AN INTEGRATED APPROACH TO THE OPTIMAL SEQUENCING OF ROBOT OPERATIONS IN A WORKCELL

by

Chetan J. Desai

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Industrial Engineering and Operations Research

APPROVED:

Subhash C. Sarin

S. C. Sarin, Ph.D, Chairman

C. C. Wang, Ph.D

A. Myklebust, Ph.D

Febuary 1, 1988

Blacksburg, Virginia

AN INTEGRATED APPROACH TO THE OPTIMAL SEQUENCING OF ROBOT OPERATIONS IN A WORKCELL

by

Chetan J. Desai

S. C. Sarin, Ph.D, Chairman

Industrial Engineering and Operations Research

(ABSTRACT)

In this research we develop an integrated approach to optimally sequence robot operation in a workcell. The workcell represents a flowshop operation with multiple robots transporting jobs among machines. A buffer of infinite capacity is available ahead of each machine. A robot transports jobs from buffers to machines and from machines to buffers. The robots used in the system are 5 joint cylindrical coordinate robots. All the robots are identical in design and capability. For the type of robot used in this study its closed form inverse kinematic solution is known. The objective is to determine the sequence of robot operations so as to minimize the total time needed to complete all jobs (known as makespan).

The integrated approach consists of determining optimal robot task sequences using a branch and bound procedure, and a graphical simulation procedure to display the robots as they perform transport operation. The branch and bound algorithm, which is an implicit enumeration scheme, is used to derive several near optimal robot task sequences. For the branch and bound algorithm, each node is a transport operation. Lower bound on the makespan is machine based and is computed at each node for further branching. The graphical simulation is used to detect interference among robots which is hard to be incorporated in the branch and bound procedure. Infeasible robot sequences are discarded and other solutions from the branch and bound procedure are displayed using the graphical simulation procedure to determine a near optimal and feasible sequence.

The integrated approach is implemented on a prototype system. A command driven general purpose graphics system MOVIE.BYU is used for the graphical simulation of the robotic workcell and robot motion. The entire system is available in an integrated environment. A powerful programming language Rexx is used to manage various programs and data. Also, intermediate Rexx programs are generated during execution to allow MOVIE.BYU to edit and display animation data on the Tektronix 41XX series of terminals. The entire system is flexible and modular to be able to be used for various different applications.

Acknowledgements

I am deeply indebted to my advisor Dr. Subhash C. Sarin for suggesting the topic area, for his guidance, support and encouragement during the course of this work. My thanks to Dr. A. Myklebust and Dr. C. C. Wang for their support and agreeing to serve on my committee.

My gratitude to Dr. Vernon O. Shanholtz and Dr. Saied Mostaghimi for supporting part of my graduate studies through a research assistantship. I am thankful to Agustin ReynaMcGlone, working with whom, I learned some of the computing techniques that were invaluable in the implementation of this research.

My stay in Blacksburg would not have been better, thanks to my friends and faculty in the IEOR department. Without the love and support of my family members, my Masters degree would not have seen reality.

Table of Contents

List of Illustrations

1.0 Introduction

An Industrial Robot is defined by RIA [36] as: "a programmable multifunctional manipulator designed to move materials, parts, tools, or special devices through variable programmed motions for the performance of a variety of tasks". Typically, industrial robots are used in hazardous environments for humans, and for repetitive and tedious tasks requiring accuracy. Specifically, robots are used in pick-n-place type of applications, welding, spray painting, assembly operations, etc.

Majority of the industrial robots are being programmed by being led through the desired sequence of motions [58]. This method of robot programming results in substantial amounts of time being spent on programming and testing applications in the workcell. As a result, alternative forms of programming techniques are being sought. Graphical simulation of robot(s) and their environment as a means of off-line programming and testing of robotic applications is one such technique that has been widely researched [7,33,59].

One of the principal reasons for graphical simulation of production cells as a means of off-line programming is to allow robots to continue production during the time of application planning and testing. This results in more time being available to the robots and the production cells for production. However, graphical simulation is not an effective optimization tool. Mathematical analysis of robotized production cells has therefore been suggested by researchers [50,51,66] to achieve efficient use of

available production time thereby to attain better production rate. Mathematical analysis techniques are also limited in application because of the complexities of the situations involved.

In this research we develop an integrated optimization and graphical simulation based procedure to design robotic workcells. Specifically, an optimal robot task sequencing procedure is integrated with the graphical simulation procedure which checks for potential interference among the robots. Such interferences are hard to detect mathematically. The proposed system is available in an integrated environment so that the infeasible solutions can be discarded and new, near-optimal solutions can be determined and displayed using simulation. The procedure is repeated until the attainment of the desired solution. In this chapter we first briefly give an overview of the various concepts used in the development of this system. The problem that is addressed is then stated, and the research objectives are defined. In the end, an outline of the thesis is given.

1.1 Classification of Robot Programming Methods

1.1.1 Lead through programming

This form of programming involves leading the robot through the desired path using the teach pendant. The path is recorded and available for future "play back". The robot motions recorded may be point-to-point or continuous path depending on the application and capability of the robot. It has been reported that

90% of robots used in industries are programmed using lead through programming [22]. In the absence of more sophisticated programming methods, lead through programming can be used for applications such as spray painting, arc welding and pick-n-place application. It is not desirable to use lead through programming for applications such as palletizing, etc. because of the amount of redundant work needed. Also, this method of programming may not be suitable for applications involving logical execution of operations.

1.1.2 Programming using textual languages

These languages have structure similar to high level computer programming languages. The programs developed would control the logic of the operations. These programs are normally developed off-line and the program development does not interfere with robot operation. Generally the logic is separated from the location data which may be recorded on-line. This method of programming takes part of the robot programming (teaching) time away from the robot workcell. One of the limitations of using this mode of programming is that the programming languages available to the user may not have all the features needed. Testing of this program would normally have to be done on the workcell.

1.1.3 Off-line programming

Off-line programming of robots would involve computing the coordinate data, function data, and cycle logic off-line. This method of programming is desirable

for sophisticated applications. But, technological and computational limitations restrict the use of off-line programming in its true sense. Off-line programming may also involve use of high level language for programming of robots. Even though off-line programming may not necessarily need graphical simulation as verification/viewing tool, many of the systems being developed for off-line programming incorporate graphical simulation.

1.2 Robot Kinematics

Robot kinematics is one of the fundamental aspects of robot motion. Hence, robot kinematics is of vital importance in the development of robotic systems. For industrial robots, our interest in kinematics is in relating the position of the robot Tool Center Point (henceforth TCP) of the robot hand with joint variables without considering the forces/moments that cause the motion. Vector and matrix algebra are normally used to describe the location of robot links with respect to fixed reference frame. A coordinate frame is associated with each robot joint. A homogeneous transformation matrix (which is a 4×4 matrix) describes the translational and rotational relationships between adjacent coordinate frames. Such a transformation was first proposed by Denavit and Harterberg [28]. Each of these transformation matrices is a trigonometric function of robot geometry, and contains complete information about the kinematic state space of the link to which it is attached.

These homogenous transformation matrices can be used to represent the position and orientation of any coordinate frame with respect to another coordinate frame

or the fixed reference frame. In general a transformation matrix $[C]$ is represented as :

$$[C] = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & o & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.1)$$

where n, o, a, and p are the normal, orientation, approach and position vectors.

This representation would be used to determine the coordinate transformation matrix of the TCP of the robot with respect to the fixed reference frame. For a 'n' joint robot, the coordinate frame of the n^{th} joint (assumed to be the TCP) is denoted by $[C_{TCP}]_0$ with '0' being the fixed reference frame.

1.2.1 Coordinate Transformation Matrices

The homogenous transformation matrices described in the previous section will be used to describe the coordinate frame of any robot joint with respect to another. Coordinate transformation $[C_m]_n$ refers to the position and orientation of joint 'm' relative to joint 'n'. In general form, $[C_m]_n$ can be expressed as :

$$[C_m]_n = \begin{bmatrix} [R_m]_n & [p_m]_n \\ f & s \end{bmatrix} \qquad (1.2)$$

where $[R_m]_n$ is a 3×3 rotation matrix, $[p_m]_n$ is a position vector, 'f' is a 1×3 perspective transformation and is normally equal to $[0, 0, 0]$, and 's' is the scaling factor [28]. The 3×3 rotation matrix does not provide any information about translation.

This necessitates the use of a fourth coordinate called the position vector. Hence the position vector is expressed in homogenous coordinates. Homogenous coordinate representation of points in a 3-D space is used in robot kinematics to include rotation as well as translations.

These transformation matrices contain all the information needed to transform a coordinate frame from location 'n' to location 'm'. Most industrial robots would contain any combination of revolute and prismatic joints. For revolute joints, the joint angle 'θ' would be the variable and for prismatic joints the translation across the joint 'd' would be the variable. For a 'n' joint robot the coordinate transformation matrix of the TCP with reference to the fixed reference frame $[C_{TCP}]_0$ can be achieved by :

$$[C_{TCP}]_0 = [C_1]_0[C_2]_1[C_3]_2....[C_{TCP}]_{n-1} \qquad (1.3)$$

There are two sub-problems associated with robot kinematics. They are forward kinematics and inverse kinematics. Both these problems are discussed briefly below.

1.2.2 Forward Kinematics

Forward kinematics involves moving the links through known angular and linear displacements. The location and orientation of the TCP is returned as output. Initially, the coordinate frames are attached to each link of the manipulator and homogenous transformation matrices $[C_1]_0, [C_2]_1, ...[C_{TCP}]_{n-1}$ are established. Conventions from Lee [28] and Zugel [69] are used to arrive at these transformations. Then the position and orientation of the TCP relative to the fixed reference frame can be determined by substituting values of all the variables and link parameters in each of the matrices

on the Right Hand Side (RHS) of equation (1.3). Since, all the link displacements are known, and can be substituted in all the adjacent transformation matrices, the position and orientation of the TCP with respect to the fixed reference frame can be determined.

1.2.3 Inverse Kinematics

Inverse kinematics requires the knowledge of position and orientation of the TCP with respect to the fixed reference frame, and the various link parameters. The joint variables are returned as output. All the elements in the Left Hand Side (LHS) matrix of equation 1.3 are input. Hence, equating the LHS of equation 1.3 with the RHS (product of 'n' matrices) of equation 1.3 would give a series of non-linear equations with 'n' variables, where 'n' is the number of joints. These equations can be used to find the inverse kinematic solutions. The inverse kinematic problem can be solved by algebraic, iterative, or geometric techniques. Closed-form inverse kinematic solutions are very desirable. If closed-form inverse kinematic solutions cannot be derived, use can be made of some iterative methods.

1.3 Graphical Simulation (of Robot Motion)

Use of interactive computer (graphical) simulation of the robots and their working environment has been suggested as a logical methodology for cost effective implementation of robotic systems and for off-line programming of robots. Various

graphical simulation systems have been developed to resolve some of the aspects of robotic systems implementation. Graphical simulation systems have been developed to solve problems such as workcell layout design, robot selection, robot design, operator training, robotics instruction, interference detection, trajectory planning, robot cycle time estimation, etc. [31,58].

Graphical simulation of robotic workcells can be done at two levels:

1. Kinematic Simulation : Simulation of robot kinematics involves simulating the motion of robot link positions without considering the forces/moments that cause the motion. Consequently, this involves constructing the robot in successive positions as it moves from one point to another. Given the initial and the target positions of the TCP of the robot, the simulation would need to compute the orientations and positions of individual links at various intermediate positions giving the robot motion simulation.
2. Dynamic Simulation : Dynamic simulation of the robot would include the effect of forces/torques generated due to the motion of masses in the simulation. Hence, the dynamic model of the robot would include the mass properties, and thereby the manipulator response to the forces/torques exerts on the joints.

1.4 Some Graphical Simulation Applications

1. Off-line programming : A set of tasks can be tested on the simulation. The off-line testing can indicate whether the objective (for example, desired production rate) can be met. A fully tested work-cycle can then be implemented on the workcell.

Off-line programming of workcells would allow production to continue, while new applications are being programmed. Also, some emergency situations can be tested on the graphical simulations. It may not be possible to test such situations on real production cell.

2. Instruction and Teaching : Graphical simulation systems can be used for operator instruction and training. There are two aspects to this application. Firstly, the robot's production time need not be used for training personnel. Secondly, an untrained operator need not put the robot and its environment in danger. The graphical simulation may be a good tool for teaching the fundamentals of robot motion and robot programming.
3. Collision Detection : Graphical simulation can construct the robot and its motion relative to the surrounding environment say, moving conveyors, stationary machines, other robots working in the same workspace, etc. The trajectory of the TCP with its payload and that of the arm can be simulated to detect possible collision. In situations where multiple robots may be moving at the same time, some means of planning a safe workcell is essential. In workcells with multiple robots with overlapping work envelopes, the risk of collision may result in trivial robot motions being used. This may result in reduced production rates (increased costs).

1.5 Efficient Use of Robotic Applications

One of the important reasons for the use of industrial robots is to reduce operating costs. Improvement in production rates can result in cost reduction. Also, there

may be time constraints on completion of jobs. These needs require efficient use of robot's production time. This would involve analyzing the application workcell to obtain optimal performance (production). There can be two levels (stages) of optimization used in designing workcells. They are :

1. Planning Stage - e.g. robot selection, layout design, etc.
2. Operational Stage - e.g. optimal sequencing of jobs and robot motions, collision free path planning, etc.

1.6 Problem Statement

Efficient use of robots in workcells is essential. For robot tending applications, the optimal sequencing of jobs can give significantly better performance compared to arbitrary sequencing or sequencing using primitive rules. Mathematical analysis of the production cell can give optimal sequencing of jobs. But, mathematical programming models may not be able to model (or overlook) all aspects related to the sequencing of operations performed by the robot in the workcell. This could result in infeasible sequences which may be identified during their implementation in the actual workcells. Therefore, the problem that is addressed here is to develop an integrated optimization procedure to determine the optimal and feasible sequences of operations performed by a robot in a workcell.

1.7 Research Objectives

Various graphical simulation systems, as a means of off-line programming or planning robotic applications; have been developed in the literature. Also, mathematical analysis of robot tasks and motions have been done to optimize production rates. The main goal of this research is to demonstrate the integration of a graphical simulation of a robotic workcell with an optimization procedure to minimize the makespan of the application workcell. Towards that goal, the objectives are to do the following:

1. Develop an optimization procedure to sequence operations performed by a robot in a workcell.

2. Develop a graphical simulation procedure to emulate the movement of robots in the workcell.

3. Develop a computer implementation of the integrated methodology on the procedures developed in 1 and 2.

4. Demonstrate the implementation of the methodology on a prototype system.

The workcell that is considered is a flowshop involving multiple robots that transport the jobs from one machine to another. The movement of the robots and their possible interaction (collision) is difficult to capture mathematically. It is this interaction that is determined through graphical simulation.

1.8 Outline of the Thesis

Chapter 2 reviews some of the approaches based on graphical simulation and mathematical programming for off-line design of robot workcells. Also, some of the problems encountered in accurate graphical simulation of robot workcells are outlined. Some relevant graphical simulation systems are reviewed. Chapter 3 specifically describes the elements of the proposed integrated system.

Chapter 4 presents the branch and bound algorithm used to determine the optimal job and robot task sequences. Specifically, the lower bound formulation of this branch and bound algorithm is presented, and the branching strategy is discussed. In the end, the implementation of the branch and bound algorithm in the integrated system is described. Chapter 5 describes the proposed integrated system and the graphical simulation of the robotic workcell. Chapter 6 presents a demonstration of the integrated system on a sample problem. All the input data needed for the system is illustrated and the system-user interactions are shown. The Chapter 7 summarizes the research conducted and identifies some of the areas for further research. The appendix at the end of this thesis provides the listing of the main control program used in the prototype system. This program is written in Rexx. The other programs have not been included in this thesis because of their large sizes.

2.0 Literature Review

In Chapter One we discussed some of the fundamental concepts involved in the development of graphical simulation of robots. In this chapter we briefly review some graphical simulation package and other aspects of the robot sequencing problem considered. This literature review is divided into following topics:

1. Graphical Simulation Systems
2. Problems in Graphical Simulation of Robot Application
3. Various Approaches to Inverse Kinematic Solutions
4. Analysis of Robotic Workcells

2.1 Graphical Simulation Systems

On-line programming of robots limits the use of robots to simpler applications [38]. With this mode of programming, the robots and the workcell will not be engaged in production when the robots are being programmed and the application is being tested. For sophisticated applications involving logical and conditional execution of tasks, it is important to be able to test the programs outside the robotic workcells [59]. A variety of robot programming languages have been developed for off-line programming of robots. Most robot programming languages do not allow program-

ming of positions and orientations of the robot [49] and thus have to be entered on-line by teaching the positions and orientations.

Graphical simulation of robots and their application workcells has been suggested by various researchers as necessary for effective robotic workcell implementations [31,33,58]. Various graphical simulation systems are being developed in industries and research institutions. The approaches taken for the development of such systems are varied. Some of the systems make use of high level robot programming languages while others have the capability of down-loading the program from the off-line computer, used to test the application, to the robot controller in appropriate format. Some of these graphical simulation systems are discussed below.

2.2 Related Graphical Simulation Systems

RoboTeach is an interactive graphics system developed at General Motors [59]. The system is mainly used for workcell layout design. The whole system is developed around a database containing the geometric models of robots, tools, parts, etc available in the GM's Corporate Graphics System (CGS). The workcell layout can be designed by specifying the position of the robot and other objects in the workcell. The most suitable layout is selected after visually seeing the implications of each layout. Robots for which closed form kinematic solutions are available are used.

The authors assert that closed form solutions have not been able to identify all the solutions for all cases [59]. The project was abandoned in 1985, with only the workcell layout function of the system being operational. The graphical simulation of

robots is based on robot kinematics and can simulate the motion of six or less degree of freedom robots. The path taken by the robot may be different from the one displayed by the simulation, as the system does not have the robot controller logic built into it. Because of this limitation, the time taken for a particular operation cannot be estimated accurately.

Leu and Mahajan have developed a graphical simulation system whose main function is cycle time estimation [31]. The system is recently being sold as I-GRIP (Interactive Robot Instruction Program). For kinematic simulation, robots with closed form inverse solutions are used. For dynamic simulation, the system uses prespecified dynamic model. They have neglected some of the forces which are expected to have little effect on the dynamic simulation. The current system is being marketed with only the kinematic simulation capability by a firm called Denab.

At McDonnel Douglas Automation Co. (MCAUTO), a graphical simulation system for evaluation of workcell layouts in terms of reach limits, cycle times, motion sequences, etc. is being developed [55]. The five main modules of the system, Place, Animate, Build, Command, and Adjust are at various stages of development. It is considered to be one of the most comprehensive robotic workcell design and off-line programming system [54] under development. The whole system is being based around the robot programming language MCL also developed by MCAUTO.

The module Place, is used to graphically create, analyze, and modify workcells using the objects from the CAD databases while the module Build is used to create and enhance the CAD databases. Any robot to be included in the database is also created using Build. Robot kinematics can be included as a routine to the Place [26].

The kinematic model would be used for the animation of robot(s) performing the tasks as directed by Command module. Animation speeds can be varied to detect possible collisions. Adjust module is being developed to do real-world cell calibration [18].

Computer Aided Manufacturing - International, Inc. (CAM-I) is involved in a project for developing a comprehensive off-line programming system for robotic workcells [58]. The system also involves developing a high level language (IRM), a CAD database, controller model, and sensor model for various robots [58]. The author forecasts the materialization of an ideal off-line graphical simulation system for robotic workcells to be at least a few years away.

GRASP is a graphical simulation package for robotic workcells developed at Renneselar Polytechnic Institute [7,8]. The system allows for robot design, robot selection and evaluation, and simulation. Besides, the system has a CAD database which can have various robots, end-effectors, workparts, etc. as entities to be used in a workcell. For this system, instead of basing the simulation purely on robot kinematics and dynamics, the author uses a cam motion profile concepts in planning robot motion [7]. This approach makes minimum demands on robot actuators, as it has the capability to scale down accelerations whenever possible. In the system the user would guide the robot around the obstacles by entering appropriate locations. The system can be used to evaluate the performance of robot arms in potential working environment. Robot-SIM is an enhanced version of GRASP being marketed by Calma Corporation [55].

IRPASS is a graphical simulation system developed at NOKIA Corp., Finland. [21,22]. The system uses a CAD database to store graphical and non-graphical infor-

mation. The system has robot dependent robot processors communicating with the CAD system. Execution of the robot controller program is directed from the CAD system. The joint parameters, I/O signals, etc. are transmitted to the CAD system in acceptable format. The simulation is guided in the teach mode through the robot controller. During the cell design phase, the signals from the robot controller are routed to the graphical simulation instead of the robot. The system can be thought of as a much lower level graphical simulation system.

Another system which is designed with a similar philosophy as IRPASS is HOPS-I [56]. The HOPS-I graphical simulation system communicates with MODEL-83KH Robot controller. This is a full fledged controller with motor drives, control electronics and ability to interface with sensors, and can control robots with maximum of 4 degree of freedom. In this case, the robot controller is used to teach the robot on the CRT screen and simulate robot operation. The same controller can be used to control the actual robot, once the robot program has been tested on HOPS-I. The authors anticipate the use of MODEL-83KH controller in future robot systems. The off-line graphical simulation is done on microcomputer.

ASEA off-line programming system uses yet another approach in their graphical simulation system [4]. Here, a communication link is established between the robot controller and the IBM PC which runs the simulation. The communication link is used to transfer the actual recorded points by the robot to the IBM PC. A high level language ALRA is used to construct the graphical simulation using the recorded points. The same language processor is available on the robot controller. A fully tested simulation program can be transferred to the robot controller for final debugging on the shop floor. Also, recorded data can be edited off-line and transferred to the robot

controller. The authors assert that true off-line programming is difficult and/or impractical given the current technology.

The graphical simulation systems such as IRPASS, HOPS-I, and ASEA cannot be used for applications such as robot selection, robot design, etc. because these systems use the already existing robot controller or controller model for their simulation system.

Li [30] has developed graphical simulation system for verifying collision free robot paths off-line. The system is based on the robot kinematics. The system has been designed for single robot workcells with no moving obstacles. The system uses motion time profiles for generating the trajectory. Point to point and continuous path motions are considered.

Geometric interference between stationary objects is checked at each intermediate step in the graphical simulation. In essence, checks are made to make sure that the convex hulls of individual links do not intersect with any of the planes of the obstacles.

Robographix is a system developed by Computervision for robotic workcell construction, creation of robot paths, and simulation of robot motion [57]. AutoSimulations, Inc. have developed a simulation system based on GPSS simulation language. It is capable of graphically simulating the functioning of manufacturing system, with or without robots [57]. RCODE is another graphical simulation system developed by SRI International [57]. The primary function of this system is collision

detection. SRI International is also working on the collision avoidance and off-line programming of robotic workcells.

As described in this section, graphical simulation of robotic systems has been done for various applications. The approaches taken in developing these graphical simulation systems are varied. These differences arise from the fact that there are lot of intricacies involved in developing these systems. In the next section, we will discuss some of the difficulties involved in developing these systems.

2.3 Problems in Graphical Simulation of Robotized Workcells

Technological limitations in various aspects such as computing inverse kinematics, inverse dynamics, collision avoidance or detection, etc in real-time or otherwise, prevent the off-line programming and testing of robot programs. Graphical simulation of robots and their working environments presents a valuable tool for verification of proposed design parameters [51]. Graphical simulation systems would allow users flexibility to test a variety of situations and configurations of the workcell before the start of production, e.g. simulation speeds can be slowed down or interrupted to detect possible collision [54].

Graphical simulation systems fall into two categories:

1. Kinematic simulation
2. Kinematic and Dynamic simulation

The kinematic simulation of robot considers the geometry of the robot arm motion with reference to the fixed reference frame. For the kinematic and dynamic simulation, besides the geometry of robot arm motion, the forces and torques exerted on the arms/joints are taken into consideration. Both simulations have been applied to problems such as layout design, cycle time estimation, instruction, collision detection, etc.

It is generally agreed that accurate planning of robotic applications is complex [5,21]. The complexity derives from the variety of sub-problems and the interaction and interdependence of various factors involved. For a fundamental problem, that of finding inverse kinematic solutions, difficulties arise because of the complexity and non-linearity of the equations [59]. It would be preferable to be able to derive a closed form solution to the inverse kinematics problem which would be able to locate all solutions. But such closed form solutions are available only for a limited number of robots with simpler geometries. Iterative methods for finding inverse solutions are not efficient and may not be suitable for real-time interactive systems.

For dynamic simulations too, the solution involves iterative numerical methods. And the solution methodology can be further complicated due to incremental errors. To make matters worse, the dynamic model would change over time, due to the nature of the application environment, and due to the type of maintenance procedures used [26]. Mahajan and Leu have presented a dynamic simulation of a six degree of freedom robot. They have used a precalculated dynamic model for their simulation [31].

Another problem, that of detecting collisions in a cluttered workspace, involves finding the path of moving parts in 3-D [5]. The implementation complexity of the problem of finding path for manipulators makes them virtually unsolvable [5], and limits their application for real-time control.

The problems discussed above and many more are interdependent, thus restricting the use of off-line programming in its true sense. Ideally, an off-line programming system should allow for planning of robotic application, testing and verification, and down loading the verified application to the controller of the robot in acceptable format. Various graphical simulation systems have been developed to solve some of the various problems and to allow at least off-line planning or testing of robotic applications.

2.4 Various Approaches to Inverse Kinematic Solutions

The problem of determining inverse kinematics is robot dependent. No unique solution methodology is available, because of inherently difficult nature of the inverse kinematics problem. Various researchers have proposed different methods to solve the inverse kinematic problem as related to a particular robot or group of robots. Closed form solutions are obviously preferable for real-time graphical simulation systems. But, robots for which closed form solutions are not available, use of iterative methods to solve inverse kinematic problem is made. In this section, a few of the various methods used by researchers to solve the inverse kinematic problem are reviewed.

Lee and Ziegler [29] have used a geometric approach to derive closed form solutions to the inverse kinematic problem. The authors assert that this approach can be used for any six or less degree of freedom robot which have all the wrist joints as revolute joints. They have shown the closed form inverse kinematic solution for the PUMA robot. The various robot arm configurations are divided into groups according to the human arm geometries. Then the first three joints of the PUMA robot are solved using the position vector from the fixed reference frame to the coordinate frame associated with the beginning or the wrist joint. The values of these three variables are then used to determine the three wrist variables.

Angeles [1] has presented an iterative method for determining inverse kinematic solutions for general, five degree of freedom robot manipulators. This method is based on derivation of kinematic model which gives six non-linear algebraic equations for the five variables. For solving the above system of equations Newton-Gauss method for finding least squares solution is used. For special geometries, the kinematic model can be used to derive closed-form solutions. But, in the absence of such solutions, this iterative method may be used for five degree of freedom robots.

Furusho and Onishi [12] present an approach that uses iterative method for finding kinematic solution for generalized open chain kinematic structure. But they have not justified it's applicability to real time graphical simulation or off-line programming of robots.

Hollerbach and Sahar [17] describe a wrist partitioned method to determine inverse kinematic positions, velocities, and acceleration for six degree of freedom robots with the robot wrist being spherical. Using this special structure, the authors show the possibility of decomposing the six variable inverse kinematic problem into two, three variable problems. The problem is decomposed into separate computations for position and orientation specifications. If the criteria that the wrist is spherical is not satisfied for a particular manipulator, then time consuming numerical approximation and convergence methods have to be used to solve the system of non-linear equations.

Tilove et al [59] make use of robots for which closed form solutions are available, in their graphical simulation system (RoboTeach). But they have done a considerable amount of work in the area of using iterative methods for finding inverse kinematic solutions. These inverse solutions take upto a few minutes of CPU time on IBM 3081 mainframes. Also, seeking such solutions requires considerable amount of pre-processing of the kinematic equations for individual robots, before implementation. They report that before such techniques can be generalized, considerable amount of work needs to be done in the area of automatic generation of kinematic equations and efficient numerical techniques.

Nakamura et al [38] have presented an algorithm to combine the computations of kinematics and dynamics of robots to eliminate duplication of computations for graphical simulation of robots. The authors have combined the best available algorithms for computing inverse kinematics and dynamics to present a single algorithm, which while using the features of individual algorithms, also eliminate duplicate computations needed at various stages.

2.5 Mathematical Analysis of Robotic Workcells

Robots may be used in applications which have highly repetitive cycles. A marginal decrease in cycle times can increase production rates significantly. There are many aspects of robotic application design where trial and error may be used in designing better application workcells. However, there could be some aspects of workcell design, where trial and error may not be possible (practical).

Mathematical modelling of application workcells can yield optimal and/or better workcell designs than those derived by trial and error and/or intuitive decisions. Mathematical programming models have been developed for optimizing such parameters as sequencing of tasks, minimizing makespan, layout design, etc. It seems logical to mathematically model the robotic applications for determining efficient ways of using the robots in the application. Few special considerations are needed for modelling robotic applications as compared to conventional manufacturing applications. Some of the mathematical programming approaches to modelling robotic applications are reviewed here.

Sarin and Wilhelm [51] have presented analysis of robotic system for designing workcell layouts. The prototype models developed suggest a structure for analyzing layout design problems which can be adapted for specific applications. Another paper by Wilhelm and Sarin [66] present structure for mathematically analyzing robotic application for sequencing robot tasks in a workcell.

Determination of sequence of robot task handling can give an indication of expected production rates.

Hitomi et al [15] present a branch-and-bound algorithm to determine the optimum job and robot operation sequence to minimize makespan. The manufacturing system considered is a flowshop with intermediate buffers with infinite capacity, and an incoming and outgoing conveyors. Multiple robots may be employed to transport jobs. The lower bound formulation is similar to that of a travelling salesman problem. Branching is conducted on robot transport operations.

A paper by Sarin [50] presents integer programs for sequencing robot activities and determining loading of orders at processing stations. Mathematical models have been developed for an application consisting of three processing stations which can be loaded with work orders. The robot transports jobs from inventory stacks to processing stations. Bauman et al [50] have developed mathematical models to determine the robot and machine utilization in the workcells where robot is involved in transporting workparts.

2.6 Summary

In this chapter, we have briefly reviewed approaches based on graphical simulation and mathematical programming for offline design of robot workcells. The approaches based on graphical simulation are not that effective for optimal design of workcells while the approaches based on mathematical programming

cannot typically capture all complexities of the robotic workcell. An integration of mathematical programming based methodologies and graphical simulation of production cells would be the next logical step for efficient workcell design. For example, the graphical simulation of workcell could be used to estimate the travel times by the robot. These travel times can be used, for example, to determine optimal sequencing of robot operations using the mathematical model of the workcell. These sequence can be fed back to the graphical simulation system for any possible infeasibilities, which may not be modelled by the mathemetical program (e.g. physical limitations of the robot, etc.). This procedure can be used iteratively to determine an optimal design. This integrated approach would significantly reduce the time expended on the actual production cell, while giving best solutions.

3.0 Elements of the Integrated System

This chapter describes the elements of the proposed integrated system. This chapter is divided into the following sections.

1. Manufacturing facility description
2. Optimal robot task sequencing
3. Kinematics of the robot used
4. General purpose graphics system MOVIE.BYU
5. Software/Languages/Hardware used

3.1 Manufacturing Facility

The system is developed for a flowshop type manufacturing facility. The various machine tools are arranged as a workcell. There is an incoming buffer station for the incoming jobs, and an outgoing buffer station for finished jobs. Multiple robots with overlapping work envelopes, transport the jobs to the machines in the workcell. Figure 1 shows the schematic sketch of the manufacturing facility being considered. The following are the assumptions for the operation of the manufacturing system:

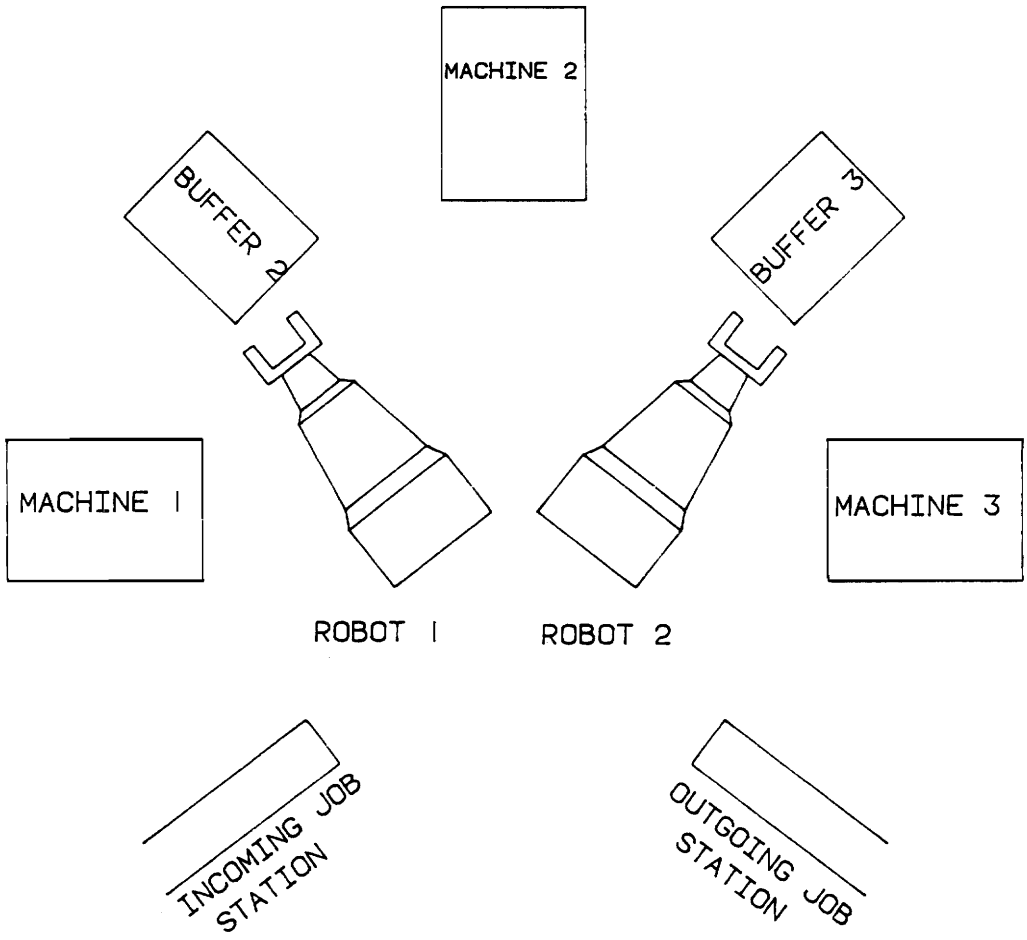


FIGURE 1. SKEMATIC SKETCH OF THE MANUFACTURING FACILITY

1. There is a buffer with infinite capacity between each adjacent pair of machines.
2. All the jobs coming into the system are required to be processed on every machine.
3. All the robots are identical in function and capabilities.
4. Each job, after being processed on a machine, goes to the buffer station, before being loaded onto the next machine.
5. All the job processing and job transport operation times are known.
6. The location of all the machines, buffers and robots are determined/known a priori.

The robot task sequencing is done based on the above assumptions. A branch and bound algorithm is used to determine the optimal sequencing of robot transport operations.

3.2 Optimal job and robot task sequencing

Optimal sequencing of robot transport operations is done using a branch and bound algorithm. The objective is to sequence the jobs and robot transport operations to minimize makespan. The bounding procedure suggested by Hitomi et al [15] has been used. Appropriate enhancements have been made to the algorithm for practical implementation of the problem being considered. The algorithm uses a pre-specified number of robots, machines, buffers, and jobs. All

the robots in the system are identical. The branch and bound algorithm will be described in chapter 4.

3.3 Robot Kinematics

The robot(s) used in the system are similar to ASEA's MHU Minor [11]. The robot(s) have five degrees of freedom and are cylindrical coordinate robots. The five joints of the robot are R-P-P-R-R (R = Revolute, P = Prismatic). The robot has a horizontal arm which is attached to the vertical axis which is firmly mounted on a fixed base. This arm can advance and retract, and also move up and down the vertical axis. In addition, the whole assembly can rotate around the base, thus describing the cylindrical workspace. Figure 2 shows the wire frame model of the robot used.

The schematic diagram of the robot used is shown in Figure 3. For obtaining the transformation matrices, the convention described in Lee [28] and Zugel [69] are used. Based on the robot arm parameters shown in Figure 3, the following transformation matrices were obtained.

$$[C_1]_0 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.1)$$

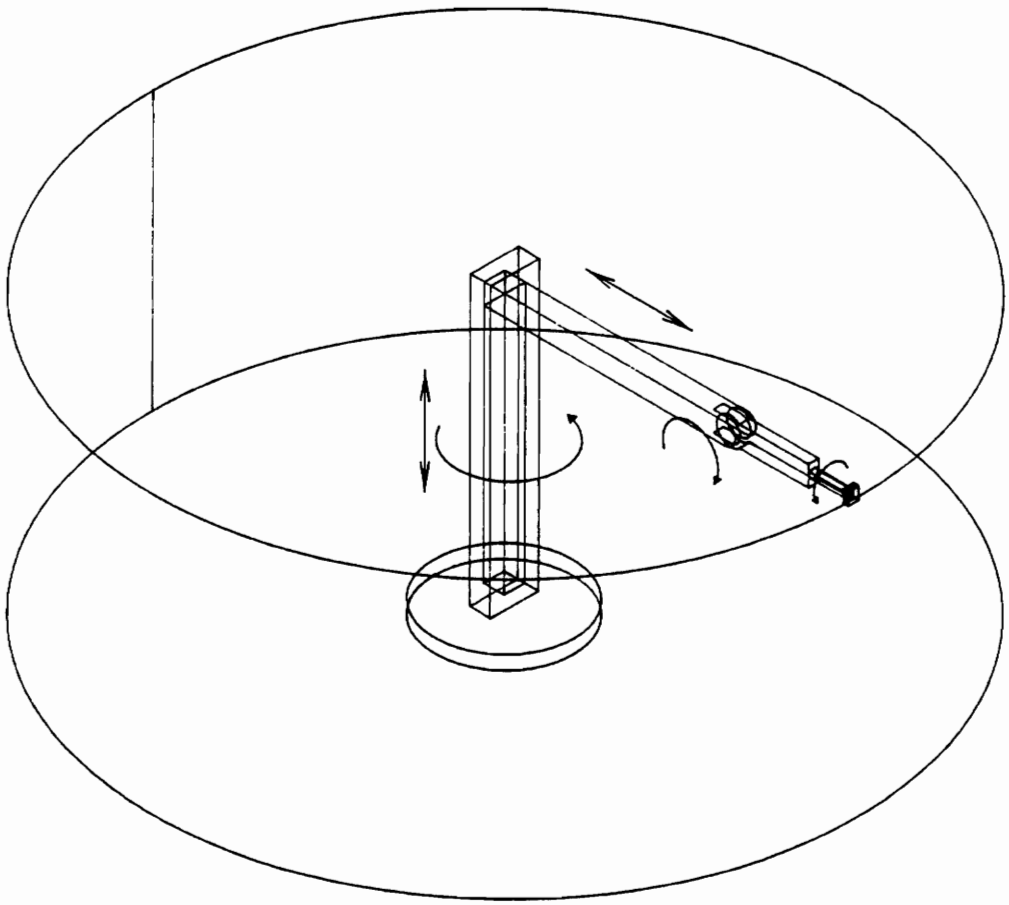
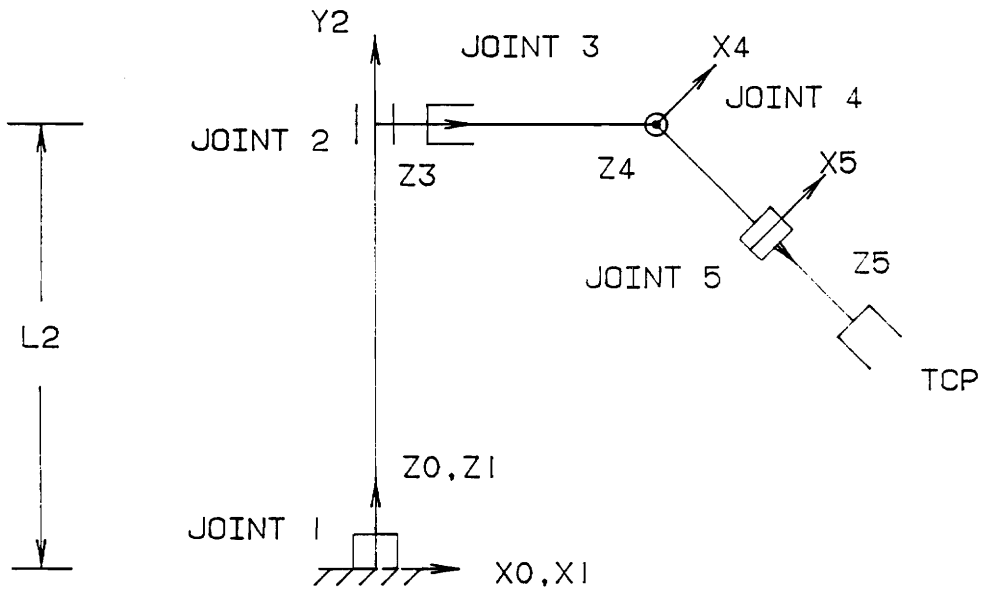


FIGURE 2. WIRE FRAME MODEL OF THE CYLINDRICAL COORDINATE ROBOT



JOINT	VARIABLE	θ	A	D	α	COS	SIN
1	θ1	θ1	0	0	0	1	0
2	D2 OR L2	90	0	L2	90	0	1
3	D3 OR L3	90	0	L3	-90	0	-1
4	θ4	θ4	0	L4	90	0	1
TCP	θ5	θ5	0	L5	0	1	0

FIGURE 3. SKEMATIC DIAGRAM OF THE CYLINDRICAL COORDINATE ROBOT

$$[C_2]_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.2)$$

$$[C_3]_2 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.3)$$

$$[C_4]_3 = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & l_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.4)$$

$$[C_{TCP}]_4 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.5)$$

The position and orientation of the TCP with respect to the base coordinate frame can be given as:

$$[C_{TCP}]_0 = [C_1]_0[C_2]_1[C_3]_2[C_4]_3[C_{TCP}]_4 \qquad (3.6)$$

Hence,

$$[C_{TCP}]_0 = \begin{bmatrix} s_5s_1 - c_1s_4c_5 & c_5s_1 + c_1s_4s_5 & c_1c_4 & (l_4 + l_5)c_1c_4 + l_3c_1 \\ -s_5c_1 - s_1s_4c_5 & -c_5c_1 + s_1s_4s_5 & s_1c_4 & (l_4 + l_5)s_1c_4 + l_3s_1 \\ c_5c_4 & -s_5c_4 & s_4 & (l_4 + l_5)s_4 + l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.7)$$

3.3.1 Forward Kinematics

For forward kinematics, pre-calculated joint variables needed to reach each buffer and machine by each robot are stored. These joint variables can be substituted in the right hand side of equation (3.7) to give position and orientation of TCP with respect to the base coordinate system.

3.3.2 Inverse Kinematics

In standard notation [28] the coordinate transformation matrix $[C_{TCP}]_0$ can be expressed as:

$$[C_{TCP}]_0 = \begin{bmatrix} n_x & o_x & a_x & R_x \\ n_y & o_y & a_y & R_y \\ n_z & o_z & a_z & R_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.8)$$

For deriving inverse kinematic solution all the elements of the matrix on the right hand side of equation (3.8) are known (as input). Closed form inverse kinematic solutions are derived and used in this system.

The closed form solutions are derived using the wrist partitioned method [17]. The geometric configuration of the robot allows breaking the inverse kinematic problem into 2 subproblems. The joints 1-3 and 4-5 are solved separately. The position of end of the horizontal arm is dependent only on the approach vector of the TCP. That gives us the following equations:

$$[R_3]_x = R_x - a_x(l_4 + l_5) \qquad (3.9)$$

$$[R_3]_y = R_y - a_y(l_4 + l_5) \qquad (3.10)$$

$$[R_3]_z = R_z - a_z(l_4 + l_5) \qquad (3.11)$$

where 'a' is the approach vector at the center of the TCP and p is the position vector of the center of TCP. The values of the R_3 vector can be used to find the values of variables of joints 1, 2, and 3. The values of joint 4 and joint 5 can be found by using the equality of the matrices in equations (3.7) and (3.8)

3.4 General Purpose Graphics System : MOVIE.BYU

MOVIE.BYU is a general purpose graphics software system developed at Brigham Young University [35]. The system consists of FORTRAN programs for display and manipulation of geometrical data represented as polygons and contour lines. The six programs are DISPLAY, UTILITY, SECTION, TITLE, MOSAIC, AND COMPOSE. The programs DISPLAY and UTILITY have been used in this research.

DISPLAY is an interactive program for the display and animation of any model composed of polygons [35]. DISPLAY can receive input in two ways. It can receive all or part of the input from a command file and it can also receive input from console. UTILITY can create and edit data. UTILITY also receives input in the same way as DISPLAY. The command structure of UTILITY is organ-

ized into three levels. Each command at command level 1 would allow only a subset of valid commands to be entered at command level 2 and so on. All data generated or edited using UTILITY can be used by DISPLAY for display and animation.

The DISPLAY program is used for display of robotic workcell and animation of robots' transport operations. The UTILITY program is a data generation and editing program. UTILITY is mainly used for editing and merging data in this system, during an execution run. A brief explanation of some of the commands and their function will be described in Chapter 6.

3.5 Hardware/Software/Languages

3.5.1 Hardware

The system is developed on a IBM 4341 system running VM/CMS. The graphical simulation is displayed on Tektronix 41XX series of graphics workstations. It can also be displayed on an IBM PC by using a Tektronix emulator for the PC.

3.5.2 Software

The software used is the UTILITY and DISPLAY programs of the MOVIE.BYU system for graphical display and editing graphical data. Tekterm, a Tektronix emulator for the IBM PC may be used for system demonstration on the IBM PC.

3.5.3 Languages

Two high level languages, viz., Rexx and FORTRAN are used in the development of the system. Rexx is an interpreted language and is mainly used for the overall control and integration of the system. FORTRAN is used in the branch and bound algorithm and for the kinematic analysis and creating graphical data in the format acceptable to MOVIE.BYU.

4.0 Branch and Bound Algorithm for Robot Task Sequencing

This chapter describes the branch and bound algorithm used to determine optimal job and robot task sequences. The bounding procedure is based on the one suggested by Hitomi et al [15]. Appropriate modifications have been made to Hitomi et al's bounding procedure. These modifications were necessary for realistic implementation of the branch and bound algorithm for the application under consideration. Here, we present the lower bound formulation for multiple robots.

The branch and bound algorithm is developed for a flowshop. The robots transport the jobs from machines to buffers and from buffers to machines. Multiple robots may be used to conduct these transport operations. Each job has to be processed on all the machines in the workcell. Below, we describe the variables used in the branch and bound algorithm.

4.1 Nomenclature

M = Set of all machines

N = Set of all jobs

L = Set of all the industrial robots in the system

M_j = Machine j (stage of flowshop)

J_i = Job i

B_1 = incoming job station

B_{M+1} = outgoing job station

B_k ($2 \leq k \leq M$) = buffer station between machines M_{k-1} and M_k

R_m = industrial robot m

R_m (τ , i, j) = Transport operation by robot m

If τ = 1, then the transport operation of job J_i is from buffer B_j to machine M_j

If τ = 2, then the transport operation of job J_i is from machine M_j to buffer B_{j+1}

OP(i,j) = operation of job J_i on machine M_j

$HT_1(i, j)$ = transport time of job J_i from B_j to machine M_j

$HT_2(i, j)$ = transport time of job J_i from M_j to buffer B_{j+1}

PT(i,j) = processing time of J_i on machine M_j

LB_j = Lower bound on machine M_j

$TS[R_m(\tau, i, j)]$ = starting time for transport operation $R_m(\tau, i, j)$ in schedule

$TE[R_m(\tau, i, j)]$ = finishing time for transport operation $R_m(\tau, i, j)$ in schedule

T_k = kth transport operation

where T_k corresponds to transport operation $R_m(\tau, i, j)$ and

TS [T_k] = TS [$R_m(\tau, i, j)$] and

TE [T_k] = TE [$R_m(\tau, i, j)$]

U_j = set of jobs J_i for which transport operation $R_m(1, i, j)$ from buffer B_j to machine M_j is not yet sequenced

U_0 = the job for which the transport operation from B_1 to M_1 was last sequenced, if $U_1 = 0$, and $U_0 = U_1$, if $U_1 \neq \phi$

p_j = job number of the job loaded on machine M_j. If no job is loaded on machine M_j, $p_j = 0$

b_j = job number of the job to be loaded on machine M_j. If there is no job on buffer B_j, $b_j = 0$

m_0 = earliest available robot

TF [R_m] = finish time of the transport operation in which robot R_m was last used

I_{tt} = Idle Travel Time required by the robot to reach a position to handle the next transport operation

$I_{tt}(R_m[\tau, i, j])$ = Idle Travel Time needed by robot R_m to reach a ready position to handle the next transport operation

$A_t(R_m[\tau, i, j])$ = earliest Available Time of robot R_m for a particular transport operation

4.2 Lower Bound Formulation for Multiple Robots

The branching is conducted on the robot transport operations. The total number of transport operations will be equal to $2 \times M \times N$. Lower bound is calculated at each node, and is the minimum time needed to complete all the jobs, regardless of what the yet undetermined transport operation sequence may be. The transport operation at the current node in the branching process is expressed as $R_m(\tau, i, j)$. For com-

puting the lower bound on the makespan value, the starting time $TS[R_m(\tau, i, j)]$ and finish time $TE[(R_m(\tau, i, j))]$ need to be calculated.

The transport operations to be performed by the robots are classified as being of two types:

1. transport operation from buffer to machine ($\tau = 1$)
2. transport operation from machine to buffer ($\tau = 2$)

If $\tau = 1$, (buffer to machine)
the current transport operation can begin after all of the following conditions are satisfied:

1. the transport operation at the current node must begin at or after the finish time of transport operation in which robot R_m was last used ($TF[R_m]$)
2. the transport operation at the current node must begin at or after the starting of previous transport operation $TS[T_{k-1}]$
3. if the transport operation at the current node involves any machine other than machine 1 (i. e. $j \geq 2$), then this transport operation can start at or after the finish time of the transport operation of job i from machine j-1 to buffer j ($TE[R_m(2, i, j-1)]$)

Thus the start and finish time of the current transport operation can be expressed as:

$$TS[T_k] = TS\ [R_m(1, i, j)] = \max \{TS[T_{k-1}], TF[R_m], TE[R_m(2, i, j-1)]\} \quad (4.1)$$

$$TE[T_k] = TS[T_k] + HT_1(i, j) \quad (4.2)$$

If $\tau = 2$, (machine to buffer)

the transport operation corresponding to current node can begin after all of the following conditions are satisfied:

1. the transport operation at the current node must begin at or after finish time of the transport operation in which robot 'm' was last used ($TF[R_m]$)
2. current transport operation must begin at or after the beginning of previous transport operation $TS[T_{k-1}]$
3. current transport operation must begin at or after the processing on job J_i on machine M_j is completed

Thus the starting time and the ending time of the current transport operation can be expressed as:

$$TS[T_k] = TS\ [R_m(2, i, j)] = \max \{TS[T_{k-1}], TF[R_m], TE[R_m(1, i, j)] + PT(i, j)\} \quad (4.3)$$

$$TE[T_k] = TS[T_k] + HT_2(i, j) \quad (4.4)$$

The lower bound at the current node is machine based. It is not necessary to consider the lower bound concerning machine M_j when $U_j = \phi$ and $p_j = 0$ (that is, there is no job waiting in buffer B_{j-1} and no job is loaded on machine M_j.

The lower bound computation at the current node could be divided into three components. The sum of these three components would give the lower bound at current node. The three components are:

1. Time elapsed till the beginning of the current transport operation
2. The transport operation time + job processing time of all the jobs $j \in U_j$ on machine M_j.
3. Time since the last job finishes its operation on machine M_j until the time that job is completed.

The first component of the Lower Bound computations would at least be equal to the maximum of the following values:

1. the starting time of the current operation (as defined earlier)
2. the earliest time robot R_{m_0} is available
3. The finish time (processing) of the job that was last loaded on machine M_j
4. The time at which the job to be processed is available on buffer B_{j-1}

This can be expressed as:

$$\max \{\text{TS } [T_k], \text{TF}[R_{m_0}], \text{TE}[R_m(1, p_j, j)] + \text{PT}(p_j, j), \text{TE}[R_m(2, b_j, \text{j-1})] \} \quad (4.5)$$

The second component of the lower bound would be equal to

$$\sum_{i \in U_j} \{ HT_1(i,j) + \text{PT}(i,j) \} \quad (4.6)$$

The third component of the lower bound computation would at least take the following amount of time:

$$\min_{k \in U_0} [HT_2(k, j) + \sum_{m=j+1} \{HT_1(k, m) + \text{PT}(k, m) + HT_2(k, m)\}] \qquad (j \leq N - 1) \quad (4.7)$$

$$\min_{k \in U_0} HT_2(k, N) \qquad (j = N) \quad (4.8)$$

Hence the lower bound on the makespan concerning machine M_j is expressed as:

$$\text{LB}_j = \max \{\text{TS } [T_k], \text{TF}[R_{m_0}], \text{TE}[R_m(1, p_j, j)] + \text{PT}(p_j, j), \text{TE}[R_m(2, b_j, \text{j-1})] \} +$$

$$+ \sum_{i \in U_j} \{ HT_1(i,j) + \text{PT}(i,j) \}$$

$$+ \left[\begin{array}{l} \min\limits_{k \in U_0} [HT_2(k, j) + \sum\limits_{m=j+1} \{HT_1(k, m) + \text{PT}(k, m) + HT_2(k, m)\}] \quad (j \leq N - 1) \\ \min\limits_{k \in U_0} HT_2(k, N) \quad (j = N) \end{array} \right] \quad (4.9)$$

The lower bound at current node would be equal to the maximum value of LB_j over all machines j and can be expressed as:

$$LB[T_k] = \max_j LB_j \quad (4.10)$$

Logical flow diagram of the procedure to compute the lower bound is shown in Figure 4.

The following are the assumptions considered by Hitomi et al [15] in formulating the machine-based lower bound:

1. The travel time required by the robot, when not handling the jobs is negligible.
2. All the transfer points in the system (machines, buffers, incoming and outgoing job stations) are accessible by all the robots tending the workcell.
3. The robot with the earliest available time is used for the transport operation.

The following modifications have been made to the assumptions described above:

1. The travel needed by the robot when not handling any workparts (also referred to as idle travel time) is considered to be proportional to the number of transfer points the robot needs to pass in the flowshop, to reach its ready position. For example, if the time needed (specified by the user) to go from one transfer point (say buffer B_1) to the next transfer point (say machine M_1) is 'm' units, then the time taken to go from M_1 to M_3 will be $4 \times m$ ($M_1 - B_2 - M_2 - B_3 - M_3$).
2. A matrix identifying the Robot - Inaccessible transport point is defined. For Hitomi et al's algorithm this matrix will be empty.
3. In a situation when a particular transport operation is accessible by more than one robot, and availability time of these robots are the same then a preferable

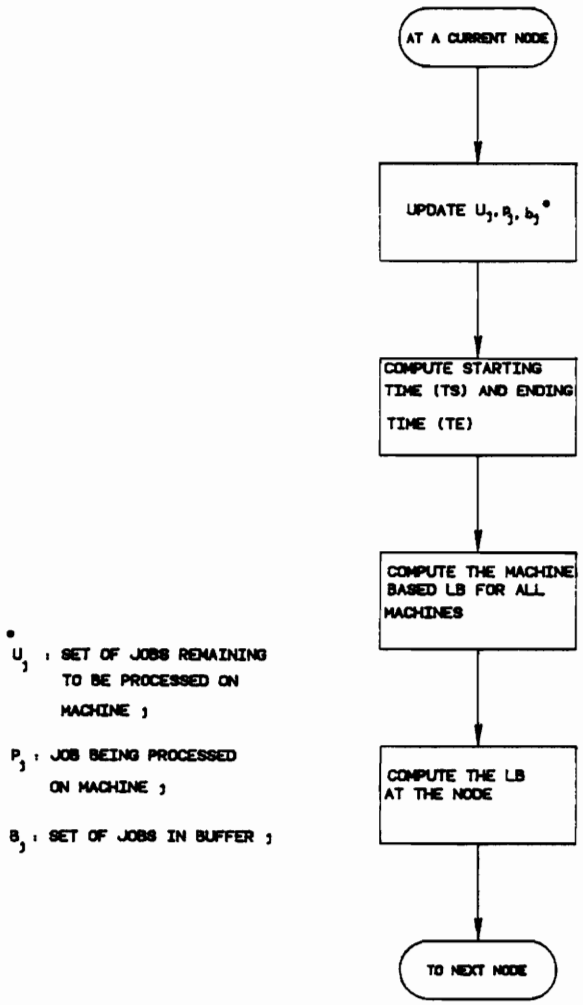


FIGURE 4. LOGICAL FLOW DIAGRAM OF THE PROCEDURE TO COMPUTE THE LOWER BOUND

robot is used for this transport operation (Inaccessibility and Preferability matrices are used in selecting the robot for a particular transport operation).

The following procedure is used at each node to accommodate the suggested modifications for the following two purposes:

1. for computing the lower bound on the makespan value and
2. to determine the robot number to be used for a particular transport operation

The logical flow diagram of the procedure is shown in Figure 5.

1. Check if the current transport operation can be handled by all the robots (Associated with each robot is an inaccessibility look-up table that is inputted to the algorithm). If yes, go to step 3, otherwise continue
2. Update the set L to be the set of robots for which the current transport operation is feasible.
3. Compute the starting time of the current transport operation using Equation 4.1 or 4.3.
4. Compute the idle travel time (l_{tt}) of each robot (in set L) to reach the ready position
5. Let A be defined as the time when one (or more) of the robot(s) can reach the ready position to conduct the next transport operation, then

   $$A = \min_{m \in L} [A_t(R_m) + l_{tt}(R_m)]$$

   a. If A > starting time of the transport operation (TS) (computed in Step 3) then the starting time of the next transport operation would be equal to A. The robot number which gave the value of A is the robot to be used.

      If more than one robot is available at time A, then go to Step 6,

   b. If $A \leq$ TS, and if more than one robot satisfy this condition then go to Step 6,

If not, Reset L (set of robots that can perform the current transport operation) to be the set of all robots and exit this routine to compute the LB of the next node (transport operation) as determined by branching algorithm.

6. Determine the set of robots which have the same starting time A. Each robot has a preferred operation look-up table. This look-up table will give the robot number to be used for the transport operation. Compute the lower bound using equation 4.9 (before proceeding to the new node) with new modified starting time, reset L to the set of all robots.

4.2.1 Branch and Bound Algorithm

New nodes are generated from every node selected for branching, This is done using the following rules:

1. If there are job(s) waiting at buffer 1 ($U_1 \neq \phi$) and if no job is loaded on machine 1 ($p_1 = 0$) then the transport operation $R_m(1, i, 1)$ of job $i \in U_0$ from the incoming job station (B_1) to machine M_1 is generated.
2. If there are job(s) waiting at buffer j ($b_j \neq 0$) and if no job is loaded on machine j ($p_j = 0$) transport operation $R_m(1, b_j, j)$ of job b_j from buffer B_j to machine M_j is generated.
3. If a job is loaded at machine j ($p_j \neq 0$) , transport operation $R_m(2, p_j, j)$ of job p_j from machine M_j to buffer B_{j+1} is generated.

   Note that if $b_j = 0$ then transport operation $R_m(1, b_j, j)$ is not generated because $b_j = 0$

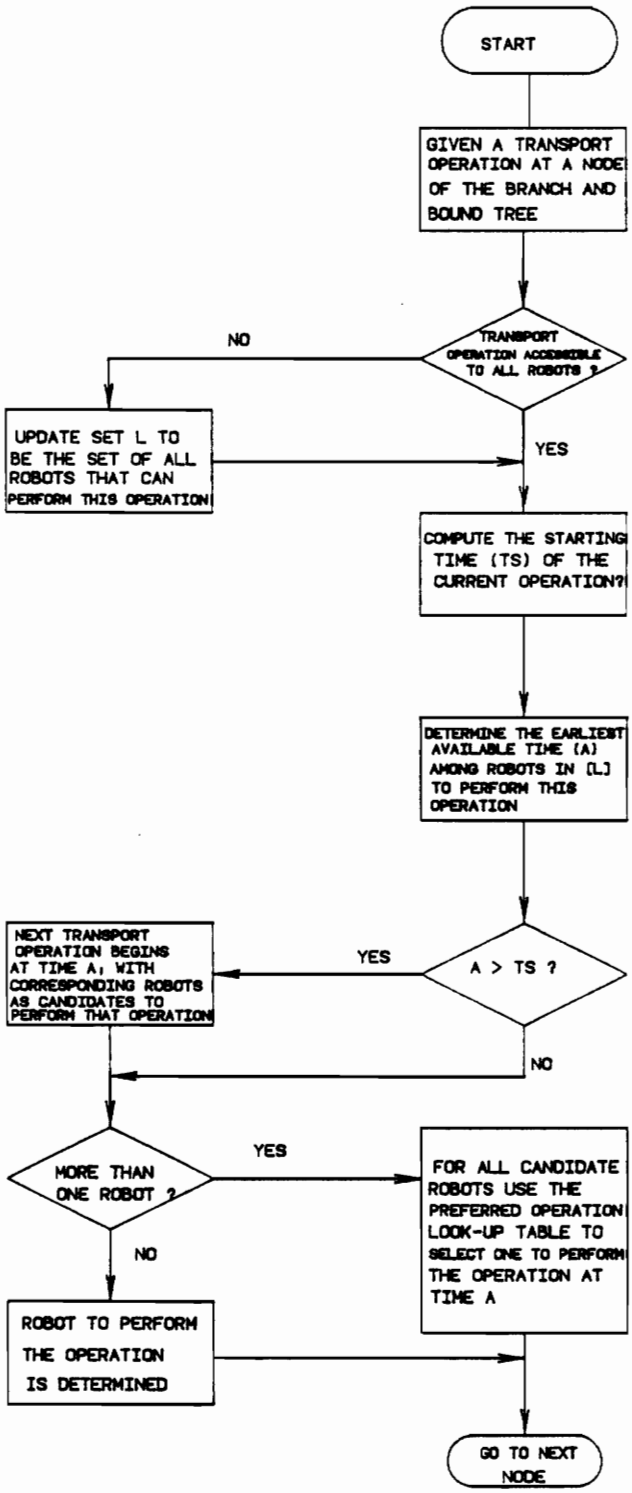


FIGURE 5. LOGICAL FLOW DIAGRAM OF THE PROCEDURE TO SELECT A ROBOT TO PERFORM A TRANSPORT OPERATION AT A NODE OF THE BRANCH AND BOUND TREE

From any node, at the most N nodes would be created. Each node is a valid transport operation. Lower bounds at each node are computed. Also the algorithm discussed above is applied at each node to accommodate the modified assumptions. The node with the lowest lower bound value is used for further branching. One complete sequence is determined by branching from the node which has the lowest lower bound at current level of branching. This complete sequence would give the upper bound. Backtracking is used to find better sequences than the sequence that gave the upper bound. At the end of the branch and bound algorithm an optimal sequence of robot transport operations and the robot numbers which should handle the transport operation are determined.

Figure 6 shows the branching of a 2 job, 2 machine hypothetical problem. The number below each node shows the lower bound. The number at the top of each node indicates the sequence in which the nodes were evaluated. The logical flow diagram of the branch and bound algorithm is shown in Figure 7.

4.2.2 N-best sequence determination

As described earlier in this chapter, one full sequence is determined by branching from the node which has the lowest LB value among all the nodes at the current level. The makespan of this sequence would give the upper bound value. This upper bound value is used to backtrack in search of better solutions. One of the strategies that can be used to determine the N-best sequences is discussed here. The upper bound established by determining one full sequence can be used to backtrack to the node which which has a LB value which is less than equal to than

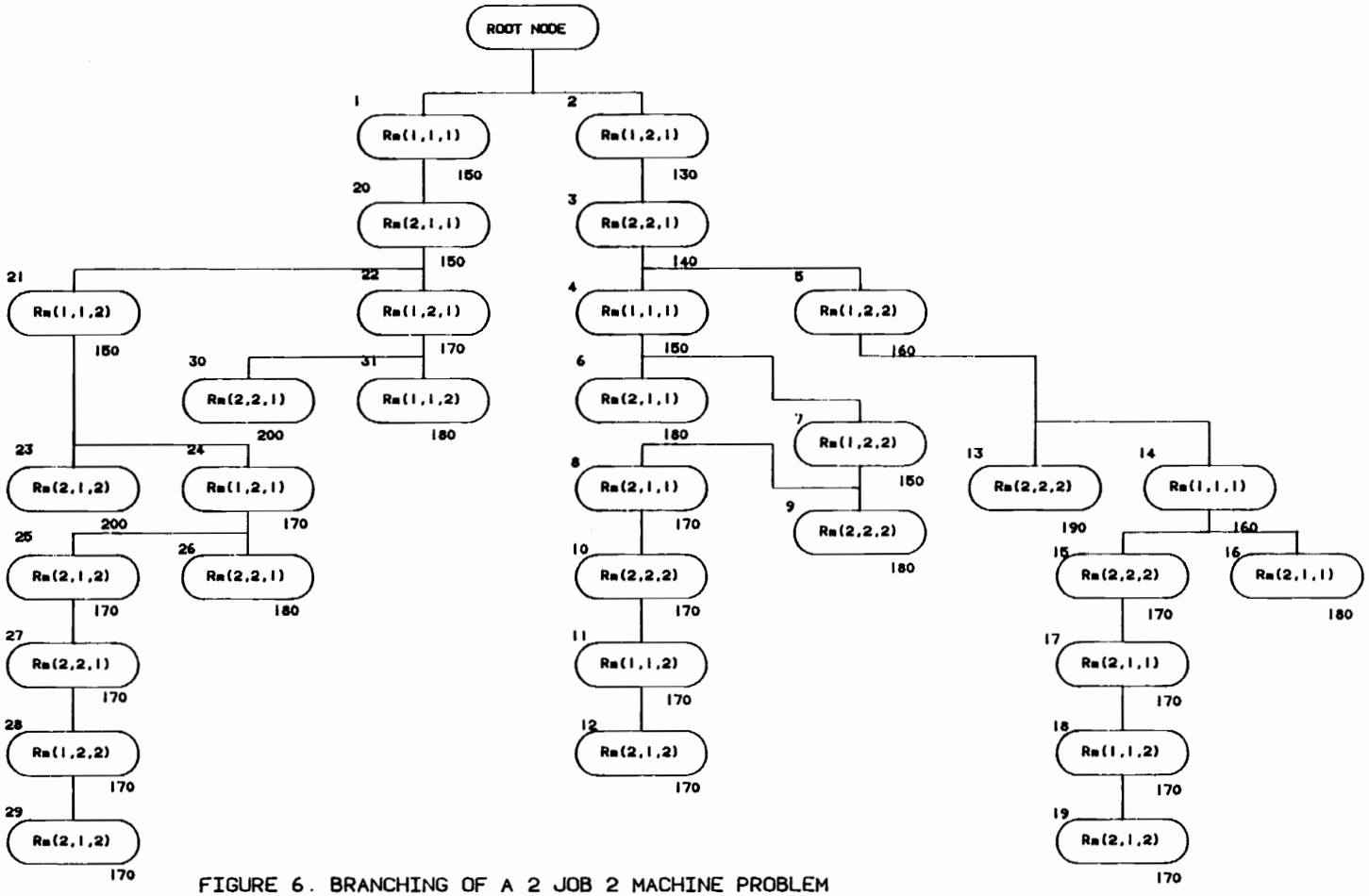


FIGURE 6. BRANCHING OF A 2 JOB 2 MACHINE PROBLEM

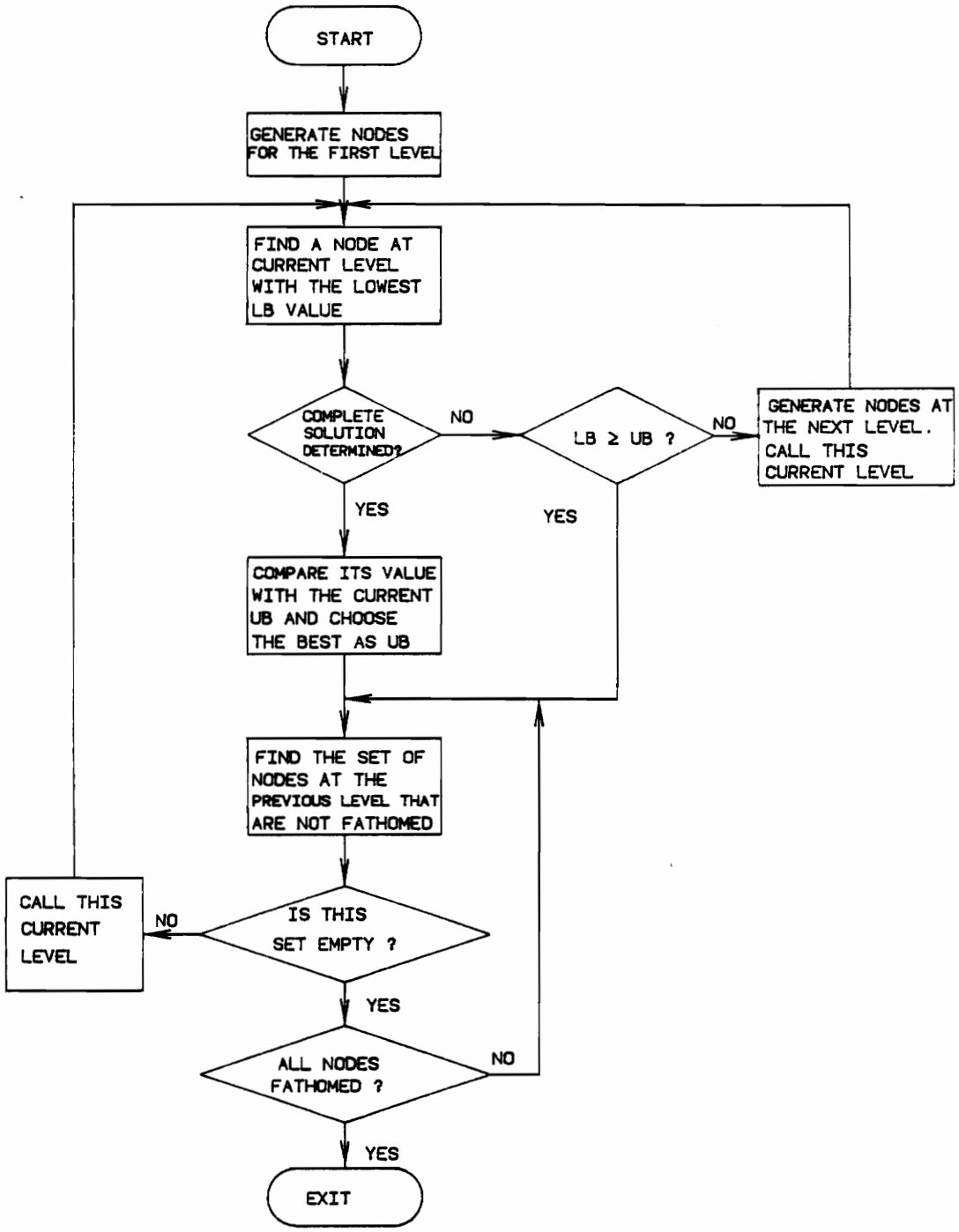


FIGURE 7. LOGICAL FLOW DIAGRAM OF THE BRANCH AND BOUND ALGORITHM

the upper bound. Further branching from this node would give another solution. The higher makespan value among these two sequences would be used as the upper bound. Continued branching in this fashion would result in new sequences. The best N sequences would be stored. The logical flow diagram of this procedure is shown in Figure 8.

The use of this strategy would result in almost exhaustive search of the tree, thereby increasing the computations tremendously. An alternate strategy is used in this research which significantly curtails the computations. Here, the makespan of the best sequence determined till now will be used as the upper bound. This strategy ensures that the optimal sequence will be determined. Also all alternate optimal solutions will be determined (if there are any). However, the additional N-1 sequences determined may not be the best sequences. The N-1 best sequences encountered during this search for the optimal solution are stored.

This strategy has its limitation. If the first sequence determined is the optimal sequence, then no other sequences may be stored, unless there are alternate optimal solutions. Also, if the optimal sequence is determined early on in the branching, the other N-1 sequences may not good solutions. This strategy was used as a trade-off to save on the computation time. The logical flow diagram of this strategy is shown in Figure 9.

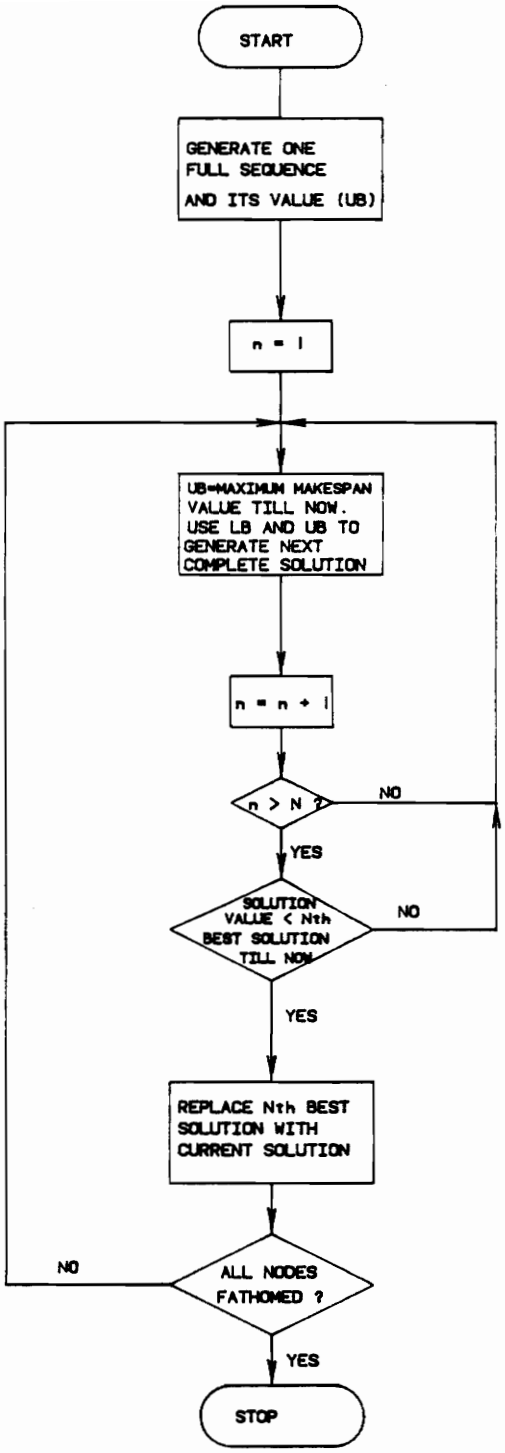


FIGURE 8. LOGICAL FLOW DIAGRAM OF THE PROCEDURE TO DERIVE THE N BEST SOLUTIONS

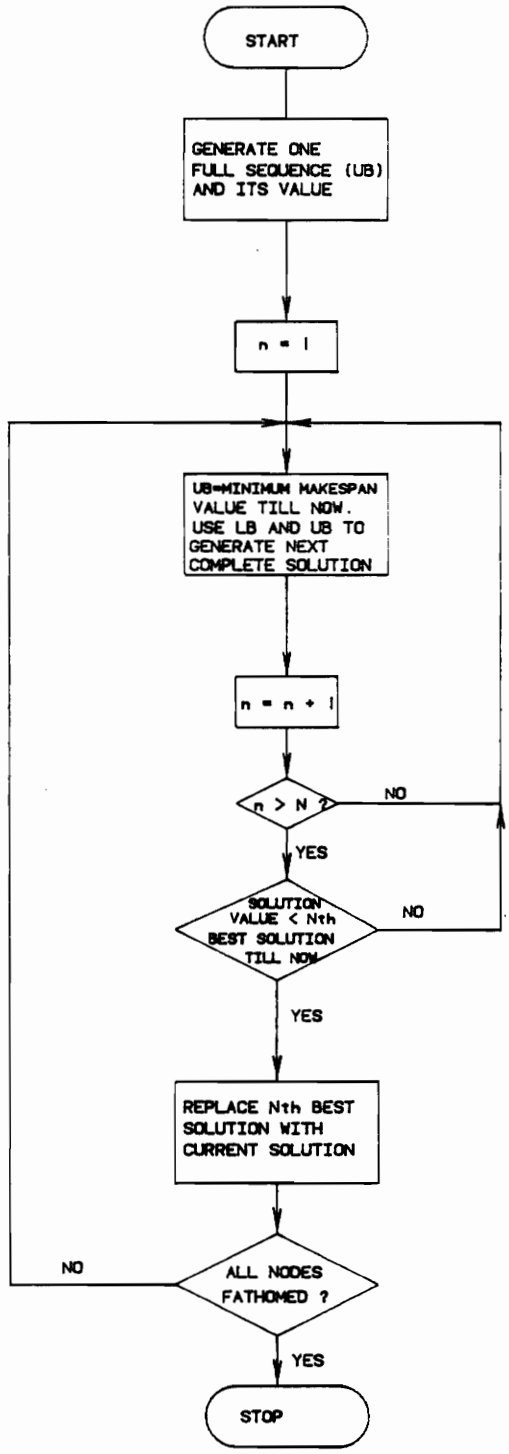


FIGURE 9. LOGICAL FLOW DIAGRAM OF THE PROCEDURE TO DERIVE THE N GOOD SOLUTIONS

4.3 Implementation of the branch and bound algorithm

The branch and bound algorithm is a set of FORTRAN programs. The main control program written in Rexx, drives the branch and bound algorithm. The following are input to the branch and bound algorithm.

1. Number of jobs
2. Number of machines (number of buffers = number of machines + 1)
3. Transport operation times needed for each transport operation
4. Job processing times of each job on every machine
5. Infeasibility matrix - indicating the set of infeasible transport operations for a particular robot
6. Preferability matrix - indicating the robot number to be preferred if the starting times of more than one robot is preferable
7. Unit idle travel time of the robot. This is the time needed for the idle robot to travel from a buffer to the next machine or vice versa.
8. The number of best sequences to be stored. This is necessary in case the optimal sequence may not be feasible because of interference among the robots.

The branch and bound program would determine the 'N' best sequences as discussed in section 4.2. Each transport operation in the sequence is identified by the following four entities:

1. the job number
2. the machine number

3. a code, which has a value of 1, if the transport operation is from buffer to machine and 2 otherwise
4. the robot number to be used in the transport operation

These sequences are stored and are available for graphical simulation.

5.0 Graphical Simulation in the Integrated System

This chapter describes the operation of the integrated system. Optimal sequencing of robot transport operations is integrated with 3-D graphical simulation of the workcell. We first describe the overall structure of the integrated system and then the graphical simulation of the workcell.

5.1 Integrated System

As described in Chapter 3, the system is developed for a flowshop. The transfer of jobs from machines to buffers and from buffers to machines is done by robots. The system uses two identical robots for the transport operations. Figure 10 shows the overview of the integrated system. The single arrows in the flow diagram describe the sequence in which the system carries out its actions.

The main control program, written in Rexx, manages all the programs and relates the appropriate programs with the appropriate data. The input to the system consists of positions for all the robots, buffers, and machines, as 3-D geometric data. Also, the transport operation times, job processing times, the number of robots, buffers, and machines in the system are input to the system. Also, the Feasibility and Preferability matrices described in Chapter 4 are input to the system.

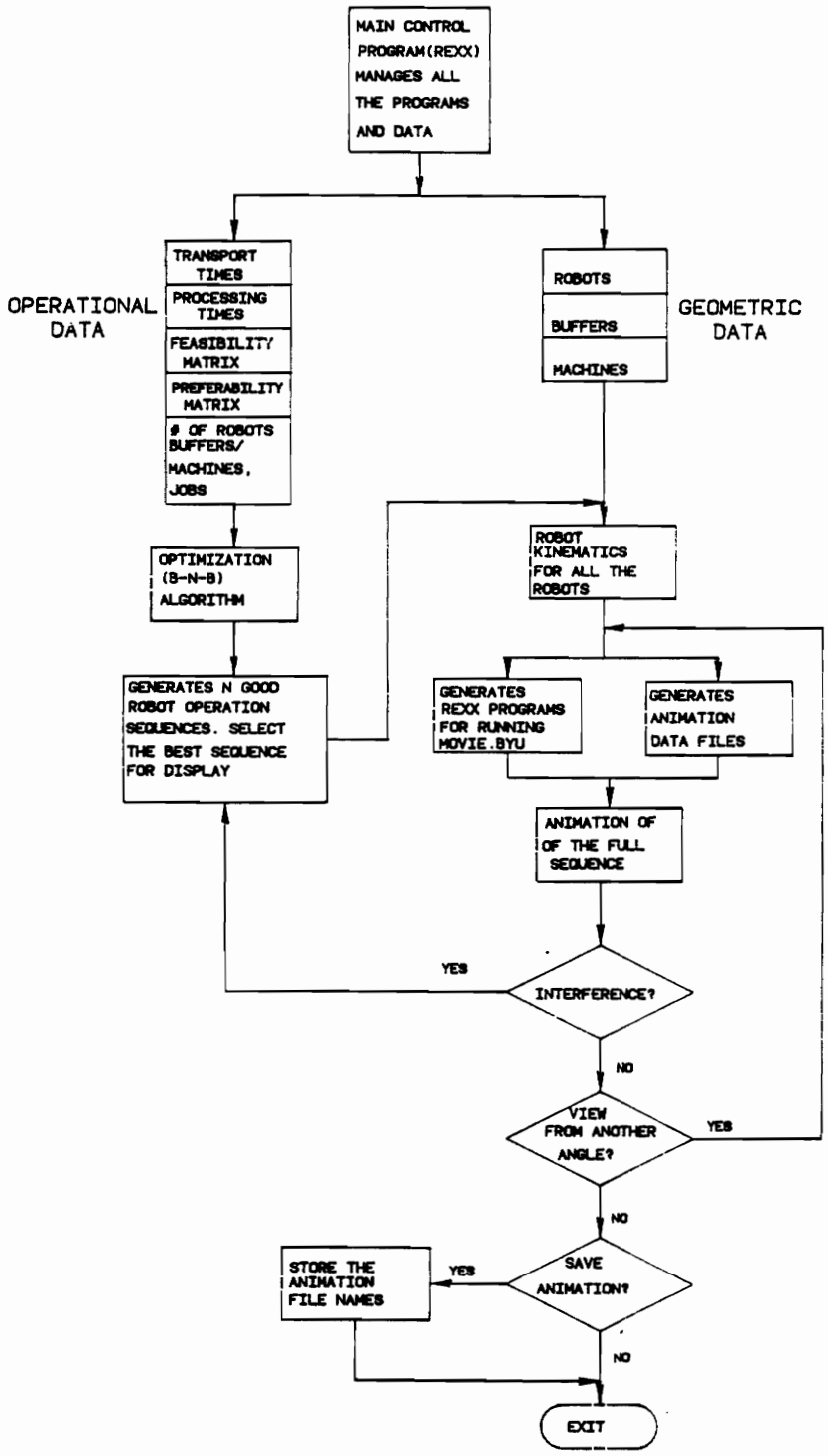


FIGURE 10. OVERVIEW OF THE INTEGRATED SYSTEM

The main control program drives the branch and bound algorithm to sequence the transport operations of the robot optimally to minimize makespan. The branch and bound algorithm generates a specified number of best sequences of robot transport operations. Every transport operation is identified with the following parameters:

1. job number
2. machine number
3. a code, with a value equal to 1 if transport operation is from buffer to machine, 2 otherwise
4. robot number of the robot which is to carry out the transport operation

The main control program then takes the optimal sequence and makes it available to the kinematics program. The kinematics program uses the geometric location data of the machines, buffers, and robots and the optimal robot transport operation sequence to generate animation data files. The kinematics programs use closed form inverse kinematic solutions. The exact function of the kinematics program and the graphical simulation will be discussed in more detail in the next section of this chapter.

The listing of the main control program, which is written in Rexx, is provided in the Appendix to this thesis. The procedure RUN_BNB drives the branch and bound algorithm for optimal robot task sequencing. The procedure RUN_KINEMA drives the kinematics program which uses the optimal robot task sequence on the graphical simulation. The procedure also selects new near optimal sequences if the optimal sequence is found to be unsuitable. Most of the other procedures interact with the user and manage all the data.

The graphical representation of the workcell and the animation of the robot(s) performing the transport operations are displayed on the Tektronix 41XX graphics workstation. The main control program helps in merging the appropriate animation data files and in making these data files available to the graphics package MOVIE.BYU. The MOVIE.BYU package can then display the robots performing the sequence of transport operations.

After displaying the sequence, the control is transferred to the user. The animation can be viewed from various user specified angles to detect the possibility of interference among the robots.

A sequence that is found unsuitable can be discarded and the next best sequence can be displayed. The procedure can be repeated until a satisfactory solution is identified. The sequence of transport operations selected in this way, would be the optimal feasible sequence. Any sequence generated can be stored for future display.

5.2 Graphical Simulation

Two major programs of the graphics system MOVIE.BYU, viz., UTILITY and DISPLAY are used for the graphical simulation. UTILITY is the data generation and editing program. UTILITY is used to generate data files for the location of all the buffers and machines. A data file containing the 3-D coordinates of the location of machines and buffers was interactively created using UTILITY. UTILITY saves this data file in the format compatible with DISPLAY. Figure 11 describes the various sections of the data file containing the 3-D coordinates of the buffers and machines. The italicized

text in the Figures are comments by the author to explain the data. A similar approach is used to generate the initial position of all the robots in the workcell and is explained in Figure 12. The user should be familiar with MOVIE.BYU to be able to create location data for machines/buffers and robot(s).

For inverse kinematics the transformation matrix $[C_{TCP}]_0$ for every individual machine and buffer relative to every robot is provided. A partial data file of these matrices is explained in Figure 13. The sequence of transport operations, transport operation times, and the job processing times are also input to the kinematics programs.

The kinematics program has all the data described above as input. The main control program loads the kinematics program. For performing a job transport operation, the program determines the following:

1. The TCP's initial position
2. The TCP's position required to reach a ready state to start the transport operation
3. The TCP's position after the transport operation (at the target location)

The transformation matrices relative to each of the above described locations are used to find the inverse kinematic solutions. The inverse kinematic solutions, will help determine the intermediate link orientations. This information is written out to data files for subsequent use by MOVIE.BYU. Along with these data files, the kinematics program also generates Rexx programs. These Rexx programs, under the control of the main program, essentially allows automatic execution of a set of commands needed by UTILITY to merge appropriate data files. This process is repeated

No. of parts	No. of nodes	No. of polygons	No. of elements in connectivity array
7	56	28	112

1. each buffer and machine is a part
2. Part 1 : Buffer 1, Part 2 : Machine 1, ...
3. each machine/buffer has 8 nodes
4. each buffer/machine has 4 polygons

1 8 9 16 17 24 25 32 33 40 41 48 49 56

The above numbers describe the parts list, that is, points 1 through 8 describe one part and so on

	Node no.	X coor.	Y coor.	Z coor.
Part 1	1	-1.00000E + 00	0.00000E + 00	0.60000E + 00
	2	-0.40000E + 00	0.00000E + 00	0.60000E + 00
	3	-0.40000E + 00	0.00000E + 00	1.00000E + 00
	4	-1.00000E + 00	0.00000E + 00	1.00000E + 00
	5	-1.00000E + 00	1.00000E + 00	0.60000E + 00
	6	-0.40000E + 00	1.00000E + 00	0.60000E + 00
	7	-0.40000E + 00	1.00000E + 00	1.00000E + 00
	8	-1.00000E + 00	1.00000E + 00	1.00000E + 00

The above values are the x, y, and z coordinates of points that describe buffer 1 Similarly, coordinates for other parts can be displayed

1 2 3 -4 5 6 7 -8 1 5 6 -2 3 7 8 -4

The numbers shown above represent the connectivity of the coordinate numbers of the nodes of buffer 1. Thus, coordinates 1, 2, 3, and 4 form a polygon and so on

Figure 11. Data needed by MOVIE.BYU to describe a part

No. of parts	No. of nodes	No. of polygons	No. of elements in connectivity array
2	12	8	12

1 6 7 12

	Node no.	X coor.	Y coor.	Z coor.
Robot 1	1	0.00000E + 00	0.00000E + 00	0.00000E + 00
	2	0.00000E + 00	0.15000E + 01	0.00000E + 00
	3	0.10000E + 01	0.15000E + 01	0.00000E + 00
	4	0.00000E + 00	0.15000E + 01	0.00000E + 00
	5	0.15000E + 01	0.15000E + 01	0.00000E + 00
	6	0.17500E + 01	0.15000E + 01	0.00000E + 00

The above six points (x, y, z coordinates) describe the line diagram of robot 1 Simlarly, coordinates for robot 2 can be displayed

1 2 4 3 5 6 7 8 10 9 11 12

Figure 12. A sample data file showing the 3-D locations of robots

Robot 1 reaching Buffer 1	Normal Vector(N)	-0.8	0.2	9.9	0.0
	Orientation(O)	0.5	0.6	9.9	0.0
	Approach(A)	-0.52	0.6	9.9	0.0
	Position(R)	-0.6	1.1	0.7	1.0

The above four lines are the normal, orientation, approach, and position vector ($[C_{TCP}]_0$) needed of robot 1, to be able to place a part in buffer 1

Robot 1 reaching Machine 1	Normal Vector(N)	0.7	0.5	9.9	0.0
	Orientation(O)	0.7	0.4	9.9	0.0
	Approach(A)	0.075	0.99	9.9	0.0
	Position(R)	0.25	1.5	0.4	1.0

Similar matrices are be described for all the buffers and machines

Robot 2 reaching Buffer 1 *NA	Normal Vector(N)	0.6	-0.6	9.9	0.0
	Orientation(O)	0.6	0.44	9.9	0.0
	Approach(A)	-0.33	0.5	9.9	0.0
	Position(R)	-2.4	1.1	0.7	1.0

Similar matrices are described for robot 2 reaching all the buffers and machines

*** NA - represents that the buffer 1 is not accessible to robot 2. However, robot 2 - buffer 1 pair should be included in the Inaccesibility matrix***

Figure 13. Transformation matrices associated with some of the machines and buffers

```
/* Exec to merge data files corresponding to the  7th transport

 operation in the sequence. Name of this file UTIL712 EXEC */
QUEUE       'MERG'
QUEUE       'POS7201'
QUEUE       '2'
QUEUE       '2'
QUEUE       '  '
QUEUE       '  '
```

Sequence of commands passed to UTILITY, The file naming convention of the data file POS7201 is interpreted by the FORTRAN as : the first step of the animation of transport opertion 7 performed by robot 2

```
QUEUE       'POS7202'
QUEUE       '2'
QUEUE       '2'
QUEUE       '  '
QUEUE       '  '
QUEUE       'POS7203'
QUEUE       '2'
QUEUE       '2'
QUEUE       '  '
QUEUE       '  '
QUEUE       'GEOM'
QUEUE       'WRIT'
QUEUE       'OPER712'
QUEUE       'N'
QUEUE       '  '
QUEUE       'EXIT'
EXIT
```

Figure 14. A partial Rexx program to merge the data files associated with robot number 2 reaching a ready position for the 7th transport operation

for all the transport operations. A sample Rexx program generated by the kinematics program, to merge data files using UTILITY, is shown in Figure 14. It should be noted that before data editing can be started, the main program would load the UTILITY module of MOVIE.BYU.

The kinematics program also generates a Rexx program which would invoke DISPLAY and execute the commands needed to display the workcell and the animation of robot transport operations. Thus, once all the appropriate data files have been merged, the sequence of transport operations are displayed one at a time. There is no interaction needed from the user until this time.

5.3 Interference Detection

Interference detection is accomplished through graphical displays. The first sequence displayed is the optimal sequence. The front view of the workcell is displayed at this time. Once the complete transport operation sequence is displayed, the control is transferred to the user. If no interference is detected, then the optimal sequence is declared feasible. If an interference is seen on the screen in the front view of the cell then there is a possibility that there is actually no interference among the robots. To detect that, the workcell can be viewed from other angles. The top and the side views can be selected for viewing. The representative angles for the top view are $x = 90$ and $z = 90$ and for the side view they are $y = 90$ and $z = 90$. The main program edits the intermediate Rexx programs to change the angle of view. The whole process is repeated for the new view angle.

If at an angle of view interference is not detected on the screen then the optimal sequence is declared feasible. If the view still shows interference, then there is a possibility of actual interference among robots. The user can specify other angles of view. If a sequence does not appear to be interference free, then the next best sequence is selected. This process is repeated till a satisfactory sequence is determined.

5.4 Summary

The Integrated System is developed on the IBM 4341 minicomputer. The graphical simulation of the workcell is done using the MOVIE.BYU package. A powerful interpreted language, Rexx, is used at various stages in the system for two main reasons:

1. Rexx does not need any explicit variable type declarations
2. Rexx can execute almost any VM/CMS (Operating System) commands during its normal execution

The whole system is modular and flexible. Minimal re-programming is expected for using the system for variety of applications and objectives. For example, the geometric representation and locations of robots, buffers, and machines are separate data files. Hence, changing the layout of the workcell is a trivial task of editing the appropriate data files (with or without UTILITY). However, the user should be familiar with the graphics system MOVIE.BYU and the format in which it reads the data files. Including a new robot type in the system would involve re-programming of the

kinematics, if a closed form inverse kinematic solution is available. The Rexx programs for driving UTILITY and DISPLAY program are generated during the kinematic program execution. This eliminates the tedious task of merging appropriate data files using UTILITY.

6.0 Demonstration of the Integrated System on a Sample problem

In this chapter we will present a sample run on the 3 machine, 3 job, and 2 robot problem. Hence, there would be four buffers in the flowshop. First, all the input needed to the problem will be illustrated. Throughout this chapter italicized text represents system prompts or display on the terminal.

6.1 System input

All the input needed by the system is listed below:

1. Transport operation and processing times

	Transport operations from buffer to machine			Transport operations from machine to buffer			Processing times		
From	B_1	B_2	B_3	M_1	M_2	M_3	M_1	M_2	M_3
Job 1	80	15	178	21	28	83	21	24	78
Job 2	12	206	23	40	30	92	40	26	80
Job 3	103	31	301	36	48	82	36	45	85

2. Infeasibility Matrix

Robot	Machine/Buffer
1	Machine 2
2	Buffer 1

The above infeasibility matrix indicates that object 4 is inaccessible to robot number 1 and object 1 is inaccessible to robot number 2. In the flowshop being considered object 4 would be machine number 2 and object 1 would be buffer number 1.

3. Preferability Matrix

Robot	Operation Type	Machine
1	2	1
2	1	2
1	1	3
1	2	3

The preferability matrix shown above would be interpreted as follows. Preferably the transport operation from machine 1 to buffer 2 should be carried out by robot number 1 and so on.

4. Number of best sequences to be generated - 5

5. Idle travel time per unit distance of travel - 5 units (can provide a matrix, if such data is available)

6. Coordinates of the initial location of the robot (created by UTILITY)

7. Coordinates of the location of the machines and buffers

8. $[C_{TCP}]_0$ matrices for the location of each of the machines and buffers relative to both the robots

6.2 Data Creation

The creation of the input data for the system is described here. One of the first things that needs to be created is the location data of all the buffers, machines, and robots. This data can be created interactively using UTILITY or using an editor. The use of UTILITY would require the user to be familiar with MOVIE.BYU. The different sections of this data are described in Figures 11 and 12 in Chapter 5. This data would be in a format compatible with MOVIE.BYU.

Once this data is created, we would need to create the $[C_{TCP}]_0$ matrices corresponding to the position and orientation of each of the robot(s). This $[C_{TCP}]_0$ matrices would be created for all the machines and buffers. This matrices are shown in Figure 13. Additionally, the infeasibility matrix and the preferability matrix should be created. Also, the idle travel time per unit distance travelled and the number of best sequences to be stored should be given in a data file. The transport operation times and the processing times would be given in a matrix as shown in the previous section. The program will prompt for the file names of all the data files.

Transport Opn. Type	Job #	Machine #	Robot #
1	1	1	1
2	1	1	1
1	3	1	1
1	1	2	2
2	1	2	2
1	1	3	2
2	3	1	1
1	2	1	1
1	3	2	2
2	2	1	1
2	3	2	2
1	2	2	2
2	1	3	1
1	3	3	1
2	2	2	2
2	3	3	1
1	2	3	2
2	2	3	2

Figure 15. The optimal transport operation sequence

6.3 Sample System Execution

This section shows the sample run of the entire system. Intermediate output is shown wherever possible. All italicized text are system prompts or messages displayed on the screen.

IROGSS

Enter the name of the file containing location of the machines and buffers

QUIT to exit the program

FILE MACHBU

Enter the name of the file containing location of the robots

QUIT to exit the program

FILE ROBOTS

Enter the name of the data file containing [C tcp] 0 matrices

QUIT to exit the program

CTCP0 DATA

Enter the name of the data file containing transport operation and processing times

QUIT to exit the program

TRANSP DATA

Enter the name of the data file containing Infeasibility matrix

QUIT to exit the program

INFEAS DATA

Enter the name of the data file containing Preferability matrix

QUIT to exit the program

PREFER DATA

The workcell contains :

1 - 3 jobs

2 - 3 machines

3 - 4 buffers

4 - 2 robots

5 - No. of best transport operation sequences to be determined = 5

6 - Idle travel time per unit travel = 5

Is all the input correct (Y/N)

Y

Optimal Solution Determined .....

The optimal solution determined is shown in Figure 15. Besides this optimal solution, four other best solutions are stored, and can be used if and when needed. After the determination of these best transport operation sequences, the kinematics programs are loaded and executed. The data corresponding to the initial position of the robot(s) are available. Also, the $[C_{TCP}]_0$ of each machine and buffer relative to all the robots are known. Hence, for each transport operation shown in Figure 15, the initial and the destination points and the robot number to be used are known. The kinematics program uses this information to solve for the intermediate orientation of robot(s) to perform the transport operation. Before the start of the kinematics program, IROGSS checks for the presence and validity of relevant input data. This part of the IROGSS is not shown here. After this process the IROGSS program issues the following commands.

EXEC LDISK MOVIE 192

LOADMOD UTILITY (NOMAP

START

(MERG)

<READ GEOM FILE>

<READ: 2 PARTS; 12 COORDINATES; 8 ELEMENTS>

<PART LIMITS>

.

.

.

<CHANGES?>

GEOM >>

Here the IROGSS links to the MOVIE disk and loads the UTILITY module of MOVIE.BYU. The intermediate EXECs created by the kinematics program stack all the commands, needed by UTILITY to merge appropriate data files, to the console stack. Utility uses these commands to have the data files ready for usage by DISPLAY module of MOVIE.BYU. The lines displayed above are the MOVIE.BYU prompts as it reads FIFO from the console stack.

LOADMOD DISPLAYT (NOMAP

START

<MOVIE SYSTEM DISPLAY>

<READ GEOM FILE>

<READ: 24 PARTS; 144 COORDINATES; 84 ELEMENTS>

<READ DISP FILE>

.

.

.

The sequence of transport operations is displayed with a pause between each transport operation. Again the lines displayed above are prompts by the DISPLAY module of MOVIE.BYU as it reads the commands from the console stack. After the full sequence is displayed the program returns with the following prompt.

Would you like to view the sequence of transport operations from

another angle (Y/N)

Y (User response)

Enter the angle of rotation about x,y,z

0 90 90 (User response)

LOADMOD DISPLAYT (NOMAP

START

<READ GEOM FILE>

.

.

<AXIS, ANGLE>

<AXIS, ANGLE>

<AXIS, ANGLE>

Animation of the transport operation sequence is displayed from the user specified angle. The drawings of intermediate stationary images in the display of one transport operation in the optimal sequence is shown in Figure 16. The drawings show interference among the robots in the front view of the display sequence. At the end of the animation sequence the program returns with the following prompt:

Would you like to view the sequence of transport operations from

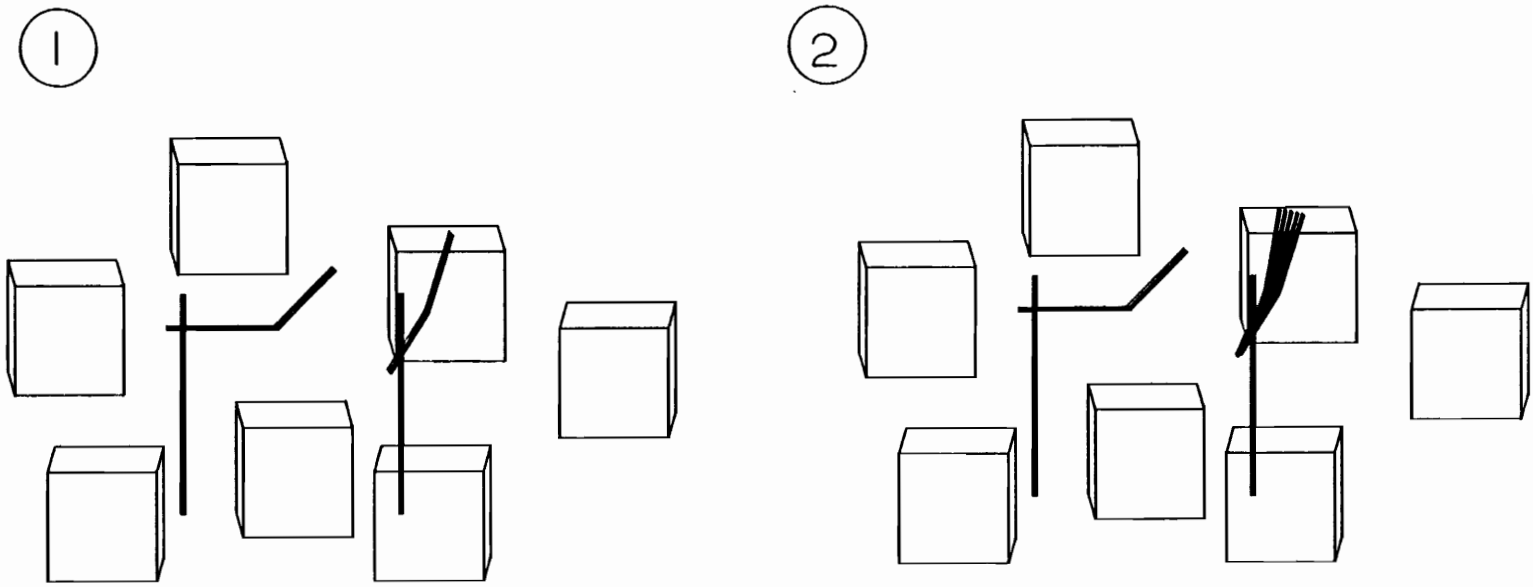


FIGURE 16. DRAWINGS OF INTERMEDIATE IMAGES IN THE DISPLAY OF A TRANSPORT OPERATION (CONTINUED ON NEXT PAGE)

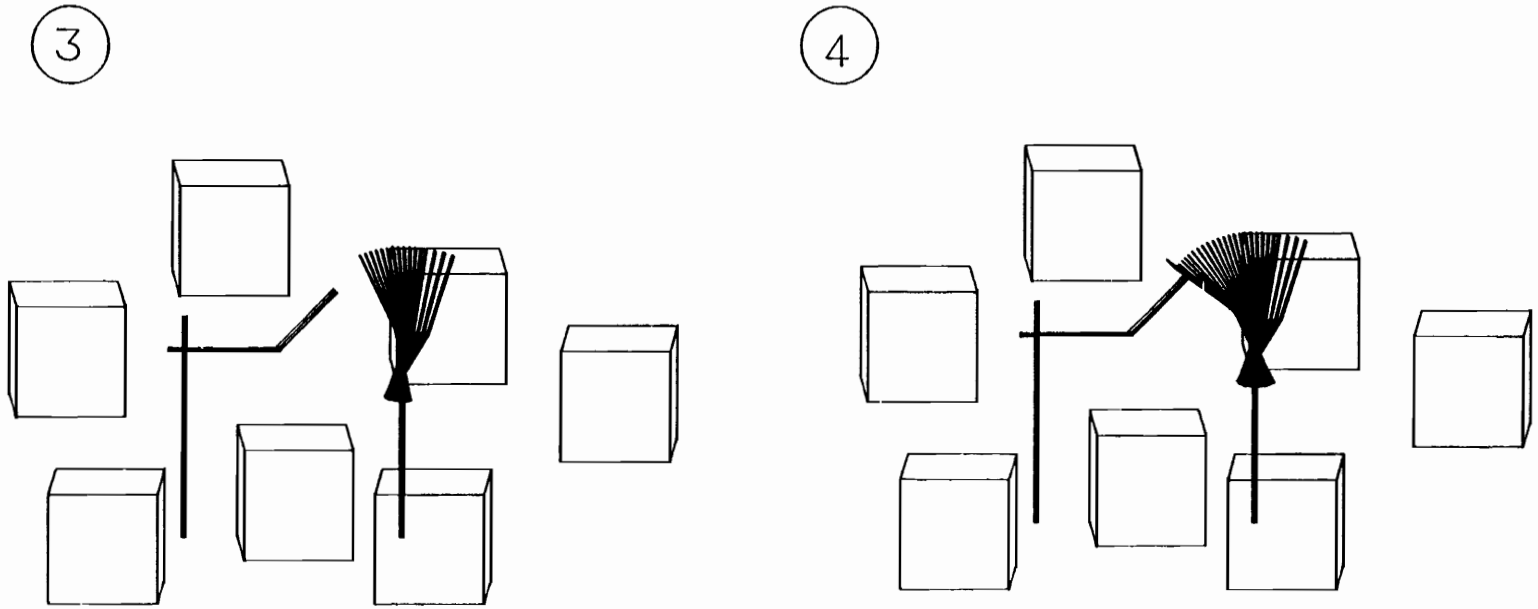


FIGURE 16. (CONT'D) DRAWINGS OF INTERMEDIATE IMAGES IN THE DISPLAY OF A TRANSPORT OPERATION. THE 4TH IMAGE SHOWS INTERFERENCE AMONG THE ROBOTS

Transport Opn. Type	Job #	Machine #	Robot #
1	3	1	1
2	3	1	1
1	2	1	1
1	3	2	2
2	2	1	1
1	1	1	1
2	3	2	2
1	2	2	2
1	3	3	1
2	1	1	2
2	2	2	2
1	1	2	2
2	1	2	2
2	3	3	1
1	2	3	2
2	2	3	2
1	1	3	1
2	1	3	1

Figure 17. Transport operation sequence of a feasible sequence

another angle (Y/N)

N (User response)

Would you like to view the next best sequence (Y/N)

Y (User response)

The next best sequence is shown in Figure 17 will be used now.

LOADMOD UTILITY (NOMAP

START

(MERG)

<READ GEOM FILE>

<READ: 2 PARTS; 12 COORDINATES; 8 ELEMENTS>

<PART LIMITS>

.

.

.

<CHANGES?>

GEOM >>

LOADMOD DISPLAYT (NOMAP

START

<MOVIE SYSTEM DISPLAY>

<READ GEOM FILE>

<READ: 24 PARTS; 144 COORDINATES; 84 ELEMENTS>

<READ DISP FILE>

.

.

.

```
3 jobs
3 machines
2 robots
5 idle travel time
314CTCP0 DATA - the [C tcp] 0 matrices
FILE 314011  (Data corresponding to transport operation 1 by robot 1)
FILE 314021
FILE 314031
FILE 314041
FILE 314051
FILE 314061
FILE 314071
FILE 314081
FILE 3140091
FILE 314101
FILE 314111
FILE 314121
FILE 314131
FILE 314141
FILE 314151
FILE 314161
FILE 314171
FILE 314181
Input data file names (for user reference only)
TRANSP DATA
INFEAS DATA
PREFER DATA
```

Figure 18. The pointer file would be used by the program REUSE to display the sequence of transport operations

The sequence of transport operation is displayed with a pause between each transport operation. After the full sequence is displayed the program returns with the following prompt.

Would you like to view the sequence of transport operations from another angle (Y/N)

N (User response)

Would you like to view the next best sequence (Y/N)

N (User response)

Would you like to store the animation data corresponding to the current sequence (Y/N)

Y (User response)

Enter a 3 character ID which would be used when calling this sequence back for display

314 (User response, here day of the year is specified as the ID)

The current transport operation sequence is stored in the file 314TRANS SEQUENCE

This file is shown in Figure 17.

The makespan corresponding to this sequence is 1119

Robot 1 was used in 11 transport operations

Robot 2 was used in 7 transport operations

Total of 0.32745 hours spent with IROGSS

The pointer file is stored in the file 314REUSE POINTER

This file is shown in Figure 18.

The program REUSE will allow the user to re-use the transport operation sequence stored. This program would prompt for the 3 character ID. Using this 3 char-

acter ID the REUSE program would look into the pointer file 314REUSE POINTER and gather all the information needed. It checks for the existence of all the needed data and re-displays the sequence. The user has the option of viewing this sequence from any angle.

7.0 Conclusions and Recommendations

This Chapter summarizes the research conducted in developing an integrated optimization procedure for the sequencing of robot operations in a workcell. Recommendations for future research in some of the aspects of robotic workcell design are also made.

7.1 Summary and Conclusions

A majority of the industrial robots are being programmed by leading the robots through the desired set of operations. This method of programming does not allow for much flexibility in the workcell design process. Workcells can be more efficient, with the use of better robot programming methods and better robot cell planning.

A branch and bound algorithm was implemented, which gives optimal sequencing of robot operations for the flowshop being considered. The branch and bound algorithm gives substantially improved solutions over the randomly generated solutions. The current prototype system uses identical cylindrical coordinate robots. The simpler geometry of the robot(s) used, allows for the derivation of "closed form" inverse kinematic solutions. The sequences generated by the branch and bound al-

gorithm are displayed on the graphical simulation of the workcell to detect interference among the robots.

The prototype system seeks to demonstrate the possibility of optimal sequencing of robot operations in a workcell. The optimal sequences can be verified on a graphical simulation to plan optimal feasible sequences. However, the graphical simulation in the proposed system is based on robot kinematics. Thus, it cannot accurately represent the path followed by the robot.

7.2 Recommendations for Further Research

An interference detection algorithm for multiple robots in a workcell can be developed. The algorithm would have to be simplified to reduce computations and be implemented in interactive systems. The implementation of such an algorithm can be interfaced with the integrated system. Availability of such an algorithm can eliminate the use of graphical simulation. However, graphical simulation system would be necessary for applications such as workcell layout design, etc.

MOVIE.BYU is a command driven graphics system. Use of some of the function based graphics system currently available can allow more flexibility in representing and changing workcells. This is especially important when cell design and system development are to done by different people. Thus the person designing the robot workcell need not be familiar with the graphics system being used.

The system can be interfaced with the actual workcell and permit switching from off-line to on-line. This kind of function would be available in many of the robot programming languages, which in turn can interact with the integrated system.

The system can be enhanced to be used for a variety of applications, such as deciding optimal workcell layouts. This would involve development of mathematical models/heuristics to derive optimal/better solutions which can be implemented in real time systems.

Bibliography

1. Angeles, J., "Iterative Kinematic Inversion of General Five-Axis Robot Manipulators", *International Journal of Robotics Research*, Vol. 4, No. 4, Winter, 1986.

2. Balestrino, A., De Maria, G., and Sciavicco, L., "An Adaptive Model Following Control for Robotic Manipulators", *Transactions of the ASME*, Vol. 105, September, 1983.

3. Boyse, J., W., and Gilchrist, J., E., "GMSolid: Interactive Modeling for Design and Analysis of Solids", *IEEE Computer Graphics and Applications*, March, 1982.

4. Brantmark, H., and Ramstrom, K., G., "ASEA Off-Line Programming System: A User Friendly Implement", *15th International Symposium on Industrial Robots*, 1985.

5. Brooks, R., A., "Planning collision free motions for pick-n-place operations", *The International Journal of Robotics Research* Vol. 2, No. 4, Winter, 1983.

6. Colson, J., C., and Perreira, N., D., "Kinematic Arrangements Used in Industrial Robots", *Conference Proceedings of the 13th International Symposium on Industrial Robots and Robots 7, Future Directions*, Vol. 2, 1983.

7. Derby, S., "Simulating Motion Elements of General-Purpose Robot Arms", *International Journal of Robotics Research*, Vol. 2, No. 1, Spring, 1983.

8. Derby, S., "GRASP: From Computer Aided Robot Design to Off-Line Programming", *Robotics Age*, Vol. 6, No. 2, Febuary, 1984.

9. Dillmann, R., "A Graphical Emulation System for Robot Design and Program Testing", *International Symposium on Industrial Robots*, 1985.

10. Fisher, E., L., Siedmann, A., and Nof, S., Y., "Robot System Analysis: Basic Concepts and Survey of Methods", *Proceedings of the Fall Industrial Engineering Conference*, 1982.

11. Flora, P., C., **International Robotics Industry Directory** Fourth Edition, Technical Database Corporation, Conroe, Texas.

12. Furusho, J., and Onishi, S., "An Efficient Approach for Solving the Inverse Kinematics of Manipulators", *15th International Symposium on Industrial Robots*, 1985.

13. Gilbert, E., G., and Johnson, D., W., "Distance Functions and Their Applications to Robot Path Planning in the Presence of Obstacles", *IEEE Journal of Robotics and Automation*, Vol. RA-1, No. 1, March, 1985.

14. Groover, M., P., Weiss, M., Nagel, R., N., and Ordey, N., G., **Industrial Robotics: Technology, Programming, and Applications**, McGraw-Hill Book Company, 1986.

15. Hitomi, K., Yoshimura, M., and Morimoto, M., "Optimal Scheduling for manufacturing systems with Industrial Robots", *Computer Integrated Manufacturing and Robots*, Winter conference of ASME, Lousiana, December, 1984.

16. Hockmam, J., F., and O'Connor, B., M., "Application of a Robot in a working cell", *Robotica*, Vol. 1, Part 3, 1984.

17. Hollerbach, J., M., and Sahar, G., "Wrist Partitioned, Inverse Kinematic Aceelerations and Manipulator Dynamics", *International Journal of Robotics Research*, Vol. 2, No. 4, Winter, 1983.

18. Howie, P., "Graphical Simulation for Off-Line Robot Programming" *Robotics Today*, Febuary, 1984.

19. Huang, B., and Milenkovic, V., "Kinematics of Minor Robot Linkage", *Conference on Robotics Research: The Next Five Years and Beyond*, 1984 .

20. Jun-shi, C., Qing-xian, X., "Computer Simulation and 2-Dimensional Display of Manipulator", *15th International Symposium on Industrial Robots*, 1985.

21. Katajamaki, M., "CAD/CAM in Robot Application Design and Simulation" *Autofact 6*, October, 1984.

22. Katajamaki, M., and Kanerva, J., "CAD/CAM - Revolutionizing Robot Applications Design", *Conference Proceedings of the 13th International Symposium on Industrial Robots and Robots 7*, 1983.

23. Kawabe, S., Ishikawa, H., Okano, A., and Matsuka, H., "Interactive Graphic Programming for Industrial Robots", *15th International Symposium of Industrial Robots*, 1985.

24. Kiel, M., J., and Myklebust, A., "Automatic 3-D Geometric modelling and animation of planar mechanisms", *Proceedings of the First International Symposium on Design and Synthesis* , July, 1984, Tokyo.

25. Kircanski, M., and Vukobratovic, M., "Computer-Aided Generation of Manipulator Kinematics Models in Symbolic Form" *15th International Symposium on Industrial Robots*, 1985.

26. Kretch, S., J., "Robotic Animation", *Proceedings of Annual Industrial Engineering Conference*, 1983.

27. Krogh, B., H., "A Generalized Potential Field Approach to Obstacle Avoidance Control", *Robotics Research: The Next Five Years and Beyond*, 1984.

28. Lee, C., S., G., "Robot Arm Kinematics, Dynamics, and Control" *Computer*, IEEE Computer Society, December, 1982.

29. Lee, C., S., G., and Ziegler, M., "A Geometric Approach in Solving the Inverse Kinematics of PUMA Robots", *Conference Proceedings of the 13th International Symposium on Industrial Robots and Robots 7, Future Directions*, Vol. 2, 1983.

30. Li, Chungwuu, "A General Robot Path Verification Simulation Sytem GRPVSS", *Unpublished M. S. Thesis*, Virginia Polytechnic Institute and State University, 1985.

31. Mahajan, R., and Leu, M., C., "Computer Graphic Simulation of Robot Kinematics and Dynamics" *Robots 8 Conference*, June 4-7, 1984.

32. Markov, M., D., Zamanov, V., B., and Nenchev, D., N., "Trajectory Modelling and Teaching of robots under Cartesian Path Control" *Proceedings of the 6th International Conference on Production Research*, 1981.

33. Matoba, H., Mohri, S., and Kogawa, T., "Robot Programming System Using Interactive Graphics", *15th International Symposium on Industrial Robots*, 1985.

34. Mertens, P., "An Industrial Palletizing System for Industrial Robots", *Unpublished M. S. Thesis*, Virginia Polytechnic Institute and State University, 1985.

35. Christiansen, H., and Stephenson, M., Brigham Young University, Provo, Utah, MOVIE.BYU, System Document, 1984.

36. Mutter, R., F., "Implementation Considerations in Process-Oriented Robotic Workcells", *International Conference on Robotics and Automation (IEEE)*, 1986.

37. Myers, D., R., and Gordon, D., F., "Kinematic Equations for Industrial manipulators", *The Industrial Robot*, September, 1982.

38. Nakamura, Y., Hanafusa, H., Yokokhji, Y., and Yoshikawa, T., "Efficient Computation and Kinematic Representation for Robot Manipulator Simulation", *15th International Symposium of Industrial Robots*, 1985.

39. Nof, S., Y., and Lechtman, H., "Robot Time and Motion System Provides Means of Evaluating Alternate Robot Work Methods" *International Journal of Production Research*, Vol. 22, No. 1, 1984.

40. O'Hara, R., P., and Gomberg, D., R., **Modern Programming Using Rexx**, Prentice-Hall, New Jersey, 1985.

41. Orin, D., E., and Tsai, Y., T., "A real-time computer architecture for inverse kinematics", *International Conference on Robotics and Automation (IEEE)*, 1986.

42. Orin, D., E., and Tsai, Y., T., "A Real-Time Computer Architecture for Inverse Kinematics", *International Conference on Robotics and Automation*, 1986.

43. Parent, M., and Laurgean, C., **Logic and Programming**, Vol. 5, Kogan Page, London, 1983.

44. Park, Y., B., Pegden, C., D., and Enscore, E., E., "A survey and evaluation of static flowshop scheduling heuristics", *International Journal of Production Research*, Vol. 22, No. 1, 1984.

45. Paul, R., P., **Robot Manipulators: Mathematics, Programming, and Control**, MIT Press, 1981.

46. Randhawa, S., U., McDowell, E., D., and Faruqu, S., "An Integer Programming Application to Solve Sequecer Mix Problems in Printed Circuit Board Production", *International Journal of Production Research*, Vol. 23, No. 3, May/June, 1985.

47. Red, E., "Minimum Distances for Robot Task Simulation", *Robotic*, Vol. 1, Part 4, 1983.

48. Red, W., E., and Truong-Cao, H., "Configuration Maps for Robot Path Planning in Two Dimensions", *Transactions of the ASME* Vol. 107, December, 1985.

49. Rembold, U., and Blume, C., "Programming Languages and Systems for Assembly Robots", *Computers in Mechanical Engineering*, January, 1984.

50. Sarin, S., C., "Mathematical Analysis of a Robotized Production Cell", *Canadian Journal of Operations Research and Information Processing* , Vol. 25, No. 1, 1987.

51. Sarin, S., C., and Wilhelm, W., E., "Prototype Models for Two-Dimensional Layout Design of Robot Systems", *IIE Transactions* , Vol. 16, No. 3, 1984.

52. Sluzek, A., "New Data Types for 4/5 Degree-of-Freedom Robot Manipulators", *15th International Symposium on Industrial Robots* , 1985.

53. Snyder, W., E., **Industrial Robots: Computer Interfacing and Control**, Prentice-Hall Inc., 1985.

54. Sol, E., J., Van Den Broek, J., A., Th., M., "Progress in CAD - tools for robot based flexible automation systems", *Computer Integrated Manufacturing and Robotics*, Winter Conference of ASME, Louisiana, December, 1984.

55. Stauffer, R., N., "Robotic System Simulation: Special Report" *Robotics Today*, June, 1984.

56. Sugiyama, T., Yaguchi, S., Yoshimura, H., and Ikemoto, M., "Robot Controller for Off-Line Programming System", *15th International Symposium on Industrial Robots*, 1985.

57. Tarvin, R., L., "An Off-Line Programming Approach", *Robotics Today*, Vol. 3, No. 2, Summer, 1981.

58. Thornhill, R., B., "Generic Programming of Robots - The CAM-I Robotics Software Project", *Robotics Research: The Next Five Years and Beyond*, 1984.

59. Tilove, R., B., Shapiro, V., and Pickett, M., S., "Modelling and Analysis of Robot Workcells in Roboteach", *Computer Integrated Manufacturing and Robots*, Winter conference of ASME, Louisiana, December, 1984.

60. Tilove, R., B., "Extending Solid Modeling Systems for Mechanism Design and Kinematic Simulation", *IEEE Computer Graphics and Applications*, June, 1983.

61. Toussaint, G., T., "Some Collision Avoidance Problems Between Spheres", *IEEE Cybernetics and Society*, 1985.

62. Waldron, K., J., Wang, S., and Bolin, S., J., "A Study of the Jacobian Matrix of Serial Manipulators", *Transactions of the ASME*, Vol. 107, June, 1985.

63. Wang, C., C., "Validating the robot simulation of rigid manipulator", *IEEE Conference on Automation* , 1987, pp. 228-232.

64. Weisbin, C., R., Barhen, J., and DeSaussure, G., "Strategy Planning by an Intelligent Machine", *Conference on Robotics Research: The Next Five Years and Beyond*, August 14-16, 1984.

65. Wiel, W., S., and Ramsthaler, D., L., "Robotic Application Software-Structure, Design, and Development", *Proceedings of the Fall Industrial Engineering Conferences*, 1982.

66. Wilhelm, W., E., and Sarin, S., C., "A structure for sequencing robot activities in machine tending applications", *International Journal of Production Research*, Vol. 23, No. 1, 1985.

67. Young, R., E., Yuan, Sue-Chea, and Sun, Y., "Development of a Robot-Centered Workstation", *Proceedings of the Fall Industrial Engineering Conference*, 1982.

68. Zapata, R., and Dauchez, P., "Co-ordinated Control of two Cooperative Manipulators: The Use of a Kinematic Model", *15th International Symposium of Industrial Robots*, 1985.

69. Zugel, J., M., "Prolog Implementation in Robot Kinematics", *Unpublished M. S. Thesis,* Virginia Polytechnic Institute and State University, 1985.

70. Duffy, J., and Crane, C., "A Displacement Analysis of the General Spatial Seven Link 7R Mechanism, *Mechanism and Machine Theory, Vol. 15, No. 3, 1980, pp. 153-169.*

71. Wang, C., C., "Necessary Vadition of Robot Simulation for Manipulator Design and Trajectory Planning", Japan-USA Symposium on Flexible Automation, 1986, pp. 159-164.

Appendix A. Appendix

The main program controls all other programs and data in the integrated system.

```
/*  Beginning of the main program */

call init 'SET'
idle_tra_time = 5
no_best_seq = 5

/* Get machine and buffer location file name */

say 'Enter the name of the file containing location of the machines and'
say 'buffers'
say 'QUIT to exit the program'

parse upper pull fn_mach ft_mach fm_mach .
if fn_mach = 'QUIT' then call getout
if fm_mach = '' then fm_mach = 'A'
call file_routines fn_mach ft_mach fm_mach

/* Get robot location file name */

say 'Enter the name of the file containing location of the robots'
say 'QUIT to exit the program'

parse upper pull fn_robo ft_robo fm_robo
if fn_robo = 'QUIT' then call getout
if fm_robo = '' then fm_robo = 'A'
call file_routines fn_robo ft_robo fm_robo

/* Get matrix data file name */

say 'Enter the name of the data file containing [C tcp] 0  matrices'
say 'QUIT to exit the program'

parse upper pull fn_matr ft_matr fm_matr
if fn_matr = 'QUIT' then call getout
if fm_matr = '' then fm_matr = 'A'
call file_routines fn_matr ft_matr fm_matr

/* Get HT and PT data file name */

say 'Enter the name of the file containing transport operation and'
say 'processing times'
say 'QUIT to exit the program'
```

```
parse upper pull fn_htpt ft_htpt fm_htpt
if fn_htpt = 'QUIT' then call getout
if fm_htpt = '' then fm_htpt = 'A'
call file_routines fn_htpt ft_htpt fm_htpt

/* Get Infeasibility matrix data file name */

say 'Enter the name of the data file containing Infeasibility matrix'
say 'QUIT to exit the program'

parse upper pull fn_infs ft_infs fm_infs
if fn_infs = 'QUIT' then call getout
if fm_infs = '' then fm_infs = 'A'
call file_routines fn_infs ft_infs fm_infs

/* Get Preferability matrix data file name */

say 'Enter the name of the data file containing Preferability matrix'
say 'QUIT to exit the program'

parse upper pull fn_pref ft_pref fm_pref
if fn_pref = 'QUIT' then call getout
if fm_pref = '' then fm_pref = 'A'
call file_routines fn_pref ft_pref fm_pref

/* Determine the number of machines and buffers */

'EXECIO 1 DISKR ' fn_mach ft_mach fm_mach ' 1'

lines_stacked = queued()
do i = 1 to lines_stacked
  pull no_of_objs .
end

no_of_machs = no_of_objs%2
no_of_buffs = no_of_objs - no_of_machs

if no_of_buffs ¬= no_of_machs+1 then do
  say 'The total number of machines+buffers should be an odd number'
  say 'The total number of machines+buffers is equal to ' no_of_objs
  say
  say 'Hit return to continue       '
  pull .
  call getout
end

/* Determine the number of robots */
```

```
'EXECIO 1 DISKR ' fn_robo ft_robo fm_robo ' 1'
lines_stacked = queued()
do i = 1 to lines_stacked
  pull no_of_robots .
end

if no_of_robots > 2 then do
  say 'The system is not setup to handle more than 2 robots '
  say 'Hit return to continue          '
  pull .
  call getout
end

/* Determine the number of jobs */

'LISTFILE ' fn_htpt ft_htpt fm_htpt '(STACK LABEL'
pull . . . . . no_of_jobs .

/* Confirm for validity of input problem determined */

do forever
  'CLEAR'
  say 'The workcell contains:'
  say '     1 -' no_of_jobs 'jobs'
  say '     2 -' no_of_machs 'machines'
  say '     3 -' no_of_buffs 'buffers'
  say '     4 -' no_of_robots 'robots'
  say '     5 - No. of best transport operation sequences to be'
  say '         determined =' no_best_seq
  say '     6 - Idle travel time per unit travel =' idle_tra_time
  say
  say 'Is all the input correct (Y/N)'
  pull respo .
  if respo ¬= 'Y' & respo ¬= 'N' then iterate
  else if respo = 'N' then do
    'CLEAR'
    say 'Hit return to continue '
    pull .
    call getout
  end
  else leave
end

'FI 07 DISK ' fn_mach ft_mach fm_mach '(LRECL 80 RECFM F'
'FI 08 DISK ' fn_robo ft_robo fm_robo '(LRECL 80 RECFM F'
'FI 09 DISK ' fn_matr ft_matr fm_matr '(LRECL 80 RECFM F'
'FI 10 DISK ' fn_htpt ft_htpt fm_htpt '(LRECL 80 RECFM F'

/* Load and Execute the Branch and Bound algorithm */
```

```
call run_bnb

call run_kinema no_best_seq
'START'
'LDISK MOVIE 192'
IF RC=0 THEN 'EXEC ANYC'
'DET 101'
exit

/* Save PF key settings and redefine to create number pad.          */

INIT:
arg option
if option='SET' then time_bg=time('R')
if option='CLEAR' then tim=time('E')
code = 0
if (option ¬= 'SET') & (option ¬= 'CLEAR') then code = 99
else do
  'EXECIO 24 CP (STRING Q PF'
  do i = 1 to 24
    pull pfk.i
    pfk.i = strip(pfk.i)
    'GLOBALV SELECT PF GET PF.'i
    if option = 'CLEAR' then do
      if word(pf.i,2)='UNDEFINED' then pf.i = word(pf.i,1)
      if word(pfk.i,3) = i then 'QUIETCP SET' pf.i
    end
    else do /* option = 'SET'*/
      key = word(pfk.i,3)
      if (key ¬= i) then 'GLOBALV SELECT PF SETL PF.'i pfk.i
      'QUIETCP SET PF'i 'IMMED' i
    end
  end
end
return code

FILE_ROUTINES:
arg fname ftype fmode
if fname = '' then call get_file
if ftype = '' then call get_file
if fmode = '' then fmode = 'A'
else do
  call file_exists? fname ftype fmode
  if result ¬= 0 then call get_file
end
'CLEAR'
call chk_file fname ftype fmode '80 F'
if result ¬= 0 then do
  'CLEAR'
  say 'File' fname ftype fmode 'has LRECL='lrecl 'and RECFM='recfm'.'
```

```
   say 'They should be LRECL=80 & RECFM=F'
   say 'Press return to continue.'
   parse pull
   call getout
end
return

/* Get the file name from the user*/

GET_FILE: Procedure Expose fname ftype fmode
do forever
   say 'What is the name of the input file? (FN FT FM) (QUIT to exit)'
   parse upper pull fname ftype fmode .
   if fname = 'QUIT' then call getout
   if fmode = '' then fmode = 'A'
   call quiet 'ESTATE' fname ftype fmode
   if result ¬= 0 then do
      'CLEAR';say 'File' fname ftype fmode 'not found.';end
   else leave
end
return

/* Checks to see if file chosen exists    */

FILE_EXISTS?: Procedure
arg fname ftype fmode
'ESTATE' fname ftype fmode
if rc ¬= 0 then do
   'CLEAR'
   say 'File' fname ftype fmode 'not found. Press return to continue.'
   parse pull
   return 8
end
return 0

/* Checks if file has lrecl of 80 recfm F */

CHK_FILE: Procedure Expose form lrecl l_s
parse upper arg fname ftype fmode .
'LISTFILE' fname ftype fmode '(STACK LABEL'
parse upper pull fname ftype fmode form lrecl recs blks dat tim label
if lrecl='80' & form='F' then retcode = 0
else retcode = 1
return retcode

/* Procedure to run the branch and bound program */

RUN_BNB: Procedure Expose idle_tra_time no_best_seq
queue format(idle_tra_time,4) format(no_best_seq,4)
'EXEC BNB'
```

```
'GLOBALV GET' bnb_rc
if bnb_rc¬ =0 then do
  say 'The branch and bound algorithm did not complete successfully'
  say 'The return code from branch and bound algorithm is ' bnb_rc
  call getout
end
say 'Optimal Solution Determined .....'
return

/* Procedure to run the kinematics program */

RUN_KINEMA: Procedure Expose no_of_jobs no_of_machs no_of_robots
        idle_tra_timetim

arg no_best_seq
no_of_t_operations = 2*no_of_jobs*no_of_machs

'EXEC LDISK MOVIE 192 501 M'

do i=1 to no_best_seq
  'FILEDEF 2 CLEAR'
  'LISTFILE SEQ'i 'DATA A (STACK LABEL'

  if rc=28 then do
    say 'The data corresponding to operation sequence' i 'not found'
    call getout
  end

  pull . . . . .nrecs .
  if nrecs¬ =no_of_t_operations then do
    say 'The sequence number' i 'has' nrecs 'transport operations.'
    say 'It is expected to have' no_of_t_operations. The program'
    sau 'cannot continue'
    call getout
  end

  'FI 02 DISK SEQ'i 'DATA A'
  'EXEC RUN_K'

  'GLOBALV GET' kinerc
  if kinerc¬ =0 then do
    say 'The program did not complete the sequence' i 'successfully'
    call getout
  end


  do forever
    'CLEAR'
    say 'Would you like to view the next best sequence (Y/N)'
    pull response .
```

```
      response = strip(response)
      if response¬ = 'Y' & response¬ = 'N' then iterate
      else leave
    end

    if response = 'N' then do
      do forever
        'CLEAR'
        say 'Would you like to store the animation data corresponding the'
        say 'current sequence (Y/N)
        pull resp .
        resp = strip(resp)
        if resp¬ = 'Y' & resp¬ = 'N' then iterate
        else leave
      end

      if resp = 'N' then call getout
      'CLEAR'
      if resp = 'Y' then do
        say 'Enter a 3 character ID which would be used when calling this'
        say 'sequence'
        say 'back for display'
        pull seqid .
        seqid = left(seqid,3)
        'EXEC WRITEO' i
        say 'The current transport operation sequence is stored in the file'
        say seqid'TRANS SEQUENCE'

        'GLOBALV GET' makesp
        'GLOBALV GET' r1
        'GLOBALV GET' r2

        say 'The makespan corresponding to this sequence is ' makesp
        say 'Robot 1 was used in ' r1 ' transport operations'
        say 'Robot 2 was used in ' r2 ' transport operations'
        say 'Total of ' format(tim,3,5) 'hours spent with IROGSS'
        say 'The pointer file is stored in the file ' seqid'TRANS SEQUENCE'
        call getout
      end
    end
  end
  return

  /* Abnormally quit the program */

  GETOUT: Procedure Expose tim
    call init 'CLEAR'
    say tim
    exit
  return
```

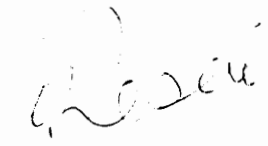
```
/* Subroutine to suppress output from CMS command */

QUIET:
  'SET CMSTYPE HT'; ''arg(1); hrc=rc; 'SET CMSTYPE RT'; return hrc
```

Vita

Chetan Desai was born in Rajkot, India on October 12, 1962 to Jyotsna and Jayant. He did his schooling in Rajkot from St. Mary's School. He joined the L. D. Engineering College at the Gujarat University, Ahmedabad, India in July 1980. He received his Bachelor of Engineering (Mechanical Engineering) degree in 1984.

From Fall of 1984 he started working towards his Masters degree in Industrial Engineering and Operations Research at Virginia Polytechnic Institute and State University. He was a graduate teaching assistant from January 1985 to June 1986 and graduate research assistant from July 1986 to June 1987. In July 1987 he joined the Agricultural Engineering Department at Virginia Polytechnic Institute and State University as a programmer/analyst. Since November 1987, he is a Research Associate with the Agricultural Engineering Department. His interests are in the fields of application software development, information systems, and manufacturing systems.



Chetan J. Desai