International Journal of Operations Research

International Journal of Operations Research Vol. 3, No. 2, 76-89 (2006)

An Intelligent Movement Sequence in Production Process for Real Time Robot Control in a Multi-Machine Manufacturing Cell

L. Hathout and S. Balakrishnan*

Department of Mechanical and Manufacturing Engineering, University of Manitoba, Winnipeg, Manitoba, Canada, R3T 2N2

Received November 2005; *Revised April 2006*; *Accepted July 2006*

Abstract—Material handling robots are routinely used for transferring parts in an interconnected multi-machine cell. The sequence of robot movements between machines is arrived at by the operator, either intuitively or by the use of some form of off-line analysis. It would be extremely beneficial if robots can be provided with additional intelligence to automatically sequence their control programs thereby optimizing production. This paper presents a decision support system that provides a real time control of a robot to meet the stated objective. The developed methodology permits the user to analyze a batch production process by considering a number of parameters that affect the throughput of parts and generates in real-time, the best sequence for a given production. This adds considerable intelligence to a standard robot controller. The parameters currently considered are: sequential versus non-sequential processing of parts; consideration of processing times on machines; the size of buffers in each machine; and the travel time of the robot. A batch production is considered to demonstrate the methodology. To the best of knowledge of the authors, robot controllers have never been provided with this kind of control intelligence. The system developed was tested on a four machine cell serviced by a single robot. The computational time required by the control software is minimal thereby facilitating real-time processing as well as control in a dynamic environment. Robot movements that prioritizes 'unloading of machines first' proved to be superior in comparison to other strategies. The number of buffers beyond a certain limit did not increase the throughput time of the batch. Non-sequential processing reduces throughput time. Loading sequence has a greater impact on sequential processing.

Keywords—Robot control, Decision support system, Flexible manufacturing cell

1. INTRODUCTION

A manufacturing cell is a cluster of machines or processes that are located in close proximity and dedicated to the manufacturing of a family of parts. Robots are being increasingly employed as material handling systems in many manufacturing cells. The scheduling problem at the cell level is characterised by a short lead-time, dynamically changing environments, the versatility of the machines, and the need for real-time decision making. Cell level scheduling needs to be: (i) responsive to changing environments; (ii) flexible in accommodating different scheduling needs; and (iii) intelligent for implementing decision support systems. With increases in computing speeds, complex control algorithms for real-time optimisation of manufacturing tasks are becoming a reality.

This paper focuses on the development of a methodology for sequencing of robot moves and sequencing of parts in a manufacturing cell by using a dynamic hierarchical decision making support system. The system is structured for a generic machine cell environment which can look at questions of buffer size, number of machines, machining times, robot move times, part processing requirements, part loading orders, and the impact of non-sequential versus sequential processing in order to improve the productivity.

The problem is one of dynamically sequencing both parts and robot moves without operator intervention. The purpose is to develop a better understanding of how the batch manufacturing time can be reduced by exploring generally occurring batch production situations. It is then extended to the development of functional control software that adds additional intelligence to the standard robot controller. The decision hierarchy has the capability to examine the effects of various control parameters and automatically restructure the sequence of loading/ unloading of machines for greater batch throughput.

2. LITERATURE REVIEW

Robot movement and part scheduling questions are known to be NP-hard, making them computationally intensive and challenging to solve. The optimal robot move sequence for a Flexible Manufacturing Cell (FMC) environment for manufacturing a batch of parts in the shortest time has been investigated (Sethi et al., 1992). However, the solution is limited to a flow line manufacturing system where parts have to visit all the three

* Corresponding author's email: balakri@cc.umanitoba.ca

stations considered and the solutions are based on cyclic scheduling. Extension of this work to parallel machines (Geismar et al., 2004) with multiple robots as well as robots with a dual-gripper has also been investigated (Sethi et al., 2004). A minimum part set (MPS) required to attain a desired production schedules has also been presented (Hall et al., 1998). The objective has been to minimize the time to produce one MPS in a cyclic production environment. A robotic cell with '*m*' machines was considered (Crama and Klundert, 1997) with an objective to find a sequence of robot moves for arriving at the minimum cycle time. A situation of no-wait constraint in which a part is moved immediately to the next machine as soon as the previous machine completes the work on the part has also been considered (Agentis, 2000). No-wait processing is typical in the steel industry and some electro-plating processes. An algorithm for improving the cycle time in a robotic cell with two machines for processing non-identical parts has also been analyzed (Aneja and Kamoun, 1999). An extension of this work (Agentis and Pacciarelli, 2000) compared the situation of identical part types in a no-wait robotic cell to a no-wait non-identical part types. The algorithm employed the travelling salesman problem' technique. Genetic algorithm based techniques have also been proposed to scheduling problems in robotic cells (Kenne and Gharbi, 2004) and (Chen et al., 2001) and presented a methodology for scheduling multiple part types for two machine cells and concluded that the problem-solving technique becomes cumbersome when the number of machines is more than two. A multiple-part type production problem on four machines by using a branch and bound technique has also been proposed (Chen et al., 2001). The technique is to first find an optimal robot cycle, and then choose a part sequence that produces the lowest cycle time. Some success was achieved when adapting the heuristics and algorithms to three machines. However, they concluded that further investigation was required to adapt the solution proposed to larger size problems.

Results from a simulation based approach for a dynamic FMC optimisation for two machines that employs different movement control logic did not reveal any new insights that could be generalized (Niemi and Davies, 1989). A two-machine cell, with buffers at each machine, for processing a batch of parts needing different processing times on each machine has been investigated (King et al., 1993). The objective is to determine the optimal sequence of robot moves to minimise the make-span of the batch. The study concluded that a branch and bound technique becomes ineffective as the number of parts increases past ten. The approach taken is to treat the problem as an open flow shop. This type of open flow-shop problem deals with non-pre-emptive shop scheduling that addresses routing of parts. It usually has the objective of minimising the make-span of a part. Moreover, a part's route is not given in advance although a predetermined processing time is known at each machine. Routes are determined by queues in front of machines and by the remaining process requirements. A linear time algorithm to find the optimal schedule for a two-machine, open-shop configuration with parts having different transportation times has also been proposed (Rebaine and Strusevich, 1999).

There has been a lot of work in modelling flexible manufacturing cells by using advanced simulation and modelling tools. The application of a Timed Place Petri-Net (TPPN) or a Coloured Petri Net (CPN) has proven to be most useful. A TPPN for solving resource allocation questions in a FMS job shop scenario using automated guided vehicles (AGV) employs a heuristic search method to determine the near optimal schedule of part processing (Cheng et al., 1994). Further work (Yalcin and Boucher 1999) used CPN to solve a FMC problem with alternative machining and alternative part sequencing and confirmed that Petri nets can be used as an effective modelling tool. An object oriented modelling approach to create open-ended simulation software for flexible manufacturing focuses on object definition and actions occurring during state transitions (Lin et al., 1994).

In these modelling studies, very little work to explore the capabilities of a tool to gain insight for improving robot move times has been demonstrated. The assumption in these papers of an infinite incoming buffer also leads to the objective of optimising the make span of a part rather than a batch of parts. The framework developed by Petri net modelling adapted to address FMC questions, promises to provide the type of information architecture for any modelling software. The papers reviewed dealing with Petri nets provide an example of how a net can be structured to handle time and place data related to objects moving through a model. These ideas proved to be valuable in developing the simulation software reported in this paper. This is equally true for observations made about implementation of object oriented coding for flexible, manufacturing related simulations.

In cases where the part loading sequence can be changed, most flexible production plans solve the loading problem first and then separately select a fixed cyclical robot move schedule based on the loading results (Moreno and Ding, 1993). Concurrently selecting and sequencing jobs in an FMS proves far more effective. The optimal assembly time for a printed circuit board (PCB) assembly where the sequence of operations is not critical has also been investigated (Kumar and Li, 1994). This work attempts to solve the problem of the robot movement as a "travelling salesman" problem. This scenario relates closely to the one presented here. However, the authors present an optimal solution for the manufacturing of just one type of PCB.

It was confirmed from the literature that concurrent loading and robot move decision-making is preferable in a dynamic FMC environment. In a specific PCB case, the implementation of non-sequential processing takes fuller advantage of the available machine capacity and results in a significant reduced throughput time. The approach used here in the development of the control program and simulation incorporates ideas from the structure proposed by prior research using Petri net modelling and object oriented programming. The methodology for

implementing the robot movement decision hierarchy also partially stems from these approaches. The hierarchy of decisions attempts to concurrently select parts for loading into the system and to sequence robot moves based on the current status of the overall system. Based on the review, the effect of variables like buffers, as well as the number of parts contained in them, loading part orders, and non-sequential processing merit further investigation. The decision support system developed as part of this work incorporates several ideas generated as a result of the literature review. The software and hardware is designed to answer issues related to optimising the batch size, number of parts and to identify sequencing aspects for real-time control.

3. LAYOUT OF EXPERIMENTAL SYSTEM

The flexible manufacturing system consists of a FMC cell and a control computer that runs custom designed software. A five degrees-of-freedom robot is used for material handling. There are four processing or machining stations, each with gravity feed buffer which can hold up to five parts as shown in Figure 1. A machine, and the buffer behind it, is occasionally called a station when the location of both is being referenced. The processing or machining times in the cell are simulated through timers in a programmable logic controller (PLC) which are triggered through sensors located on each of the four buffers. There is one input buffer for parts entering the cell and a drop-off location for the finished parts. The robot controller has the ability to store up to ninety-nine unique programs that are controlled by a master program. This master program responds to commands from the input/output (I/O) board located in the external personal computer. The board will be controlled by the decisions from the support system software.

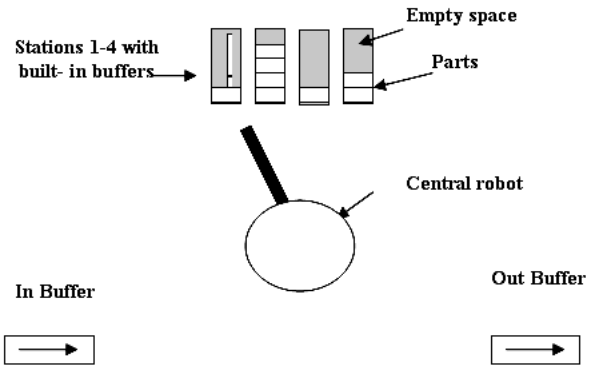

Figure 1. Cell layout.

The I/O board reads the status of the timers in the PLC and the status of the sensors at the processing stations. The latter indicate whether each station is occupied. Based on the information read by the I/O interface, the software determines what the robot's next move should be and sets the corresponding output bit on the I/O board. The master program for the robot controller reads the status of the input bit, which controls the use of one of the many subprograms. The subprograms correspond to different tasks that the robot is required to perform. A single program written in C++ language controls the FMC both in an on-line/real time environment as well as in an off-line/simulation mode. The program was written to test the effect of different variables on the total throughput. The program makes decisions as to which task to perform next based on input criteria, the decision hierarchy, FMC variables and the existing feedback from the FMC. The custom program developed has the ability to read user input batch data for production through the FMC and then simulate or run the process. User can step through the simulation one step at a time and run all possible sequences for a batch. At the end of each program, statistics are displayed that indicate each station's utilization; it's occupied and free times. The program has the option to save data in a user-defined file at the end of each run. When multiple tests are conducted, data is filed automatically in a pre-specified file. All variables related to the given test are also included in this file and can be selected for real-time robot implementation.

3.1 Structure of cell program

The cell program was designed to provide the user with the option of choosing two types of production cycles. The first type prioritises loading parts into the system; and the second prioritises unloading. One of the following actions is taken during each cycle: load, shift part from one station to the next, unload, or move the robot and wait at the next station needing unloading or shift a part to its next operation. The order in which these choices are presented depends on whether *unload* or *load* priority is selected. *Unload* priority first looks at the option of unloading a part. If this is not possible, then it considers shifting a part from one station to the next. If shifting is not an option, then it may load a part. If all else is impossible the final choice is to move the robot to a station that has almost finished machining and waiting. The *load* priority first looks at the option of loading a part, then shifting a part from one station to the next, followed by unloading, and lastly the moving and waiting command is considered.

During each cycle, the program's clock is incremented by the total time it takes for the robot to move. When the move and wait action is called, the clock is incremented by the time it takes the robot to move plus the time remaining for processing the part at the station to which the robot moves. As the main clock is incremented, the timers (representing the appropriate machining times) corresponding to the stations processing parts decrements. There are also two queues, the 'done station' queue and the 'almost done station' or 'working station' queue. These queues keep track of which part is done first and which parts are to be done next. In the case of a station finishing processing when the clock is incremented, the station number gets added to the 'done station queue', and that station number is removed from the working station queue. For example, if the 'working station queue' indicates that

machine 3 is done machining first, followed by stations 1 and 4, the working queue is 314. If machine 2 is holding a finished part, the 'done' queue is 2. If the timer is incremented by an amount which is equal or more than the time required to finishing machining the part at machine 3, then the done queue is 23 and the working queue becomes 14.

Each part moving through the system is entered in a matrix as shown in Figure 2. An individual part has an identification number and nine variables of information defined when the part enters the system. The first five variables are Boolean variables that identify the stations that a particular part needs to visit. These variables could also be turned into integer variables and used to store unique machining times for each station.

As the part visits each station for processing, the integer value representing the status changes, and binary numbers 0 and 1 are used to represent the state. The sixth variable monitors how many stations that the robot still has to visit by reading the binary number created by the first five Boolean variables. For example, if the first five variables describing a part are [0, 1, 1, 0, 1], then the sixth variable is 2 + 4 + 16 = 22. If the third operation is complete, the sixth variable is 18. The seventh variable indicates the part number, which would be used if the part needs to be unloaded to a part-specific buffer. The eighth variable gives the current station at which the part is located. The ninth variable indicates whether the part is in transition (i.e. held by the robot), in a buffer, being worked on, waiting for unloading or is out of the system. Each of these states has a number affiliated to it. There is also a buffer matrix that keeps track of the part queues at each station. Figure 3 provides a graphical representation of these matrices. Of the four operations that can be performed in each cycle, the load operation is the most complicated. The load operation triggers a subroutine that determines whether there is any part that can enter the system. The primary requirement for loading a part into the cell is the availability of space to load the part onto a required machine, or into a limited capacity buffer. The program has been developed to accommodate two types of part processing, sequential and non-sequential processing. The loading rules for both these processes are the same. However, when parts are loaded sequentially, the option of which part to choose is more restricted. The following paragraph explains the hierarchy of the part selection priorities.

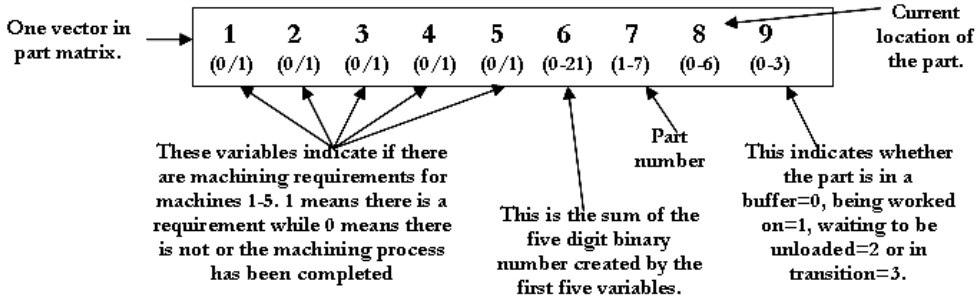

Figure 2. Job matrix.

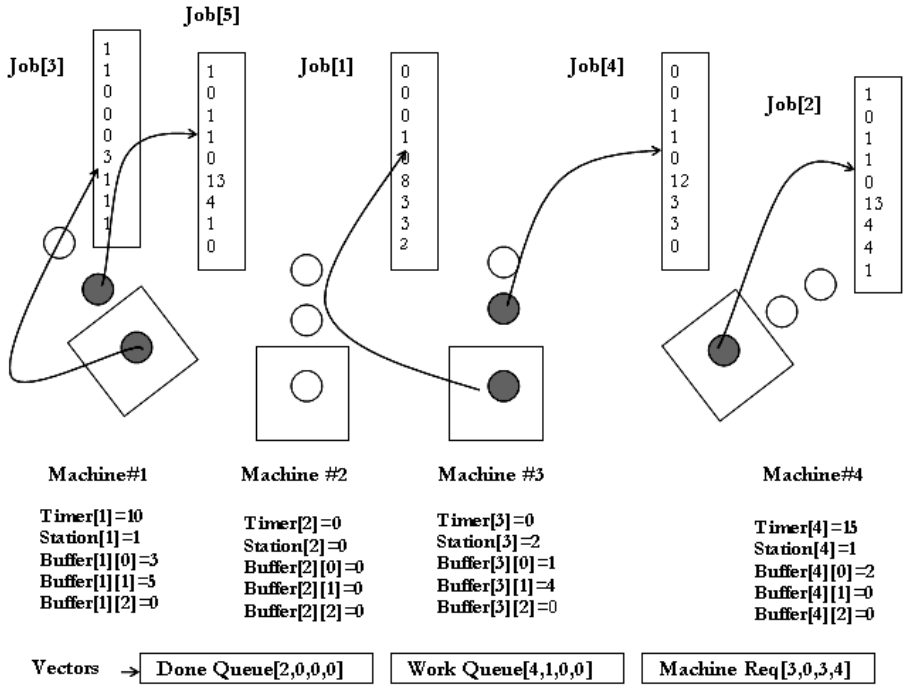

Figure 3. Matrix example.

In the sequential case, a part can be loaded only if the first machine in the series of machines that the part needs to go to is free. Given that there may be more than one part that is a suitable candidate for loading, there is an order of priority for selecting a part. The first priority is given to the first part in the order of feasible parts which needs the currently least occupied station (e.g. a station having the fewest waiting parts). If there is more than one station with the fewest parts (e.g. two stations that are occupied by one part while the other stations have two or more), priority goes to the station that is most in demand by the parts in the current batch. If no clear priority is found, then the same search is performed for stations that have one more occupancy.

The process of part selection for the non-sequential case goes through the same order of priorities as in the previously defined sequential process. However, in this case, a part is a candidate providing one of its machining requirements is met. There are two additional criteria in the case of non-sequential processing. The part to be loaded, after having met all the other requirements, must go through a conflict check. The conflict check is a subroutine that ensures that, if a part is loaded into the system, it will not cause a blockage to the further processing of parts. For example, consider a situation when two parts of the same type need to visit two machines, and both machines have no buffers. If the parts are loaded consecutively on machines 1 and 2, there is no way for either part to proceed to the other station. This scenario is defined as a conflict situation, which needs to be avoided or resolved.

The other unique feature of non-sequential loading is that the cell can be loaded to capacity. This feature means that all the free buffer space in the cell can be loaded in such a way that no process is in conflict. This is not desirable because it tends to reduce the number of options available for the movement of parts and it usually leaves only one free buffer space for manoeuvring. Consequently, the total batch time is increased greatly. This situation does not arise in sequential processing. However, capping current machining requirements of parts in the system by making it a part loading condition solves the problem of insufficient manoeuvring room. The number of parts, requiring a particular machine, permitted to enter into the cell, is fixed as the number of buffers at that machining station plus two. For example: if there are two buffers at each of three stations, then the current requirements of the parts in the cell cannot be more than four requests for processing on machine 1, four requests on machine 2, and four requests on machine 3. The choice of setting the cap at the number of buffers plus two was determined based on numerical experimentation performed by using an exhaustive search algorithm. Values greater or less than two proved to be unsatisfactory for reasons given below. The first arrangement, where no cap was imposed, allowed the cell to be loaded to its maximum capacity. This arrangement made it possible to load the cell to the point where any additional part would completely block the system. It made shifting parts difficult and led to the conclusion that loading the system to maximum capacity resulted in buffers being used more as storage locations than temporary transfer points. The chosen cap of the number of buffers plus two provided the best compromise between overloading the system and switching to an 'unload always' rule, regardless of the number of buffers. In conclusion, to benefit from the buffers, the time spent by parts in the buffers should be minimal.

The shift load, and unload routines are the other places in the program in which a part selection hierarchy is used. In both cases a decision needs to be made as to which of '*x*' number of finished parts is to be unloaded or shifted first. Here, again, priority is given to the parts at a station that is in high demand. If demand among stations with finished parts is equal, then the 'done parts queue' is referred to, in order to see which part will be finished first. Based on these criteria, a decision is made as to which part to unload next. In the case of shifting a part from station to station, the same two criteria are examined. First, the part being shifted can be shifted only to a station that has free buffer space. Second, a conflict check is performed in the case of non-sequential processing.

3.2 Hierarchy of decisions

Having stated the basic criteria for making decisions regarding robot moves and part selection rules, this section provide a clear understanding of how these decisions affect the batch processing time. This section also demonstrates the differences between the loading and unloading rules, as well as the non-sequential and sequential loading rules. The differences are demonstrated in four step-by-step analyses of state changes in one particular robot loading and sequencing example. The example considered consists of three part types A, B, and C to be produced in the cell. The data for batch considered (Batch #1) is shown in Table 1.

Table 1. Data for batch #1

Part type	Number to be produced	Station 1	Station 2	Station 3	Station 4
		Processing time=15 s	Processing time= 15 s	Processing time=15 s	Processing time=15 s
A	2	X	X		
B	2		X	X	
C	2			X	X

The robot's move times are each 5 seconds between stations. The stations that the part has to visit are indicated by an 'X'. A step by step account of the processing of this batch with an objective to highlight the difference between the robot's movements chosen when an unload priority is used versus those when the load priority is used to process parts sequentially is given next. For example, part B needs to visit station 2 before proceeding to station 3. Tables 2a and 2b present the time history of production generated from an analysis for sequential loading and unloading. The 'time' column gives the time that the action in the 'robot action' column was finished. The 'other actions' column indicates if any other action occurred at that time, such as a part has been finished machining or has moved from one buffer space to another. The 'machine req' (machining requirements) column has four numbers. A given number represents the number of processing requirements at each station that is needed by all the parts currently in the system. This information is used to decide which parts enter the cell next. Values in 'machine req' only change when a part no longer needs to be machined at a station and it is no longer occupying that station. The last column shows the done queue, which indicates the order in which the stations have finished their machining jobs. The following notations are used to describe the station changes. "*L*" indicates that the robot has moved to the input buffer, picked up a new part and deposited it at a station.

"U" indicates that the robot has picked up a part and dropped it off at the systems out buffer. "S" indicates that the robot has picked-up a part and dropped it off at a station. Parts are identified by a part letter followed by a number to indicate which specific one of a part type is being referenced (e.g. A1 or A2). A station, S, and its affiliated buffers, are designated by the notation S*xy*. The '*y*' indicates what buffer position the part has taken at station '*x*'. Moreover $y = 0$ implies that the part at a station is currently being machined or that machining at station '*x*' has completed processing this part. If there is no letter in front of a part name but a station is identified after the part, then the part has shifted up one place in the buffer space. For example if a part B number 3 (B3) was loaded on station 3 buffer 2 (S32) and the part at station 3 buffer 0 (S30) was unloaded, then part B3 would move to S31. 'Done' means the part has completed processing at a station.

The two examples differ only in the priority rule used for the robot's moves. Sequential loading always loads parts if there is place to load a part. If not, the algorithm considers part shifts and, if this is also not possible, then it considers unloading parts. In sequential unloading, the hierarchy of robot actions is the reverse — first the possibility of unloading a part is considered, then shifting a part, and then loading parts. These two cases produce sequences of events that are different even though the total batch time is the same. The robot's move time forms the bottleneck in these examples. Examining the 'done queue' list shows this. If it tends to have one or more parts waiting, it is because the robot cannot keep up with the demand for its action.

Once the first level of decision making is determined (whether to load, shift unload or wait), the second level of decision making comes into play. Here the program decides which part is to be loaded, shifted or unloaded. The order in which parts are loaded is determined by the batch demand for a machine and whether space is available in the FMC for a new part. To explain the term batch demand for a machine, consider 50 unprocessed parts of which 30 require machine 3, 20 require machine 4 and 10 require machine 2. Given this batch, the control program first checks if there is a part which can be loaded on machine 3; if not, it checks if there is a part which needs machine 4 and then machine 2. In this example, the machines most in demand are 3 and 4. Hence, they are loaded first by parts that require them, in this case part B and part C.

One important decision that has to be made in this model is whether or not to continue to adhere to a set of rules or to make special conditions for unique cases. For example consider a situation wherein the robot has just loaded the buffer of one station with a part and the part being machined at that station is finished. In this case it may be beneficial to take advantage of the robot's position to shift that part rather than addressing the next part in the queue according to the hierarchy set out. This is illustrated in bold lettering (at time 35) in the load example of Table 2a. Through experimentation, it is determined that the benefit of implementing such a policy is limited to cases where the robot's move time is the bottleneck. Otherwise imposing such a policy is no longer beneficial; rather, it increases the batch throughput time. The table shown is automatically generated by the control software. Data similar to the one shown in Table 2a and 2b were also generated for non-sequential load and unload cases.

The non-sequential unload showed a unique case wherein in which no unload, shift or load operations are possible. Hence, the robot moves to the station that would first be done machining a part and then the robot waits for the next required action. This situation did occur when no more parts are to be machined at a station nor are there any more parts to be loaded. The control methodology then looks ahead in the working queue to pick a station that has a completed part and move the robot to that location. This kind of "look ahead" feature reduces the robot's movement time by having it take advantage of "free time".

In the sequential load examples given in Table 2a, the order in which parts are loaded is B C A B C A and all the parts are loaded immediately, meaning the first six robot actions load parts into the cell. In the non-sequential load example, the load order is 'ABACC' and only after two shifts occur is the final part B loaded. The reason that the order is different is because two parts can be loaded immediately on station 2, namely parts A and B. However, if there is a restriction requiring the parts to be processed in order, then part A cannot start on station 2. It is also

Table 2a. Time history of sequential loading

Time (s)	Robot action	Other Actions	Machine req	Done queue
5	L B1-S20		0110	0000
15	L C1-S30		0121	0000
20		B1 done	0121	2000
25	L A1-S10		1221	2000
30		C1 done	1221	2300
35	LB2-S21		1331	2300
40		A1 done	1331	2310
45	L C2-S31		1342	2310
55	L A2-S11		2442	2310
60		C2-S30	2432	2310
65	S C1-C40		2432	2100
70		B2-S20	2332	2100
75	S B2-S31	C2 done	2332	1300
75		B1-S30	2322	1300
80	S C2-S41	C1 done	2322	1400
85		B2 done	2322	1420
85		A2 S10	1322	4200
90	S A1-S21	B1 done	1322	4230
90		A1-S20	1222	4230
95	S B2-S31		1222	4300
100		C2 S40	1221	3000
105	U C1	A2 done	1221	3100
105		A1 done	1221	3120
115	S A2-S21	C2 done	0221	3240
120		B2-S30	0211	3240
125	U B1		0211	2400
130		A2-S20	0111	2400
135	U A1	B2 done	0111	4300
145	U C2	A2 done	0110	3200
155	U B2		0100	2000
165	C A2		0000	0000

Table 2b. Time history of sequential unloading

Time (s)	Robot action	Other Actions	Machine req	Done queue
5	L B1-S20		0110	0000
15	L C1-S30		0121	0000
20		B1 done	0121	2000
25	L A1-S10		1221	0000
30		C1 done	1221	2300
35	S B1-S31		1121	3000
40		A1 done	1121	3100
40		B1-S30	1121	3100
45	S C1-S40		1111	1000
55	S A1-S20	B1 done	0111	3000
60		C1 done	0111	3400
65	U B1		0101	4000
75		A1 done	0101	4200
75	UC1		0100	2000
80	UA1		0000	0000
90	L B1-S20		0110	0000
95	L C1-S30		0121	0000
100		B1 done	0121	2000
105	L A1-S10		1221	0000
110		C1 done	1221	2300
115	S B1-S31		1121	3000
120		A1 done	1121	3100
120		B1-S30	1121	3100
125	S C1-S40		1111	1000
130	S A1-S20	B1 done	0111	3000
135		C1 done	0111	3400
145	U B1		0101	4000
155		A1 done	0101	4200
160	UC1		0100	2000
165	UA1		0000	0000

important to observe that the non-sequential loading immediately load all six parts because, in this case, if the last part B is added then further movement is blocked. This situation does not occur in sequential loading. In the sequential processing the bottleneck machine regulates the cell's capacity. In the non-sequential loading, there is a possibility to stifle movement in the FMC by overloading it with parts. Hence, an additional condition is added to the load rule that restricts the number of parts that need a specific machine to two.

This section has provided an overview of the software developed to handle dynamic FMC control problems on-line or off-line for simulation purposes. The key programming decisions have been outlined as well as the overall system's structure. The next section describes various aspects related to the implementation of the proposed cell control strategy.

4. EXPERIMENT AND ANALYSIS

Prior to detail analysis, the functionality of the on-line FMC control system was verified through several experimental runs. All robot movements corresponded appropriately to the programmed decision structures. Feedback loops from the system also functioned appropriately. The throughput from the simulation when compared to the actual experimental run showed a discrepancy of ±5%. This variation is a result of the slight time delay between when the external interface initiates a request for a subprogram and the actual execution by the robot controller, a typical situation found in robot controllers.

The next test compares the results of individually processing part types (all part As then all part Bs and so on) versus concurrently processing parts. The significance of concurrent processing over non-concurrent processing seems to be influenced greatly by the total order and the process that each part type has to undergo. Hence, the benefit of concurrent processing is composition dependent. The following examples corroborate this assertion. The load priority is used in all cases. If each part type in the batch shown in Table 3 (Batch #2) is processed individually, the total processing time is 3180 seconds. In comparison, the total throughput for the same forty parts produced concurrently is 1085 seconds.

This batch is an extreme case in which each part has only one process requirement. Producing all of one part type at a time resulted in three idle machines. The batch throughput time becomes the process time of each part on a specified station plus the time for the robot to load, and unload the part. Hence, the concurrent processing of the parts proves to be very beneficial and reduced the

throughput time by almost a third.

Considering another batch (Batch #3) whose data is shown in Table 4, the total processing time is 4580 seconds when the parts are processed individually. The total throughput time for the same forty parts produced concurrently is 4395 seconds. This case is the opposite of previous case, namely, although they may still be unique, they need to visit the same three stations. Concurrent processing of parts has very little effect in this case. The minor improvement shown is due to the fact that concurrently processing the four part types can be done without interruption caused by the completion of all part A's and starting the next batch of part B's.

The next analysis considered a more realistic model of the types of parts that would be processed in a FMS. In this case, parts need at least a couple of the machining processes in the FMC. The batch data considered (Batch #4) is shown in Table 5.

If each part type shown above is processed individually, the total processing time is 4520 seconds. The total throughput for the same forty parts produced concurrently is 4310 seconds. Here, concurrently processing parts results in a 5% improvement. The results from an analysis when the batch processing requirements for the data shown in Table 5 were reduced by 50% (case 2) and 75% (case 3), respectively are shown in Table 6.

There are two trends related to the 'total machining requirements' of a batch. The first shows that, as the total batch processing time is reduced, the closer the total throughput to the total machining requirements. As the processing is completed more quickly, the robot is in greater demand compared to when the robot's moves occur while all the machines are busy machining parts. Reducing the machining time leaves the stations idle while the robot is moving. Essentially the robot becomes the bottleneck. This observation is true for both the concurrent processing and for processing by part. The other observable trend is the influence of the processing time on the improvements afforded by concurrent processing. Column 4 of the results shown above also indicates a clear reduction of benefits brought through concurrent processing as the ratio of the processing to robot move times reduces.

What these analyses show is that the ability to concurrently process parts with different processing requirements in the same FMC reduces the throughput time of the batch in comparison to processing all one part type followed by another part type. Hence, the ability to dynamically respond to different part processing requests

Table 3. Data for batch #2

Part type	Number to be produced	Station 1 Processing time=40 s	Station 2 Processing time= 90 s	Station 3 Processing time=70 s	Station 4 Processing time=60 s
A	10	X			
B	10		X		
C	10			X	
D	10				X

Table 4. Data for batch #3

Part type	Number to be produced	Station 1 Processing time=40 s	Station 2 Processing time= 90 s	Station 3 Processing time=70 s	Station 4 Processing time=60 s
A	10	X	X	X	
B	10	X	X	X	
C	10	X	X	X	
D	10	X	X	X	

Table 5. Data for batch #4

Part type	Number to be produced	Station 1 Processing time=40 s	Station 2 Processing time= 90 s	Station 3 Processing time=70 s	Station 4 Processing time=60 s
A	10	X	X		
B	10		X	X	
C	10	X	X	X	
D	10		X	X	X

Table 6. Results from analysis for batch #4

Case	Total machining time required (s)	Total throughput time (s) by part	Total Throughput time(s) concurrent	%improvement	% decrease for part throughput	% decrease for concurrent throughput
1	7100	4520	4310	4.65	------	------
2	3550	2630	2530	3.80	41.8	41.3
3	1740	1669	1638	1.86	36.5	35.3

Table 7. Data for batch #5

Part type	Number to be produced	Station 1	Station 2	Station 3	Station 4
		Processing time=5 to 100 s	Processing time=5 to 100 s	Processing time=5 to 100 s	Processing time=5 to 100 s
A	5	X	X		
B	5		X	X	
C	5			X	X

and FMC conditions takes fuller advantage of the capacity of a FMC over a fixed robot move cycle for processing one part type at a time. It can also be argued that developing fixed cycles for a combination of part types would produce a superior batch throughput to that afforded by separately processing single part types. However, the flexibility that this dynamic response allows is still likely to produce better results. Proof of this assertion is beyond the scope of this work.

The next set of analysis examined the effect of increasing the machining time while the robot's move time stays at 5 seconds for all inter-cell movements. The data considered is shown in Table 7 (Batch #5). The objective is to observe the effect of changing the ratio of the machining to robot move times on the total production time under the following, individually applied conditions. The robot prioritizes unloading parts; parts are processed sequentially; or the robot prioritizes loading parts; parts are processed sequentially; or the robot prioritizes unloading parts; parts are processed non-sequentially; or the robot prioritizes loading parts; parts are processed sequentially. In all four cases the 15 parts identified in the data shown below are to be produced. The processing time for each test was incremented by 5 seconds on each test. The summary of the results are shown in Figure 4, as well as in Table 7. The results shown in Figure 4 as well as Table 8 indicate that, in both the sequential and non-sequential cases, prioritizing the unload results in a lower total batch time than prioritizing the loading when the ratio of the machining and robot move times is about six or more. This test also indicates that non-sequential processing of parts is faster than sequential process. When the average machining time is less than about six times the robot's move time, there seems to be no observable pattern. The observations made about this particular experiment are generally applicable. For all variations in batch sizes and part requirements, the unload priority gives a lower total time than when the load is prioritized. This observation holds true regardless of the number of buffers or parts. What cannot be generalized is at what ratio of the average machining time to robot move time does the system become stable (i.e. achieves a constant relationship). Thus far, it has been observed that, for the parameters used in these simulation studies, all batch simulations become stable under a machining to robot move time ratio of around 10: 1.

The next set of tests evaluated the effect of buffer sizes for both non-sequential and sequential unloading. The same batch order employed in the previous analysis was used and, as in the previous cases, 15 parts were manufactured with the machining time incremented progressively by 5 seconds from 5 to 100 seconds. The results from this analysis are shown in Figures 5a and 5b. From Figure 5a and 5b, it would appear that no significant improvement is produced by increasing the buffer size to more than two in the non-sequential case and more than four in the sequential case. The sequential example shows that, having more buffers in the FMC, ensures that stations past the bottleneck have less chance of remaining idle. In both examples, since there are only 15 parts in the batch, this inevitably decreases the need for buffers. The tests show that the point where more buffers become beneficial is related directly to the average machining time. For example, it is only after the average machining time is greater then 65 seconds that there is a benefit to having four rather than three buffers in the non-sequential case. In the analogous sequential case there is a benefit after 60 seconds. In both cases, as the ratio of robot move time to machining time increases, the more the robot sits idle. If extra buffers are available, the robot can take advantage of this idle time to fill the buffers with parts. This ensures that when a part is removed from a machine there is always a new part immediately available for machining. By decreasing machine idle time, the total batch throughput is also decreased.

Table 8. Machining time data

Machining time(s)	Non-sequential		Sequential	
	Load	Unload	load	unload
5	405	425	425	430
10	415	445	425	445
15	415	440	420	420
20	410	450	445	410
25	425	455	455	440
30	460	465	495	465
35	475	455	520	455
40	495	480	565	530
45	530	510	610	605
50	570	565	655	580
55	615	600	710	610
60	660	645	765	710
65	705	695	820	775
70	755	745	875	825
75	805	795	930	875
80	855	845	990	930
85	905	895	1050	985
90	955	945	1110	1040
95	1005	995	1170	1095
100	1055	1045	1230	1150

The next experiment examined the difference between non-sequential and sequential part processing. Four different batch orders were tested. The composition of each order is intended to examine the effect of having different bottleneck machines in the cell. All the orders

demand equal machining times and each station has two buffers. In all tests, 40 parts were processed (10 of each part type A, B, C, D). Moreover, each part had the same total machining time requirements for all cases, although these requirements were not necessarily on the same machines for each case. In the four different cases considered, each specific machine requires 90 seconds to machine a part. However, the machines may have different total machining demands (meaning machine 1 processes 20 parts while machine 2 processes 30 parts). The details of the batch requirements are given (Batch #6 through #9) in Table 9.

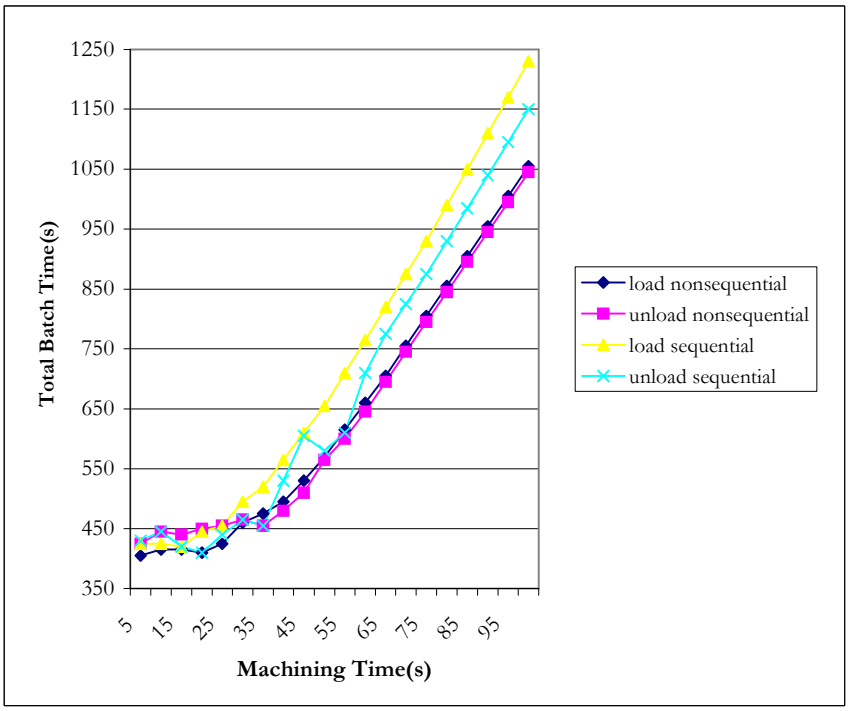

Figure 4. Influence of machining times on load and unload rules.

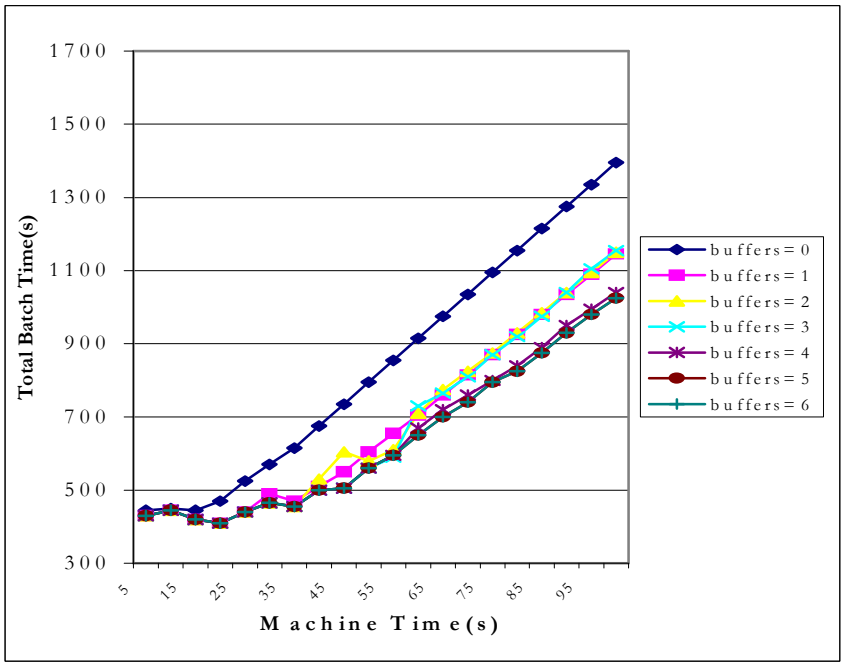

Figure 5a. Influence of buffers on sequential processing.

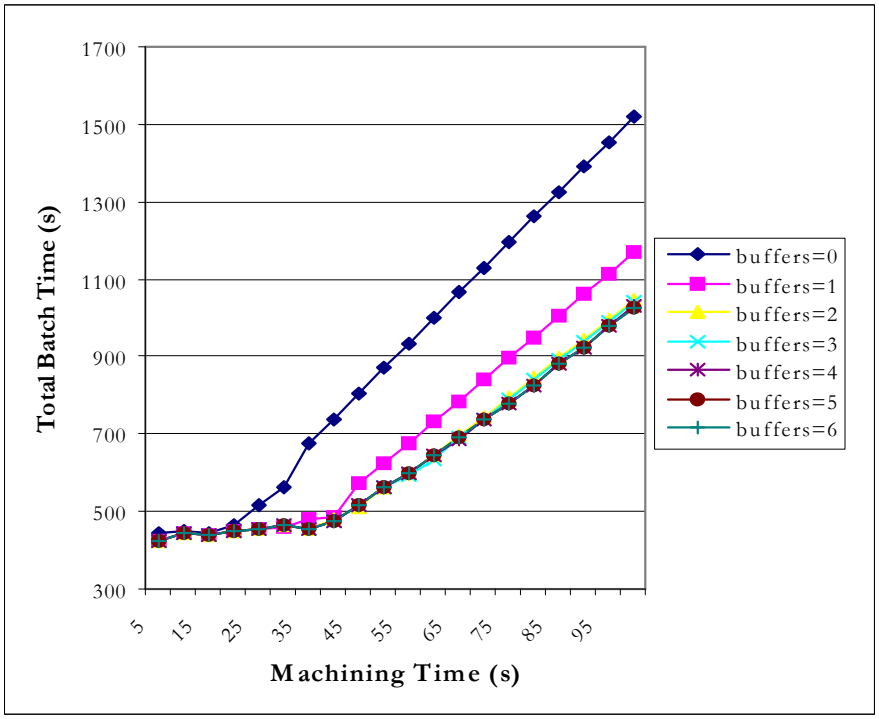

Figure 5b. Influence of buffers on non-sequential processing.

Table 9. Batch order data

Batch#6 Part type	Number to be produced	Station 1 Processing time=90 s	Station 2 Processing time= 90 s	Station 3 Processing time=90 s	Station 4 Processing time=90 s
A	10	X	X		
B	10	X		X	
C	10			X	X
D	10	X	X		X
Batch#7 Part type	**Number to be produced**	**Station 1 Processing time=90 s**	**Station 2 Processing time= 90 s**	**Station 3 Processing time=90 s**	**Station 4 Processing time=90 s**
A	10	X	X		
B	10		X	X	
C	10			X	X
D	10	X	X		X
Batch#8 Part type	**Number to be produced**	**Station 1 Processing time=90 s**	**Station 2 Processing time= 90 s**	**Station 3 Processing time=90 s**	**Station 4 Processing time=90 s**
A	10	X	X		
B	10		X	X	
C	10			X	X
D	10	X		X	X
Batch#9 Part type	**Number to be produced**	**Station 1 Processing time=90 s**	**Station 2 Processing time= 90 s**	**Station 3 Processing time=90 s**	**Station 4 Processing time=90 s**
A	10	X	X		
B	10		X		X
C	10			X	X
D	10	X		X	X

The results for the four batch orders are summarized in Table 10. What is clear from this table is that non-sequential processing is always faster than sequential processing. However, what needs to be explained is the noticeable variation in the throughput times given that all the batch cases have the same total machining time. Through experimentation, these variations have been attributed to the location of the bottleneck machine. Indeed, the batch requirements were chosen specifically to demonstrate this effect. In all cases, three of the machines are required to machine 20 parts while the fourth is required to machine 30 parts. Clearly the machine with the additional load corresponds to the bottleneck. The batch orders 6 through 9 each have a different machine forming the bottleneck.

From the results shown in Table 10, it would seem that non-sequential processing is slightly faster if the bottleneck machines are used first in the process. However, given that the processing is non-sequential, the machine order should, intuitively, not influence the results. This first observation indeed turns out to be incorrect. The reason for the variation in the batch times will be explored further in the next experiment, which links the part loading order to this throughput variation. In regard to the throughput times of sequential processing, this example indicates that the location of the bottleneck machine should hardly affect the batch's throughput time. Through further experimentation (in the next section), it has been found that the loading order of the parts, as well as the location of the bottleneck machine, does affect the total throughput time. The next experiment will demonstrate that, in the sequential case, when all the different orders parts can be introduced into the system, the throughput times are greater, on average, when the bottleneck machine is at the front of the queue of machines. Moreover, the throughput time decreases as the bottleneck machine progresses to the end of this queue.

Table 10. Summary

Batch Order	Non-sequential (in seconds)	Sequential (in seconds)
1	2710	2900
2	2710	3030
3	2790	3035
4	2790	3010

The last experiment will address the issue of the loading sequence of parts. As it stands, the program is capable of searching the parts that remain to be processed in order to find a suitable fit with the current state of the FMC. For example, if three machines are busy and the fourth machine is idle, the program searches the parts remaining to be processed to see if any part requires machining at the free station. Consequently, the order of parts introduced to the system is superseded by the requirements of the cell. However, in most cases, several parts may need the machines that are currently free or have free buffer space. This situation is where the 'search order' that the program uses for checking the remaining part types for loading suitability becomes critical. Until now, all the parts were processed in the same order that they entered the system (i.e. A, B, C, and D). Indeed, there is no obvious reason for selecting one part over another, or pairing parts in a specific manner.

This test will examine how alternative loading sequences of parts affects the total throughput of a batch. Using the same scenarios presented, twenty-four individual but different experiments were conducted. Each experiment involved a different loading sequence. The twenty-four experiments were conducted by using both the sequential and non-sequential rules. The results are presented in the Table 11. The first, most observable result from Table 11 is that there is a marked variation in the batch throughput time that clearly depends on the order of the loading sequence. In the case of non-sequential processing, this variation is around 1% while, in the sequential case, variations are between 1.6 and 7.8%. This lower variability supports the conclusions given earlier that non-sequential processing is more beneficial than sequential processing. Lower batch throughput times are also obtained consistently by using non-sequential processing. This difference is also indicated by the average processing times listed at the bottom of Table 11. Furthermore it is clear that the best non-sequential results do not correspond to the best sequential results.

Overall, several patterns are observable in Table 11, such as the interchangeable nature of part type 1 and 3 in the non-sequential case. However, what has not been determined is how to predict when a specific loading sequence results in a higher than average throughput time, or how to select a sequence which would produce the lowest throughput time. Various hypotheses were perused to correlate the loading patterns and throughput results but no conclusive pattern was found. This problem requires further investigation. The first, most observable result from Table 11 is that there is a marked variation in the batch throughput time that clearly depends on the order of the loading sequence. In the case of non-sequential processing, this variation is around 1% while, in the sequential case, variations are between 1.6 and 7.8%. This lower variability supports the conclusions given earlier that non-sequential processing is more beneficial than sequential processing. Lower batch throughput times are also obtained consistently by using non-sequential processing. This difference is also indicated by the average processing times listed at the bottom of Table 11. Furthermore it is clear that the best non-sequential results do not correspond to the best sequential results.

Overall, several patterns are observable in Table 11, such as the interchangeable nature of part type 1 and 3 in the non-sequential case. However, what has not been determined is how to predict when a specific loading sequence results in a higher than average throughput time, or how to select a sequence which would produce the lowest throughput time. Various hypotheses were perused to correlate the loading patterns and throughput results but no conclusive pattern was found. This problem requires further investigation.

Table 11. Load sequence (results in seconds)

Non-sequential					Sequential				
Sequence	case 1	case 2	case 3	case 4	Sequence	case 1	case 2	case 3	case 4
1234000	2710	2710	2790	2790	1234000	2900	3030	3035	2935
1243000	2725	2725	2805	2805	1243000	2900	3030	3035	2935
1324000	2710	2710	2770	2770	1324000	2900	3030	3035	2935
1342000	2710	2710	2770	2770	1342000	2805	3030	3035	2935
1423000	2725	2725	2710	2710	1423000	2805	3030	3035	2935
1432000	2710	2710	2795	2795	1432000	2805	3030	3035	2935
2134000	2710	2710	2710	2710	2134000	3470	3030	3035	2935
2143000	2725	2725	2710	2710	2143000	3470	3030	3035	2935
2314000	2710	2710	2710	2710	2314000	3470	3030	3035	2935
2341000	2770	2770	2725	2725	2341000	3375	3130	2845	2815
2413000	2865	2865	2725	2725	2413000	3375	3130	2845	2815
2431000	2865	2865	2725	2725	2431000	3375	3130	2845	2815
3124000	2710	2710	2770	2770	3124000	2900	3030	3035	2935
3142000	2710	2710	2770	2770	3142000	2805	3030	3035	2935
3214000	2720	2720	2725	2725	3214000	3470	3030	3035	2935
3241000	2780	2780	2750	2750	3241000	3375	3130	2845	2815
3412000	2710	2710	2785	2785	3412000	3305	3130	2845	2815
3421000	2715	2715	2750	2750	3421000	3225	3130	2845	2815
4123000	2785	2785	2760	2760	4123000	3305	3130	2845	2815
4132000	2710	2710	2710	2710	4132000	3305	3130	2845	2815
4213000	2795	2795	2725	2725	4213000	3225	3130	2845	2815
4231000	2795	2795	2725	2725	4231000	3225	3130	2845	2815
4312000	2710	2710	2710	2710	4312000	3305	3130	2845	2815
4321000	2720	2720	2725	2725	4321000	3225	3130	2845	2815
Highest	2865	2865	2805	2805	Highest	3470	3130	3035	2935
Lowest	2710	2710	2710	2710	Lowest	2805	3030	2845	2815
Standard Deviation	48.4	48.4	31.4	31.4	Standard Deviation	249.8	51.1	97.0	61.3
Average	2741.5	2741.5	2743.8	2743.8	Average	3180.1	3080.3	2940.3	2875.0

5. CONCLUSIONS

The objective of this work is to enhance the notion of flexibility in "flexible" manufacturing cells so that a cell's utility can be enhanced by using a computer implemented scheduler. This aim has been achieved by allowing multiple parts to be processed concurrently without using predetermined cycles of robot movement. The program developed to control the FMC performs reliably. It provided tools for dynamically selecting parts and controlling the robot's movements to complete complex batch demands with lower throughput times than is possible by processing one part type at a time. The other significant contribution was to develop a dynamic manufacturing cell which could process parts non-sequentially, an issue which has not been addressed adequately in the literature.

The simulation program, developed in conjunction with the control software, allows users to experiment with a multitude of variables that exist in the FMC environment prior to selecting a strategy that best fits a production run. Experiments using this software demonstrated its potential as a tool for examining different FMC control heuristics, and the effect of buffers on the throughput as well as part loading order questions. From these experiments a general understanding of the complexity of this enhanced flexible environment can be gained. The first critical issue that the simulation software helped to verify was that, in all cases, the concurrent processing of parts is more desirable than processing parts in cycles when trying to minimize the batch throughput time. The software also helped to study the relationship between robot's moves to machining times. When this ratio was low (between 1:1 and 1:10), the throughput times were less predictable due to the changes in robot's path. However, it was observed that this relationship became more predictable once the ratio of the robot's move to machining times was higher than 1:10. At this point all robot movements occurred while all the stations were engaged in machining. This resulted in a steady increase in the throughput time. The software also

helped to confirm that, from the perspective of batch throughput time, using an 'unload always' robot movement rule is consistently superior (albeit marginally) to a 'load always' rule. Numerical simulations also suggested that there was a limit to the number of buffers that could be added to reduce the throughput time. It was shown that, after a point, buffers began to work more as a storage device than as dynamic transfer points. Increasing the number of buffers can enhance the performance of the cell but excess work in progress in arguably not desirable.

The question of non-sequential processing versus sequential processing was also examined using the simulation software. The results show that non-sequential processing, when possible, reduces batch throughput times, thereby increasing the utilization of a cell. Factors that increase or decrease the significance of this improvement include the number of machines in the FMC and the location of the "bottle neck machine" or the machine most in demand in the machining processing cycle. The final set of experiments showed the effect on the batch throughput time of the sequence in which parts are introduced into the cell. Consideration of sequence in which parts are introduced proves to be significant depending on the process requirements of the parts in the batch. An experiment of 24 different part sequences was tested with four different batch orders of 40 parts. Each batch required a total of 135 minutes of machining. As a result of the different part orders there was a 1% variation in the batch throughput time using non-sequential processing, and a 7.8% variation in sequential processing. Overall, the loading sequence had a greater effect on sequential processing than that produced by non-sequential processing. However, no easily generalized patterns were obvious for selecting the sequence that generates the smallest batch throughput time. Thus far, the only way to determine the best sequence in which to load part is to simulate all combinations and allow the software to choose the best part sequence for a given scenario which can not be done in real time.

REFERENCES

1. Aneja, Y. and Kamoun, H. (1999). Scheduling of parts and robot activities in a two machine robotic cell. *Computers and Operations Researc*h, 26(4): 297-312.
2. Agentis, A. (2000). Scheduling no-wait robotic cells with two and three machines. *European Journal of Operational Research*, 123: 303-314.
3. Agentis, A and Pacciarelli, D. (2000). Part sequencing in three-machine no-wait robotic cells. *Operations Research Letters*, 27: 185-192.
4. Chen, S., Chen, L., and Lin, L. (2001). Knowledge-based support for simulation analysis of manufacturing cells. *Computers in Industry*, 44: 33-49.
5. Chen, H., Chu, C, and Proth, J.M. (1997). Sequencing of parts in robotic cells. *The International Journal of Flexible Manufacturing Systems*, 9: 81-104.
6. Cheng, C.W, Sun, T.H. and Fu, L.C. (1994). Petri-net based modeling and scheduling of a flexible manufacturing system. *IEEE International Conference on Robotics and Automation*, pp. 513-519.
7. Crama, Y. and Klundert, J.V.D. (1997). Cyclic scheduling of identical parts in a robotic cell. *Operations Research*, 45: 952-965.
8. Geismar, H.N., Sriskandarajah, C., and Ramanan, N. (2004). Increasing throughput for robotic cells with parallel machines and multiple robots. *IEEE Transactions on Automation Science and Engineering*, 1: 84-89.
9. Hall, N.G., Kamoun, H., and Sriskandarajah, C. (1998). Scheduling in robotic cells: complexity and steady state analysis. *European Journal of Operations Research*, 109: 43-65.
10. Kenne, J.P. and A. Gharbi, A. (2004). A simulation optimization based control policy for failure prone one-machine, two-product manufacturing systems. *Computers and Industrial Engineering*, 46: 285-292.
11. King, R.E., Hodgson, T.J., and Chafee, F.W. (1993). Robot task scheduling in a flexible manufacturing cell. *IIE Transactions*, 25(2): 80-87.
12. Kumar, R., and Li, H. (1994). Assembly time Optimization of PCB Assembly. *Proceedings of the American Control Conference,* Baltimore Maryland, pp. 306-310.
13. Lin, L., Wakabayashi, M., and Adiga, S. (1994). Object-oriented modeling and implementation of control software for a robotic flexible manufacturing cell. *Robotics & Computer-Integrated Manufacturing*, 11(1): 1-12.
14. Moreno, A.A. and Ding, F.Y. (1993). A constructive heuristic algorithm for concurrently selecting and sequencing jobs in an FMS environment. *International Journal of Production Research*, 31(5): 1157-1169.
15. Niemi, E. and Davies, B.J. (1989), Simulation of an optimizing FMS-cell control system. *Robotics & Computer-Integrated Manufacturing*, 5(2/3): 229-234.
16. Rebaine, D. and Strusevich, V.A. (1999). Two machine open shop scheduling with special transportation times. *Journal of the Operational Research Society*, 50: 756-764.
17. Sethi, S.P., Sriskandarajah, C., Sorger, G., Blazewicz, J., and Kubiak, W. (1992). Sequencing of parts and robot moves in a robotic cell. *The International Journal of Flexible Manufacturing Systems*, 4: 331-358.
18. Sethi, S.P., Chandrasekaran, R., Drobouchevitch, I., and Sriskandarajah, C. (2004). Scheduling multiple parts in a robotic cell served by a dual-gripper robot. *Operations Research*, 52: 65-82.
19. Yalcin, A. and Boucher, T.O. (1999). An architecture for flexible manufacturing cells with alternative machining and alternative sequencing. *IEEE Transactions on Robotics and Automations*, 15(6): 1126-1130.