contain values, pointers to other frames, sets of rules, or procedures. Figure 3.3 shows a frame representation for a milling operation [Fox 1983].

4. **Logic.** Logic provides another way to represent knowledge. Predicate calculus is a common logical system. The elementary unit in predicate logic is an object. Statements about objects are called predicates. For example:

   machine status (milling_center, up).
   operation (part_1, drilling).
FIGURE 3.2 AN EXAMPLE OF A SEMANTIC NETWORK
((milling-operation

  INSTANCE: operation
  WORK-CENTER: milling-center 2
  DURATION: ((INSTANCE: time-interval
              DURATION: 15 )
  NEXT-OPERATION: drilling-operation
  Refined-by: milling-setup milling-run
  Includes: milling-setup milling-run
  Enabled-by: enable-milling ))

((process-wrench
  SUB-STATE-OF: enable-milling
  (INSTANCE: process
   POSSESSOR: milling-center
   POSSESSION: wrench
   OVERLAP: milling-operation
   INCLUDE: milling-setup ) ) )

Figure 3.3 'A Frame Representation of Milling Operation
mean that the status of milling center is up, and the operation of part 1 is drilling.

An important consideration of representing knowledge is the choice of primitive attributes of the problem domain that can be represented in the knowledge base. It is possible to represent the knowledge in a variety of ways using different semantic primitives. For example, in PROLOG (which uses the predicate logic for knowledge representation), the queue of parts can be represented in the following ways:

1. queue (machine_1, [part_1, part_2, part_3]).
2. part_1 (machine_queue_1, first).
   part_2 (machine_queue_1, second).
   part_3 (machine_queue_1, third).

3.3 STRATEGIES FOR DRAWING INFERENCES: THE INFERENC ENGINE

The inference engine of an expert system contains the inference strategies and controls that an expert uses when he or she manipulates the facts and rules. Inference and control strategies guide an expert system as it uses the facts and rules stored in its knowledge base, and the information it acquires from the user. During consultation with a user, the inference engine of an expert system performs two major tasks [Harmon and King 1984]. First, it examines existing facts and rules, and add new facts when possible. Second, it decides the order in which inferences are made. The inference portion of an expert system uses one or more of the following approaches:
1. **Modus ponens.** This approach says that when A is known to be true and if a rule states, "If A, then B," it is valid to conclude that B is true. In other words, if the premises of a rule is true then we are comfortable to believe the conclusion.

2. **Reasoning about uncertainty.** Unlike conventional programs, an expert system is capable of dealing with incomplete or missing information. MYCIN (an expert system for medical consultation) has an inference engine that can manage the degrees of certainty [Harmon and King 1984] in the following ways:
   
a. Facts may be concluded by more than one rule. A combining function blends the certainty factors.

   b. Compound premises (those with more than one clause joined by the operations AND or OR) may test uncertain facts. An uncertain premise leads to an uncertain conclusion.

   c. Rules themselves may be less than definite.

3. **Resolution.** Resolution is a way to find out whether a new fact is valid, given a set of logical statements. Resolution is designed to work with formulas in clausal form. Given two clauses related in an appropriate way, it will generate a new clause that is a consequence of them.
Resolution strategies are used in logical systems such as PROLOG.

In addition to the inference procedure, an expert system also consists of a control procedure in its inference engine. The control portion of the inference engine serves two purposes: (1) to decide where the reasoning process is to begin in the knowledge base, and (2) to resolve conflicts when alternative lines of reasoning emerge. Some of the control strategies used are:

1. **Chaining.** Sometimes also called goal-directed systems, backward chaining systems start from the goal and work "backward" through the subgoals in an effort to choose an answer. Most existing expert systems use this approach [Waterman 1986]. Forward chaining, on the other hand, first examines the premises of the rules to see whether they are true, given the information. If so, then the conclusions are added to the list of facts known to be true and the system examines the rules again. Forward chaining systems are sometimes called data-driven systems.

2. **Search.** Search techniques are important for expert systems. Both forward and backward chaining systems employ some kind of search in drawing conclusions from the knowledge base. Two major search techniques that are most commonly used are depth-first search and breadth-first
search. In a depth-first search, the inference engine, if given a choice of continuing the search from several nodes, always decides to choose a deepest one. A deepest node is one that is farthest from the start node. In contrast to the depth-first search, the breadth-first search chooses to first visit those nodes that are closest to the start node.

3. Monotonicity. Monotonicity refers to facts or knowledge that become true and remain true, and the growth of the amount of true information during the consultation period with the expert system [Harmon and King 1984]. In a monotonic reasoning system, all values concluded for an attribute remain true for the duration of the consultation session. In a nonmonotonic reasoning system, facts that are true may be retracted. It is hard to implement a nonmonotonic reasoning system because it needs to track down all of the implications that are based on a fact in the knowledge base. Most expert systems today support monotonic reasoning but allow only carefully controlled types of nonmonotonic reasoning [Harmon and King 1984].
3.4 CLASSIFICATION OF EXPERT SYSTEMS

Expert systems can be classified according to the tasks they perform. These are usually the tasks that experts perform [Stefik et. al. 1982]:

1. **Interpretation.** Interpretation is the analysis of data to determine their meaning. Some of the problems associated with this task are time varying data, erroneous data and missing data. Examples of such expert systems are FALCON, used for process control, and GAMMA, used for interpreting gamma-ray activation spectra in physics.

2. **Diagnosis.** Diagnosis is the task of fault-finding in a system or determination of disease state in medicine. Problems that are associated with this task are inaccessible, expensive, or dangerous data to be measured, intermittent faults, etc. Examples of such expert systems are DELTA which is used for identifying and correcting malfunctions in diesel electric locomotives, and MYCIN, used for medical diagnosis.

3. **Monitoring.** Monitoring involves continuously interpreting signals, setting off alarms when intervention is required. The key problem in this task is the context dependency of an alarm condition. Consequently, the monitoring systems have to vary signal expectations with time and situation.
Examples of such systems are NAVEX, used for space applications, and YES/MVS, used for monitoring computer system performance.

4. **Prediction.** Prediction means to forecast the course of the future from the past and present. Some problems with this task include the integration of incomplete information, contingency of events, and sensitivity to variations in input data for multiple possible future states. Examples of such expert systems are WILLARD which helps meteorologists forecast the likelihood of severe thunderstorms, and I&W, used for military prediction of the place and time an armed conflict will occur.

5. **Planning.** Planning involves developing a course of action that can be carried out to achieve goals. Key problems in this task are large solution space, overwhelming details, interacting subproblems, and coordination of hierarchical planning. Examples of the existing expert systems are ISIS which is used for manufacturing planning, and KNEECAP which aids in the planning of crew activity on board the space shuttle orbiter.

6. **Design.** Design is the process of making specifications to create objects that satisfy particular requirements. Problems associated in this area are constraints, multiple
possibilities, complexity and interconnected relationship of subproblems in a design. Examples of such expert systems are XCON that configures VAX 11/780 computer systems, CRITTER for VLSI circuit design, and FADES for facilities layout and design.

3.5 PROLOG: PROGRAMMING IN LOGIC

PROLOG is an artificial intelligence language that was first introduced by Alain Colmerauer and his associates in France at the University of Marseilles in 1972. Later David Warren and his associates developed the first efficient PROLOG interpreter and compiler at the University of Edinburgh, United Kingdom. Since then, PROLOG has become increasingly popular and is accepted in many countries besides the UK and France. The Hungarian government has encouraged extensive use of PROLOG in industry. In 1981, the Japanese government announced the Fifth-Generation Computer project and has adopted PROLOG as the primary language for the operating system of the supercomputers they hope to build. The acceptance of PROLOG in the United States, however, is rather slow. This is due to the fact that LISP, which is the first artificial intelligence language developed by John McCarthy at MIT in 1958, has long been faithfully used for artificial intelligence research in the United States. However, after the announcement of the Japanese Fifth-Generation project, American artificial intelligence researchers and others became aware of the beauty and power of PROLOG. Since then, PROLOG seems to be steadily gaining in popularity in the United States.
PROLOG is designed for symbolic rather than simply numerical computation. It has its roots in mathematical logic, mainly the predicate calculus. Whereas conventional languages such as FORTRAN, BASIC, PASCAL, and even LISP are procedurally oriented, PROLOG introduces the descriptive or declarative view. Conventional languages are still how-type languages of which modern common LISP is the champion [Winston 1984]. Common LISP is expressive but how to do something is still what the LISP programmer is allowed to be expressive about. PROLOG, on the other hand, is a language that clearly breaks away from the how-type languages, encouraging the programmer to describe situations and problems, not the detailed means of how the problems are to be solved. This declarative approach has made PROLOG a superior language in terms of ease of use, self-documenting, readability, and size of program code.

Some of the outstanding features which make PROLOG a powerful but simple to use programming language are [Coelho 1982]:

1. A declarative semantics inherited from logic in addition to the usual procedural semantics.

2. Identity of form of program and data — in other words there is no essential difference between data and program, and a program can execute or examine other programs (a powerful feature of meta-logical programming).

3. The input and output arguments of program do not have to be distinguished in advance but may vary from one call to
another.

4. Programs may have multiple inputs and outputs.

5. Built-in "inference engine" and relational database. Programs may generate, through backtracking, a sequence of alternative results. This provides a high level of interaction.

6. Sophisticated pattern matching that supports artificial intelligence applications.

7. Incomplete data structures (i.e., structures containing free variables) may be returned. PROLOG is very efficient at list processing.

8. Lastly but definitely not least, PROLOG is flexible and is much easier to learn and implement than LISP, the natural rival to PROLOG.

In fact, PROLOG is much more than simply a programming language. Advanced implementation of PROLOG such as the PROLOG-2 is indeed an environment. It supports modular programming and advanced editing and debugging facilities. In addition, PROLOG-2 has the capability to communicate with other software or languages, achieved by an advanced interfacing facility.

3.6 KNOWLEDGE BASED SIMULATION

Knowledge based simulation is relatively new in the domain of simulation. A knowledge based simulation is a rules-based simulation
system that combines the traditional simulation approach with the expert system technique. This approach uses behavioral rules that describe a situation. ROSS, a Rule-Oriented Simulation System developed at RAND, is an example of such a system. ROSS consists of 75 behavioral rules, extracted from experts. It is used for simulation of military air battles [Klahr and Faught 1980]. Futo and Szeredi [1984] describe a simple simulation model for bank robbery problems using PROLOG. A rather detailed discussion on discrete event simulation using concurrent PROLOG can be found in Cleary et. al. [1986].

Some of the advantages of knowledge based simulation over the traditional simulation software such as GPSS on SIMAN are [Ben-Arie 1985):

1. **Modular.** The simulation is made of modules. Each module describes the rules and behavior of the event being modeled. Rules can be added or deleted in a module with ease. Removal of some modules from the system is not necessarily fatal.

2. **Backtracking.** One of the major advantages of knowledge based simulation is the capability of generating multiple solutions or alternatives within a single simulation run. In SIMAN, for example, at least the experimental frame of

---

4Concurrent PROLOG is a very powerful subset of PROLOG. It allows for expressing concurrency and for constructing concurrent programs such as operating systems.
the simulation model has to be changed in order to generate different results.

3. Built-in relational database. The database of the system is within the simulation model.

4. Automated data analysis. Data analysis at the end of simulation run can be automated. The knowledge based simulation system will provide rating of a given scenario from a user specified perspective, detect the "forces" acting upon specified performance parameters, and suggest possible cause of action for a scenario that satisfies a user specified goal.
CHAPTER IV
THE RES: ROUTING EXPERT SYSTEM

4.1 THE EXPERT SYSTEM STRUCTURE

The purpose of the Routing Expert System (RES) is to provide a control system which can automatically solve complex routing problems in a CIMS as well as a human expert could. The RES should be able to utilize all the information available in a CIMS to generate the best routing decisions for parts, maintain a balanced load for the machining centers, and to achieve the best performance of the system in case of machine failures. The structure of RES is shown in Figure 4.1.

As can be seen in Figure 4.1, the RES is composed of three parts. The knowledge contains all the static and dynamic information associated with the CIMS. It interacts with the process controller of the CIMS and collects data from the shop floor. The heuristic knowledge decides the best routes for incoming parts that need to be routed. This level of knowledge provides real-time control commands for the CIMS. The meta knowledge is like a human supervisor in the CIMS. It observes and assesses the performance of the CIMS and makes overriding decisions if necessary. This level of knowledge also deals with unpredictable circumstances such as when a machine breaks down.

In addition to the RES, a simulated CIMS model is also included. This model represents a real computer integrated manufac-
turing system which consists of computer controlled machining centers, robots, automated guided vehicle system and a means for pilot testing the RES before it is actually implemented.

4.2 KNOWLEDGE REPRESENTATION

As discussed in Chapter III, knowledge representation is an important issue in developing an expert system. In this system, two types of knowledge exist: facts and rules. The knowledge representation strategy used for the facts and rules of this expert system is in predicate logic form which is provided in PROLOG. In PROLOG the facts or assertions are represented as follow:

\[
\text{part_process (a, [milling, reaming, grooving, inspection])}
\]
\[
\text{buffer\_size (machining\_center\_1,5).}
\]

These say the part process of "a" is milling, reaming, grooving and inspection, and the machining_center 1 has a buffer size of 5. As can be seen in the examples, PROLOG represents facts in two parts, the one outside the parenthesis is called the predicate of the statement; the one inside the parenthesis is the arguments of the statement. The number of arguments in a predicate can be as many as needed. The arguments can also be represented in the form of lists, numbers, letters, words, variables or even free variables.

Although it is essential to state facts in the way described, the power of PROLOG becomes greatly extended when rules are for-
mulated, for it is rules that allow the inference mechanism (inference engine) of PROLOG to operate. In PROLOG, the ":-" represents "if", "," represents "and", and ";" represents "or". If desired the words "if", "and", and "or" can be used (after we define them) instead of the symbols. An example of a rule represented in PROLOG is:

\[
part\_arrival\ (Part):- \\
\hspace{1cm} find\_processes\ (Part), \\
\hspace{1cm} record\_arrival\_time\ (Part), \\
\hspace{1cm} find\_machine\ (Part, Machine), \\
\hspace{1cm} check\_schedule\ (Part, Machine).
\]

In PROLOG, all capital letters represent variables. The variables in a rule are all local. That means the same variables can be used in other rules as well. A free or blank variable is represented in PROLOG as ".". It is often used in situations where one needs to recognize the existence of an argument, but does not wish to instantiate it to a constant value. PROLOG tries to satisfy the goals and thus generate solutions by pattern matching through a depth-first search, backward chaining process.

In addition to representing facts and rules in PROLOG, the expert system also contains pattern-directed systems. Pattern-directed systems are special programming approaches that are composed of pattern directed modules. Each module is defined by (1) a precondition pattern, and (2) an action to be executed if the data environ-
ment matches the pattern. Pattern-directed systems are important in representing the meta knowledge of the expert system.

4.3 KNOWLEDGE ACQUISITION

In order to develop the proposed expert system and the knowledge based simulation model, the following steps are taken:

1. A hypothetical computer integrated manufacturing system (which is based on the automated manufacturing research facility at National Bureau of Standards) was conceptualized.

2. The a priori knowledge and the desired behavior of the CIMS was then encoded in IF-THEN production rules. The rules form the basic part of the simulation model. They were embedded as PROLOG clauses.

3. A knowledge base that contains the facts of the CIMS was developed. It contains all the static and dynamic information of the CIMS. The facts were encoded in predicate logic form available in PROLOG.

4. The simulation model and the knowledge base provide the foundation for the next phase of development of the expert system: the acquisition of the heuristic knowledge and the meta knowledge.
5. The available artificial intelligence search techniques were thoroughly reviewed to determine the most appropriate search technique to be used for the expert system. The search technique forms the basis of the heuristic knowledge that serves as the control system for the CIMS.

6. After the heuristic knowledge was acquired, several trial runs were carried out and more knowledge about the system behavior was retrieved from the simulation runs. It was found that the heuristic knowledge did not take care of parts in a failed machine very well but was excellent in making decisions for incoming parts. So a second level knowledge, the meta knowledge, was built.

7. The meta knowledge greatly improves the performance and remedy the weaknesses of the heuristic knowledge. The meta knowledge is mainly based on CIMS supervisors' common sense that deals with unpredictable events such as machine failures.

4.4 THE KNOWLEDGE BASE

The knowledge base is an important part of the expert system. It provides all the "knowledge" or information about the CIMS for the control system. It stores the system states, the properties

---

5Since PROLOG is often associated with Expert and Knowledge Based Systems, the name "Knowledge Base" is adopted instead of the name "database."
of the system components, the process plans for the parts and interacts with the CIMS. It is composed of a static knowledge base and a dynamic knowledge base. The static knowledge is the source of information that describes the processes and the CIMS. Examples are the description of the machining centers, the process time for the different operations on the various machines, buffer size of each machine, the initial and final conditions for the simulation and other known in advance values. The knowledge base is represented as facts in PROLOG. Examples of such facts are:

```
machine_code (automated_inspection_center,i).

which represents the automated inspection center as the machine code "i".
```

```
part_process (e,[boring *[v/3,t1/3,t2/2],
       slotting*[h1/7,h2/7,v/9,t1/6,t2/6],
       inspection*[i/6])*[loading* f/1]).
```

This predicate specifies the part name, the processes it needs to take and the time on each feasible machine.

The dynamic knowledge base stores the data that represent the state of the CIMS. It contains the queue sizes and contents, the status of the machines, the status of robots, the current processes performed on each part, the remaining processes of each part, and other time varying data. Examples for this part are:

```
queue(h1,[b/drilling/5/8,f/milling/8/13,d/reaming/2/21]).
```
This predicate specifies the parts in the queue of machine h1, the processes that are going to be performed, their processing time and the waiting time in the queue at that moment.

\[
\text{machine\_status (h1,up).}
\]

This says that machine h1 is in operational condition.

\[
\text{current\_process (m,turning,v).}
\]

This predicate represents that part m is currently performing turning operation on machine v.

In addition to the predicates that dynamically describe the CIMS, there are also predicates that collect information for statistical analysis such as the queue length of each machine, cumulative time spent in the system for each part, and the cumulative quantity of parts produced.

4.5 THE HEURISTIC KNOWLEDGE

Overview

The flexibility of routing in CIMS represents a problem associated with large state space. For the routing domains in CIMS, the number of alternatives to be explored is so high that the problem of complexity often becomes critical. Figure 4.2 shows a state space for the routing of a part. If each node in the state space has m successors, then the number of paths of length n from the start node is
FIGURE 4-2 STATE SPACE FOR THE ROUTING OF A PART
\[m^n\] (assuming no cycles). Thus the set of candidate solution paths grows exponentially with their length, which leads to what is called the combinatorial explosion. Therefore, the search procedure should use some problem specific knowledge ("rules of the game") to decide what is the most promising path to proceed at each stage of the search. Problem specific knowledge is heuristic. Search that uses heuristic is called heuristically guided search or simply heuristic search.

In the routing domain, the problem specific knowledge can be the availability of the machines, the queues that each machine has, the type of process to be done next, the capability of the machines to perform the task and some other information. This information has to be evaluated based on some measure of performance in order to reduce the number of possible solutions generated.

Before going into the development of the heuristic knowledge, it is worthwhile to briefly discuss some of the available search techniques that are used in the artificial intelligence domain, as well as in the operations research domain. Search techniques have been widely used as the basic tools in many artificial intelligence research and programs. Some of the more commonly known artificial intelligence search techniques are algorithm A* (discussed in next section), algorithm B*, AND/OR search, alpha-beta algorithm, breadth-first search, and depth-first search (the last two search techniques are defined in Chapter III). Each of these search techniques has its own
specialized applications. Algorithm B* is a variation of algorithm A* wherein probability theory is integrated with the search algorithm [Pearl 1984]. AND/OR search is used in problems that can be decomposed into mutually independent subproblems. Examples of such problems include route finding, symbolic integration, theorem proving, etc. Alpha-beta algorithm (also known as minimax procedure) is widely used in many chess-playing programs. This algorithm generates feasible game moves by reducing the branching factor (i.e. number of branches stemming from each internal node) of a tree to its square root (to avoid exhaustive search). The knowledge of an alpha-beta based chess program takes three forms: (1) evaluation function, (2) tree-pruning heuristics and (3) quiescence heuristics. The evaluation function computes the best value by disregarding some less promising continuations (based on tree-pruning heuristics) and extending the search in threatening positions beyond the depth limit until a quiescent position is reached (based on quiescence heuristics). It is interesting to compare the best chess programs with human chess masters. Powerful chess programs often search millions (and more) of positions before deciding the best move to play. Human chess masters, however, typically search just a few tens of positions, at most a few hundred. Despite this apparent inferiority, human chess masters usually beat powerful chess programs without too much effort [Pearl 1984]. The masters' advantage lies in their knowledge, which far exceeds that contained in the programs. This shows that the enormous advantage in the calculating power cannot completely compensate the lack of knowledge.
In operations research, dynamic programming, branch and bound, and the shortest route algorithm are the commonly used search techniques that are concerned with optimizing a function. Again these techniques are applied in their specialized areas. Branch and bound is a special kind of integer programming technique that uses a fathoming criteria for eliminating non-promising branches. The process of branching and bounding reduces the search and enables a near-optimal or optimal solution to be reached. In dynamic programming, the state space is decomposed into stages and a functional equation for the stages is defined. The process of finding the optimal solution is similar to branch and bound method. In this case, the best solution for each stage is always maintained and by advancing stage by stage, the goal will eventually be reached. The shortest route algorithm is concerned with finding the shortest route from an origin to a destination through a connecting network, given the distance associated with the branches of the network. There are many variations to solve the shortest-route problem. The simplest one is to fan out from the origin, successively identifying the shortest route to each of the nodes of the network in the ascending order of their shortest distances from the origin. The problem is solved when the destination node is reached.

Although artificial intelligence and operations research are two separate areas of domain, it is clear that they do share a common viewpoint i.e., to search for the best solution to a problem. These
two fields can indeed be married. Some papers that discuss the blending of these two fields of applications can be found in Marcus [1984] and Hawkins [1986].

**The Search Algorithm**

The heuristic knowledge of the expert system is similar to chess programs and is composed of two parts: algorithm A* or the best-first search algorithm and the problem specific knowledge for machine routing in the CIMS. The best-first search algorithm used in the heuristic knowledge is a variation of the famous A* algorithm [Nilsson 1980]. It uses the problem specific knowledge (which contain a heuristic estimate) defined for the nodes in the state space and always continue the search from the most promising node in the candidate set until the goal node is reached. The goal node for the best-first search algorithm is, of course, to reach the automatic inspection station as early as possible.

Assume that the cost function for the algorithm is defined for the arcs of state space. So \( c(n,n') \) is the cost of moving from a node \( n \) to its successor \( n' \) in the state space. The evaluation function \( f(n) \) for partial solutions can be represented as

\[
f(n) = g(n) + h(n)
\]

where \( g(n) \) is the estimate of the cost of the optimal path from \( s \) to \( n \); and \( h(n) \) is the heuristic estimate of the cost of an optimal path
from n to t. s is the start node of the search and t is the terminate node (Figure 4.3).

A more general definition of f-values can be extended from nodes to trees. For a single node tree (a leaf), n, the original definition remains the same. For a tree, T, whose root is n, and n's successors are \( m_1, m_2, m_3, \ldots, m_i \), the definition is:

\[
f(T) = \min_i\{f(m_i)\}
\]

The start node for the parts in CIMS is the automatic loading/unloading station and terminate node or the goal node is the automatic inspection station. We can interpret the evaluation function as follows. When a node n is encountered by the search process, a path from s (the start node) to n must have already been found, and its cost can be computed as the sum of the arc costs on the path. The other term, \( h(n) \), is more problematic because the state space between n and t (the terminate node) has not been explored by the search at that point yet. So \( h(n) \) is typically a heuristic guess, based on the algorithm's general knowledge about the problem (the routing problem).

The search process of best-first search algorithm involves an activate-deactivate mechanism that always deals with the currently most promising alternative among the competing subtrees. Subtrees have subtrees and these are explored by subprocesses of subprocesses and so on (Figure 4.4). The currently most promising alternative is always maintained and the search process keeps expanding. When the f
Figure 4.3 The cost of the cheapest path from s to t via n for a single node tree (a leaf).

Figure 4.4 The relation of expanding Tree from n until the f-value exceeds Bound results in Tree1.
values change so that some other alternative becomes more promising, then the activity is switched to this alternative. The search process ends when the goal node is encountered.

In order to avoid a total enumeration of all possible paths, the best-first search algorithm uses the available knowledge about the system defined by problem specific knowledge ("rules of the game"). These knowledge define the problem and also convey heuristic information about how to solve the problem. This part of the heuristic knowledge describes the relationship of the system such as the precedence constraint for the processes of each part, the set of machines that can be processed on for the next operation, and the cost associated with the alternative machine chosen. Accordingly, the routing problem of the system can be formulated as a state-space search problem as follows:

- states are partial routes;
- a successor state of some partial route is obtained by determining the next process and the machines that can be operated on;
- the start state is the loading/unloading station;
- the goal state is the inspection station;
- the cost of a solution (which is to be minimized) is the finishing time of all processes of a goal route;
- accordingly, the cost of a transition between two partial routes is the processing time plus the current waiting time of the next machine in the sequence.
All the information required for describing the problem specific knowledge is obtained from the knowledge base, as part of the requirements for the heuristic knowledge. In addition, the heuristic knowledge contains a heuristic function which will provide a very efficient guidance to best-first search algorithm. The function satisfies the principle of admissibility and will hence guarantee an optimal route. Principle of admissibility is an important property of the best-first search algorithm. It is defined as follows [Nilsson 1980]:

A heuristic search algorithm is said to be admissible if it always produces an optimal solution (that is, a minimum-cost path) provided a solution exists at all (a goal node exists). All solutions produced through the search can be considered admissible if the first solution found is optimal. Let, for each node n in the state space, $h^*(n)$ denote the cost of an optimal path from n to a goal node. A theorem about admissibility says: A search algorithm that uses a heuristic function h such that for all nodes n in the state space

$$h(n) \leq h^*(n)$$

is admissible.

The result of this principle is of great practical value. Even if we do not know the exact value of $h^*$, we just have to find a lower bound of $h^*$ and use it as h. If we knew $h^*$, we would use $h^*$
itself and search algorithm using $h^*$ finds an optimal solution directly, without any backtracking at all. There is, however, a trivial lower bound, namely:

$$h(n) = 0, \text{ for all } n \text{ in the state space}$$

This indeed guarantees admissibility. The disadvantage of $h = 0$ is that it has no heuristic power and does not provide any guidance for the search. In fact, best-first search algorithm using $h = 0$ behaves similar to the breadth-first search and results in high complexity. The heuristic knowledge, therefore, must have a lower bound of $h^*$ to ensure admissibility, and should be as close as possible to $h^*$ to ensure efficiency. In order to satisfy these two constraints, the heuristic knowledge must be very "informed." It is desirable to use all the information available to obtain an accurate heuristic function for the search algorithm. The heuristic function used for the best-first search algorithm is based on the minimum due time of parts for the remaining processes to be completed. It is formulated as follows:

$$\text{Due Time} = T + \sum_{n}^{t} \min\{Pt_n\} + \sum_{n}^{t} \min\{Wt_n\}$$

where $T$ is the current time (of the simulation run); $\min\{Pt_n\}$ is the minimum processing time among the alternative machines of succeeding process at current node $n$; and $\min\{Wt_n\}$ is the minimum current waiting time among the alternative machines of succeeding process at the current node $n$. The minimum due time for the part is the earliest
time that the part can finish all its processes (that is to reach the inspection station as early as possible). The heuristic estimate used satisfies the principle of admissibility, and will therefore, guarantee that the solution generated by the best-first search algorithm is the optimum routes. The heuristic knowledge is run whenever a part arrives and the solutions generated are in real time.

**Example**

In this example, a part just arrives at the loading/unloading station, waiting to be routed for processing. From the process plan of the part (which is stored in the static knowledge base of the expert system), there are four processes (including the inspection) to be performed on the part. Process 1 can be done on five machines (nodes 1, 2, 3, 4, 5 in Figure 4.5). The following process can be performed on machines (nodes) 2, 3, 4. Process 3 needs one of the three machines: nodes 1, 4, or 5. The last operation is done on the inspection station (the goal node). Each arc in the state space or the tree has a cost which is represented as a tuple \((P_t, W_t)\) where \(P_t\) is the processing time of the part at the machine and \(W_t\) is the current waiting time the queue. As defined earlier, the evaluation function for the tree of a route for the part is:

\[
f(T) = \min \{f(m_i)\}
\]

\[
f(m_i) = g(m_i) + h(m_i)
\]

where: \(g(m_i) = P_t m_i + W_t m_i\), is the transition cost between two partial routes, \(h(m_i)\) is the heuristic estimate of the minimum due time for the part as defined before.
FIGURE 4.5 STATE SPACE FOR THE EXAMPLE
Given: current time of the system = 40

the bound of the state space (the tree) = 1000.

initial cost = 0

1. The heuristic search process begins, at node 0. The successor nodes of node 0 (process 1) are: nodes 1, 2, 3, 4, and 5. Their costs, \( g(m) \) are:

\[
\begin{align*}
g(m_1) &= c(0,1) = 8 + 6 = 14 \\
g(m_2) &= c(0,2) = 8 + 8 = 16 \\
g(m_3) &= c(0,3) = 11 + 1 = 12 \\
g(m_4) &= c(0,4) = 9 + 12 = 21 \\
g(m_5) &= c(0,5) = 10 + 1 = 11
\end{align*}
\]

2. To calculate the f-values of the successor nodes of node 0, we need to estimate due time of the part for the remaining processes. Since the remaining processes (and the feasible machines) are the same for all of the successor nodes, the heuristic estimate is also the same for the nodes:

\[
h(m) = 40 + \sum_{m} \min\{Pt_m\} + \sum_{m} \min\{wt_m\}
\]

\[
h(m) = 40 + (7 + 5 + 3) + (1 + 1 + 3) = 60
\]

Therefore, the f-values are:

\[
\begin{align*}
f(m_1) &= 14 + 60 = 74 \\
f(m_2) &= 16 + 60 = 76 \\
f(m_3) &= 12 + 60 = 72
\end{align*}
\]
3. The best f-value of process 1 is $f(m_5)$. The search process will then proceed from node 5 of process 1. The most promising route so far is 0 - 5. The successor nodes of node 5 (process 2) are: nodes 2, 3, and 4. Their costs are:

\[ c(5,2) = 7 + 8 = 15 \]
\[ c(5,3) = 8 + 1 = 9 \]
\[ c(5,4) = 10 + 12 = 22 \]

The cumulative costs from the most promising route for the successor nodes are:

\[ g(m_2) = 11 + 15 = 26 \]
\[ g(m_3) = 11 + 9 = 20 \]
\[ g(m_4) = 11 + 22 = 33 \]

4. The heuristic estimate of the due time for the remaining processes from node 5 of process 1 is

\[ h(m) = 40 + (5 + 3) + (1 + 3) = 52 \]

Therefore, the f-values are

\[ f(m_2) = 26 + 52 = 78 \]
\[ f(m_3) = 20 + 52 = 72 \]
\[ f(m_4) = 33 + 52 = 85 \]
5. The best f-value of process 2 is $f(m_3)$, and the most promising route so far is then 0-5-3. Next the search process proceeds from node 3 of process 2 (of node 5). The successor nodes are nodes 1, 4, and 5. Their costs are

\[ c(3,1) = 7 + 6 = 13 \]
\[ c(3,4) = 5 + 12 = 17 \]
\[ c(3,5) = 9 + 1 = 10 \]

The cumulative costs from the most promising routes for the nodes are:

\[ g(m_1) = 20 + 13 = 33 \]
\[ g(m_4) = 20 + 17 = 37 \]
\[ g(m_5) = 20 + 10 = 30 \]

6. The heuristic estimate is

\[ h(m) = 40 + 3 + 3 = 46 \]

and the f-values are

\[ f(m_1) = 33 + 46 = 79 \]
\[ f(m_4) = 37 + 46 = 83 \]
\[ f(m_5) = 30 + 46 = 76 \]

7. The best f-value of process 3 is $f(m_5)$. Since the next process is inspection, the search process terminates with the final routes of machines 0-5-3-5-inspection station. The costs of these routes is
76 which is the minimum due time that the part will complete all its processes (at current time 40).

4.6 THE META KNOWLEDGE

The meta knowledge is an advanced implementation of the expert system, incorporating many powerful PROLOG programming features such as second-order programming, meta-logical programming, pattern-directed programming, and it also includes a meta interpreter for the expert system. Second-order programming enhances the first-order logic of PROLOG wherein quantification over a number of predicates is possible [Sterling and Shapiro 1986]. First-order logic talks about individuals, whereas second-order logic allows sets and their properties to be manipulated. Second-order programming is important when generating multiple solutions is desired. Meta-logical programming, on the other hand, allows a program to examine and execute other programs. They are used to test and check for certain characteristics and to manipulate terms in the dynamic knowledge base, and thus can be used for the expert system to supervise the performance of the CIMS.

The meta knowledge of the expert system is composed of three parts: (1) pattern-directed modules, (2) a meta interpreter, and (3) a meta rule for the arrival of parts. Pattern-directed modules (or systems) are different from conventional systems (namely the subrou-

---

6 meta knowledge means knowledge about knowledge.
tines) in that the modules of the system do not call each other according to a fixed, explicitly predefined scheme. Instead, they are "called" by patterns that occur in their data environment. Conventional subroutines are sequential and deterministic in terms of flow of execution, whereas the flow of execution of a pattern-directed system is parallel and non-deterministic.

A pattern-directed module is defined by:

(1) a precondition pattern, and

(2) an action to be executed if the data environment matches the pattern.

The execution of program modules is triggered by patterns that occur in the system's environment. The data environment is, of course, the knowledge base of the expert system. Pattern-directed systems provide a high degree of modularity and are especially desirable in systems with complex knowledge bases because it is difficult to predict in advance all the interactions between individual pieces of knowledge in the knowledge base [Bratko 1986].

Similar to other existing expert systems, the knowledge source contained in the meta knowledge of this expert system can be modified or expanded whenever necessary. The current version of the meta knowledge contains four pattern-directed modules. The preconditions and actions for these pattern-directed modules were decided by "common sense" (and were also suggested by Dr. Joseph Elgomayel, Professor of
Industrial Engineering at Purdue University, an expert in the field of computer integrated manufacturing systems).

MODULE [1]

Module 1 says that if a machine is down and the estimated repair time is greater than the waiting time in the alternative machine, then reroute the parts to the alternative machine.

MODULE [2]

Module 2 says that if, after all the parts have been introduced to the system, there is a machine in the system starving for parts, then check the load of the other machines. If there are parts that can be rerouted to the starved machine, then reroute the parts to the machine.

MODULE [3]

This module says that if the queue length of the machine has reached the maximum buffer size of the machine, then supply the machine with the part with the shortest processing time in the queue of the machine in order to reduce the queue length.

MODULE [4]

Module 4 says that if a part is assigned (by the heuristic knowledge) to a machine that has reached its maximum buffer size capacity, then reroute this part to alternative machines with the minimum cost (processing time + waiting time).
Each of these pattern-directed modules consists of a set of rules that utilize second-order predicates and meta-logical predicates for manipulating the knowledge base (and thus the CIMS). The syntax of the pattern-directed modules are:

Condition Part ———> Action Part

The condition part is a list of conditions:

[condition1, condition2, condition3,....]

The action part is a list of actions:

[Action1, Action2, Action3....]

Each condition and action is simply PROLOG goal in the system. The precondition is satisfied if all the goals in the list are satisfied. Similarly, to execute an action list, all the actions in the list have to be executed.

In addition to the pattern-directed modules, the meta knowledge also contains a meta interpreter for resolving conflict when more than one module can be executed at the same time. The conflict resolution strategy used in this interpreter is rather simple in that it always executes the first potentially active pattern-directed module in the order as they are written. In this case, the most important pattern-directed module is ordered at the top, followed by the next most important module and so on.
The meta interpreter of the meta knowledge always keeps track of the performance of the system, which is reflected in the dynamic knowledge base. When the conditions of the system meet with the preconditions of the modules, then the associated actions will be taken. In this way the meta knowledge is acting like a human supervisor, it may override the decisions already made by the heuristic knowledge regarding to the routes the parts should take. The objectives of the meta knowledge are, therefore, to be adaptive and to maintain a balanced load for the machining centers, and to achieve the best performance of the system in case of machine failures.

The last part of the meta knowledge is concerned with the introduction of new arrival for a part whenever the part finishes all the manufacturing operations (initially all parts are introduced to the system at the same time). A meta rule is used to determine the best time to introduce a new arrival of the part to the system. The meta rule observes the behavior of the system and the interval of new arrival is affected by the failure rate and the load of the system. The arrival rate is of pull type and is dynamic.

4.7 THE KNOWLEDGE BASED SIMULATION MODEL FOR THE CIMS

In creating the RES, it happens that a knowledge based simulation model for a CIMS was developed. A simulation model for a CIMS is needed in order to implement the RES, instead of trying it in a real system. The knowledge based simulation model differs from the tradi-
tional simulation approach in that the knowledge based simulation model is entirely rules-based and is more descriptive.

The behavior of the CIMS is easily described by PROLOG using rules represented by predicate logic. These behavioral rules describe actions or events that if happen trigger other actions. The knowledge based simulation model developed can basically be decomposed into two major parts: (1) the modeling knowledge and (2) the simulation driver.

The modeling knowledge consists of four major types of events: (1) part arrival, (2) process finish, (3) machine failures, and (4) machine back to operational, that occur in the CIMS. These events are modeled in the forms of production rules using PROLOG, and are simulated by the simulation driver. The behavioral rules for the four events are:

1. **Rules for part arrival event.** These rules describe the arrival of a part and what are the actions to be taken when a part arrives. For example:

   IF a part arrives

   THEN check the processes that need to be performed,
   finds its routes (by the heuristic knowledge),
   route to the first machine,
   create current process for the part or
   add to the queue of the machine,
   schedule end of process for the part.
2. Rules for process finish event. These rules describe what should be done when a part finishes a process. For example:

IF a part finishes a process
THEN remove it from the machine,
schedule next part in the queue (FIFO order),
find the next process and machine for the part,
create current process for the part or add to the queue of the next machine,
schedule end of process for the part.

3. Rules for machine failures event. The time between breakdowns of each machine is an exponentially distributed random variable (except the inspection station). The mean time between failures can be specified by the user as desired. The first machine breakdown event for all the machines is scheduled at the beginning of a simulation run. The rules say:

IF a machine is down
THEN change the status of the machine,
estimate the repair time for the machine,
deactivate the robot,
delay the process finish time for the current part,
delay the waiting time in the queue.
4. **Rules for machine back to operational event.** The time between this event is a random variable of exponential distribution with mean equal to 20 minutes. This event and the machine breakdown event form a cycle of alternate runtimes and downtimes that takes care of the failure-repair chain. The rules say:

IF a machine is back to operational

THEN change the status of the machine,

activate the robot,

continue with the process or

schedule next part in the queue,

generate next machine failure event.

The four events just described represent the primary events for the simulation model. Each of these events trigger lower level events, lower level events are further decomposed into lower level events until the lowest level events are reached and change the data structures in the knowledge base. All of these events are represented as rules. Any rule that triggers leads to an action. This action changes the state of the system, the entity attributes, and set membership as needed. All of these changes are reflected in the dynamic knowledge base of the expert system. Examples of such changes might be a part moves from the queue to a process, enters different queues, robot becomes idle, etc. Changes in the knowledge base are utilized through predicates that are defined as rules. Examples are rules for
updating queue time, rules for updating machine status, etc. In addition, there are also predicates for generating exponential distribution and random numbers as required by the machine failure and back-to-operational events.

The other part of the knowledge based simulation model is the simulation driver which is responsible for the simulation run. The simulation driver is composed of five parts. These include rules to perform a simulation run, rules for sorting the event list, rules for performing an event in the event list, the event list itself, and the simulation clock. The event list maintains a list of future events with their times. The events are generated by the modeling knowledge and the simulation clock is activated by the events generated in the event list. The simulation process is terminated at the predefined end of simulation time in the static knowledge base.

The knowledge based simulation model described uses the event-scheduling approach for discrete event simulation. It is based entirely on rules written in PROLOG. It is descriptive and modular, and enables the RES to react to the simulated events, instead of real ones. The detailed specifications of the RES and the simulation model can be found in Khaw [1986].

4.8 THE HYPOTHETICAL COMPUTER INTEGRATED MANUFACTURING SYSTEM

Since it is not feasible to implement the RES in real CIMS, a simulated hypothetical CIMS is required. The design of the hypotheti-
The CIMS is based on the automated manufacturing research facility of the National Bureau of Standards [Simpson, et. al. 1982]. The CIMS is composed of two CNC horizontal machining stations, one CNC vertical machining station, two CNC turning stations, and one automatic inspection station as shown in Figure 4.5. Each station has a small buffer storage for incoming parts and all the loading/unloading operations are performed by robots on the stations. When a part finishes all its operations, it is moved to the inspection station before transporting to the assembly area.

The input to the CIMS are 15 parts that need to be machined. The number of processes required are up to five processes. Since the machines are rather general purpose, they can perform a variety of operations. However, the efficiency of performing an operation on the machines is different. This flexibility is the main attribute to the complexity of the routing problem. The objective of the RES is to maximize the production rate of the CIMS with very little or no human input at all.
Figure 4.6 The Hypothetical Computer Integrated Manufacturing System (modified from the automated manufacturing research facility of the National Bureau of Standards)
CHAPTER V

RESULTS OF EXPERIMENTS

5.1 OBJECTIVES

This chapter discusses the validation of the knowledge based simulation model and evaluates alternative designs for the Routing Expert System and compares the performances obtained from these designs with those obtained from random routing policy. In addition, the expert system is also evaluated without the meta knowledge part of the expert system, as well as adding the meta knowledge to the random routing policy. In the first case, the meta knowledge of the expert system is disconnected whereas in the latter case a layer of meta knowledge is added. The objectives of these experiments are to determine the effectiveness of the decisions made by the meta knowledge on top of the decisions made by the heuristic knowledge and to investigate how well the meta knowledge performs under messy and noisy conditions (e.g. when machines are overloaded with parts or when machines are down). In addition the experiments are also performed to determine how sensitive system performance is to: (1) the machine failures rate, and (2) the processing time. The performance criteria used are (1) production rate for each part type, (2) average queue length at each machine, and (3) average time spent in the system per part. All the experiments are run for 240 simulated minutes. The summary of results is given in Appendix A.
5.2 RANDOM ROUTING POLICY

The random routing policy is used to provide a benchmark for the comparison of results obtained from RES and alternate RES designs. In this policy, parts select the machine arbitrarily from the list of candidate machines for their operations. In practice, the use of random routing is quite common. In this case, all the parts wait in a common buffer storage. When a machine is available then a part is routed to the machine (mostly FIFO order). After the process is done, the part is again waiting in the buffer area until a machine is available for next processing. This approach does share the load among all the machines equally and the system is quite balanced.

5.3 VALIDATION TESTING OF THE SIMULATION MODEL

The purpose of validation testing is to ensure that the simulation model faithfully depicts the actual CIMS that it was designed to represent. Once the fidelity of the simulation model has been established, data collected from it can be used as a substitute for actual system. This will permit the results of experiments performed on the simulation model to be applied to the system it was designed to represent. There are two major issues in the validation of the simulation model. The first concerns whether the structure of the simulation model captures the important aspects of the structure of the system being simulated. The second major aspect of validation concerns the stochastic behavior of the simulation model.
The appropriateness of the structure of the simulation model was established by having two members of the Industrial and Systems Engineering faculty at Ohio University who are familiar with CIMS, review it. The remainder of this discussion of validation testing will focus on stochastic behavior of the simulation model, such that statistically valid conclusions can be drawn from it.

In order for statistically valid conclusions to be drawn, the stochastic behavior of the simulation model must meet two criteria. The first concerns with the independence of each observation (data). The data collected successively should be random, i.e. not correlated. The second criteria concerns with the steady state condition for the data collected. In this simulation model, all parts are introduced to the system at the same time in order to minimize the time required to reach steady state. This assumption is better than that of a system that starts with "empty and idle" condition.

To determine if the observations are independent, three tests were performed. The tests are runs tests up and down, runs tests above and below mean, and time series analysis. Time series analysis is also used to study steady state behavior of the system by fitting autoregressive moving average models, i.e. ARMA (n,n-1) models. When n = 0, then the data will be free from patterns associated with non-steady state behavior. This will also show that the data is random and free of trend. The routing policies used for the tests are the RES and random routing with meta knowledge. Both of the routing poli-
cies were simulated for 960 minutes with intervals of 30 minutes each. The data collected were the number of parts produced at each time period.

The runs tests up and down and runs tests above and below mean are hypothesis tests for randomness where

\[ H_0: \text{Observations are random} \]

\[ H_1: \text{Observations are not random}. \]

If an alpha error of .05 is selected, the critical value for a two sided test is \( Z = 1.96 \). The results of the runs tests for the two policies are:

<table>
<thead>
<tr>
<th>Policies</th>
<th>Runs Above and Below Mean</th>
<th>Runs Up and Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>The RES</td>
<td>-0.16</td>
<td>NO</td>
</tr>
<tr>
<td>Random with Meta</td>
<td>-0.08</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>Z</td>
<td>Reject?</td>
</tr>
<tr>
<td></td>
<td>-0.75</td>
<td>Z</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>Rejects?</td>
</tr>
<tr>
<td></td>
<td>-0.15</td>
<td>NO</td>
</tr>
</tbody>
</table>

Based on this information, the null hypothesis that the observations are random cannot be rejected. This satisfies the independence property of the random number.

Time series analysis was also used for the purpose of testing the data for independence and for testing whether or not a steady state had been reached. The statistical analysis involved using Pandit and Wu sequential F-tests for determining the most appropriate
If data follows a ARMA \((n,n-1)\) model with \(n = 0\), then it follows that simulation model was in steady state and the data was random. The results of the computer output are summarized below:

<table>
<thead>
<tr>
<th>Routing Policies</th>
<th>Sum of Squares Error (SSE) of ARMA order</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(0,0) (1,0) (2,0) (3,0) (1,1) (2,1)</td>
</tr>
<tr>
<td>The RES Random with Meta</td>
<td>96.97 96.92 95.43 94.66 92.66 92.52</td>
</tr>
<tr>
<td></td>
<td>104.84 100.85 99.42 95.20 94.56 97.63</td>
</tr>
</tbody>
</table>

The results of the sequential F-tests are:

<table>
<thead>
<tr>
<th>Checking for the Adequate ARMA Model</th>
<th>(F_{\text{critical}})</th>
<th>The RES F-criterion Model</th>
<th>Random with Meta F-criterion model</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,0) vs. (1,0)</td>
<td>(F_{.95(1,30)} = 4.17)</td>
<td>0.015 (0,0)</td>
<td>1.146 (0,0)</td>
</tr>
<tr>
<td>(0,0) vs. (2,0)</td>
<td>(F_{.95(2,30)} = 3.32)</td>
<td>0.225 (0,0)</td>
<td>0.763 (0,0)</td>
</tr>
<tr>
<td>(0,0) vs. (3,0)</td>
<td>(F_{.95(3,30)} = 2.92)</td>
<td>0.220 (0,0)</td>
<td>1.506 (0,0)</td>
</tr>
<tr>
<td>(0,0) vs. (1,1)</td>
<td>(F_{.95(2,30)} = 3.32)</td>
<td>0.651 (0,0)</td>
<td>1.522 (0,0)</td>
</tr>
<tr>
<td>(0,0) vs. (2,1)</td>
<td>(F_{.95(3,30)} = 2.92)</td>
<td>0.433 (0,0)</td>
<td>0.665 (0,0)</td>
</tr>
</tbody>
</table>

Based on the above information, the adequate model for both of the routing policies is ARMA \((0,0)\). The reduction of SSE by fitting higher order models is not significant. This indicates that the simulation model was in steady state condition during the period for which the observations were generated.
5.4 SENSITIVITY TO MACHINE FAILURES

The purpose of investigating the sensitivity of this parameter is to see how well the RES performs under different machine failures and no machine failure conditions. Three machine failures conditions are simulated: machines failures of exponentially distributed with mean equal to 60 and 120 respectively, and no machine failures. In practice, machine failures are common problems in CIMS. However, we are interested in assessing the performance of the heuristic knowledge and the meta knowledge of the RES. A key question is whether the meta knowledge will be necessary when the machines are operating reliably.

(1) Number of Parts Produced

The meta knowledge of the expert system is designed to be highly adaptive to the CIMS environment. It is particularly reactive to any interruption occurring in the system. It also has the overriding authority regarding the routes a part should follow. The heuristic knowledge, on the other hand, is also adaptive to the CIMS environment. It has full look ahead\(^1\) capability and always produces the best routing decisions based on the current state of the system. However, there is a potential for trouble. Once a part is assigned a route, it has to follow that route and the processing of parts in a machine is in FIFO order. Machine failures will cause the parts currently being processed to be delayed. In addition, parts in the

\(^1\)Look ahead means how deep is the search algorithm (in the heuristic knowledge) searches through the tree i.e., the number of processes that the parts need to take in the future. Full look ahead considers all the processes of the part in the future.
queue of the failed machine will also be delayed. The length of the repair time will determine if new arrivals should be routed to the failed machine. In this manner the heuristic knowledge is able to efficiently deal with the problem of machine failure so as to achieve the performance of the system. However, under a no machine failure condition, the situation becomes much simpler. The decisions already made by the heuristic knowledge are the best at all times, and thus the decisions are of high quality and further improvement may not be possible.

From the results of the experiments (see Figure 5.1), it is clear that the total quantity of parts produced for the RES is better than the RES without the meta knowledge in all of the three machine failures conditions. The added meta knowledge to the random routing policy greatly improves the performance of just the random routing policy. The performance of the random routing policy with the added meta knowledge produces almost as many parts as the RES, and is consistently better than the RES without the meta knowledge. The results clearly show the capability of the meta knowledge in terms of dealing with machine failures, balancing the load of the system, and trying to improve the performance of the system constantly.

(2) Average Queue Length

With the exception of the inspection station, the average queue length at each machining center is shorter when there is no
FIGURE 5.1 NUMBERS OF PARTS PRODUCED UNDER MACHINE FAILURES AND NO FAILURES CONDITIONS (ORIGINAL PROCESS PLAN)
machine failure, than the queue length when machines may fail (see Figure 5.5). This is reasonable because there is no delay when no machine fails. As shown in Figures 5.2, 5.3 and 5.4, the RES has the most balanced queue length. The random routing policy is the worst case. The RES, with its high performance knowledge, successfully compensates for the failures of machines and is able to balance the queues and keep them as short as possible. The heuristic knowledge of the RES basically considers a failure as a very long queue and routes parts around it. In a sense, it recognizes which machine is currently a bottleneck machine and tries to avoid it. The meta knowledge, on the other hand, is like a human supervisor in the CIMS. It checks and sees which machine is overloaded, which part needs to be rerouted because of the limit of the buffer storage, and compares the repair time of a failed machine with the waiting time of other machines to decide if the parts in the queue of a failed machine need to be rerouted. In a cooperative sense, the heuristic knowledge tries to provide the best routes for the parts and the meta knowledge tries to improve them if possible.

For all the different machine failure conditions, the difference between the maximum queue length and minimum queue length using the RES is the smallest (see Table 5.7 on page 108). This shows that the RES is able to maintain a balanced load for the machine. The queue lengths at the inspection station for both the RES and the RES without the meta knowledge, and random routing with meta knowledge,
FIGURE 5.2 AVERAGE QUEUE LENGTH FOR ORIGINAL PROCESS PLAN WITH FAILURE RATE OF EXPON(60)
FIGURE 5.3 AVERAGE QUEUE LENGTH FOR ORIGINAL PROCESS PLAN UNDER NO MACHINE FAILURES CONDITION.
FIGURE 5.4 AVERAGE QUEUE LENGTH FOR ORIGINAL PROCESS PLAN WITH MACHINE FAILURE RATE OF EXPON 120
FIGURE 5.5 AVERAGE QUEUE LENGTH WITH DIFFERENT FAILURE RATES FOR ROUTING EXPERT SYSTEM.
however, are surprisingly high. Since there is no machine failure, a part can be done faster without any delay. The completed parts accumulated in the queue waiting to be inspected. This resulted in a bottleneck situation for the inspection station. A second inspection station might be needed under no failure condition. The meta knowledge, on the other hand, consists of a pattern-directed module that will process the part with the shortest processing time first, when buffer storage of the machine reaches the maximum capacity. This will reduce the load of the machine faster. The queue length of the inspection station using the random routing policy is also considerably higher with no failure condition. Basically, the average queue length for each machine increases with higher machine failure rate (except for the inspection station) in all of the cases.

(3) Average Time Spent in System (Mean Finishing Time) Per Part

This performance criterion measures the average total time that a part needs to complete all the manufacturing processes required. The average time spent in the system per part is much shorter in no failure condition than in failure condition. From the results shown in Table 5.2, the performance of the RES without the meta knowledge is much closer to the performance of the RES under no failure condition. It is interesting to see that the random routing with the added meta knowledge performs better than the RES without meta knowledge under all of the machine failure conditions. However, the RES's (with the meta knowledge) performance is better than that by
### Table 5.1 Number of Parts Produced under Different Machine Failure Conditions for the Original Process Plan

<table>
<thead>
<tr>
<th>FAILURE RATE</th>
<th>RES</th>
<th>RES w/o META</th>
<th>RANDOM with META</th>
<th>RANDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO FAILURES</td>
<td>68</td>
<td>63</td>
<td>65</td>
<td>57</td>
</tr>
<tr>
<td>EXPON (120)</td>
<td>64</td>
<td>59</td>
<td>61</td>
<td>53</td>
</tr>
<tr>
<td>EXPON (60)</td>
<td>53</td>
<td>45</td>
<td>51</td>
<td>38</td>
</tr>
</tbody>
</table>

### Table 5.2 Average Time Spent in System Per Part under Different Machine Failure Conditions for the Original Process Plan (with standard deviation)

<table>
<thead>
<tr>
<th>FAILURE RATE</th>
<th>RES</th>
<th>RES w/o META</th>
<th>RANDOM with META</th>
<th>RANDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO FAILURES</td>
<td>48.13</td>
<td>51.71</td>
<td>49.48</td>
<td>54.29</td>
</tr>
<tr>
<td></td>
<td>(13.28)</td>
<td>(15.58)</td>
<td>(14.51)</td>
<td>(17.67)</td>
</tr>
<tr>
<td>EXPON (120)</td>
<td>52.05</td>
<td>63.59</td>
<td>53.92</td>
<td>71.33</td>
</tr>
<tr>
<td></td>
<td>(16.28)</td>
<td>(25.35)</td>
<td>(17.29)</td>
<td>(37.98)</td>
</tr>
<tr>
<td>EXPON (60)</td>
<td>57.45</td>
<td>67.09</td>
<td>63.30</td>
<td>91.18</td>
</tr>
<tr>
<td></td>
<td>(22.97)</td>
<td>(32.35)</td>
<td>(23.30)</td>
<td>(51.60)</td>
</tr>
</tbody>
</table>
the random routing with meta knowledge. Again this shows that the meta knowledge is able to improve the performance of the heuristic knowledge even when the machines are working reliably. The average time spent in the system for the random routing policy is surprisingly good under no failure condition. Table 5.2 also shows that the standard deviation of the measurements on average time spent in system per part by the RES is the least among the routing policies. The standard deviation is important, from a practical point of view, in that it is related to the predictability of individual part's finishing time.

5.5 SENSITIVITY TO CHANGES IN PROCESSING TIME

The purpose of investigating the sensitivity of this parameter is to evaluate the influence of some static properties of the manufacturing system. The goal of computer integrated manufacturing system is to reduce the various time elements in the product life cycle. Specifically, the main concerns of computer integrated manufacturing are to increase productivity by reducing the time required to produce one unit of product and the time associated with planning and setting up for each batch of production [Groover and Zimmers 1984]. Therefore, it is important for the expert system to perform consistently and reliably even when technological changes (e.g. faster machines that require less processing time, etc.) exist. This experiment also serves to evaluate the performance of the meta knowledge when processing time for a part is long (such as in a job shop type system). In order to evaluate this parameter, three simulation models using different process plans were run:
1. The original process plan (see Appendix F). The average processing time for each operation is about 7 minutes.

2. The processing time of each machine for each process requires half that of the original process plan. The average processing time for each operation is about 3.5 minutes.

3. All the processes of the machines require (except the inspection station) twice the time of the original process plan. The average processing time for each operation is about 15 minutes.

The influence of this parameter is evaluated with respect to the performance measures mentioned earlier.

(1) **Number of Parts Produced**

Based on the results shown in Tables 5.1, 5.3, and 5.5, the number of parts produced using the RES was consistently higher than the number of parts produced using other policies. Under the original process plan, the RES without the meta knowledge and the random routing policy produced about 90% and 80% respectively of the number of parts produced by using the RES. However, when the meta knowledge was added to the random routing policy, a significant improvement was observed. In this case, the number of parts produced by the random routing with meta knowledge is very close to that of the RES and is, in fact, better than the RES without the meta knowledge. When faster
Table 5.3  Numbers of Parts Produced under Different Machine Failure Conditions for Process Plan 1

<table>
<thead>
<tr>
<th>FAILURE RATE :</th>
<th>RES</th>
<th>RES w/o META</th>
<th>RANDOM with META</th>
<th>RANDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO FAILURES</td>
<td>142</td>
<td>134</td>
<td>137</td>
<td>115</td>
</tr>
<tr>
<td>EXPO (120)</td>
<td>133</td>
<td>110</td>
<td>127</td>
<td>92</td>
</tr>
<tr>
<td>EXPO (60)</td>
<td>112</td>
<td>88</td>
<td>109</td>
<td>56</td>
</tr>
</tbody>
</table>

Table 5.4  Average Time Spent in System Per Part under Different Machine Failure Conditions for Process Plan 1

<table>
<thead>
<tr>
<th>FAILURE RATE :</th>
<th>RES</th>
<th>RES w/o META</th>
<th>RANDOM with META</th>
<th>RANDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO FAILURES</td>
<td>22.36</td>
<td>23.29</td>
<td>22.89</td>
<td>27.29</td>
</tr>
<tr>
<td></td>
<td>(6.41)</td>
<td>(4.86)</td>
<td>(4.92)</td>
<td>(8.04)</td>
</tr>
<tr>
<td>EXPO (120)</td>
<td>22.70</td>
<td>27.98</td>
<td>24.57</td>
<td>34.93</td>
</tr>
<tr>
<td></td>
<td>(6.61)</td>
<td>(12.60)</td>
<td>(8.12)</td>
<td>(14.47)</td>
</tr>
<tr>
<td>EXPO (60)</td>
<td>28.14</td>
<td>36.67</td>
<td>27.11</td>
<td>52.15</td>
</tr>
<tr>
<td></td>
<td>(9.95)</td>
<td>(14.61)</td>
<td>(11.64)</td>
<td>(29.36)</td>
</tr>
</tbody>
</table>
Table 5.5  Numbers of Parts Produced under Different Machine Failure Conditions for Process Plan 2

<table>
<thead>
<tr>
<th>FAILURE RATE</th>
<th>RES</th>
<th>RES w/o META</th>
<th>RANDOM with META</th>
<th>RANDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO FAILURES</td>
<td>35</td>
<td>35</td>
<td>34</td>
<td>27</td>
</tr>
<tr>
<td>EXPON (120)</td>
<td>31</td>
<td>30</td>
<td>31</td>
<td>23</td>
</tr>
<tr>
<td>EXPON (60)</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 5.6  Average Time Spent in System Per Part under Different Machine Failure Conditions for Process Plan 2

<table>
<thead>
<tr>
<th>FAILURE RATE</th>
<th>RES</th>
<th>RES w/o META</th>
<th>RANDOM with META</th>
<th>RANDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO FAILURES</td>
<td>91.25</td>
<td>89.34</td>
<td>93.52</td>
<td>110.16</td>
</tr>
<tr>
<td></td>
<td>(31.25)</td>
<td>(32.73)</td>
<td>(42.20)</td>
<td>(54.13)</td>
</tr>
<tr>
<td>EXPON (120)</td>
<td>103.93</td>
<td>110.76</td>
<td>107.01</td>
<td>135.96</td>
</tr>
<tr>
<td></td>
<td>(40.68)</td>
<td>(50.90)</td>
<td>(57.54)</td>
<td>(77.44)</td>
</tr>
<tr>
<td>EXPON (60)</td>
<td>122.57</td>
<td>117.25</td>
<td>119.69</td>
<td>149.80</td>
</tr>
<tr>
<td></td>
<td>(50.58)</td>
<td>(59.14)</td>
<td>(52.22)</td>
<td>(80.79)</td>
</tr>
</tbody>
</table>
machines were used (process plan 1), bigger differences among the routing policies were observed. In this case, the number of parts produced by the RES without the meta knowledge and random routing policy were about 85% and 68% respectively of the number of parts produced by using the RES. The difference between the RES and random routing with meta knowledge, however, is about the same. In both of the process plans, the number of parts produced by random routing with meta knowledge was about 96% of the parts produced by the RES. The results also show that, by reducing the processing time of manufacturing operations (which is one of the major goals of computer integrated manufacturing systems), the RES in general, and the meta knowledge in particular can further improve the productivity of the CIMS. When the processing time is short (i.e., when the cycle time of the part is short), the system has to deal with more routing decisions. The meta knowledge of the RES is capable of making more good judgement of the decisions generated by the heuristic knowledge and decides which part should be rerouted. In this way the RES is able to make more good decisions to achieve the best performance of the system. It is interesting to note that the meta knowledge, when added to the random routing policy, performs much better than just using the random routing policy. This clearly shows the capability of the meta knowledge in improving the overall performance of the system when the system is messy.

When processing time of a part is long (such as jobs found in a typical job shop), the differences among the RES, the RES without
the meta knowledge, and the random routing with meta knowledge are very small. The random routing policy alone is the worse in all cases. With long processing time, the number of routing decisions needed is reduced and the response of the system is slow. In a sense, the meta knowledge (which acts like a human supervisor) is more relaxing because things change too slowly. Another explanation to the similarity of the number of parts produced for the RES and the RES without the meta knowledge is that, for all other things being equal, the chances for the meta knowledge to make overriding decisions are less. Parts stay in the machines longer and a longer waiting time in the queue is expected. When the meta knowledge checks the conditions of the system to see if there is any machine idle or if the queue of a machine is empty, the chance of that situation happening is less (pattern-directed module no. 2 takes care of this situation). In addition, when a machine is down and the repair time, when compared to the waiting time in the alternative machines, is not favorable, the decision will be not to reroute the parts in the failed machine (pattern-directed module no. 1 takes care of the situation when a machine fails). In a sense, the decisions made at the heuristic knowledge are good enough to provide the best performance of the system. In the case of random routing policy, however, there is still plenty of room for further improvement to the system. The meta knowledge did override the decisions generated by the random routing policy and did provide good rerouting decisions to achieve the best performance of the system.
Average Queue Length

Basically, the average queue length at each machining center increases with longer parts processing time (except at the inspection station). When the processing time of parts is short, the waiting time at each machining center is shorter than that of the case when the processing time of parts is long. Figure 5.6 shows the increase of the average queue length at each machining center for the RES for the three different process plans under machine failure rate of exponentially distributed with mean equal to 120 minutes. The queue length at the inspection station, however, behaves in the opposite with the machining centers. When the processing time of parts is increased, the queue at the inspection station is shorter. This can be explained by the fact that with longer processing time, the finishing time of the parts will also be longer. Thus fewer parts accumulate at the inspection station. Figures 5.4, 5.7 and 5.8 show that the queues of the RES, the RES with the meta knowledge, and the random routing with meta knowledge respectively are generally balanced and with fewer fluctuations. Among the three policies, the RES has the most consistent (least fluctuated) queues. The random routing policy is the worst in all cases. However, when a layer of meta knowledge is added to it, a significant improvement is observed (see the results of the queues generated by random routing with meta knowledge). Again, this shows the high performance of the meta knowledge which is able to recognize the states of the system and
FIGURE 5.6 AVERAGE QUEUE LENGTH FOR DIFFERENT PROCESS PLANS FOR ROUTING EXPERT SYSTEM WITH MACHINE FAILURE RATE OF EXPON 120.
FIGURE 5.7 AVERAGE QUEUE LENGTH FOR PROCESS PLAN 1 WITH MACHINE FAILURE RATE OF EXPON 120
FIGURE 5.8 AVERAGE QUEUE LENGTH FOR PROCESS PLAN 2
WITH MACHINE FAILURE RATE OF EXPON 1.20
tries to balance the queues and compensates for the failures of machines as much as possible. Tables 5.7, 5.8 and 5.9 provide more results of the experiments.

(3) **Average Time Spent in System Per Part**

The average time spent in system (or the mean finishing time) per part increases with longer processing time per part. From the results shown in Tables 5.2, 5.4 and 5.6, there is basically no difference in the average time spent in system per part among the RES, the RES without the meta knowledge, and the random routing with meta knowledge for each of the process plans when there are no machine failures. On the whole, the RES is the best among the routing policies. When compared to the RES without the meta knowledge, the RES is better in seven out of the nine cases (for the three process plans). When compared to random routing with meta knowledge policy, the RES is still better in seven out of the nine cases. Nevertheless, when the random routing with meta knowledge is compared to the RES without the meta knowledge, the former is better in seven out of the nine cases. The results from the tables also show that the standard deviation of the measurement on average time spent in system per part by the RES is still the best. When compared to random routing with meta knowledge, the RES is better in eight out of the nine cases. When compared to the RES without the meta knowledge, the RES is also better in eight out of the nine cases. However, when random routing with meta knowledge is compared to the RES without the meta knowledge, it is
Table 5.7  Maximum and Minimum Queue Length at the
Production Area for the Original Process Plan

<table>
<thead>
<tr>
<th>FAILURE RATE</th>
<th>RES</th>
<th>RES w/o META</th>
<th>RANDOM with META</th>
<th>RANDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO FAILURES</td>
<td>0.87</td>
<td>1.33</td>
<td>1.15</td>
<td>2.42</td>
</tr>
<tr>
<td></td>
<td>0.46</td>
<td>0.38</td>
<td>0.60</td>
<td>0.33</td>
</tr>
<tr>
<td>EXPON (120)</td>
<td>1.32</td>
<td>1.68</td>
<td>1.86</td>
<td>4.63</td>
</tr>
<tr>
<td></td>
<td>0.73</td>
<td>0.69</td>
<td>0.81</td>
<td>0.23</td>
</tr>
<tr>
<td>EXPON (60)</td>
<td>2.02</td>
<td>3.30</td>
<td>2.41</td>
<td>4.38</td>
</tr>
<tr>
<td></td>
<td>0.99</td>
<td>0.52</td>
<td>0.93</td>
<td>0.27</td>
</tr>
</tbody>
</table>
Table 5.8  Maximum and Minimum Queue Length at the
Production Area for Process Plan 1

<table>
<thead>
<tr>
<th>FAILURE RATE</th>
<th>RES</th>
<th>RES w/o META</th>
<th>RANDOM with META</th>
<th>RANDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO FAILURES</td>
<td>0.95</td>
<td>0.97</td>
<td>0.74</td>
<td>3.58</td>
</tr>
<tr>
<td></td>
<td>0.55</td>
<td>0.50</td>
<td>0.41</td>
<td>0.27</td>
</tr>
<tr>
<td>EXPON (120)</td>
<td>1.23</td>
<td>4.16</td>
<td>1.12</td>
<td>3.58</td>
</tr>
<tr>
<td></td>
<td>0.74</td>
<td>0.22</td>
<td>0.51</td>
<td>0.27</td>
</tr>
<tr>
<td>EXPON (60)</td>
<td>1.73</td>
<td>3.82</td>
<td>1.50</td>
<td>4.90</td>
</tr>
<tr>
<td></td>
<td>0.91</td>
<td>0.17</td>
<td>0.58</td>
<td>0.20</td>
</tr>
</tbody>
</table>
Table 5.9  Maximum and Minimum Queue Length at the
Production Area for Process Plan 2

<table>
<thead>
<tr>
<th>FAILURE RATE</th>
<th>RES</th>
<th>RES w/o META</th>
<th>RANDOM with META</th>
<th>RANDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO FAILURES</td>
<td>2.07</td>
<td>2.21</td>
<td>2.20</td>
<td>5.59</td>
</tr>
<tr>
<td></td>
<td>1.24</td>
<td>1.10</td>
<td>1.18</td>
<td>0.41</td>
</tr>
<tr>
<td>EXPON (120)</td>
<td>2.58</td>
<td>2.45</td>
<td>3.34</td>
<td>5.53</td>
</tr>
<tr>
<td></td>
<td>1.44</td>
<td>1.00</td>
<td>1.07</td>
<td>0.67</td>
</tr>
<tr>
<td>EXPON (60)</td>
<td>2.14</td>
<td>2.79</td>
<td>2.88</td>
<td>5.51</td>
</tr>
<tr>
<td></td>
<td>1.12</td>
<td>1.19</td>
<td>1.03</td>
<td>0.36</td>
</tr>
</tbody>
</table>
interesting to notice that the former is more favorable. In this case, the performance of random routing with meta knowledge is better than that of the RES without the meta knowledge in six out of the nine cases. Again, the results have proved the capability of the meta knowledge that has played a significant role in improving the performance of the random routing policy. The random routing policy alone is far inferior when compared to all other routing policies.
CHAPTER VI

CONCLUSIONS

6.1 SUMMARY OF RESEARCH

This research introduced the RES, a Routing Expert System for solving part routing problems in computer integrated manufacturing systems. The RES consists of three parts: (1) a knowledge base, (2) a heuristic knowledge, and (3) a meta knowledge. In order to implement the RES, a knowledge based simulation model for a CIMS was also developed. Both the RES and the knowledge based simulation model were written in PROLOG. Initially when the expert system was built, several trial runs were made and the results were observed and improved with added rules. The RES is highly intelligent and adaptive, its heuristic knowledge uses the artificial intelligence search technique to decide the best routes for parts in real time. The meta knowledge of the RES is truly an expert system. Its knowledge source is nothing but the common sense of a CIMS supervisor whose expertise increases with experience. The performance of the RES was compared to that of the random routing policy. In order to further investigate the performance of the RES, the RES itself was also evaluated with the RES without the meta knowledge, and random routing with the meta knowledge. A total of thirty six simulation experiments were run for the sensitivity analyses of machine failures and of changes in processing time with regards to the RES, the RES without the meta
knowledge, random routing with the meta knowledge, and random routing policy. The results were analyzed and discussed.

6.2 SUMMARY OF FINDINGS

The results of the experiments clearly showed that the control of the CIMS was done much better by the expert system. In all of the cases, the expert system outperformed the random routing policy with regards to the measuring performance criteria of the number of parts produced, average queue length, and average time spent in the system. An interesting finding in the experiments, though, is the role played by the meta knowledge of the expert system. At first it was thought that the meta knowledge acts as little more than a system watchdog. It was also thought that the artificial intelligence based heuristic knowledge of the expert system is good enough to handle the complex routing problem efficiently. However, it turned out that the RES performed much better than the RES without the meta knowledge even under no machine failure condition. The evidence is even more obvious when the processing time of parts is short (i.e. when faster machines are used). The high performance of the meta knowledge is clearly demonstrated when the meta knowledge was added to the random routing policy. It takes care of the mess generated by the random routing policy and greatly improves the performance of the system. This is an important finding because a sophisticated algorithm such as the artificial intelligence based best-first search algorithm as well as other commonly used routing policy can be further improved quite easily by
just human's common sense (of which the meta knowledge is based). Further research in this area, though, is still needed.

6.3 CONTRIBUTIONS OF RESEARCH

1. A Routing Expert System (RES) was developed for intelligent decision making and real time control of part routing in a computer integrated manufacturing system. This expert system is a hybrid system which means it generates solutions whenever needed (the heuristic knowledge) and it also contains predefined conditions which, when matched with the conditions of the situation, will lead to actions (the meta knowledge). It has been demonstrated that this expert system is consistent and reliable, and it can achieve better performance than at least the random routing policy. In a dynamic and complicated CIMS, the control system must be highly flexible and adaptive. The expert system approach to control such a system is very promising in terms of increasing the productivity of the CIMS, and, more importantly, to realize a truly unmanned computer integrated manufacturing system. It is predictable that the next generation CIMS and lightless plants will heavily rely on sophisticated expert systems for controlling such facilities. The role of the human will be reduced to as little as just the system's watchdog.
2. To implement the developed expert system, a knowledge based simulation model was also constructed. This simulation model is entirely rules-based and is highly modular and descriptive. The developed knowledge based simulation model provides an excellent alternative and insight into the next generation simulation system which will have features such as automated data analysis and multiple alternatives generation. The knowledge based simulation model also provides a good foundation for further development of more complicated simulation models which might include automated guided vehicle systems (AGVS), automated storage and retrieval systems (ASRS), and automated assembly stations. Rules to develop such complex systems are for future research problems.

3. The use of PROLOG language as an artificial intelligence language has been demonstrated. In this research, both the RES and the knowledge based simulation model are entirely written in PROLOG. Undoubtedly, PROLOG is much easier to learn and use than other traditional programming languages such as FORTRAN, PASCAL or LISP. More importantly, PROLOG has much more to offer than just a programming language. Advanced PROLOG, such as PROLOG-2, is in fact an environment. With its many advanced facilities, PROLOG-2 should be seriously considered for any applica-
tion software development, especially for artificial intelligence applications.

4. A total of 97 predicates (rules and facts) have been developed and defined for both the RES and the knowledge based simulation model. A few of them are provided by the PROLOG-2 system library and they are listed in Appendix E. These predicates are modular and they represent rules and facts in PROLOG clauses. Each clause is relatively autonomous and it may trigger other clauses as defined in the body of the clause itself.

6.4 RECOMMENDATIONS FOR FUTURE RESEARCH

Some of the work developed in this research can be further extended. These include:

1. A more sophisticated meta interpreter for the meta knowledge. Since the knowledge base in this expert system is relatively small, the meta interpreter used is very efficient. But for a very large knowledge base, the conflict resolution strategy employed in this meta interpreter will not be able to handle it efficiently. A more advanced meta interpreter should be based on more meta rules.

2. Considering due date as a major constraint to the expert system. In this case, the expert system has to consider
and recognize the importance of different parts. Parts might be assigned some kind of priority index by the meta knowledge. In addition, the heuristic knowledge has to be properly modified to incorporate different levels of importance for the parts. This can be done by modifying the problem specific knowledge for the routing portion of the heuristic knowledge.

3. Considering higher level of flexibility problems in CIMS. Higher levels of flexibilities include the flexibility of sequencing of operations, the flexibility of different operation sets for parts, and the flexibility of part mix in the CIMS. More rules have to be built to integrate these problems.

4. Adding learning to the expert system. Such an expert system is able to solve new problems in a computer integrated manufacturing system based on past "experience" of the expert system. This capability will be the most prominent feature of the next generation expert system. The author feels that integration of artificial intelligence technique with statistical pattern recognition approach will play an important role in achieving this type of learning.

5. Interfacing distributed processing technology with expert system into the CIMS. Traditionally, most CIMS use a sin-
gle host computer for a centralized CIMS control system. These will create the following problems:

(a) Time sharing of computer resources limits the sophistication of the control system (such as the RES) since top priority for access to computer resources is usually given to real time control of the computer system.

(b) Significant degradation of the CIMS is possible when a crisis (interruption) occurs. Therefore, a local area network (LAN) type of control system is necessary for the expert system to fully utilize its expertise and knowledge. Interfacing expert system with LAN for a real CIMS control system is an exciting research problem.
REFERENCES
REFERENCES


### TABLES FOR AVERAGE QUEUE LENGTH

<table>
<thead>
<tr>
<th>CASE</th>
<th>H1</th>
<th>H2</th>
<th>V</th>
<th>T1</th>
<th>T2</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ORIGINAL PROCESS PLAN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE RATE</td>
<td>1.50</td>
<td>2.02</td>
<td>0.99</td>
<td>1.17</td>
<td>1.32</td>
<td>1.19</td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURES</td>
<td>0.68</td>
<td>0.58</td>
<td>0.46</td>
<td>0.87</td>
<td>0.79</td>
<td>5.04</td>
</tr>
<tr>
<td><strong>ROUTING EXPERT SYSTEM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE RATE</td>
<td>1.09</td>
<td>0.93</td>
<td>0.73</td>
<td>1.3</td>
<td>1.32</td>
<td>2.73</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ROUTING EXPERT SYSTEM: NO META KNOWLEDGE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE RATE</td>
<td>3.30</td>
<td>0.52</td>
<td>0.59</td>
<td>2.00</td>
<td>1.80</td>
<td>0.78</td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURES</td>
<td>0.86</td>
<td>0.60</td>
<td>0.38</td>
<td>1.33</td>
<td>0.88</td>
<td>4.75</td>
</tr>
<tr>
<td>FAILURE RATE</td>
<td>1.34</td>
<td>0.85</td>
<td>0.69</td>
<td>1.68</td>
<td>1.28</td>
<td>2.27</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### RANDOM ROUTING POLICY WITH META KNOWLEDGE

<table>
<thead>
<tr>
<th>Case</th>
<th>H1</th>
<th>H2</th>
<th>V</th>
<th>T1</th>
<th>T2</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Rate</td>
<td>1.34</td>
<td>1.05</td>
<td>0.93</td>
<td>2.41</td>
<td>1.61</td>
<td>0.60</td>
</tr>
<tr>
<td>Expon 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Failures</td>
<td>0.83</td>
<td>0.75</td>
<td>0.60</td>
<td>1.15</td>
<td>1.01</td>
<td>4.0</td>
</tr>
<tr>
<td>Failure Rate</td>
<td>1.19</td>
<td>1.16</td>
<td>0.81</td>
<td>1.86</td>
<td>1.30</td>
<td>1.94</td>
</tr>
<tr>
<td>Expon 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### RANDOM ROUTING POLICY

<table>
<thead>
<tr>
<th>Case</th>
<th>H1</th>
<th>H2</th>
<th>V</th>
<th>T1</th>
<th>T2</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Rate</td>
<td>3.71</td>
<td>0.44</td>
<td>0.27</td>
<td>4.38</td>
<td>0.79</td>
<td>0.30</td>
</tr>
<tr>
<td>Expon 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Failure</td>
<td>1.98</td>
<td>1.95</td>
<td>0.33</td>
<td>2.42</td>
<td>0.93</td>
<td>1.52</td>
</tr>
<tr>
<td>Failure Rate</td>
<td>2.23</td>
<td>0.98</td>
<td>0.23</td>
<td>4.63</td>
<td>0.65</td>
<td>0.57</td>
</tr>
<tr>
<td>Expon 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### PROCESS PLAN 1

#### ROUTING EXPERT SYSTEM

<table>
<thead>
<tr>
<th>CASE</th>
<th>H1</th>
<th>H2</th>
<th>V</th>
<th>T1</th>
<th>T2</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE RATE</td>
<td>0.91</td>
<td>1.73</td>
<td>0.92</td>
<td>1.44</td>
<td>1.34</td>
<td>1.00</td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURES</td>
<td>0.58</td>
<td>0.61</td>
<td>0.55</td>
<td>0.86</td>
<td>0.95</td>
<td>3.90</td>
</tr>
<tr>
<td>FAILURE RATE</td>
<td>1.03</td>
<td>0.74</td>
<td>0.76</td>
<td>1.07</td>
<td>1.23</td>
<td>1.93</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### ROUTING EXPERT SYSTEM: NO META KNOWLEDGE

<table>
<thead>
<tr>
<th>CASE</th>
<th>H1</th>
<th>H2</th>
<th>V</th>
<th>T1</th>
<th>T2</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE RATE</td>
<td>3.82</td>
<td>0.17</td>
<td>2.04</td>
<td>0.73</td>
<td>1.63</td>
<td>0.39</td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURES</td>
<td>0.79</td>
<td>0.78</td>
<td>0.50</td>
<td>0.97</td>
<td>0.95</td>
<td>3.90</td>
</tr>
<tr>
<td>FAILURE RATE</td>
<td>4.16</td>
<td>0.22</td>
<td>0.22</td>
<td>1.18</td>
<td>0.71</td>
<td>1.09</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### RANDOM ROUTING POLICY WITH META KNOWLEDGE

<table>
<thead>
<tr>
<th>CASE</th>
<th>H1</th>
<th>H2</th>
<th>V</th>
<th>T1</th>
<th>T2</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FAILURE RATE</strong></td>
<td>1.07</td>
<td>0.90</td>
<td>0.58</td>
<td>1.50</td>
<td>1.36</td>
<td>0.87</td>
</tr>
<tr>
<td><strong>EXPON 60</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NO FAILURES</strong></td>
<td>0.48</td>
<td>0.47</td>
<td>0.41</td>
<td>0.73</td>
<td>0.74</td>
<td>4.68</td>
</tr>
<tr>
<td><strong>FAILURE RATE</strong></td>
<td>0.69</td>
<td>0.60</td>
<td>0.51</td>
<td>1.12</td>
<td>0.91</td>
<td>3.04</td>
</tr>
<tr>
<td><strong>EXPON 120</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### RANDOM ROUTING POLICY

<table>
<thead>
<tr>
<th>CASE</th>
<th>H1</th>
<th>H2</th>
<th>V</th>
<th>T1</th>
<th>T2</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FAILURE RATE</strong></td>
<td>4.90</td>
<td>0.44</td>
<td>0.20</td>
<td>2.94</td>
<td>1.07</td>
<td>0.13</td>
</tr>
<tr>
<td><strong>EXPON 60</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>NO FAILURES</strong></td>
<td>0.97</td>
<td>2.13</td>
<td>0.27</td>
<td>3.58</td>
<td>0.69</td>
<td>1.09</td>
</tr>
<tr>
<td><strong>FAILURE RATE</strong></td>
<td>1.19</td>
<td>2.53</td>
<td>0.21</td>
<td>3.23</td>
<td>0.83</td>
<td>0.73</td>
</tr>
<tr>
<td><strong>EXPON 120</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### PROCESS PLAN 2

#### ROUTING EXPERT SYSTEM

<table>
<thead>
<tr>
<th>CASE</th>
<th>H1</th>
<th>H2</th>
<th>V</th>
<th>T1</th>
<th>T2</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE RATE</td>
<td>1.81</td>
<td>1.75</td>
<td>1.12</td>
<td>2.02</td>
<td>2.14</td>
<td>0.12</td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE RATE</td>
<td>1.86</td>
<td>1.24</td>
<td>1.46</td>
<td>2.04</td>
<td>2.07</td>
<td>0.30</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE RATE</td>
<td>1.69</td>
<td>1.45</td>
<td>1.44</td>
<td>2.58</td>
<td>1.89</td>
<td>0.18</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### ROUTING EXPERT SYSTEM: NO META KNOWLEDGE

<table>
<thead>
<tr>
<th>CASE</th>
<th>H1</th>
<th>H2</th>
<th>V</th>
<th>T1</th>
<th>T2</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE RATE</td>
<td>1.88</td>
<td>1.69</td>
<td>1.19</td>
<td>2.79</td>
<td>1.56</td>
<td>0.10</td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURES</td>
<td>1.87</td>
<td>1.10</td>
<td>1.63</td>
<td>1.93</td>
<td>2.21</td>
<td>0.24</td>
</tr>
<tr>
<td>FAILRE RATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE RATE</td>
<td>2.05</td>
<td>1.00</td>
<td>1.61</td>
<td>2.45</td>
<td>1.84</td>
<td>0.25</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CASE</td>
<td>H1</td>
<td>H2</td>
<td>V</td>
<td>T1</td>
<td>T2</td>
<td>I</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>FAILURE RATE EXPON 60</td>
<td>2.28</td>
<td>1.36</td>
<td>1.03</td>
<td>2.88</td>
<td>1.59</td>
<td>0.03</td>
</tr>
<tr>
<td>NO FAILURES</td>
<td>1.62</td>
<td>1.75</td>
<td>1.18</td>
<td>2.08</td>
<td>2.20</td>
<td>0.17</td>
</tr>
<tr>
<td>FAILURE RATE EXPON 120</td>
<td>1.45</td>
<td>1.07</td>
<td>1.19</td>
<td>3.34</td>
<td>1.41</td>
<td>0.12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CASE</th>
<th>H1</th>
<th>H2</th>
<th>V</th>
<th>T1</th>
<th>T2</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE RATE EXPON 60</td>
<td>3.43</td>
<td>0.36</td>
<td>0.41</td>
<td>5.51</td>
<td>0.71</td>
<td>0.00</td>
</tr>
<tr>
<td>NO FAILURES</td>
<td>2.98</td>
<td>0.61</td>
<td>0.41</td>
<td>5.59</td>
<td>0.54</td>
<td>0.04</td>
</tr>
<tr>
<td>FAILURE RATE EXPON 120</td>
<td>1.83</td>
<td>1.02</td>
<td>0.56</td>
<td>5.53</td>
<td>0.67</td>
<td>0.05</td>
</tr>
</tbody>
</table>
REMARK ROUTING WITH META KNOWLEDGE

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE R.</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURE</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>FAILURE R.</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

REMARK ROUTING POLICY

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE R.</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURE</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>FAILURE R.</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLES FOR QUANTITY PRODUCED

<table>
<thead>
<tr>
<th>ORIGINAL PROCESS PLAN</th>
<th>ROUTING EXPERT SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CASE</strong></td>
<td><strong>FAILURE R.</strong></td>
</tr>
<tr>
<td><strong>EXPO</strong></td>
<td>120</td>
</tr>
<tr>
<td><strong>EXPO</strong></td>
<td>60</td>
</tr>
<tr>
<td><strong>EXPO</strong></td>
<td>120</td>
</tr>
<tr>
<td><strong>EXPO</strong></td>
<td>120</td>
</tr>
</tbody>
</table>

**ROUTING EXPERT SYSTEM: NO META KNOWLEDGE**

| **EXPO**           | 60                    |         |
| **EXPO**           | 120                   |         |
| **EXPO**           | 120                   |         |
### PROCESS PLAN 1

#### ROUTING EXPERT SYSTEM

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure R.</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Expon 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Failure R. | 7 | 9 | 12 | 11 | 11 | 1 | 11 | 7 | 9 | 10 | 7 | 11 | 5 | 12 | 10 |
| Expon 120 | |

#### ROUTING EXPERT SYSTEM: NO META KNOWLEDGE

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure R.</td>
<td>5</td>
<td>3</td>
<td>11</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Expon 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Failure R. | 8 | 9 | 11 | 10 | 10 | 9 | 9 | 7 | 9 | 9 | 7 | 10 | 6 | 11 | 9 |
| Expon 120 | |

| Failure R. | 5 | 7 | 12 | 8 | 11 | 8 | 9 | 5 | 8 | 7 | 8 | 8 | 3 | 0 | 11 |
| Expon 120 | |
### RANDOM ROUTING WITH META KNOWLEDGE

| CASE | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FAILURE R. | 8 | 7 | 8 | 10 | 11 | 9 | 1 | 6 | 6 | 8 | 4 | 11 | 2 | 8 | 10 |
| EXPON 60 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

| CASE | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NO FAILURE | 7 | 9 | 10 | 11 | 10 | 10 | 7 | 9 | 9 | 7 | 11 | 6 | 11 | 10 |
| EXPON 120 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

### RANDOM ROUTING POLICY

| CASE | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FAILURE R. | 2 | 4 | 8 | 5 | 5 | 2 | 5 | 2 | 5 | 4 | 2 | 5 | 2 | 1 | 4 |
| EXPON 60 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

| CASE | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NO FAILURE | 6 | 6 | 9 | 10 | 9 | 8 | 9 | 5 | 7 | 9 | 6 | 8 | 4 | 12 | 7 |
| EXPON 120 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

| CASE | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FAILURE R. | 5 | 6 | 12 | 6 | 10 | 6 | 6 | 6 | 5 | 7 | 4 | 8 | 3 | 8 | 0 |
| EXPON 120 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
### PROCESS PLAN 2

#### ROUTING EXPERT SYSTEM

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure R.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expon</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Failure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failure R.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expon</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### ROUTING EXPERT SYSTEM: NO META KNOWLEDGE

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure R.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expon</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Failure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failure R.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expon</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### RANDOM ROUTING WITH META KNOWLEDGE

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE R.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPTON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE R.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPTON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### RANDOM ROUTING POLICY

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE R.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPTON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE R.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPTON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLES FOR AVERAGE TIME SPENT IN SYSTEM PER PART

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>I</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>I</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| FAILURE R. | 160.5 | 165.5 | 135.7 | 140.8 | 46 | 72 | 132.5 | 89 | 161.7 | 47 | 166.3 | 140.3 | 117 | 153.2 | 34.2 |
| EXPON 60 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

| NO FAILURE | 69 | 143.8 | 36 | 139.6 | 152.5 | 142.8 | 138.2 | 57 | 149.8 | 144.6 | 65 | 141.8 | 192.5 | 146.8 | 38.6 |
| EXPON 120 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

| FAILURE R. | 165.7 | 146.8 | 36 | 142.8 | 145.2 | 51 | 142.4 | 171.7 | 149.5 | 143.3 | 166.3 | 134.2 | 195.5 | 140.8 | 49.8 |
| EXPON 120 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

| FAILURE R. | 165.7 | 164.8 | 36 | 142.8 | 145.2 | 51 | 142.4 | 171.7 | 149.5 | 143.3 | 166.3 | 134.2 | 195.5 | 140.8 | 49.8 |
| EXPON 120 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

### ROUTING EXPERT SYSTEM; NO META KNOWLEDGE

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>I</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>I</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| FAILURE R. | 64 | 175.3 | 129.1 | 52 | 148.5 | 94 | 128.8 | 127 | 156.2 | 150.5 | 192.5 | 159.5 | 134 | 132.7 | 62.3 |
| EXPON 60 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

| NO FAILURE | 56.5 | 142.2 | 36 | 37 | 148.5 | 149.8 | 40 | 77 | 141.4 | 146.2 | 70 | 147.5 | 90 | 43 | 50.5 |
| EXPON 120 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

| FAILURE R. | 173.7 | 140.6 | 131.5 | 137.4 | 153.5 | 144.2 | 39 | 194.5 | 143.4 | 144.3 | 164.3 | 150.8 | 1240 | 146.8 | 50 |
| EXPON 120 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
### RANDOM ROUTING POLICY WITH META KNOWLEDGE

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failure R.</td>
<td>55.3:62</td>
<td>133.3:56</td>
<td>143.4:98</td>
<td>138.6:76</td>
<td>7182.5:72.7</td>
<td>199.5:34.3</td>
<td>193.5:37.4</td>
<td>66.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expon 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Failure</td>
<td>51.5:45.4</td>
<td>36.2</td>
<td>39</td>
<td>143.4</td>
<td>153.8</td>
<td>37.6</td>
<td>165.3</td>
<td>146.8</td>
<td>41</td>
<td>168.7</td>
<td>39.6</td>
<td>188.5</td>
<td>38.6</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>Failure R.</td>
<td>63.3:44.5</td>
<td>149.5</td>
<td>146.5</td>
<td>140.4</td>
<td>141.8</td>
<td>143.8</td>
<td>77</td>
<td>144.2</td>
<td>50</td>
<td>99</td>
<td>41</td>
<td>74</td>
<td>38</td>
<td>55.8</td>
<td></td>
</tr>
<tr>
<td>Expon 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### RANDOM ROUTING POLICY

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failure R.</td>
<td>60.5:85</td>
<td>154.2</td>
<td>36.2</td>
<td>168.3</td>
<td>153</td>
<td>148.7</td>
<td>132</td>
<td>100</td>
<td>151.2</td>
<td>185</td>
<td>39.6</td>
<td>189</td>
<td>55</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>Expon 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Failure</td>
<td>51.5:59.3</td>
<td>133.7</td>
<td>143.2</td>
<td>44</td>
<td>153.7</td>
<td>133.2</td>
<td>177.5</td>
<td>165.3</td>
<td>149.7</td>
<td>75</td>
<td>138.4</td>
<td>192.5</td>
<td>35.4</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>Failure R.</td>
<td>159.3</td>
<td>163.7</td>
<td>35.3</td>
<td>32.8</td>
<td>144.5</td>
<td>105</td>
<td>145.3</td>
<td>111</td>
<td>72.7</td>
<td>155.8</td>
<td>109</td>
<td>151.5</td>
<td>170</td>
<td>36</td>
<td>77.3</td>
</tr>
<tr>
<td>Expon 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLES FOR AVERAGE TIME SPENT IN SYSTEM PER PART

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE R.</td>
<td>160.5</td>
<td>65.5</td>
<td>135.7</td>
<td>140.8</td>
<td>46</td>
<td>72</td>
<td>132.5</td>
<td>89</td>
<td>161.7</td>
<td>47</td>
<td>166.3</td>
<td>140.3</td>
<td>41.7</td>
<td>117</td>
<td>53.2</td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURE</td>
<td>69</td>
<td>143.8</td>
<td>36</td>
<td>139.6</td>
<td>52.5</td>
<td>142.8</td>
<td>38.2</td>
<td>57</td>
<td>149.8</td>
<td>144.6</td>
<td>65</td>
<td>141.8</td>
<td>142.5</td>
<td>146.8</td>
<td>38.6</td>
</tr>
<tr>
<td>FAILURE R.</td>
<td>165.7</td>
<td>146.8</td>
<td>36</td>
<td>142.8</td>
<td>145.2</td>
<td>51</td>
<td>142.4</td>
<td>171.7</td>
<td>149.5</td>
<td>143.3</td>
<td>146.3</td>
<td>1134</td>
<td>195.5</td>
<td>140.8</td>
<td>49.8</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE R.</td>
<td>64</td>
<td>175.3</td>
<td>129.1</td>
<td>52</td>
<td>148.5</td>
<td>94</td>
<td>128.8</td>
<td>127</td>
<td>156.2</td>
<td>150.5</td>
<td>192.5</td>
<td>59.5</td>
<td>134</td>
<td>32.7</td>
<td>62.3</td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURE</td>
<td>56.5</td>
<td>142.2</td>
<td>36</td>
<td>137</td>
<td>148.5</td>
<td>49.8</td>
<td>40</td>
<td>77</td>
<td>141.4</td>
<td>146.2</td>
<td>70</td>
<td>147.5</td>
<td>40</td>
<td>43</td>
<td>50.5</td>
</tr>
<tr>
<td>FAILURE R.</td>
<td>173.7</td>
<td>140.6</td>
<td>131.5</td>
<td>137.4</td>
<td>153.5</td>
<td>144.2</td>
<td>39</td>
<td>194.5</td>
<td>143.4</td>
<td>144.3</td>
<td>164.3</td>
<td>150.8</td>
<td>240</td>
<td>146.8</td>
<td>50</td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CASE</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
</tr>
<tr>
<td>------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPON 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURE: 27.8:32.6:21.1:20.8:21.7:25.8:20.6:37:26.1:23.7:34.5:22.8:47.5:16.5:30.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPON 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLES FOR AVERAGE TIME SPENT IN SYSTEM PER PART

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE R.1161</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPO 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE R.1127</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPO 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE R.1169</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPO 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO FAILURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAILURE R.1169</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPO 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### RANDOM ROUTING POLICY WITH META KNOWLEDGE

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure R:</td>
<td>78</td>
<td>176.5</td>
<td>95</td>
<td>162.7</td>
<td>1113</td>
<td>1114</td>
<td>1102</td>
<td>142</td>
<td>1240</td>
<td>1104</td>
<td>1212</td>
<td>1108</td>
<td>1180</td>
<td>171.7</td>
<td>98</td>
</tr>
<tr>
<td>Expon 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Failure:</td>
<td>118</td>
<td>82</td>
<td>149.3</td>
<td>58</td>
<td>68.3</td>
<td>171.3</td>
<td>160.7</td>
<td>128</td>
<td>178.5</td>
<td>88</td>
<td>176</td>
<td>65</td>
<td>186</td>
<td>63.7</td>
<td>110</td>
</tr>
<tr>
<td>Expon 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failure R:</td>
<td>114</td>
<td>176.7</td>
<td>81</td>
<td>152.3</td>
<td>172.5</td>
<td>197.5</td>
<td>172.3</td>
<td>1104</td>
<td>1240</td>
<td>89</td>
<td>1239</td>
<td>165.7</td>
<td>133</td>
<td>170.7</td>
<td>98</td>
</tr>
<tr>
<td>Expon 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### RANDOM ROUTING POLICY

<table>
<thead>
<tr>
<th>CASE</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure R:</td>
<td>116</td>
<td>148</td>
<td>82</td>
<td>49</td>
<td>88</td>
<td>1240</td>
<td>54</td>
<td>1214</td>
<td>1240</td>
<td>1229</td>
<td>1240</td>
<td>151.5</td>
<td>1240</td>
<td>146.5</td>
<td>159</td>
</tr>
<tr>
<td>Expon 60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Failure:</td>
<td>104</td>
<td>102</td>
<td>73</td>
<td>76</td>
<td>62</td>
<td>1109</td>
<td>160.3</td>
<td>158</td>
<td>1131</td>
<td>185.5</td>
<td>194</td>
<td>60</td>
<td>1240</td>
<td>56.5</td>
<td>141</td>
</tr>
<tr>
<td>Expon 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failure R:</td>
<td>105</td>
<td>123</td>
<td>162.7</td>
<td>148.3</td>
<td>95</td>
<td>1223</td>
<td>146.5</td>
<td>1204</td>
<td>1240</td>
<td>107</td>
<td>1240</td>
<td>150.7</td>
<td>1240</td>
<td>167.7</td>
<td>187</td>
</tr>
<tr>
<td>Expon 120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DATA FOR VALIDATION TESTING OF THE SIMULATION MODEL

ROUTING EXPERT SYSTEM WITH FAILURE RATE OF EXPON 60

<table>
<thead>
<tr>
<th>TIME INTERVAL (MIN)</th>
<th>CUMMULATIVE PRODUCED</th>
<th># OF PARTS PRODUCED</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>90</td>
<td>17</td>
<td>9</td>
</tr>
<tr>
<td>120</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>150</td>
<td>33</td>
<td>8</td>
</tr>
<tr>
<td>180</td>
<td>41</td>
<td>8</td>
</tr>
<tr>
<td>210</td>
<td>46</td>
<td>5</td>
</tr>
<tr>
<td>240</td>
<td>55</td>
<td>9</td>
</tr>
<tr>
<td>270</td>
<td>63</td>
<td>8</td>
</tr>
<tr>
<td>300</td>
<td>68</td>
<td>5</td>
</tr>
<tr>
<td>330</td>
<td>76</td>
<td>8</td>
</tr>
<tr>
<td>360</td>
<td>80</td>
<td>4</td>
</tr>
<tr>
<td>390</td>
<td>87</td>
<td>7</td>
</tr>
<tr>
<td>420</td>
<td>93</td>
<td>6</td>
</tr>
<tr>
<td>450</td>
<td>100</td>
<td>7</td>
</tr>
<tr>
<td>480</td>
<td>105</td>
<td>5</td>
</tr>
<tr>
<td>510</td>
<td>111</td>
<td>6</td>
</tr>
<tr>
<td>540</td>
<td>120</td>
<td>9</td>
</tr>
<tr>
<td>570</td>
<td>130</td>
<td>10</td>
</tr>
<tr>
<td>600</td>
<td>138</td>
<td>8</td>
</tr>
<tr>
<td>630</td>
<td>142</td>
<td>4</td>
</tr>
<tr>
<td>660</td>
<td>149</td>
<td>7</td>
</tr>
<tr>
<td>690</td>
<td>157</td>
<td>8</td>
</tr>
<tr>
<td>720</td>
<td>166</td>
<td>9</td>
</tr>
<tr>
<td>750</td>
<td>173</td>
<td>7</td>
</tr>
<tr>
<td>780</td>
<td>181</td>
<td>8</td>
</tr>
<tr>
<td>810</td>
<td>187</td>
<td>6</td>
</tr>
<tr>
<td>840</td>
<td>194</td>
<td>7</td>
</tr>
<tr>
<td>870</td>
<td>202</td>
<td>8</td>
</tr>
<tr>
<td>900</td>
<td>208</td>
<td>6</td>
</tr>
<tr>
<td>930</td>
<td>214</td>
<td>6</td>
</tr>
</tbody>
</table>
ROUTING EXPERT SYSTEM: NO META KNOWLEDGE

WITH FAILURE RATE OF EXPON 60

<table>
<thead>
<tr>
<th>TIME INTERVAL (MIN)</th>
<th>CUMMULATIVE PRODUCED</th>
<th># OF PARTS PRODUCED</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>90</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>120</td>
<td>22</td>
<td>8</td>
</tr>
<tr>
<td>150</td>
<td>29</td>
<td>7</td>
</tr>
<tr>
<td>180</td>
<td>35</td>
<td>6</td>
</tr>
<tr>
<td>210</td>
<td>44</td>
<td>9</td>
</tr>
<tr>
<td>240</td>
<td>51</td>
<td>7</td>
</tr>
<tr>
<td>270</td>
<td>59</td>
<td>8</td>
</tr>
<tr>
<td>300</td>
<td>62</td>
<td>3</td>
</tr>
<tr>
<td>330</td>
<td>71</td>
<td>9</td>
</tr>
<tr>
<td>360</td>
<td>76</td>
<td>5</td>
</tr>
<tr>
<td>390</td>
<td>82</td>
<td>6</td>
</tr>
<tr>
<td>420</td>
<td>87</td>
<td>5</td>
</tr>
<tr>
<td>450</td>
<td>92</td>
<td>5</td>
</tr>
<tr>
<td>480</td>
<td>98</td>
<td>6</td>
</tr>
<tr>
<td>510</td>
<td>107</td>
<td>9</td>
</tr>
<tr>
<td>540</td>
<td>115</td>
<td>8</td>
</tr>
<tr>
<td>570</td>
<td>121</td>
<td>6</td>
</tr>
<tr>
<td>600</td>
<td>127</td>
<td>6</td>
</tr>
<tr>
<td>630</td>
<td>130</td>
<td>3</td>
</tr>
<tr>
<td>660</td>
<td>135</td>
<td>5</td>
</tr>
<tr>
<td>690</td>
<td>142</td>
<td>7</td>
</tr>
<tr>
<td>720</td>
<td>148</td>
<td>6</td>
</tr>
<tr>
<td>750</td>
<td>152</td>
<td>4</td>
</tr>
<tr>
<td>780</td>
<td>158</td>
<td>6</td>
</tr>
<tr>
<td>810</td>
<td>161</td>
<td>3</td>
</tr>
<tr>
<td>840</td>
<td>168</td>
<td>7</td>
</tr>
<tr>
<td>870</td>
<td>173</td>
<td>5</td>
</tr>
<tr>
<td>900</td>
<td>179</td>
<td>6</td>
</tr>
<tr>
<td>930</td>
<td>185</td>
<td>6</td>
</tr>
<tr>
<td>960</td>
<td>191</td>
<td>6</td>
</tr>
</tbody>
</table>
RANDOM ROUTING WITH META KNOWLEDGE

WITH FAILURE RATE OF EXPON 60

<table>
<thead>
<tr>
<th>TIME INTERVAL (MIN)</th>
<th>CUMMULATIVE PRODUCED</th>
<th># OF PARTS PRODUCED</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>60</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>90</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>120</td>
<td>25</td>
<td>9</td>
</tr>
<tr>
<td>150</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>180</td>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>210</td>
<td>44</td>
<td>4</td>
</tr>
<tr>
<td>240</td>
<td>53</td>
<td>9</td>
</tr>
<tr>
<td>270</td>
<td>60</td>
<td>7</td>
</tr>
<tr>
<td>300</td>
<td>67</td>
<td>7</td>
</tr>
<tr>
<td>330</td>
<td>71</td>
<td>4</td>
</tr>
<tr>
<td>360</td>
<td>77</td>
<td>6</td>
</tr>
<tr>
<td>390</td>
<td>83</td>
<td>6</td>
</tr>
<tr>
<td>420</td>
<td>91</td>
<td>8</td>
</tr>
<tr>
<td>450</td>
<td>97</td>
<td>6</td>
</tr>
<tr>
<td>480</td>
<td>102</td>
<td>5</td>
</tr>
<tr>
<td>510</td>
<td>111</td>
<td>9</td>
</tr>
<tr>
<td>540</td>
<td>119</td>
<td>8</td>
</tr>
<tr>
<td>570</td>
<td>129</td>
<td>10</td>
</tr>
<tr>
<td>600</td>
<td>136</td>
<td>7</td>
</tr>
<tr>
<td>630</td>
<td>139</td>
<td>3</td>
</tr>
<tr>
<td>660</td>
<td>146</td>
<td>7</td>
</tr>
<tr>
<td>690</td>
<td>153</td>
<td>7</td>
</tr>
<tr>
<td>720</td>
<td>162</td>
<td>9</td>
</tr>
<tr>
<td>750</td>
<td>169</td>
<td>7</td>
</tr>
<tr>
<td>780</td>
<td>177</td>
<td>8</td>
</tr>
<tr>
<td>810</td>
<td>185</td>
<td>8</td>
</tr>
<tr>
<td>840</td>
<td>192</td>
<td>7</td>
</tr>
<tr>
<td>870</td>
<td>200</td>
<td>8</td>
</tr>
<tr>
<td>900</td>
<td>206</td>
<td>6</td>
</tr>
<tr>
<td>930</td>
<td>211</td>
<td>5</td>
</tr>
</tbody>
</table>
RANDOM ROUTING POLICY WITH FAILURE RATE OF EXPON 60

<table>
<thead>
<tr>
<th>TIME INTERVAL (MIN)</th>
<th>CUMMULATIVE PRODUCED</th>
<th># OF PARTS PRODUCED</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>60</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>90</td>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>120</td>
<td>17</td>
<td>4</td>
</tr>
<tr>
<td>150</td>
<td>21</td>
<td>4</td>
</tr>
<tr>
<td>180</td>
<td>27</td>
<td>6</td>
</tr>
<tr>
<td>210</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>240</td>
<td>35</td>
<td>3</td>
</tr>
<tr>
<td>270</td>
<td>38</td>
<td>3</td>
</tr>
<tr>
<td>300</td>
<td>45</td>
<td>7</td>
</tr>
<tr>
<td>330</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>360</td>
<td>53</td>
<td>3</td>
</tr>
<tr>
<td>390</td>
<td>59</td>
<td>6</td>
</tr>
<tr>
<td>420</td>
<td>65</td>
<td>6</td>
</tr>
<tr>
<td>450</td>
<td>67</td>
<td>2</td>
</tr>
<tr>
<td>480</td>
<td>73</td>
<td>6</td>
</tr>
<tr>
<td>510</td>
<td>82</td>
<td>9</td>
</tr>
<tr>
<td>540</td>
<td>88</td>
<td>6</td>
</tr>
<tr>
<td>570</td>
<td>95</td>
<td>7</td>
</tr>
<tr>
<td>600</td>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>630</td>
<td>103</td>
<td>3</td>
</tr>
<tr>
<td>660</td>
<td>108</td>
<td>5</td>
</tr>
<tr>
<td>690</td>
<td>115</td>
<td>7</td>
</tr>
<tr>
<td>720</td>
<td>121</td>
<td>6</td>
</tr>
<tr>
<td>750</td>
<td>127</td>
<td>6</td>
</tr>
<tr>
<td>780</td>
<td>130</td>
<td>3</td>
</tr>
<tr>
<td>810</td>
<td>137</td>
<td>7</td>
</tr>
<tr>
<td>840</td>
<td>144</td>
<td>7</td>
</tr>
<tr>
<td>870</td>
<td>147</td>
<td>3</td>
</tr>
<tr>
<td>900</td>
<td>154</td>
<td>7</td>
</tr>
<tr>
<td>930</td>
<td>162</td>
<td>8</td>
</tr>
<tr>
<td>960</td>
<td>168</td>
<td>6</td>
</tr>
</tbody>
</table>
APPENDIX B

LISTINGS OF THE ROUTING EXPERT SYSTEM
This expert system is written in PROLOG-2 and is run on an IBM PC-AT. It consists of three parts:
(1) A Heuristic Knowledge Part
(2) A Meta Knowledge Part, and
(3) A Knowledge Base for the CIMS

The Heuristic Knowledge is composed of two parts:
(1) A Best First Search Algorithm or A* Algorithm
(2) A Problem Specific Knowledge for Routing

best_routes(Start, Solution):-
biggest(Big),
proveed([], leaf(Start, 0/0), Big, _, yes, S),
reverse(S, Sol),
decision(Sol, Solution).
/* Decision for the routes */

decision([],[]).

decision(Sol,Solution):- /* Decision for the routes */
   efface(First,Sol,Rest),
   arg(2,First,S1),
   decision(Rest,S2),
   append(S1,S2,Solution).

/* proceed(P, Tree, Bound, Tree1, Solved, Solution). 
expands a current subtree as long as the f-value of this tree 
remains less or equal to Bound. The arguments of proceed are: 
P Path between the start node and Tree 
Tree Current search tree 
Bound f-limit for expansion of Tree 
Tree1 Tree expanded within Bound and the f-value of Tree1 
is greater than Bound 
Solved Indicator whose value is 'yes', 'no' or 'never' 
Solution A solution path from the start node to the goal node */

proceed(P,leaf(N,-),_,_,yes,[N:P]):- goal(N).

proceed(P,leaf(N,F/G),Bound,Tree1,Solved,Solution):-
   F <= Bound,
   /* The predicate "bagof" generates successor nodes of N together 
with the costs of the arcs between N and the successor nodes */
   bagof(M/Cost,(s(N,M,Cost),not member(M,P)),Succ),!,
   successor_list(G,Succ,Ts), /* Competing list of subtrees */
   bestf_value(Ts,F1), /* Best f-value for each subtree */
   proceed(P,t(N,F1/G,Ts),Bound,Tree1,Solved,Solution);
   Solved = never). /* No successors - dead end */

proceed(P,t(N,F/G,[T:Ts]),Bound,Tree1,Solved,Solution):-
   F <= Bound,
   bestf_value(Ts,BF),
   min(Bound,BF,Bound1),
   proceed([N:P],T,Bound1,T1,Solved1,Solution),
   continue(P,t(N,F/G,[T1:Ts]),Bound,Tree1,Solved1,Solved,Solution).

proceed(_,t(_,_,[]),_,_,never,_)!.

proceed(_,Tree,Bound,Tree,no,_):-
   f(Tree,F), F > Bound. /* Cannot grow - bound exceeded */

/* The predicate "continue" will continue the search if no solution 
is found, otherwise a solution is returned */

continue(_,_,_,_,yes,yes,Solution).

continue(P,t(N,F/G,[T1:Ts]),Bound,Tree1,Solved1,Solved,Solution):-
   (Solved1 = no, insert(T1,Ts,NTs);
    Solved1 = never, NTs = Ts),
   bestf_value(NTs,F1),
   proceed(P,t(N,F1/G,NTs),Bound,Tree1,Solved,Solution).
This predicate makes a list of subtrees from the successor nodes and computes their f-value and g-value.*/

successor_list([],[],[]).

successor_list(G0,[N*B/C:NCs],Ts):-
  G is G0 + C,
  h(N,H),                  /* Heuristic term h(N) */
  F is G + H,
  successor_list(G0,NCs,Ts1),
  insert(leaf(N*B,F/G),Ts1,Ts).

insert(T,Ts,[T:Ts]):-  /* Insert a leaf into a tree */
  f(T,F),
  bestf_value(Ts,F1),
  F =< F1, !.

insert(T,[T1:Ts],[T1:Ts1]):-  insert(T,Ts,Ts1).

/* Extract f-value */

f(leaf(_,F/_),F).                /* f-value of a leaf */

f(t(_,F/_,_),F).                 /* f-value of a tree */

bestf_value([T:_],F):-
  f(T,F).                         /* Best f-value of a list of trees */

bestf_value([],Big):-
  biggest(Big).                  /* No trees: bad f-value */

min(X,Y,X):-  X =< Y, !.

min(X,Y,Y).

/* Problem Specific Knowledge for Routing */
/* Rules for successor relationship of partial routes */

s(Process~machine*~,ProMach+B~Cost):-
  efface(P1*MList,Process_machine,ProMach),
  member(M/T,MList),
  queuelength(M,Wait_time),
  attach(P1*M/T,A,B),
  Cost is Wait_time + T.

queuelength(Machine,0):-
  queue(Machine,[]).

queuelength(Machine,Totaltime):-
  queue(Machine,WaitingList),
  last(_,_,/Totaltime,WaitingList).
attach(P=M/T,[1],[P=M/T]).

goal(_*[inspection*+/._]).    */ The goal node */

/* Rules for heuristic estimates of the minimum due time of part */

h(Pro,Due_time):-
current_time(T),
sum_minptime(Pro,MinP),
sum_min_qtime(Pro,MinQ),
Due_time is (MinQ + MinP + T).

sum_minptime([],0).  /* Sum of minimum process time */

sum_minptime([Pro:Processes],MinP):-
  arg(2,Pro,A),
  minP(A,T),
  sum_minptime(Processes,Ptime),
  MinP is Ptime + T.

minP([_/T1:Rest],T):-     /* Minimum process time */
  minP(Rest,T),
  minimum(T,T1).

minP([_/T:_],T).

sum_min_qtime([],0).  /* Sum of minimum queue time */

sum_min_qtime([Pro:Processes],MinQ):-
  arg(2,Pro,B),
  minQ(B,T),
  sum_min_qtime(Processes,Qtime),
  MinQ is Qtime + T.

minQ([M_/Rest],T):-    /* Minimum queue time */
  queue_length(M,T1),
  minQ(Rest,T),
  minimum(T,T1).

minQ([M_/:_],T):-
  queue_length(M,T).

minimum(X,Y):-
  X =< Y.

/*****  The Meta Knowledge  *****/

/ *  The Meta Knowledge is composed of three parts:  *
(1) Pattern-directed Modules
(2) A Meta Interpreter
(3) Meta Rules for new arrivals of parts  */
/* Pattern Directed Modules */

/* This module says that if a machine is down and the estimated repair time is greater than the waiting time in the alternative machine, then reroute the parts to the alternative machine. */

machine_status(Machine, down),
down_time(Machine, T),
queue(Machine, List),
last(Part/P/\_\_\_, List),
part_process(Part, P_list),
arg(1, P_list, L1),
member(P\*L, L),
member(M/T\_\_L),
M \= Machine,
not(machine_status(M, down)),
queue_length(M, Wait_time),
Wait_time < T -->

[not(current_process(Part, \_\_\_\_)),
retract(processes(Part, Pro)),
effect(P\*Machine/\_\_\_, Pro, Processes),
adding(P\*M/T\_\_\_, Processes, NewPro),
asserta(processes(Part, NewPro)),
retract(queue(Machine, List)),
effect(Part/P/\_\_\_, List, List\_\_),
asserta(queue(Machine, List\_\_)),
check_schedule(Part, P\*M/T\_\_),relax].

/* This module says that if, after all the parts have been introduced to the system, there is a machine in the system starving for parts, then check the load of the other machines. If there is a part that can be rerouted to the starved machine, then reroute the part to the starved machine. */

[(robot(M, idle);
queue(M, [])),
M \= i,
queue(Machine, List),
length(List, N),
N >= 1] -->

[last(Part/Pro/\_\_\_, List),
not(current_process(Part, \_\_\_\_)),
part_process(Part, P_list),
arg(1, P_list, L1),
member(Pro\*L, L1),
member(M/T\_\_L),
retract(queue(Machine, List)),
effect(Part/Pro/\_\_\_, List, NewList),
asserta(queue(Machine, NewList)),
retract(processes(Part, Process)),
effect(Pro\*Machine/\_\_\_, Process, P1),
adding(Pro\*M/T\_\_, P1, NewPro),
asserta(processes(Part, NewPro)),
check_schedule(Part, Pro\*M/T\_),relax].
/* This module says if the queue length of the machine has reached
the maximum buffer size of the machine then supply the machine
with the shortest processing time in the queue of machine in
order to reduce the queue length */

[queue(Machine,List),
 length(List,N),
  buffer_size(Machine,Size),
  N = Size] --->
[supply_part_with_spt(Part,MACHINE),
 efface(Part,List,List1),
   adding(Part,List1,List2),
   retract(queue(Machine,List)),
   asserta(queue(Machine,List2)),relax].

/* This module says if the queue length of the machine is greater
than the maximum buffer size capacity, then reroute the last
part of the machine to alternate machine with the minimum cost */

[queue(Machine,List),
 length(List,N),
  buffer_size(Machine,Size),
  Machine \= i,
  N > Size,
  last(Part/Pr/...,List),
  part_process(Part,P_list),
  arg(1,P_list,L1),
  member(Pro*L,L1),
  member(M/T1,L),
  M \= Machine,
  buffer_size(M,Buffer),
  queue(M,Q),
  length(Q,Q_size),
  Q_size < Buffer] --->
  [not(current_process(Part,_,_)),
   retract(processes(Part,P)),
   efface(Pro*MACHINE/_,P,Processes),
   adding(Pro*M/T1,Processes,NewPro),
   asserta(processes(Part,NewPro)),
   retract(queue(Machine,List)),
   efface(Part/Pro/...,List,List1),
   asserta(queue(Machine,List1)),
   check_schedule(Part,Pro*M/T1),relax].

/* If no other things need to be considered, then RELAXxxx... but
keep watching!!! */

[] --- [relax].

/* Meta Interpreter for the pattern-directed modules */

run:-(current_time(Time),
  Time > 1,
Condition \rightarrow Action,
test(Condition),
execute(Action);true),!.

test([]).           /* Test the conditions */

/test([First:Rest]):-
    call(First),
    test(Rest).

execute([relax]):!.
/* Execute the actions */

execute([]):-!,run.

execute([First:Rest]):-
    call(First),
    execute(Rest).

/* These meta rules will assign appropriate time for the
   introduction of new arrival to the system based on the
   system load and failure rate observed */

assign_value(Value):-
    down_number(X),
    system_load(Y),
    Value is fix(X + Y/3) + 1,!.

system_load(Load):-
    findall(L,queue(M,L),List),
    flatten(List,NewList),
    length(NewList,Load),!.

/* This is a second-order predicate that finds all the elements
   that satisfy the Goal and collect them into a list */

findall(X,Goal,ItemList):-
    call(Goal),
    assert(stack(X)),
    fail;
    assert(stack(bottom)),
    collect(ItemList).

collect(L):-
    retract(stack(X)),!,
    (X == bottom,!,
    L = [];
    L = [X:Rest],
    collect(Rest)).

supply_part_with_spt(Part,Machine):-
    queue(Machine,List),
    spt(Part,List),!.
/* Rule to find the shortest processing time of parts */
spt(Part/Process/PT/Wt.[P/Pro/T/W:Rest]):-
spt(Part/Process/PT/Wt,Rest).
   less(Part/Process/PT/Wt,P/Pro/T/W).
spt(Part/P/PT/W,[Part/P/PT/W:_.]).
less(.,_/PT/_,_/_/T/_,):
   PT =< T.
adding(X,[],X)).
adding(X,[A:Rest],[X,A:Rest]).

/*** The Knowledge Base ***/
/* The Knowledge Base is composed of two parts:
   (1) A Static Knowledge Base
   (2) A Dynamic Knowledge Base */

/*** The Static Knowledge Base ***/
/* Machine code for the machining centers */

machine_code(horizontal_milling_center_1,h1).
machine_code(horizontal_milling_center_2,h2).
machine_code(vertical_milling_center,v).
machine_code(turning_center_1,t1).
machine_code(turning_center_2,t2).
machine_code(automated_inspection_center,i).
machine_code(automated>Loading_center,l).

/* The process plans of the parts */

part_process(a,[milling*[h1/6,h2/5,v/5],
   reaming*[h1/6,h2/6,v/5],
   grooving*[t1/9,t2/9],
   inspection*[i/3][loading*1/1]).

part_process(b,[boring*[h1/B,h2/B,v/11,t1/9,t2/10],
   surface_grinding*[h1/B,h2/B],
   inspection*[i/5][loading*1/1]).

part_process(c,[turning*[t1/14,t2/11],
   inspection*[i/5][loading*1/1]).

part_process(d,[milling*[h1/8,h2/9],
   slotting*[h1/4,h2/4,v/3,t1/6,t2/6],
   inspection*[i/3][loading*1/1]).

part_process(e,[boring*[v/3,t1/3,t2/2],
   slotting*[h1/9,h2/9,t1/6,t2/6],
   inspection*[i/5][loading*1/1]).
part_process(f,[turning*[t1/7,t2/7],
milling*[h1/7,h2/7,v/9,t1/6,t2/6],
inspection*[i/6])*loading*1/1).

part_process(g,[drilling*[h1/9,h2/9,v/8,t1/9,t2/9],
reaming*[h1/5,h2/5,v/6],
inspection*[i/3])*loading*1/4).

part_process(h,[surface_grinding*[h1/8,h2/8],
grooving*[t1/9,t2/8],
slotting*[h1/10,h2/10,t1/8,t2/8],
inspection*[i/2])*loading*1/1).

part_process(i,[turning*[t1/8,t2/8],
reaming*[h1/10,h2/10,v/8],
inspection*[i/3])*loading*1/1).

part_process(j,[milling*[h1/9,h2/10,v/10],
surface_grinding*[h1/6,h2/7],
inspection*[i/3])*loading*1/1).

part_process(k,[turning*[t1/8,t2/8],
reaming*[h1/10,h2/8,v/9],
grooving*[t1/9,t2/8],
inspection*[i/2])*loading*1/1).

part_process(l,[milling*[h1/9,h2/8,v/11],
drilling*[h1/4,h2/4,v/5,t1/5,t2/5],
inspection*[i/2])*loading*1/1).

part_process(m,[turning*[t1/9,t2/10],
tapping*[h1/6,h2/6,t1/9,t2/9],
surface_grinding*[h1/4,h2/3],
grooving*[t1/4,t2/3],
inspection*[i/4])*loading*1/1).

part_process(n,[milling*[h1/6,h2/6,v/6],
reaming*[h1/7,h2/7,v/5],
inspection*[i/3])*loading*1/1).

part_process(o,[turning*[t1/6,t2/8],
threading*[t1/4,t2/3],
inspection*[i/1])*loading*1/1).

/* Buffer size of the machining centers */

buffer_size(h1,4).
buffer_size(h2,4).
buffer_size(v,4).
buffer_size(t1,4).
buffer_size(t2,4).
buffer_size(i,4).
/* End of Simulation Time */
end_of_simulation_time(240).

/** The Dynamic Knowledge Base **/

/* Initial conditions of the system */
/* Contents of queue at each machine */

queue(h1,[]).
queue(h2,[]).
queue(v,[]).
queue(t1,[]).
queue(t2,[]).
queue(i,[]).

/* Machine Status */
machine_status(h1,up).
machine_status(h2,up).
machine_status(v,up).
machine_status(t1,up).
machine_status(t2,up).
machine_status(i,up).

/* Robot Status */
robot(h1,idle).
robot(h2,idle).
robot(v,idle).
robot(t1,idle).
robot(t2,idle).
robot(i,idle).

/* Number of Machine Down */
down_number(0).

/* Statistics Collection */
/* Cummulative queue length at each machine */

q_length(h1,0).
q_length(h2,0).
q_length(v,0).
q_length(t1,0).
q_length(t2,0).
q_length(i,0).
/** Cumulative time spent in system for each part */

cum_time_spent_in_system(a,0).
cum_time_spent_in_system(b,0).
cum_time_spent_in_system(c,0).
cum_time_spent_in_system(d,0).
cum_time_spent_in_system(e,0).
cum_time_spent_in_system(f,0).
cum_time_spent_in_system(g,0).
cum_time_spent_in_system(h,0).
cum_time_spent_in_system(i,0).
cum_time_spent_in_system(j,0).
cum_time_spent_in_system(k,0).
cum_time_spent_in_system(l,0).
cum_time_spent_in_system(m,0).
cum_time_spent_in_system(n,0).
cum_time_spent_in_system(o,0).

/** Quantity of parts produced */

quantity_produced(a,0).
quantity_produced(b,0).
quantity_produced(c,0).
quantity_produced(d,0).
quantity_produced(e,0).
quantity_produced(f,0).
quantity_produced(g,0).
quantity_produced(h,0).
quantity_produced(i,0).
quantity_produced(j,0).
quantity_produced(k,0).
quantity_produced(l,0).
quantity_produced(m,0).
quantity_produced(n,0).
quantity_produced(o,0).
APPENDIX C

LISTINGS OF THE KNOWLEDGE BASED SIMULATION MODEL
KNOWLEDGE BASED SIMULATION MODEL FOR CIMS

This Simulation Model consists of two parts:
(1) A Modeling Knowledge
(2) A Simulation Driver

Hardware Requirements for the PC:
This simulation model requires a 8087 or 80287 math co-processor for the logarithm function needed in generating exponential distribution. A hard disk is highly recommended for running the RES and the simulation model for extra storage and speed.

**********************************************************************************

/**** The Simulation Driver  ****/
/*
* The Simulation Driver is composed of five parts:
* (1) A simulation clock
* (2) An event file
* (3) Rules to perform a simulation run
* (4) Rules to perform an event in the event file
* (5) Rules for sorting the event file */
/*
* The simulation clock */
current_time(0).
/* The event file */

event_list([l*[part_arrival,a],l*[part_arrival,b],l*[part_arrival,c], l*[part_arrival,d],l*[part_arrival,e],l*[part_arrival,f], l*[part_arrival,g],l*[part_arrival,h],l*[part_arrival,i], l*[part_arrival,j],l*[part_arrival,k],l*[part_arrival,l], l*[part_arrival,m],l*[part_arrival,n],l*[part_arrival,o]]).

/* Rules to perform the simulation run */
simulate:-
  repeat,
  schedule_machine_failure,
  perform_event,
  run,          /* Run the Meta Knowledge */
  (end_of_simulation_time(T),
   current_time(T1),
   T1 >= T,!)
write("wait!"), tab(1), fail).

/* Schedule machine failure event at the beginning of simulation */
schedule_machine_failure:-
current_time(T),
T \= 0, !;
machine_is_operational(h1),
machine_is_operational(h2),
machine_is_operational(v),
machine_is_operational(t1),
machine_is_operational(t2), !.

/* Rules for performing an event in the event file */
perform_event:-
  retract(event_list(List)),
  remove(Event,List,Rest),
  asserta(event_list(Rest)),
  arg(1,Event,T),
  arg(2,Event,X),
  update_current_time(T),
  G = .. X,
  call(G), !.

remove(First,[First],[]).
remove(First,[First:Rest],Rest):- !.

update_current_time(T):-
  retract(current_time(T1)),
  asserta(current_time(T)),
  Elapsed_time is T - T1,
  update_down_time(Elapsed_time),
  update_queue_time(Elapsed_time), !.

update_down_time(T):-
  retract(down_time(M,T1)),
  T2 is T1 - T,
  asserta(down_time(M,T2)),
  fail;true, !.

update_queue_time(T):-
  retract(queue(Machine,List)),
  update(List,T,NewList),
  asserta(queue(Machine,NewList)),
  retract(q_length(Machine,L)),
  length(NewList,N),
  L1 is L + (N*T),
  asserta(q_length(Machine,L1)),
  fail;true, !.

update([],_,[]).
update([Part/Process/Pt/Wt:Wt1,L,T],[Part/Process/Pt/Wt1:L1]):-
    Wt1 is Wt - T,
    update(L,T,L1).
/* Rules for sorting the event list */
quicksort(List,Sorted):-
    quicksort2(List,Sorted-[]).
quicksort2([],Z-Z).
quicksort2([X:Tail],A1-Z2):-
    split1(X,Tail,Small,Big),
quicksort2(Small,A1-[X:A2]),
quicksort2(Big,A2-Z2).
split1(X,[Y:],[],[]).
split1(X,[Y:Tail],[Y:Small],Big):-
gt(X,Y),!,
split1(X,Tail,Small,Big).
split1(X,[Y:Tail],Small,[Y:Big]):-
    split1(X,Tail,Small,Big).
gt(X*,Y*):- X > Y.

/*** The Modeling Knowledge ***/
/* The Modeling Knowledge is composed of four parts: 
(1) Rules for part arrival event 
(2) Rules for process finish event 
(3) Rules for machine breakdown event 
(4) Rules for machine back to operational event */

/* Rules for Part Arrival Event */
part_arrival(Part):-
    find_processes(Part,Processes),
    best_routes(Processes,Routes), /* From the Heuristic Knowledge */
    arg2(Routes,RestRoutes),
    asserta(processes(Part,RestRoutes)),
    record_arrival_time(Part),
    find_machine(Part,Machine), /* Start to route */
    check_schedule(Part,Machine),!.

find_processes(Part,Processes):-
    part_process(Part,Processes),true,!.

record_arrival_time(Part):-
    current_time(T),
    assert(arrival_time(Part,T)).
add_to_queue(Part,Pro*MachinelT):-
    retract(queue(Machine,Cl)),
    asserta(queue(Machine,CPart/Pro/T/T3))
retract(queue(Machine,List)),
    last(-/-/-/Wtl,List),
    Wt is Wtl + T,
    add(Part/Pro/T/Wt,List,Listl),
    asserta(queue(Machine,Listl))).

create~current~process(Part~Proces~*Machine/~~:-
    machine-status(Machine,up),
    retract(robot(Machine,idle)),
    asserta(robot(Machine,busy)),
    asserta(current~process(Part,Process,Machine, humiliad machine, idle),
    asserta(current~process(Part,Process,Machine)),
    (not queue(Machine,[]),
    retract(queue(Machine,List)),
    efface(Part/Process/__/List,List)),
    true),!.
    asserta(queue(Machine,List))).
asserta(queue(Machine,Listl)));
asserta(queue(Machine,Listl))
/* Rules for Process Finish Event */

process~finish(Part,Process*Machine):-
    retract(processes(Part,Process*Machine/__)), /* Last process */
    remove_from_machine(Process*Machine),
    schedule_next_part(Process*Machine),
    request_new_part(Part),
    time_spent_in_system(Part), /* Collect statistics */
    update_production_rate(Part),!.

    time_spent_in_system(Part):-
    retract(arrival_time(Part,T)),
    current_time(Time),
    Total_time_used is Time - T,
    retract(cum_time_spent_in_system(Part,Cum_time)),
    Latest_cum_time is Cum_time + Total_time_used,
    asserta(cum_time_spent_in_system(Part,Latest_cum_time)).

update_production_rate(Part):-
    retract(quantity_produced(Part,Rate)),
    NewRate is Rate + 1,
    asserta(quantity_produced(Part,NewRate)),!.

process~finish(Part,Machine):-
    remove_from_machine(Machine),
    schedule_next_part(Machine),
    find_next_machine(Part,NextM),
    check_schedule(Part,NextM),!.
find_next_machine(Part, Pro*Machine/T) :-
  retract(processes(Part, Processes)),
  remove(_, Processes, Remaining_processes),
  next_process_machine(Pro*Machine/T, Remaining_processes),
  asserta(processes(Part, Remaining_processes)), !.

check_schedule(Part, Machine) :-
  (create_current_process(Part, Machine),
   schedule_end_process(Part, Machine);
   add_to_queue(Part, Machine)), !.

remove_from_machine(Pro*Machine) :-
  retract(current_process(_, Pro, Machine)),
  retract(robot(Machine, busy)),
  asserta(robot(Machine, idle)), !.

schedule_next_part(_*Machine) :-
  queue(Machine, t), true.

schedule_next_part(_*Machine) :-
  queue(Machine, List),
  next_part(Nextpart, List),
  find_next_process(Nextpart, ProM),
  create_current_process(Nextpart, ProM),
  schedule_end_process(Nextpart, ProM), !.

next_part(X, [X/_, /_/i_/i_]).

find_next_process(Part, Process*Machine/T) :-
  processes(Part, List),
  next_process_machine(Process*Machine/T, List), !.

next_process_machine(P*M/T, [P*M/T]_).}

schedule_end_process(Part, Process*Machine/T) :-
  current_time(Current_time),
  End_time is Current_time + T,
  retract(event_list(List)),
  add(End_time*[process_finish, Part, Process*Machine], List, List1),
  quicksort(List1, Sorted),
  asserta(event_list(Sorted)), !.

request_new_part(Part) :-
  current_time(T),
  assign_value(Value), /* Assigned by the Meta Knowledge */
  Arrival_time is T + Value,
  retract(event_list(List)),
  add(Arrival_time*[part_arrival, Part], List, List1),
  quicksort(List1, Sorted),
  asserta(event_list(Sorted)), !.
/* Rules for Machine Breakdown Event */

machine_failure(Machine):-
exponent(X),
Down_time is fix(20 * X),
asserta(down_time(Machine, Down_time)),
machine_is_down(Machine, Down_time),
current_time(T),
Total_down_time is T + Down_time,
retract(event_list(List)),
add(Total_down_time*[machine_is_operational, Machine], List, List1),
quicksort(List1, Sorted),
asserta(event_list(Sorted)),!.

machine_is_down(Machine, Time):-
update_machine_status(Machine),
delay_queue_time(Machine, Time),
retract(down_number(X)),
NewX is X + 1,
asserta(down_number(NewX)),
(retract(robot(Machine, busy)),
current_process(Part, Process, Machine),
retract(event_list(E_List)),
efface(T*[process_finish, Part, Process*Machine], E_List, List1),
New_time is T + Time,
add(New_time*[process_finish, Part, Process*Machine], List1, List2),
quicksort(List2, Sorted),
asserta(event_list(Sorted));
retract(robot(Machine, idle))),!.

update_machine_status(Machine):-
retract(machine_status(Machine, up)),
asserta(machine_status(Machine, down)).

delay_queue_time(Machine, Time):-
retract(queue(Machine, List)),
delay(List, Time, NewList),
asserta(queue(Machine, NewList)),!.

delay([], []).

delay([Part/Pro/T/Wt:L], Down_time, [Part/Pro/T/Wt1:L1]):-
Wt1 is Wt + Down_time,
delay(L, Down_time, L1).

/* Rules for Machine Back to Operational Event */

machine_is_operational(Machine):-
not current_time(0),
machine_is_up(Machine),
exponent(X),
Operational_time is fix(60 * X),
current_time(T),
Total_operational_time is T + Operational_time,
retract(event_list(List)),
add(Total_operational_time*[machine_failure, Machine], List, List1),
quicksort(List1, Sorted),
asserta(event_list(Sorted)),!.
machine_is_operational(Machine):-
  exon(X),
  Operational_time is fix(60 * X),
  retract(event_list(List)),
  add(Operational_time*[machine_failure,Machine],List,List1),
  quicksort(List1,Sorted),
  asserta(event_list(Sorted)),!.

machine_is_up(Machine):-
  retract(machine_status(Machine,down)),
  asserta(machine_status(Machine,up)),
  retract(down_number(X)),
  NewX is X - 1,
  asserta(down_number(NewX)),
  retract(down_time(Machine,_,)),
  (current_process(_,_,Machine),
  asserta(robot(Machine,busy)));
asserta(robot(Machine, idle)),
  schedule_next_part(*Machine)),!.

add(X,[],[X]).
add(X,[A:List],[A:List1]):-
  add(X,List,List1).

/* Exponential Distribution */
exon(X):-
  random(R),
  X is -log(1-R), !.

/* Random Number Generation */
seed(199).

random(X):-
  retract(seed(S)),
  N is ((S * 181) mod 32768) + 1,
  asserta(seed(N)),
  X is N/32768, !.
APPENDIX D

LISTINGS OF THE RANDOM ROUTING POLICY
RANDOM ROUTING POLICY

In this routing policy, the heuristic knowledge and the meta knowledge (except the meta rule for new arrivals of parts) portions of the Routing Expert System are disconnected. Only the knowledge base of the expert system is available. The routes for this policy are generated arbitrarily using a random number generator for the knowledge based simulation model.

The Knowledge Base: Static

Machine code used in the knowledge base

machine_code(horizontal_milling_center_1,h1).
machine_code(horizontal_milling_center_2,h2).
machine_code(vertical_milling_center,v).
machine_code(turning_center_1,t1).
machine_code(turning_center_2,t2).
machine_code(automated_inspection_center,i).
machine_code(automated>Loading_center_1).

This is the process plans for the parts

part_process(a,[milling*[h1/6,h2/5,v/5],
               reaming*[h1/6,h2/6,v/5],
               grooving*[t1/9,t2/9],
               inspection*[i/3]*[loading*l/1]]).

part_process(b,[boring*[h1/8,h2/8,v/11,t1/9,t2/10],
               surface_grinding*[h1/8,h2/8],
               inspection*[i/5]*[loading*l/1]]).

part_process(c,[turning*[t1/14,t2/11],
               inspection*[i/5]*[loading*l/1]]).

part_process(d,[milling*[h1/8,h2/9],
               slotting*[h1/4,h2/4,v/3,t1/6,t2/6],
               inspection*[i/3]*[loading*l/1]]).

part_process(e,[boring*[v/3,t1/3,t2/2],
               slotting*[h1/9,h2/9,t1/6,t2/6],
               inspection*[i/5]*[loading*l/1]].
part_process(f,[turning*[t1/7,t2/7],
milling*[h1/7,h2/7,v/9,t1/6,t2/6],
inspection*[i/6]]*[loading*1/1]).

part_process(g,[drilling*[h1/9,h2/9,v/8,t1/9,t2/9],
reaming*[h1/5,h2/5,v/6],
inspection*[i/3]]*[loading*1/4]).

part_process(h,[surface_grinding*[h1/8,h2/8],
grooving*[t1/9,t2/8],
slotting*[h1/10,h2/10,t1/8,t2/8],
inspection*[i/2]]*[loading*1/1]).

part_process(i,[turning*[t1/8,t2/8],
reaming*[h1/10,h2/10,v/8],
inspection*[i/3]]*[loading*1/1]).

part_process(j,[milling*[h1/9,h2/10,v/10],
surface_grinding*[h1/6,h2/7],
inspection*[i/3]]*[loading*1/1]).

part_process(k,[turning*[t1/8,t2/8],
reaming*[h1/10,h2/8,v/9],
grooving*[t1/9,t2/8],
inspection*[i/2]]*[loading*1/1]).

part_process(l,[milling*[h1/9,h2/8,v/11],
drilling*[h1/4,h2/4,v/5,t1/5,t2/5],
inspection*[i/2]]*[loading*1/1]).

part_process(m,[turning*[t1/9,t2/10],
tapping*[h1/6,h2/6,t1/9,t2/9],
surface_grinding*[h1/4,h2/3],
grooving*[t1/4,t2/3],
inspection*[i/4]]*[loading*1/1]).

part_process(n,[milling*[h1/6,h2/6,v/6],
reaming*[h1/7,h2/7,v/5],
inspection*[i/3]]*[loading*1/1]).

part_process(o,[turning*[t1/6,t2/8],
threading*[t1/4,t2/3],
inspection*[i/1]]*[loading*1/1]).

/* Buffer Size of the Machining Centers */
buffer_size(h1,5).
buffer_size(h2,5).
buffer_size(v,5).
buffer_size(t1,5).
buffer_size(t2,5).
buffer_size(i,5).

/* End of Simulation Time */
end_of_simulation_time(240).
/* The Knowledge Base: Dynamic */

/* Initial conditions of the system */
/* Contents of queue at each machine */
queue(h1,[]).
queue(h2,[]).
queue(v,[]).
queue(t1,[]).
queue(t2,[]).
queue(i,[]).

/* Simulation Clock */
current_time(0).

/* Machine Status */
machine_status(h1,up).
machine_status(h2,up).
machine_status(v,up).
machine_status(t1,up).
machine_status(t2,up).
machine_status(i,up).

/* Robot Status */
robot(h1,idle).
robot(h2,idle).
robot(v,idle).
robot(t1,idle).
robot(t2,idle).
robot(i,idle).

/* Number of Machine Down */
down_number(0).

/* Statistics Collection */
/* Cummulative queue length at each machine */
q_length(h1,0).
q_length(h2,0).
q_length(v,0).
q_length(t1,0).
q_length(t2,0).
q_length(i,0).

/* Cummulative time spent in system for each part */
cum_time_spent_in_system(a,0). cum_time_spent_in_system(b,0).
cum_time_spent_in_system(c,0). cum_time_spent_in_system(d,0).
cum_time_spent_in_system(e,0). cum_time_spent_in_system(f,0).
cum_time_spent_in_system(g,0). cum_time_spent_in_system(h,0).
cum_time_spent_in_system(i,0). cum_time_spent_in_system(j,0).
cum_time_spent_in_system(k,0). cum_time_spent_in_system(l,0).
cum_time_spent_in_system(m,0). cum_time_spent_in_system(n,0).
cum_time_spent_in_system(o,0).

/* Quantity of parts produced */
quantity_produced(a,0). quantity_produced(b,0).
quantity_produced(c,0). quantity_produced(d,0).
quantity_produced(e,0). quantity_produced(f,0).
quantity_produced(g,0). quantity_produced(h,0).
quantity_produced(i,0). quantity_produced(j,0).
quantity_produced(k,0). quantity_produced(l,0).
quantity_produced(m,0). quantity_produced(n,0).
quantity_produced(o,0).

/* Event List */
event_list([1*[part_arrival,a],1*[part_arrival,b],1*[part_arrival,c],
1*[part_arrival,d],1*[part_arrival,e],1*[part_arrival,f],
1*[part_arrival,g],1*[part_arrival,h],1*[part_arrival,i],
1*[part_arrival,j],1*[part_arrival,k],1*[part_arrival,l],
1*[part_arrival,m],1*[part_arrival,n],1*[part_arrival,o])

*******************************************************************************
*                                                                         *
* KNOWLEDGE BASED SIMULATION MODEL FOR CIMS                                 *
*                                                                         *
* This simulation model is the same as the one before but for the reason of completeness, it is also included here for the simulation of random routing policy. *
*                                                                         *
*******************************************************************************

/* Rules to perform the simulation run */
simulate:-
    repeat,
    schedule_machine_failure,
    perform_event,
    (end_of_simulation_time(T),
     current_time(T1),
     T1 >= T,!!;
     write("wait!"),tab(1),
     fail).

/* Schedule machine failure event at the beginning of simulation */
schedule_machine_failure:-
current_time(T),
/* Rules for performing an event in the event file */

perform_event:-
  retract(event_list(List)),
  remove(Event,List,Rest),
  asserta(event_list(Rest)),
  arg(1,Event,T),
  arg(2,Event,X),
  update_current_time(T),
  G = .. X,
  call(G),!.

remove(First,[First],[]).
remove(First,[First;Rest],Rest):-!.

update_current_time(T):-
  retract(current_time(T1)),
  asserta(current_time(T)),
  Elapsed_time is T - T1,
  update_down_time(Elapsed_time),
  update_queue_time(Elapsed_time),!.

update_down_time(T):-
  (retract(down_time(M,T1)),
   T2 is T1 - T,
   asserta(down_time(M,T2)),
   fail;true),!.

update_queue_time(T):-
  (retract(queue(Machine,List)),
   update(List,T,NewList),
   asserta(queue(Machine,NewList)),
   retract(q_length(Machine,L)),
   length(NewList,N),
   L1 is L + (N*T),
   asserta(q_length(Machine,L1)),
   fail;true),!.

update([],_,[]).

update([Part/Process/Pt/Wt:1],T,[Part/Process/Pt/Wt1:L1]):-
  Wt1 is Wt - T,
  update(L,T,L1).
Rules for the Random Routing Policy

random_routes(Processes, Routes) :-
    member(P*M, Processes),
    length(M, N),
    random_route(N, Random),
    nmember(Machine/T, M, Random),
    assertz(found(P*Machine/T)),
    fail;
    assertz(found(mark)),
    collect_found(Routes).

seed1(199).

random_route(R, X) :-
    retract(seed1(S)),
    X is (S mod R) + 1,
    NewSeed is (181 * S + 1) mod 32768,
    asserta(seed1(NewSeed)), !.

collect_found(List) :-
    retract(found(X)), !,
    (X == mark, !,
     List = [],
     List = [X!Rest],
     collect_found(Rest)).

Rules for Part Arrival Event

part_arrival(Part) :-
    find_processes(Part, Processes),
    arg(1, Processes, Pro),
    random_routes(Pro, Routes), /* Found by random routing policy */
    asserta(processes(Part, Routes)),
    record_arrival_time(Part),
    find_machine(Part, Machine),
    check_schedule(Part, Machine), !.

find_processes(Part, Processes) :-
    part_process(Part, Processes), true, !.

record_arrival_time(Part) :-
    current_time(T),
    assert(arrival_time(Part, T)).

find_machine(Part, Pro*Machine/T) :-
    processes(Part, List),
    find(Pro*Machine/T, List), !.

find(Pro*M/T, [Pro*M/T!_]).

add_to_queue(Part, Pro*Machine/T) :-
    (retract(queue(Machine, [])),
     asserta(queue(Machine, [Part/Pro/T/T]));
retract(queue(Machine,List)),
last(\_/_/\_/Wt1,List),
Wt is Wt1 + T,
add(Part/Pro/T/Wt,List,List1),
asserta(queue(Machine,List1)),!.
create_current_process(Part,Process*Machine/_):-
machine_status(Machine,up),
retract(robot(Machine,idle)),
asserta(robot(Machine,busy)),
asserta(current_process(Part,Process,Machine)),
(not queue(Machine,[]),
retract(queue(Machine,List)),
efface(Part/Process/_/_/List,List1),
asserta(queue(Machine,List1));true),!.
/* Rules for Process Finish Event */
process_finish(Part,Process*Machine):-
retract(processes(Part,[Process*Machine/_])), /* Last process */
remove_from_machine(Process*Machine),
schedule_next_part(Proceess*Machine),
schedule_next_part(Process*Machine),
time_spent_in_system(Part), /* Collect statistics */
update_production_rate(Part),!.
time_spent_in_system(Part):-
retract(arrival_time(Part,T)),
current_time(Time),
Total_time_used is Time - T,
retract(cum_time_spent_in_system(Part,Cum_time)),
Latest_cum_time is Cum_time + Total_time_used,
asserta(cum_time_spent_in_system(Part,Current_cum_time)).
update_production_rate(Part):-
retract(quantity_produced(Part,Rate)),
NewRate is Rate + 1,
asserta(quantity_produced(Part,NewRate)),!.
process_finish(Part,Machine):-
remove_from_machine(Machine),
schedule_next_part(Machine),
find_next_machine(Part,NextM),
check_schedule(Part,NextM),!.
find_next_machine(Part,Pro*Machine/T):-
retract(processes(Part,Processes)),
remove(Pro*Processes,Remaining_processes),
next_process_machine(Pro*Machine/T,Remaining_processes),
asserta(processes(Part,Remaining_processes)),!.
check_schedule(Part,Machine):-
(create_current_process(Part,Machine),
schedule_end_process(Part,Machine);
add_to_queue(Part,Machine)),!.
\begin{verbatim}
remove_from_machine(Pro*Machine):-
  retract(current_process(_,Pro,Machine)),
  retract(robot(Machine,busy)),
  asserta(robot(Machine,idle)),!.

schedule_next_part(_.*Machine):-
  queue(Machine,[]),true.

schedule_next_part(_.*Machine):-
  queue(Machine,List),
  next_part(Nextpart,List),
  find_next_process(Nextpart,ProM),
  create_current_process(Nextpart,ProM),
  schedule_end_process(Nextpart,ProM),!.

next_part(X,[X/\]/_/\_\_]).

find_next_process(Part,Process*Machine/T):-
  processes(Part,List),
  next_process_machine(Process*Machine/T,List),!.

next_process_machine(P*M/T,[P*M/T:_/\_]).

schedule_end_process(Part,Process*Machine/T):-
  current_time(Current_time),
  End_time is Current_time + T,
  retract(event_list(List)),
  add(End_time*[process_finish,Part,Process*Machine],List,Listl),
  quicksort(Listl,Sorted),
  asserta(event_list(Sorted)),!.

request_new_part(Part):-
  current_time(T),
  assign_value(Value),
  Arrival_time is T + Value,
  retract(event_list(List)),
  add(Arrival_time*[part_arrival,Part],List,Listl),
  quicksort(Listl,Sorted),
  asserta(event_list(Sorted)),!.

/* Meta rules for new arrivals of parts */

assign_value(Value):-
  down_number(X),
  system_load(Y),
  Value is fix(X + Y/3) + 1,!.

system_load(Load):-
  findall(L,queue(M,L),List),
  flatten(List,NewList),
  length(NewList,Load),!.

findall(X,Goal,Xlist):-
  call(Goal),
\end{verbatim}
assertz(stack(X)),
fail;
assertz(stack(bottom)),
collect(X:list).

collect(L):-
  retract(stack(X)),!,
  (X == bottom,!,
   L = [];
   L = [X:Rest],
   collect(Rest)).

/* Rules for Machine Breakdown Event */

machine_failure(Machine):-
expon(X),
Down_time is fix(20 * X),
asserta(down_time(Machine,Down_time)),
machine_is_down(Machine,Down_time),
current_time(T),
Total_down_time is T + Down_time,
retract(event_list(List)),
add(Total_down_time*[machine_is_operational,Machine],List,List1),
 quicksort(List1,Sorted),
asserta(event_list(Sorted)),!.

machine_is_down(Machine,Time):-
update_machine_status(Machine),
delay_queue_time(Machine,Time),
retract(down_number(X)),
NewX is X + 1,
asserta(down_number(NewX)),
(retract(robot(Machine,busy)),
current_process(Part,Process,Machine),
retract(event_list(E_list)),
efface(T*[process_finish,Part,Process*Machine],E_list,List1),
New_time is T + Time,
add(New_time*[process_finish,Part,Process*Machine],List1,List2),
 quicksort(List2,Sorted),
asserta(event_list(Sorted));
retract(robot(Machine,idle))),!.

update_machine_status(Machine):-
retract(machine_status(Machine,up)),
asserta(machine_status(Machine,down)).

delay_queue_time(Machine,Time):-
retract(queue(Machine,List)),
delay(List,Time,NewList),
asserta(queue(Machine,NewList)),!.

delay([],_,[]).
delay([Part/Pro/T/Wt:L1],Down_time,[Part/Pro/T/Wt1:L1]):-
    Wt1 is Wt + Down_time,
    delay(L1,Down_time,L1).

/* Rules for Machine Back to Operational Event */

machine_is_operational(Machine):-
    not current_time(0),
    machine_is_up(Machine),
    expon(X),
    Operational_time is fix(60 * X),
    current_time(T),
    Total_operational_time is T + Operational_time,
    retract(event_list(List)),
    add(Total_operational_time*Machine_failure,Machine,List,List1),
    quicksort(List1,Sorted),
    asserta(event_list(Sorted)),!.

machine_is_operational(Machine):-
    expon(X),
    Operational_time is fix(60 * X),
    retract(event_list(List)),
    add(Operational_time*Machine_failure,Machine,List,List1),
    quicksort(List1,Sorted),
    asserta(event_list(Sorted)),!.

machine_is_up(Machine):-
    retract(machine_status(Machine,down)),
    asserta(machine_status(Machine,up)),
    retract(down_number(X)),
    NewX is X - 1,
    asserta(down_number(NewX)),
    retract(down_time(Machine,_)),
    (current_process(_,_,Machine),
    asserta(robot(Machine,busy));
    asserta(robot(Machine,idle)),
    schedule_next_part(_*Machine)),!.

add(X,[],X).

add(X,[A:List],[A:List1]):-
    add(X,List,List1).

/* Exponential Distribution */

expon(X):-
    random(R),
    X is -log(1-R), !.

/* Random Number Generation */

seed(199).

random(X):-
    retract(seed(S)),
    expon(X).
N is ((S * 181) mod 32768) + 1,
asserta(seed(N)),
X is N/32768, !.

/* Rules for sorting the event list */

quicksort(List,Sorted):-
quicksort2(List,Sorted-[]).

quicksort2([],Z-Z).

quicksort2([X;Tail],A1-Z2):-
split1(X,Tail,Small,Big),
quicksort2(Small,A1-[X:A2]),
quicksort2(Big,A2-Z2).

split1(X,[],[],[]).

split1(X,[Y;Tail],[Y;Small],Big):-
gt(X,Y),!,
split1(X,Tail,Small,Big).

split1(X,[Y;Tail],Small,[Y;Big]):-
split1(X,Tail,Small,Big).

gt(X_,Y_):- X > Y.
APPENDIX E

PREDICATES PROVIDED BY THE PROLOG-2 SYSTEM LIBRARY
PREDICATES PROVIDED BY THE PROLOG-2 SYSTEM LIBRARY

A number of the more commonly used predicates for developing the expert system and the knowledge based simulation model are provided by the PROLOG-2 system library. They are listed and defined as follows:

**member**

member is used to test for membership and to generate lists with a given thing as member. It is defined as:

```
member(Element,[Element|_]).
member(Element,[_|List]):-
    member(Element,List).
```

**nmember**

nmember is used either to extract the Nth element of a list to test for member and also get the position in the list. It is defined as:

```
nmember(Element,[Element|_],1).
nmember(Element,[_|List],Number):-
    nmember(Element,List,Num_sofar),
    Number is Num_sofar + 1.
```

**last**

last is used to extract the last element of a list. It is defined as:

```
last(Element,[Element]).
last(Element,[_|Rest]):-
    last(Element,Rest).
```

**append**

append is used to join two lists together. It is defined as:

```
append([],List,List).
append([Head|Tail],List,[Head|List1]):-
    append(Tail,List,List1).
```
efface  efface is used to remove an element from a list. To be precise, it removes the first occurrence of an element desired to be removed in a list. It is not resatisfiable. It is defined as:

\[ \text{efface}(\text{Element}, [\text{Element}; \text{Rest}], \text{Rest}) : - !. \]

\[ \text{efface}(\text{Element}, [\text{Element}, \text{List1}, \text{List1}, \text{List2}]): - \text{efface}(\text{Element}, \text{List1}, \text{List2}). \]

flatten  flatten is used to flatten a list that contains other lists as members. The process of flattening a list involves adding all members of such lists to the main list in place of the lists. It is defined as:

\[ \text{flatten}([], []) : - !. \]

\[ \text{flatten}([\text{Head}; \text{Rest}], [\text{Head}; \text{Newlist}]): - \\
\text{not islist(Head),}
\quad \text{flatten(Rest, Newlist)}, !. \]

\[ \text{flatten}([\text{Head}; \text{Rest}], \text{Newlist}): - \\
\quad \text{flatten(Head, List1)}, \\
\quad \text{flatten(Rest, List2)}, \\
\quad \text{append(List1, List2, Newlist)}, !. \]

reverse  reverse is used for reversing a list. It accumulates the reversed list by using a subsidiary predicate and is thus able to access lists entirely from the front. It is defined as:

\[ \text{reverse}([\text{List1}, \text{List2}]): - \\
\text{qrev(\text{List1}, [], \text{List2})}. \]

\[ \text{qrev}([\text{Head}; \text{Tail}], \text{Sofar}, \text{Result}): - \\
\text{qrev(\text{Tail}, [\text{Head}; \text{Sofar}], \text{Result})}. \]

\[ \text{qrev}([], \text{Result}, \text{Result}). \]
APPENDIX F

PROCESS PLANS FOR THE PARTS
### PROCESS PLANS

<table>
<thead>
<tr>
<th>PART</th>
<th>PROCESS</th>
<th>MACHINE WITH PROCESSING TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>H1</td>
</tr>
<tr>
<td>A</td>
<td>MILLING</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>REAMING</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GROOVING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>BORING</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>GRINDING</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>TURNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inspection</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>MILLING</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>SLOTTING</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>BORING</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>SLOTTING</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>TURNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MILLING</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>DRILLING</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>REAMING</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>GRINDING</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>GROOVING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SLOTTING</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>TURNING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REAMING</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>MILLING</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>GRINDING</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TURNING</td>
<td>10</td>
</tr>
<tr>
<td>----</td>
<td>---------</td>
<td>----</td>
</tr>
<tr>
<td>K</td>
<td>REAMING</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>GROOVING</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---------</td>
<td>----</td>
</tr>
<tr>
<td>L</td>
<td>MILLING</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>DRILLING</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---------</td>
<td>----</td>
</tr>
<tr>
<td>M</td>
<td>TURNING</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>TAPPING</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>GRINDING</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>GROOVING</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---------</td>
<td>----</td>
</tr>
<tr>
<td>N</td>
<td>MILLING</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>REAMING</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---------</td>
<td>----</td>
</tr>
<tr>
<td>O</td>
<td>TURNING</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>THREADING</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>INSPECTION</td>
<td></td>
</tr>
</tbody>
</table>
ABSTRACT

This research introduces the RES, a Routing Expert System for solving part routing problems in computer integrated manufacturing systems (CIMS). The RES consists of three parts: (1) knowledge base, (2) heuristic knowledge, and (3) meta knowledge. The knowledge base contains all the static and dynamic information of the CIMS. The heuristic knowledge consists of a heuristic search strategy that decides the best routes for parts in real time. The meta knowledge acts like a human supervisor. It assesses the performance and decisions made by the heuristic knowledge, and based on the behavior the system, may make overriding decisions so that the best performance of the system can be achieved. In order to implement the RES, a knowledge based simulation model for the CIMS is also developed. Both the RES and the knowledge based simulation model are written in PROLOG and are run on an IBM PC-AT. When compared to random routing policy, the RES is clearly far more superior even when the meta knowledge part of the RES is disconnected. An interesting result observed is that when the random routing policy is added with the meta knowledge, significant improvements of the system can be achieved. The expert system developed has the potential to be used for the control system of a truly unmanned CIMS.