THESIS REPORT
Ph.D.
Institute for Systems Research

Reconfigurable Control in Discrete Event Dynamic Systems Applied to Manufacturing Systems

by J.S. Dhingra
Advisor: G.L. Blankenship

The Institute for Systems Research is supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Ph.D. 93-3

Reconfigurable control in Discrete Event Dynamic Systems applied to Manufacturing Systems

by

Jastej Singh Dhingra

Thesis submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1992

Advisory Committee:

Professor Gilmer L. Blankenship, Chairman
Professor Steven I. Marcus
Professor Prakash Narayan
Professor Guangming Zhang
Professor Michael C. Fu

ABSTRACT

Production management in an automated manufacturing system entails the implementation of the following two decision functions: Operational Planning and Resource Allocation, and Production Control. In this work, we present scheduling and production control algorithms for manufacturing systems. We present a general manufacturing system model and formalize the concept of a schedule as a single sequence of operations. Using a general performance measure we formulate the operations scheduling problem as a combinatorial optimization problem. A simulated annealing based optimization algorithm is developed for this job shop scheduling problem. We present a three level hierarchical on-line reconfigurable control scheme. For the "process" level control, we present a reactive operations scheduling scheme. Based on the control specification at the process level, the lower "operation" level parameters are defined. We present operation control algorithms for both continuous and batch mode processing. Using dynamic programming principles, we present a Quasi-Variational Inequality based impulse control algorithm for online control of processing rates for a single batch. Perturbation Analysis, in conjunction with stochastic approximation techniques, is used for continuous mode, online processing rate control. Algorithm extensions to other discrete event systems are also discussed.

CHAPTER
ONE

Introduction

1.1 Introduction

A flexible manufacturing system consists of a network of multi-purpose machines which process raw material, in a predefined process sequence, to produce a set of finished products (or parts). The multi-purpose machines are, in general, numerically controlled and are equipped with automatic tool-changing systems. The parts are routed, to the different machines, using one of the several automated material handling systems such as robots, conveyer systems, trolleys etc. The state of a manufacturing system, at any time instant, can be defined in terms of the operations in progress, along with the involved resources. The state of the system changes at the occurrence of an "event". Start/end of an operation, machine failure/repair etc., are examples of such events. These events can occur at deterministic (start/finish of an operation) or random (machine failure/repair) times.

The traditional systems theory has mainly been devoted to the study of systems evolving continuously in time. Such systems satisfy certain energy conservation and motion laws, and are modeled using difference or differential equations involving continuous valued, possibly random, variables. It is difficult to model the complex man-made dynamic systems using ordinary or partial differential equations. These systems are event driven in a symbolic state space, i.e., the occurrence of events causes the system to move from one logical state to another at discrete, possibly random, instants in time. The system dynamics are characterized by sequences of symbolic or logical events along with their discrete and random occurrence times. A manufacturing system, with its event driven dynamics, is a perfect example of such a discrete event system. Other examples of such systems are: computer/communication networks, traffic systems, order processing systems, power relay networks etc. All these examples are operationally different, but the underlying event driven dynamics are similar. These systems are considered to be part of the ever growing, large family of *Discrete Event Dynamic Systems* (DEDS).

Discrete event systems have event driven, piecewise constant trajectories. The state sequence is represented by constant segments of the trajectory: and

the duration of these segments represent the holding times for each state. The inherently discrete, physical states of the system, combined with the "continuous time" evolution (holding times are continuous variables) of the system, makes DEDS analysis a difficult task. For a more realistic formulation we also have to consider the transient behaviour, stochastic disturbances and the interacting dynamics between various components of the system. As the number of physical states grows exponentially with the increase in number of components, we have to take into account the computational aspects of the analysis.

For these reasons, it is very difficult to provide a unified analytical approach for the study of discrete event systems, which could support the development of control strategies for such systems. In this thesis we will present, using manufacturing systems as a test case, online control algorithms for discrete event systems. The need for planning and control in this "net-structured event-driven large scale service system"[1] stems from the desire for improved productivity, which can be defined in terms of meeting order due-dates while maintaining low work-in-process and finished-parts inventory, and efficient resource utilization. Although the control algorithms presented are specific to manufacturing systems, it is straight forward to adapt the algorithms for control of other discrete event systems.

1.2 Discrete Event Dynamic Systems: A survey

The diversity and the interdisciplinary nature of the field of discrete event systems stems from the variety of application areas. Although researchers have used techniques from operations research and computer science to study discrete event systems, only in the past few years has a serious effort been made to present a unified systems theoretic approach for understanding of such systems. Continuous time, continuous variable control theory is useful for suggesting modeling, analysis and control paradigms for discrete event systems. By drawing on the wealth of modeling and analysis techniques from the continuous case, extensions and analogs for the discrete event case can be obtained. Examples of such extensions include use of perturbation analysis techniques for performance evaluation and control [25], "linear" time-invariant models [10], stability, controllability and observability using supervisory control [42] etc. In this section, we will present a synopsis of the current research in the field of discrete event systems.

The study of discrete event dynamic systems can be divided into three main areas namely, modeling, analysis and control. As in the continuous variable dynamical system (CVDS) case, where differential/difference equations are used for modeling, the need for a strong modeling formalism for the discrete event case can not be overemphasised. A good model is not only useful in definition, description and verification of system behaviour but also, in general. can be helpful in analysis and control of DEDS. Though several modeling formalisms

	Timed	Untimed
Logical	Timed Transition Models Timed Petri Nets	Finite State Machines Petri Nets
Algebraic	Min-Max Algebra	Finitely Recursive Processes
Performance	Queuing Models; Simulation Models	

Table 1.1: Classification of DEDS models.

have been developed, no unified approach (analogous to differential equations for the CVDS case) has emerged which could generate satisfactory models for all types of discrete event systems. The diversity in the types of models for DEDS reflects the diverse fields in which DEDS appear and the different aspects of interest for each DEDS.

The modeling formalisms can be categorized on the basis of timed (or quantitative) models or untimed (qualitative) models. Quantitative or timed models address timing constraints (event occurrence times) and are useful in performance evaluation and optimization of DEDS. The untimed models deal with the sequence of events and address qualitative questions like reachability, deadlock, observability etc. The models can also be grouped into logical, algebraic or performance models. This distinction arises on the basis of the methodology used to describe DEDS behaviour. As the name suggests, logical models use logical descriptions for system behaviour, while algebraic models define the DEDS event sequences on the basis of algebraic operations on the present state and past event sequencing. Performance based models include queuing models and simulation models. These models use a probabilistic approach for modeling event occurrences and use statistical techniques for performance analysis. Table 1.2 (from [27]) summarizes the different models and their classification. In the remainder of this section, we will present brief descriptions of the various modeling formalisms along with their applications in analysis and control of discrete event systems.

Petri Nets provide an asynchronous, concurrency based, structural modeling formalism for discrete event systems. The nodes and transition arcs of a Petri Net, along with the token initiated firing of transitions, make it ideally suited for modeling discrete event systems. Timed Petri Net models permit a deterministic or stochastic firing time for each transition. The firing time concept supports modeling of various delays in a discrete system and provides a timed. logical modeling formalism. Sajkowski [45], Hillon and Proth [24], Hatano *et al* [23]

and Baccelli *et al* [2] have used variations of timed Petri nets (also referred to as time-augmented Petri Nets) for modeling DEDS. Hillon and Proth have used the timed event graph theory to model job-shop systems and to "evaluate the steady state performance of the system under a deterministic and cyclic production process." Baccelli *et al* have used stochastic marked graphs for flow analysis and studied the steady state behaviour. Sajkowski used time-augmented Petri net for modeling and verification of communication protocols.

Ramadge and Wonhan [42, 43] developed DEDS models using formal language theory. The set of desired DEDS sample paths is modeled as a generator of a formal language. On the basis of the formal language description, qualitative issues such as existence, uniqueness and definition of controllers may be posed. The concepts of target language and controllable sub-language (i.e. set of target sequences and controllable sub-sequences respectively) lead to the specification of a supervisor/controller for general discrete event systems. Ostroff and Wonham [36, 37] proposed a timed transition model for discrete event systems and have used real time temporal logic for system definition and controller specification.

Cohen *et al* [10, 11] have presented algebraic models for discrete event systems. The systems are modeled as timed event graphs, which are a special case of timed Petri nets. The discrete event system behaviour, represented by state evolution equations using *min-max* algebra, is shown to be linear for the deterministic case. Hence, the discrete event system can be modeled as a linear, time-invariant, finite-dimensional system. This linear algebraic formulation allows extension of certain results from conventional linear systems theory to the discrete event case.

Inan and Varaiya [28] have used Finitely Recursive Process (FRP) models. These models are based on the communicating sequential processes developed by Hoare. The FRP models specify the process by a set of recursion equations and the system trace is specified by recursive construction in time. The FRP formulation has greater descriptive power than Petri nets ([28]) and allows for an easier software implementation for simulation and performance evaluation.

Queuing theory is a natural candidate for DEDS modeling. Baccelli and Makowski [3] have used classical queuing theory, along with stochastic ordering and ergodic theory, to model and analyze queuing systems with synchronization constraints. The rich literature of Markov process analysis techniques is an added incentive for the use of queuing theory based DEDS models. Glynn [17] has used Generalized Semi-Markov Process (GSMP) formalism for DEDS modeling. The GSMP approach attempts to provide a mathematical framework to discrete event simulation. The mathematical formulation has a discrete component, which is representative of the "physical" state of the simulation, and a continuous component used for "time" representation. The occurrence of events, coupled with event lifetimes, determine the next state. This in turn leads to re-evaluation of event lifetimes and thus the system evolves in time.

Modern technology has led to increased computational power for the user, which in turn led to the development of various object oriented simulation languages. QNAP2, SIM++, SIMSCRIPT, SLAM, SIMAN, etc., are few examples of simulation languages which support implementation, verification and testing of different analysis and control algorithms for discrete event system models.

The modeling formalisms discussed above are useful for better understanding of the DEDS behaviour. They also help in i) identification of the different control issues; and ii) development of appropriate control policies. In the remainder of this section, we will present some control schemes which originate from the underlying modeling formalism used for DEDS description.

As mentioned earlier, Ramadge and Wonhom [42] have used a formal language based modeling scheme for description of DEDS. They developed the *Supervisory control* theory in which, based on the formal language model of the discrete event system, the supervisor (or controller) is defined as the largest *controllable* language contained in the given *legal* language. Using formal language concepts, various control theoretic ideas such as controllability, observability, aggregation, decentralized or hierárchical control etc., have been investigated. Supervisory control theory provides qualitative analysis for discrete event systems and separates the open loop dynamics (discrete event system) from the feedback control (supervisor/controller).

Ostroff and Wonham [36, 37] have used real time temporal logic as an assertion language for verification of the system behaviour under the given control specifications. The temporal construction is only used for specification of the control problem while the dynamics of the system is described using the timed transition model. This formal language based technique can be used to analyze qualitative problems as well as quantitative time based issues.

Passino and Antsaklis [38] have used deterministic, augmented automata, which can model costs for occurrence of events, as DEDS models. The discrete event system is modeled in terms of a *valid* behaviour. The *allowed* behaviour is defined by the desired characteristics of the DES, and is contained in the *valid* behaviour set. The optimal control problem is presented as choice of input sequences such that the DEDS behaviour is *allowable* and also minimizes some associated cost. The optimal controller is defined as a minimum cost controller among all possible solutions to the controller synthesis problem. The $\mathbf{A}^*$ heuristic search algorithm is used for finding the optimal controller.

Gershwin [16, 30] used the idea of grouping of events in a "frequency of occurrence" spectrum, to propose a *hierarchical* control strategy for planning/control of manufacturing systems. This hierarchical grouping of events allows for feedback control of events in the same level and also defines control for lower levels (i.e higher frequency events). Using dynamic programming arguments it was shown that the optimal production policy is to produce parts at the maximum allowable rate towards the "hedging point" inventory level. This hedging point inventory level is an accumulated finished parts inventory which allows to

"hedge" against future failures at minimal cost. Once the hedging point level is achieved, the inventory should be maintained at that level until the machine state changes. On similar lines, Perkins and Kumar [39], Bielecki and Kumar [5] developed a class of scheduling policies for general flexible manufacturing systems. Sharifnia [46] used the concept of hedging points from the hierarchical control theory, for production control in manufacturing systems.

Perturbation analysis has previously been used for sensitivity analysis in linear continuous time systems. Ho, Cao, Gong and Suri [25, 26, 9, 19, 18, 48, 50] have extended those ideas and applied them to discrete event dynamic systems. Parameter sensitivities for DEDS are calculated from observation of a single sample path. Sub-divisions of Perturbation analysis into Infinitesimal PA, Extended Infinitesimal PA, Smoothed PA and Finite PA originate from the same basic premise, but with different assumptions and different applications. Leung and Suri [31] have used Perturbation analysis for online optimization of certain parameters for a G/G/1 queue. The basic idea is to use perturbation analysis to obtain cost gradient estimates. The estimated gradient values are then used in stochastic approximation algorithms for on-line optimization.

Scheduling theory can be considered as an integral part of discrete event system control and planning. Scheduling problems arise naturally in discrete event systems. The need for increased productivity in manufacturing systems, improved link/resource utilization in computer/communication networks etc. – all these are reasons for use of scheduling as a planning tool. Scheduling can be considered as a procedure for system resource allocation over time. Based on the system model, scheduling problems can be classified as stochastic or deterministic. The majority of the research in deterministic scheduling theory formulates the scheduling problem as a combinatorial optimization/integer programming problem. The performance measure is maximized under resource allocation and timing constraints. The optimization problem can be solved using any of the existing techniques namely, exhaustive search, heuristic algorithms, dynamic programming etc. Stochastic scheduling allows for introduction of random characteristics which arise naturally in real world systems.

Scheduling is a *feed-forward* control procedure. In case of deterministic system models, the scheduling process takes into account the predicted future states for complete schedule generation. This is *predictive* scheduling. For stochastic models, if the actual present system state, which may differ from the predicted future state, is taken into consideration for future scheduling, then it is termed as *reactive* scheduling. The scheduling problem in all discrete event systems is similar, except for some minor, problem specific issues. The main thrust in recent research has been on finding efficient techniques for solution of the combinatorial problem. A comprehensive bibliography and literature survey on scheduling is available in [35].

Our literature survey on research in discrete event systems is not comprehensive. It is meant to convey an idea of the issues involved and the various

approaches being used. In the next section, we will present a description of a general flexible manufacturing system and a brief discussion on the various issues involved in manufacturing system control.

1.3 Manufacturing System Scheduling and Control

A flexible manufacturing system is composed of a network of machines, each capable of performing a variety of operations on different sets of parts. The parts are processed in a predefined sequence of operations, performed on different machines. The manufacturing system can produce parts using one of the three modes of operation: batch mode, continuous mode or hybrid mode. Batch mode operation[1] involves processing a single part type, over a given time interval (determined by the batch size), and then switching to process another part type. Batch mode processing is used if there is a significant setup time involved for the machine to switch from processing one part type to another. Continuous mode processing is used in cases where the processing for different part types is very similar and changeovers do not require a significant reconfiguration of the machine setup. A hybrid mode is possible in cases were sets of physical machines can be grouped to form "virtual machines." In such systems, we have batch mode processing for inter-virtual-machine operation and continuous mode processing (single product) within each virtual-machine. The grouping can be done on the basis of setup-times for changeover from one process configuration to another. Depending on the mode of operation, the parts are transported from one machine to another using robots, automated guided vehicles, conveyers or trolleys, or in some cases manually.

The manufacturing system is configured to process specific sets of products or parts. Associated to each part is a process flow, which defines the sequence of processes or operations to be performed, on the raw material, to produce the finished part. The constituent operations of a process flow can be performed only on a certain set of allowed machines. Hence, the process flow can be written in terms of a set of operation sequences, with each sequence corresponding to a distinct combination of allowed machines on which the operations can be done. Each operation on a machine has an associated set-up time and a processing time. This setup time corresponds to the time required for switching the machine configuration from one part processing to another. The processing time, as the name suggests, is the time required to process a single part on the machine.

A batch[2] is a physical order, of a pre-specified number of units of a specific part type, to be produced by a given due-date. Based on the batch size and machine assignments, we can define physical operations, along with associated

[1]This is distinct from the bulk processing, as in the case of IC wafer fabrication, in which a machine can process many wafers in parallel.

[2]No connection to the batch-mode processing discussed earlier. The usage will be clear from context.

setup and processing time, which are performed on different machines. The physical operations defined on each machine, under process flow precedence constraints, can be sequenced on that machine. On the basis of the setup times and the operation processing time (total time to process the number of parts in that batch) and the operation sequencing, we can assign the start and finish times for each operation. This cumulative task of machine assignment and operation sequencing, which includes assignment of operation start and finish times, is what is referred to as "Scheduling" in the manufacturing literature.

The problem of machine assignment in scheduling, also referred to as part "routing," arises naturally in this flexible manufacturing environment. As mentioned earlier, it may be possible to perform a particular operation of a batch on one of given set of allowable machines. The processing and setup times for that operation may differ on each of those machines. The intuitive choice would be to assign it to a machine with the smallest processing and setup times, but under certain load conditions, it might be more logical to schedule it on a slower machine. This problem then can be considered as a combinatorial optimization problem, analogous to the "network" flow problem in computer networks. The set of possible solutions to this combinatorial problem grows exponentially as the manufacturing system flexibility (i.e. capability of the constituent machines to process different parts) increases. On the basis of machine assignments, each machine has an assigned set of operations to be performed. Operation sequencing is the (optimal) ordering of these operations. The optimality of the ordering can be defined in terms of the minimization of some cost defined as a function of batch tardiness, work-in-process, machine utilization etc.

Optimal operations scheduling in a manufacturing system is an NP-complete problem, i.e., the problem solving time grows exponentially with the size of the problem. Common methods, for solving these problems, are based on finding a sub-optimal solution using a heuristic algorithm and then verifying the "closeness" of the solution to the optimal one. The closeness can be defined in terms of the difference between the associated costs for the two solutions.

As is apparent from the discussion above, scheduling is a form of feed-forward predictive control of the manufacturing system. In an ideal, deterministic case, this feed forward control would be sufficient to optimally manage the manufacturing system. Since real manufacturing systems are characterized by random occurrences like machine failures, varying demands, random processing and setup times etc., this predictive control will not be optimal online. Feedback control is required to deal with the uncertainity.

The manufacturing system can be controlled on the *planning level*, or on the *production level*. In the scheduling context, there are two possible ways of solving this control problem: a) *Predictive scheduling*: Schedule operations taking into consideration the expected random machine failures and the expected random repair times, random part processing times, etc., or b) *Reactive Scheduling*: Use the present system state, as feedback information, to schedule future operations

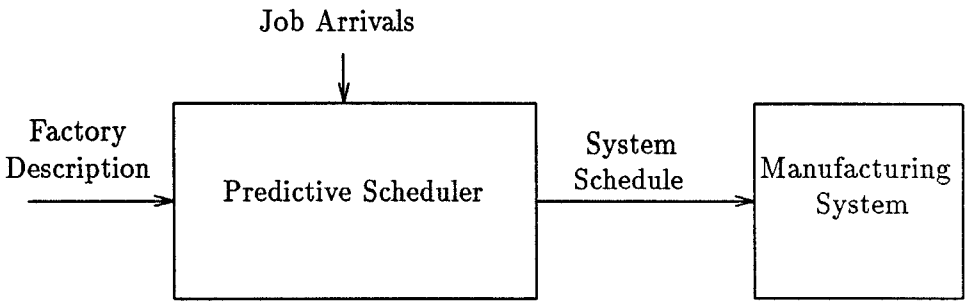


Figure 1.1: Predictive Scheduling based Control

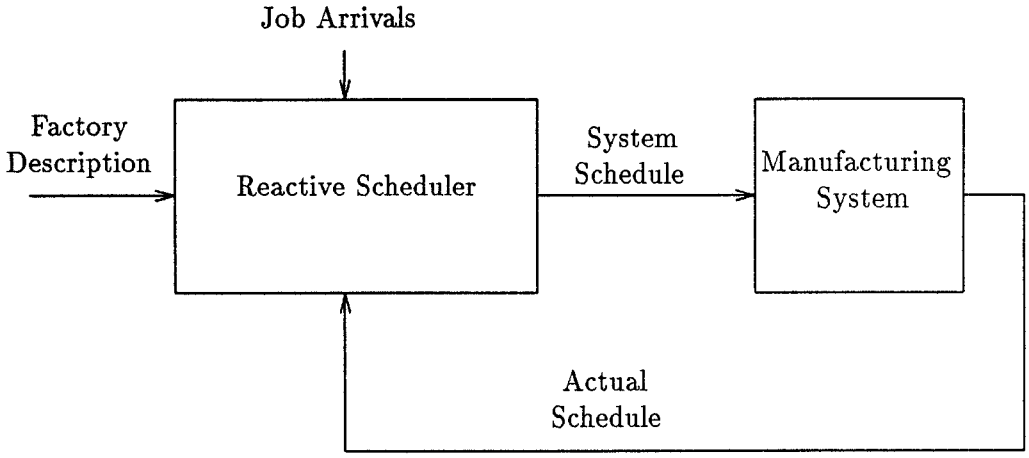


Figure 1.2: Reactive Scheduling based Control

on the different machines. The schematic representations for the two scheduling schemes are shown in Figure 1.1 and 1.2 respectively.

At the production level, the basic strategy is to generate a schedule and then re-configure the manufacturing system to achieve minimum deviation from the desired schedule. This reconfiguration entails optimal adjustment of system parameters. For example, it might be possible to operate a machine at a faster speed so that the present operation is finished on time and does not affect the subsequent operations. As another example, in case of a machine failure, the waiting batch operations could be routed to other slower machines. The control information is passed along to the predictive scheduler, so that the changes can be taken into account for scheduling future order arrivals. Figure 1.3 depicts an example production control scheme.

1.4 Research Approach and Organization

Production management in an automated manufacturing system entails the implementation of the following two decision functions: Operational planning and resource allocation, and production control. The first decision procedure in-

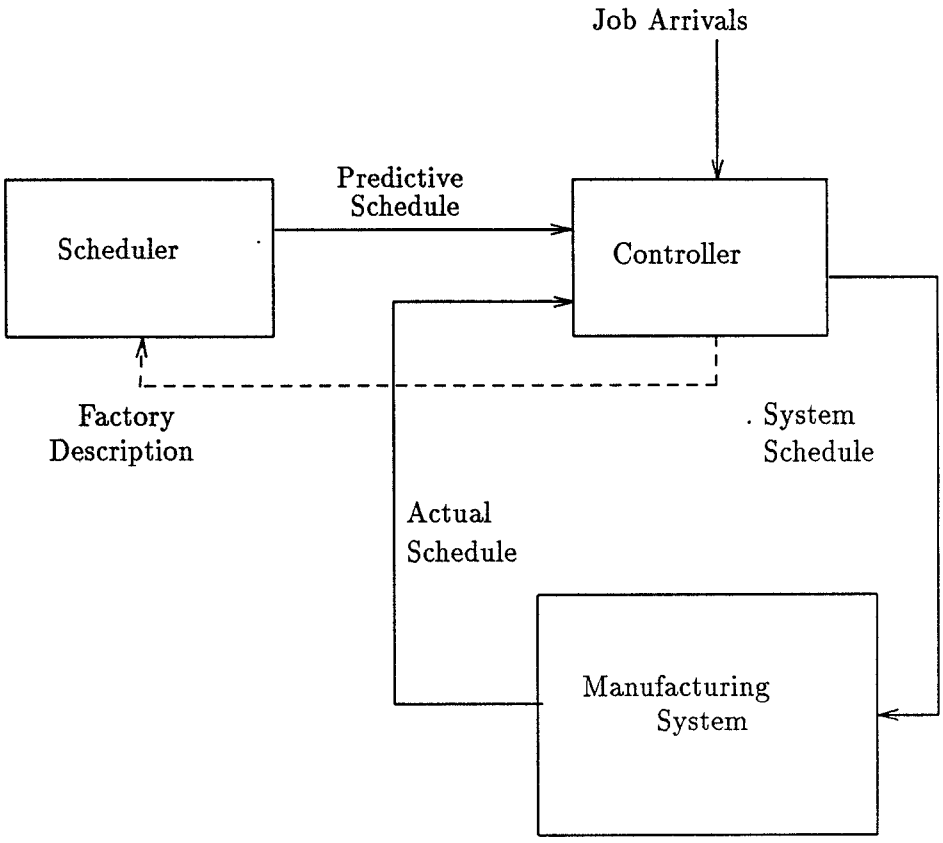


Figure 1.3: Example Production Control Scheme

volves allocation, in time, of raw material and tool resources for each order. It also includes optimal, future sequencing of operations on different machines. The optimality of the operation sequences can be defined in terms of constraint optimization of an appropriate performance criteria. The constraints are defined in terms of the order due-dates, resource availability, the machine-network layout etc.

The second decision function, production control, refers to the feedback control of the actual production process. The actual production process is continuously compared to the planned production schedule and appropriate corrective action is taken to minimize deviation from the planned production schedule.

In this thesis, we will present algorithms for both operational planning and production control of manufacturing systems. The emphasis, in this thesis will be on development of control algorithms which can be implemented in a real flexible manufacturing system.

In chapter 2, we will present a detailed description of the Texas Instruments PC board assembly unit at Johnson City, which has provided the motivation for this research. Using this manufacturing system as an example case, we will present a mathematical model for a general manufacturing system. This will

form the basis for development of a simulated annealing based scheduling algorithm. The scheduling methodology presented is general and can be used for scheduling other manufacturing systems. This will be the operational planning control component of our research. The experience gained during design, implementation and installation of scheduling software at the manufacturing facility will be highlighted.

In chapter 3, we will present a three level hierarchical control structure for online production control of manufacturing system. We will discuss the essential concepts of general hierarchical schemes and their advantages. We will next identify the various control parameters of interest and present our three level control scheme. As a prelude to algorithm specification and testing, we will introduce a hybrid (with both continuous and batch mode processing) manufacturing system example.

Based on the three level hierarchical structure, chapter 4 will be devoted to development of algorithm for the middle level, i.e., the *process control* level. A reactive operations scheduling scheme will be presented. The predictive scheduler, discussed in chapter 2, in conjunction with the manufacturing system and a reactive controller, will be used for the process control algorithm. In the end, we will present results on application of the algorithm for the example test case.

In chapter 5, we will present control algorithms for the lower level, *operation control.* Since our test case includes both continuous and batch mode processing, we will present algorithms for both cases. For the batch processing mode, we will present a dynamic programming based control algorithm for online impulse control of processing time for a single operation. Perturbation analysis, in conjunction with stochastic approximation, will be used for online control of processing time, for the continuous processing mode. The results on algorithm application will also be presented.

In chapter 6, we will present a summary of our research contribution. We will discuss, in the context of real manufacturing systems, the application aspects of the algorithms. Future work and extensions/adaptation of the algorithms, for control of other discrete event systems, will conclude the chapter and this thesis.

CHAPTER
TWO

Operations Scheduling

2.1 Introduction

In the previous chapter, we introduced the concept of operations scheduling as a form of *planning* control for manufacturing system. In this chapter, we present a operations scheduling scheme for feed-forward control of manufacturing system. In Section 2.2 we present a detailed description of the Texas Instruments electronic circuit board assembly unit at Johnson City, TN, which has provided the motivation for this research. This manufacturing system provides the basis for development of a general mathematical model for manufacturing system description (Section 2.3). Based on the mathematical model, we present the operations scheduling problem as a combinatorial optimization problem (Section 2.4). Using a variation of simulated annealing method for optimization, a predictive scheduling algorithm is presented in Section 2.5.

The features of the scheduling software $\mathcal{ABES}$ (Annealing Based Experiment in Scheduling), which was developed for scheduling the surface mount machines in the TI facility, will be presented in Section 2.6. We use the Texas Instruments manufacturing unit as an example case to illustrate the methodolgy. The manufacturing system model and the simulated annealing based scheduling scheme presented are very general and can be applied for control of other manufacturing and discrete event systems.

2.2 TI Custom Manufacturing Unit

The Texas Instruments PC board assembly facility at Johnson City, TN assembles circuit boards for external customers and also for their Industrial Automation division. The Custom Manufacturing Business consists of three main units: surface mount, through-hole, and quality control. Both single and double sided boards are manufactured at this facility. A typical process flow for a double sided board is shown in Figure 2.1. For a single sided board the second surface mount operation is not needed. The surface mount unit is fully automated and has a lower operational cost than the through hole and quality control units which are semi-automated. The installed capacity of the surface mount unit is

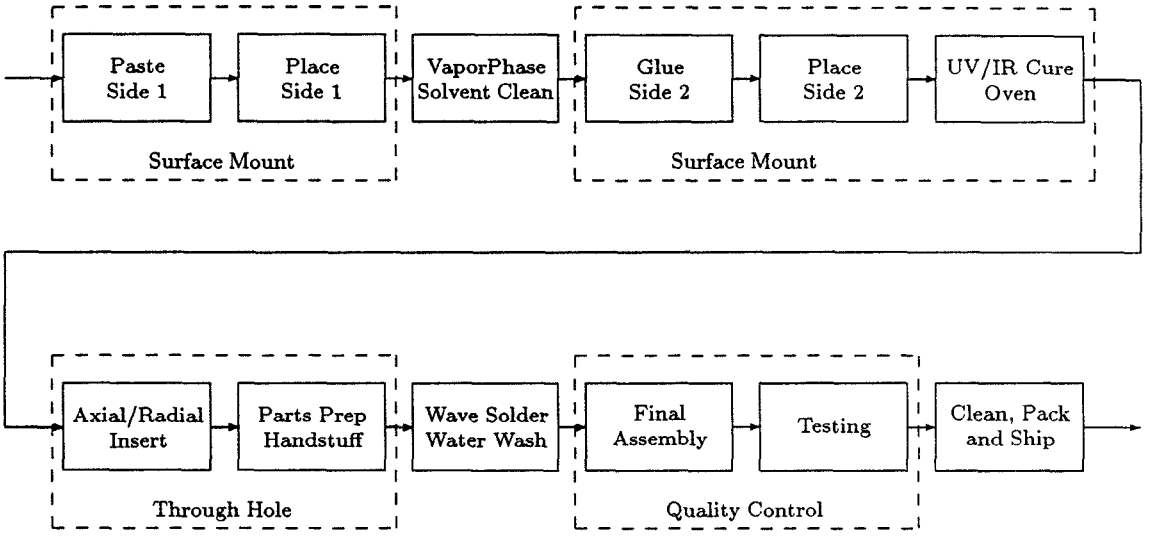


Figure 2.1: A typical process flow

less than the other two units and is generally the bottleneck region in board process flow. Therefore, we shall focus on scheduling the surface mount unit.

The surface mount unit consists of six machine cells called SMT lines. Of these six lines, lines 1, 2, 3, and 4 are dedicated to top-side processing, while the remaining two (5 and 6) are exclusively used for bottom side processing. The layout of the surface mount unit is shown in Figure 2.2. The top side processing lines, each consist of a screen-printer machine and a pair of component placement machines, while each of the remaining two lines have a glue-machine and a component placement machine followed by an Ultraviolet-Infrared Cure machine. The four SMT lines, 1 through 4, are connected through a conveyor system to the vapor phase and solvent clean machines. The boards, in general, are processed top side first on one of the four SMT lines and after being cleaned, are transported, via trolleys, to lines 5 or 6 for bottom side processing. The processing of the boards is continuous in each line, but the topside-processed and solvent cleaned boards are transported for bottom side processing in batches. Thus, the system has a mix of *continuous* and *batch flow* processing, which makes the scheduling job even tougher.

Each board has a *principal process flow* and a set of *alternate process flows*, which are determined by the characteristics of the parts being used in produc-

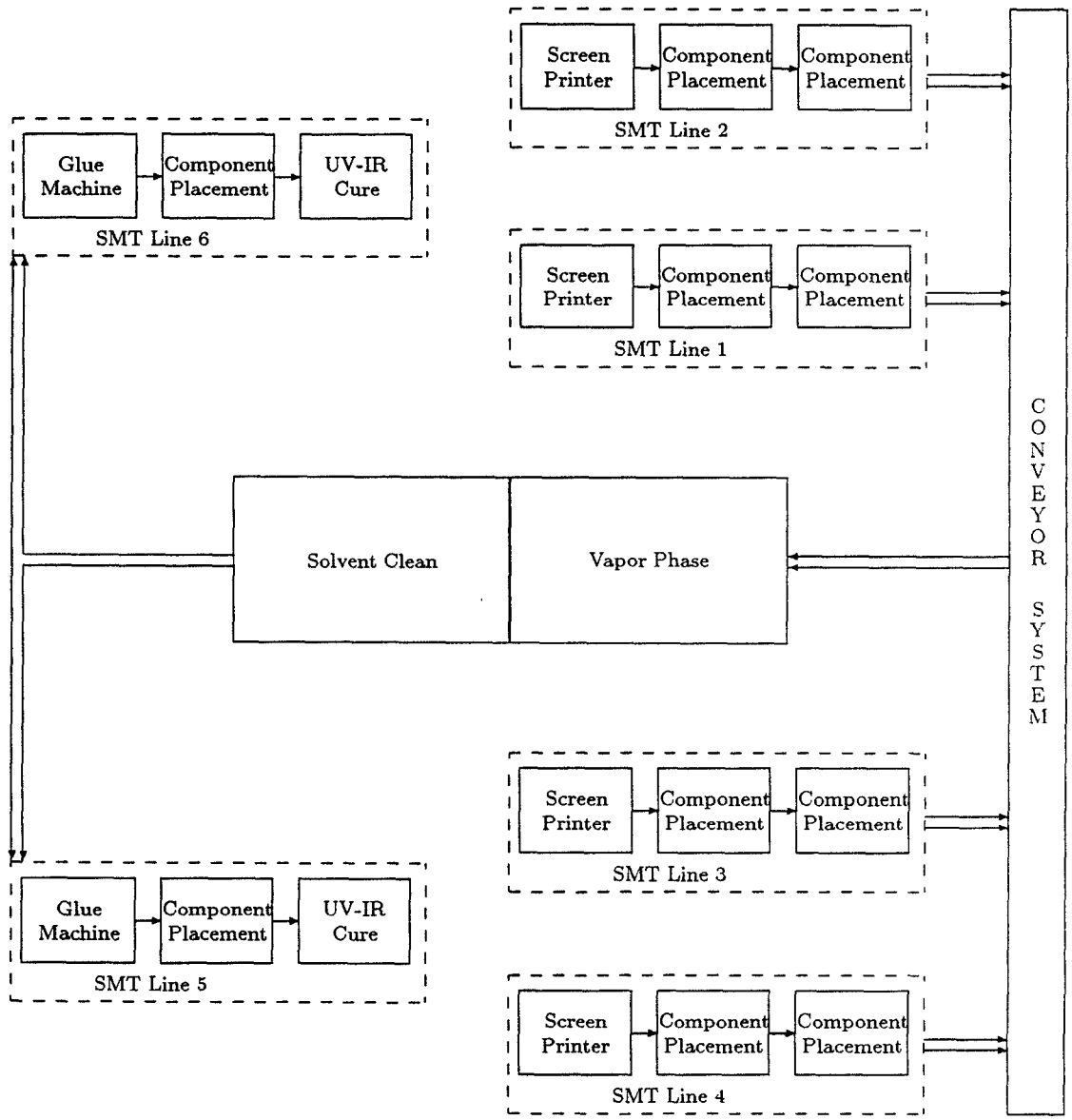


Figure 2.2: Surface Mount Unit

tion. For example, if many fine pitch parts are being used on a particular board, then it can be assembled only on one of the four SMT lines for topside and similarly can be processed only on one of the two lines for bottom side processing. Hence, we only have a principal process flow and no alternate flows for this particular board. On the other extreme, there are certain boards which can be manufactured on any of the lines. Thus, a double sided board which can be run on any of the 4 lines for topside processing and either of the two lines (5 or 6) for bottom side processing has 8 possible process flow configurations. We assign any one of those process flows to be the principal one and the rest are alternate flows.

The Custom Manufacturing Business produces about 50 different boards. Depending on the customer demand and parts availability, it handles 30 different orders, or batches, per week on an average. The number of operations to be scheduled for each batch varies between 3 and 6 depending on the number of sides to be processed. Therefore, in principle we have about 100 operations to be scheduled each week.

The concept of "board families" plays an important role in schedule generation. If two batches of two different boards, belonging to the same board family, are scheduled successively on the same surface mount line, the setup time for the succeeding operation is reduced to a fraction of the original setup time. This occurs when the two boards have similar sets of components to be used on that particular line and so only a partial teardown/setup is required. These reduced setup times lead to increased machine utilization and higher productivity.

In certain batch orders, it is possible, that the raw material is not available at the time of the order. Hence, the operations can only be scheduled after the parts are made available. All these additional constraints, i.e., non-availability of the raw parts, machine availability over time, board family based reduced set-up times, machine assignment for each operation, the large number of operations to be scheduled in a given time frame etc., makes optimal scheduling a formidable task.

The surface mount machines are scheduled manually. Parts availability and the customer demand date are the main criteria used for scheduling. The different operations for various batches are then scheduled appropriately, on the 6 lines, so as to meet the due-date requirements. Approximate board processing times and machine setup times are used to determine the production run for each batch. This process of schedule generation is time consuming and tedious. Moreover, the quality of the schedule is heavily dependent on the experience and expertise of the scheduler.

The Texas Instruments assembly unit provided us deep insight into the actual functioning of a real manufacturing system. The various operational issues, as discussed above, and their role in schedule generation provided us with a better understanding of the complexity and the intricate nature of flexible manufacturing system scheduling. The assembly unit provided a benchmark for the

development of a general manufacturing system model. The modeling framework, along with the scheduling experience, were helpful in the development of a general purpose scheduling software $\mathcal{ABES}$. The onsite visit, for the $\mathcal{ABES}$ installation phase, was further useful in understanding of the different control issues involved in manufacturing system. This information was beneficial for development of online control algorithms for manufacturing systems.

2.3 Manufacturing System Model

Motivated by the flexible manufacturing system described above, we present a general, deterministic manufacturing system model. Though the model presented here is based on the TI assembly unit, it can be used to model different flexible manufacturing systems.

We model the factory as a set of physical machines $\mathcal{M} = \{m_1, m_2, ..., m_{|\mathcal{M}|}\}$. A machine $m_i \in \mathcal{M}$ can be a single physical machine or a virtual machine (ordered collection of machines which process parts in tandem). We will not distinguish between the concept of physical or virtual machine. For our purposes the machine is a processing station with a single inflow and outflow of parts. The manufacturing system produces a set of parts $\mathcal{P} = \{p_1, p_2, \ldots, p_{|\mathcal{P}|}\}$. To manufacture part p_i a sequence of operations, $< O_k^i >_{k=1}^{N_i}$, must be performed in order. That is, operation O_k^i must begin before operation O_{k+1}^i. As discussed in Section 2.2, it may be possible to perform a particular operation, say O_k^i, on any machine from a given subset of $\mathcal{M}$. On this basis, we can associate, to each part type p_i, a set of process flows $\mathcal{F}_i = \{f_1^i, f_2^i, \ldots f_{|\mathcal{F}_i|}^i\}$. A process flow f_l^i is defined as a sequence of machine operations $< {}_jO_k^i >_{k=1}^{|f_l^i|}$ where ${}_jO_k^i$ represents the k^{th} prototype operation in the process flow for part p_i, performed on machine m_j.

Let $\mathcal{O}$ denote the set of all operations, and let $\mathcal{SO}$ denote the set of finite sequences of distinct operations. The parts are produced in batches and we may have to schedule several batches for each part. Hence, there is also a set of *batches*, $\mathcal{B} = \{b_1, b_2, ..., b_{|\mathcal{B}|}\}$. Each batch contains many instances of a single part. For each batch b, of device i, there is a sequence of *operation instances*, denoted $< (b, O_k^i) >_{k=1}^{N_i}$; (b, O_k^i) represents operation O_k^i acting on all parts in batch b. We denote the set of operation instances as $\mathcal{OI}$, and the set of finite sequences of operation instances as $\mathcal{SOI}$. The selection of a particular process flow for a given batch is equivalent to the machine assignment for each constituent operation instance of that batch. Further, each batch b_i has an earliest start time, a due time, and a priority chosen from a finite set $\mathcal{W}$.

Each operation has a machine on which it must be performed and involves a *principal setup time.* The idea of *method family* is to group different part types into "families". Based on the operation sequencing on a machine, if two operations of the same *method family* are performed in succession on a machine,

the actual time for a setup change is a fixed fraction of the principal setup time. This allows for the reduced setup representation discussed in section 2.2.

There are two ways to transfer parts from one operation to the next. In one case, the entire batch is processed by the first operation before it begins at the next. In the other, parts are "pipelined" from one operation to another. In this case, the second operation begins after a short delay, called the *transfer delay.* When batch processing is used instead of "pipelining," we say the transfer delay has the value BATCH.

Below we present this formally in terms of logical relations, which are, for our purposes, functions mapping one set into another. We define several operators to make the notation concise.

Operators

$\underline{in} : \mathcal{O} \times \mathcal{SO} \rightarrow \{$ true, false $\}$
$(o \ \underline{in} \ seq) =$ true $\Leftrightarrow seq = < o_1, o_2, ..., o_n > \bigwedge o = o_i$ for some i

$(\underline{follows}, \underline{in}) : \mathcal{O} \times \mathcal{O} \times \mathcal{SO} \rightarrow \{$ true, false $\}$
$(o_i \ \underline{follows} \ o_j \underline{in} \ seq)=$ true $\Leftrightarrow seq = < k_1, k_2, ..., k_n > \bigwedge k_p = o_j \bigwedge k_{p+1} = o_i$ for some p.

Again, a similar definition applies for
$(\underline{follows}, \underline{in}) : \mathcal{OI} \times \mathcal{OI} \times \mathcal{SOI} \rightarrow \{$ true, false $\}$

Distinction between the operators with identical names will be clear from context.

Relations

$op\_seq \ \mathcal{D} \rightarrow \mathcal{SO}$	The sequence of operations used to manufacture a part, subject to $o \ \underline{in} \ op\_seq(b) \bigwedge o \ \underline{in} \ op\_seq(a) \Rightarrow a = b$
$part : \mathcal{B} \rightarrow \mathcal{D}$	The part type of a given batch.
$qty : \mathcal{B} \rightarrow Z^+$	$qty(b)$ is the quantity of batch b
$earliest\_start : \mathcal{B} \rightarrow \Re$	Batch b cannot start before $earliest\_start(b)$.
$due\_time : \mathcal{B} \rightarrow \Re$	Batch b is due at $due\_time(b)$.
$priority : \mathcal{B} \rightarrow \mathcal{W}$	This is used by the cost function to be defined later.
$proc\_time : \mathcal{O} \rightarrow \Re^+$	Processing time per part for an operation.

$principal\_setup\_time : \mathcal{O} \rightarrow \Re^+$	The time required to setup the machine for a given operation.
$setup\_time : \mathcal{OI} \times \mathcal{OI} \rightarrow \Re^+$	Setup time between two operations
$transfer\_delay : \mathcal{O} \rightarrow \{\text{BATCH}\} \bigcup \Re^+$	The transit delay from the start of. an operation to the start of the next
$transfer\_time : \mathcal{OI} \rightarrow \Re^+$	Delay in start of operation
$method\_family : \mathcal{P} \rightarrow \mathcal{MD}$	$\mathcal{MD}$ is the set of Method family

Derived Set definitions

$\mathcal{OI} \equiv \{(b, o) | b \in \mathcal{B} \bigwedge o \ \underline{in} \ op\_seq(part(b))\}$

$\mathcal{SOI} \equiv \{< k_1, k_2, ..., k_n > | k_i \neq k_j \text{ when } i \neq j \bigwedge k_i \in \mathcal{OI} \text{ for all } i, j\}$

These are *operation instances* and *sequences of operation instances*, respectively.

2.4 Schedule Definition

Based on the manufacturing system model presented in the previous section, we formalize the concept of a schedule as a single sequence of operations. We present a general performance measure for this single sequence schedule representation, and then use it to formulate the job shop scheduling problem as an optimization problem.

A *schedule* is an assignment of a process flow to each batch (the routing problem), and a sequence of operation instances to each physical machine (the sequencing problem). We identify the schedule, S, with a graph, $\mathbf{G_S} = (\mathbf{N}, \mathbf{E_S})$, as follows. The vertices are all operation instances, $\mathbf{N} = \mathcal{OI}$. The set of edges, $\mathbf{E_S}$, comes from two sources. For each batch b, and each k, there is an edge in the graph from vertex $(b, O_k^{part(b)})$ to vertex $(b, O_{k+1}^{part(b)})$. For each machine, the sequence of operation instances assigned to it similarly defines a set of edges in the graph. A schedule is *feasible* if the graph so generated is acyclic. The precise definition of a schedule and its graph are as follows:

A *schedule*, denoted $S = (assign\_flow, mach\_op\_seq)$, is an ordered pair of relations which satisfies the following requirements:

$assign\_flow : \mathcal{B} \rightarrow \mathcal{F}$

$assign\_flow(b) \in \mathcal{F}_{part(b)}$ for every batch $b \in S$

$mach\_op\_seq : \mathcal{M} \rightarrow \mathcal{SOI}$

for any $(b, o) \in \mathcal{OI}$, there exists $m \in \mathcal{M}$ with $(b, o) \ \underline{in} \ mach\_op\_seq(m)$

Notation:

$\mathcal{S}$ = the set of feasible schedules.

$\mathbf{N} = \mathcal{OI}$ = set of vertices

$\mathbf{E}_S \subseteq \{(u,v)$, where $u, v \in \mathcal{OI}\}$ = directed edges for schedule S.

The graph $\mathbf{G}_S = (\mathbf{N}, \mathbf{E}_S)$ associated with schedule S satisfies:

(1) for $(b,o),(b,p) \in \mathcal{OI}, p$ *follows* o *in* $op\_seq(part(b)) \Longrightarrow ((b,o),(b,p)) \in \mathbf{E}_S$

(2) for $m \in \mathcal{PM}, (b^1,o^1),(b^2,o^2) \in \mathcal{OI}$,
(b^2,o^2) *follows* (b^1,o^1) *in* $mach\_op\_seq(m) \Rightarrow ((b^1,o^1),(b^2,o^2)) \in \mathbf{E}_S$

(3) Every edge in $\mathbf{E}_S$ is obtained from either (1) or (2)

As described earlier, a schedule consists of a method assignment and a sequence of operation instances for each physical machine. It is algorithmically and conceptually simpler to consider an alternative representation for the second part. Instead of using a separate sequence of operation instances for each machine, we use a single sequence containing all the operation instances. The space, $\mathcal{S}$, of feasible schedules is, then, replaced by $\mathcal{S}_{new} \equiv \mathcal{F} \times \mathcal{S}_n$, where $\mathcal{S}_n$ is the group of permutations of $n = |\mathcal{OI}|$ operation instances, and $\mathcal{F}$ is the union of sets of process flows. Our new representation for a schedule $S = (assign\_flow, mach\_op\_seq)$ is $S_{new} = (assign\_flow, \sigma)$, where $\sigma \in \mathcal{S}_n$ is a sequence of the operation instances.

We can determine σ from S, albeit nonuniquely, as follows. Form the graph $\mathbf{G}_S$ for S. Let α be an empty string of vertices of $\mathbf{G}_S$. Since S is feasible, $\mathbf{G}_S$ is acyclic, and there is a vertex which has no incoming arcs. Delete this vertex from the graph, and append it to the end of α. Also delete all of its outgoing arcs. The graph remaining is still acyclic. This process may be repeated until all vertices have been transferred to α. Now the sequence in α is a permutation of the elements of $\mathcal{OI}$, with the property that if there was a path from vertex A to vertex B in $\mathbf{G}_S$, then A appears earlier in α than does B.

The sequence of operation instances for a given batch, b, may be found from such a permutation by deleting from the list those operation instances belonging to a different batch. We call the sequence remaining the *filter* of α onto batch b, and denote this $filter_b(\alpha)$. Note that we must have:

$$filter_b(\alpha) = < (b, O_k^{part(b)}) >_{k=1}^{N_{part(b)}}$$

That is, the order of the operation instances must match that specified by $op\_seq(part(b))$. Similarly, the sequence of operation instances for physical machine m may be found by deleting any operation instance (b,o) not scheduled on machine m.

To illustrate this point consider a simple two machine (m_1 and m_2), two batch (b_1 and b_2) problem. Each batch consists of two operations. In batch b_1 the operations are sequenced first on m_1 and then on m_2; the reverse order is used by batch b_2. If operation ${}_bo_i^m$ represents the i^{th} operation of batch b to be performed on machine m, then, for the above example, a possible schedule operation sequence is given by $< {}_1o_1^1, {}_2o_1^2, {}_1o_2^2, {}_2o_2^1 >$. Since operations ${}_1o_1^1$ and ${}_2o_2^1$ are for the same machine, this schedule operation sequence indicates that ${}_1o_1^1$ must complete before ${}_2o_2^1$ begins. This schedule operation sequence and the corresponding derived machine schedules are shown in Fig 2.3. The arrows denote the operation sequence for each batch. The same schedule often can be derived from other schedule operation sequences, such as $< {}_1o_1^1, {}_2o_1^2, {}_2o_2^1, {}_1o_2^2 >$ for our example.

The times when operation instances begin and end can be computed from the schedule. Each operation is assumed to begin as soon as all its preceding operations have taken place. The algorithm for computing these times, along with the associated sub-algorithms, are presented below.

Transfer time calculation

algorithm $tranfer\_time((b, o))$
  if $transfer\_delay((b, o)$ =BATCH
    return $(qty(b) \times proc\_time(o))$
  else
    return $transfer\_delay((b, o))$
  endif
end_algorithm

Setup time calculation

algorithm $setup\_time((b_1, o_1), (b_2, o_2))$
  $part_i = part(b_i)$, for $i = 1, 2$.
  if $method\_family(part_1) = method\_family(part_2)$
    return β $principal\_setup\_time(o_2)$ /* for some β, $0 < \beta < 1$ */
  else
    return $principal\_setup\_time(o_2)$
  endif
end_algorithm

Start and finish time calculation

if $\{(b_2, o_2) \in \mathcal{OI} | ((b_2, o_2), (b, o)) \in \mathbf{E_s}\} = \emptyset$
  $t_s(b, o) = earliest\_start((b, o))$
else
  $t_s((b, o)) =$

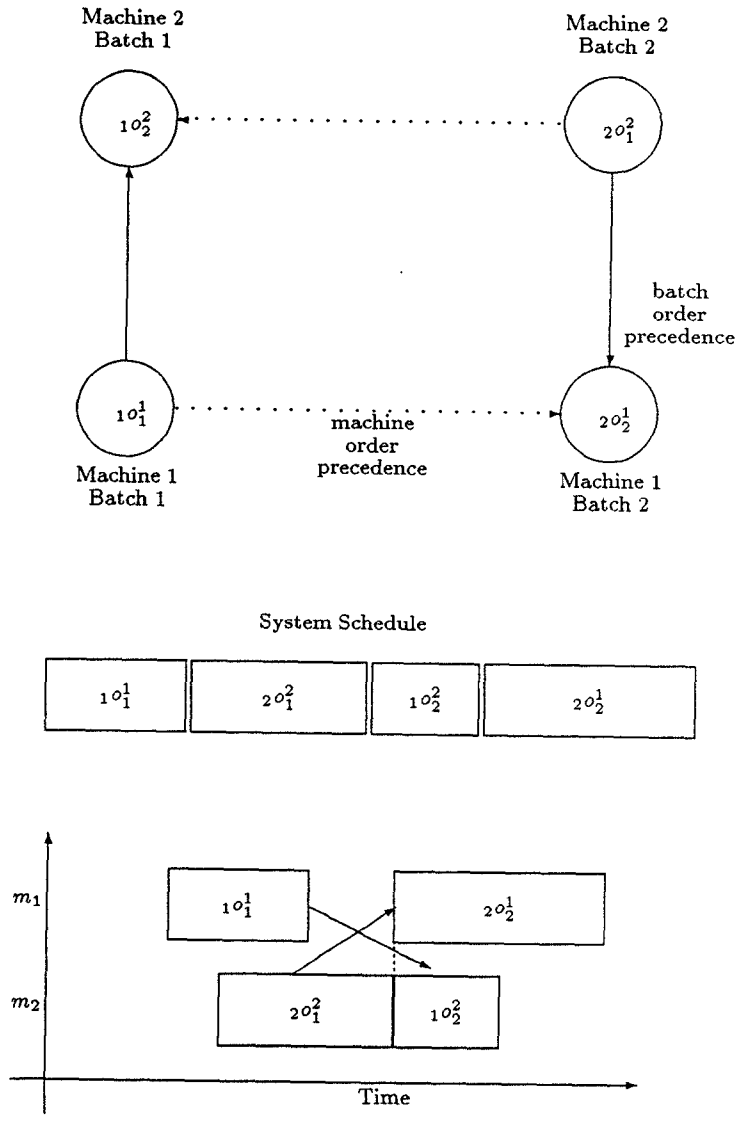


Figure 2.3: Derived machine schedules.

$$\max_{\{(b_2,o_2)|((b_2,o_2),(b,o))\in \mathbf{E_S}\}} \begin{cases} t_s((b_2,o_2)) + transfer\_time((b_2,o_2)) & \text{if } b_2 = b \\ t_f((b_2,o_2)) + setup\_time((b_2,o_2),(b,o)) & \text{otherwise} \end{cases}$$

$$t_f((b,o)) = \max\{t_s((b,o)) + qty(b) \times proc\_time(o), \max_{\{(b,o_2)\in\mathcal{OI}|o \text{ } \underline{follows} \text{ } o_2 \text{ } \underline{in} \text{ } op\_seq(part(b))\}}(t_f(b,o_2))\}$$

A performance measure for a schedule can then be computed from the start and finish times. Any performance measure which is computable from the start and finish times of all operations is permitted.

Our optimization problem is to **maximize the performance measure over the set of feasible schedules**, or equivalently, to **minimize the associated cost.**

$\mathcal{TV} = \{$ all maps $\mathcal{OI} \rightarrow \Re\}$ NOTE: $\mathcal{TV}$ stands for Time Vector.

$t_s : \mathcal{OI} \rightarrow \mathcal{TV}$

$t_f : \mathcal{OI} \rightarrow \mathcal{TV}$

$c : \mathcal{TV} \times \mathcal{TV} \rightarrow \Re$ This is the cost function.

For notational simplicity, we write $\mathcal{C}(S)$ for $\mathcal{C}(t_s, t_f)$. We can now pose the following optimization problem.

$$\mathcal{C}^\star = \arg\min_{S\in\mathcal{S}} \mathcal{C}(S)$$

2.5 Simulated Annealing Based Optimization

In the previous section, we represented a schedule as a single sequence of operations. The schedule optimization problem was posed as a maximization of a performance measure (minimization of an associated cost) over the set of feasible schedules. This optimization problem, and scheduling problems in general, are part of a larger group of *combinatorial optimization* problems. Combinatorial optimization problems deal with cost function maximization (or minimization) on a discrete problem domain.

As discussed in chapter 1, deterministic scheduling problems like many of the combinatorial optimization problems, are NP-complete. That is, the computational complexity (and hence the computation time) grows exponentially with the problem size. As a consequence, practical NP-complete problems, including our scheduling problem, can not be solved exactly in a reasonable time. Possible methods of finding an exact solution include exhaustive search, dynamic programming and branch-and-bound methods. These methods are based on enumeration of the possible solutions. In case of exhaustive search, as the name

suggests, all solutions are evaluated to obtain a globally optimal solution. The other two cases involve a restricted search procedure. On the basis of the problem structure, certain solutions are eliminated from the search process. This leads to a reduction in the search domain size and reduction in computation time. These exact solution techniques are not very effective for solving practical problems, due to the large search domains associated to those problems.

Heuristic methods rely on finding sub-optimal solutions and then verifying the closeness (in terms of appropriate probabilistic bounds) of the solution to the "unknown" optimal solution. Rule-based expert systems, simulated annealing etc. are some of the popular heuristics based combinatorial optimization techniques.

Rule based expert systems use problem specific heuristics for reduction of the size of the problem domain. The efficiency of these techniques is heavily dependent on the selection of the heuristics or rules. Moreover, the problem based heuristic specification does not allow for a general solution method.

Simulated annealing uses stochastic search in a deterministic problem domain. The iterative search procedure is based on selection of trial solutions, and then accepting (or rejecting) the trial solutions based on a probabilistic structure. The probabilistic structure tends to move the search process towards a global optimum, without getting entrapped in a local optimum. Simulated annealing provides good sub-optimal solutions, but at the cost of long execution times.

Simulated annealing has been used extensively for solution of combinatorial optimization. It has been used for solving circuit layout and routing optimization problems. The extensive literature, general acceptance of the technique, and the natural adaptation for a two level process (discussed later in this section) makes it an ideal choice for our problem. Although simulated annealing methods have been applied to job shop scheduling problems [40], our approach is technically different and more general. We will not get into the detailed theoretical analysis of simulated annealing. The interested reader can refer to [35] and the references mentioned therein. We are more interested in using this technique for solving our optimization problem.

There are two major components to simulated annealing based optimal solution search. The first component is the *neighbourhood* definition. Simulated annealing relies on random selection of a trial solution from the set of neighbouring solutions. The second aspect is the choice of appropriate performance measure (or cost function) for the solution, which needs to be maximized (minimized). In the remainder of this section, we will present the performance measure, the neighbourhood structure used, and other implementational details.

In operations scheduling, performance measures of interest include due date performance, work in process inventories, and machine utilization. In our implementation, we permit a cost function including each of these elements, and we use a variation of "simulated annealing" to optimize it. The scheduling problem

we consider has two related but distinguishable components, namely routing and sequencing. For this reason a two level stochastic descent method is natural.

Simulated annealing treats each schedule as a state in a discrete time finite state Markov chain. The transition probabilities of the Markov chain are selected so that the state drifts toward the lower cost schedules.

The simulated annealing algorithm uses a Markov process originally described by Metropolis [32]. An initial state (schedule), X_0, is chosen at random. Next, a nearby "trial" state, X_1^{trial}, is chosen by some probabilistic rule, and the two are compared. If the "trial" state has lower cost, it is accepted, with probability 1, as the new state of the Markov chain. Otherwise it is accepted with a lower probability determined by the relative costs. More precisely,

$$\begin{aligned} \Pr\{X_{k+1} = X_{k+1}^{trial} | X_{k+1}^{trial}, X_k\} &= \begin{cases} 1 & \text{if } c(X_{k+1}^{trial}) < c(X_k) \\ e^{-[c(X_{k+1}^{trial}) - c(X_k)]/T_k} & \text{otherwise} \end{cases} \\ \Pr\{X_{k+1} = X_k | X_{k+1}^{trial}, X_k\} &= 1 - \Pr\{X_{k+1} = X_{k+1}^{trial} | X_{k+1}^{trial}, X_k\} \end{aligned}$$

Here the value, T_k is called the "temperature," and the sequence $\{T_k\}_{k=0}^{\infty}$ is called the temperature schedule. If the trial state is only accepted when it has lower cost, the algorithm will remain in any local minimum it may find. This is the case when the temperature is zero. A nonzero temperature, however, gives the Markov chain the opportunity to escape from local minima. We will refer to a Markov process evolving in this manner as a Metropolis process. Simulated annealing is a special metropolis process in which $T_k \rightarrow 0$ slowly, so that the state of the Markov chain converges in probability to the collection of states with globally minimum cost[33]. In practice, since we can keep track of the best state visited so far, we are more interested in having ever reached a minimum than reaching it and staying there.

As mentioned earlier, we have a two level Metropolis process, corresponding to the two components of schedule generation. The Markov chain $< X_k >_{k=0}^{\infty}$ has $\mathcal{S}_{new}$ as its state space. We call $< X_k >_{k=0}^{\infty}$ the *high level* Metropolis process, and we define a so-called *low level* process to generate the random variable X_{k+1}^{trial}. We generate the initial state S_0^k, for the low level Metropolis process $< S_i^k >_{i=0}^{N}$, by using a different process flow assignment, but the same permutation of $\mathcal{OI}$ as X_k. The subsequent states of the Markov process, $< S_i^k >_{i=1}^{N}$ are generated using the same flow assignment but the permutation of $\mathcal{OI}$ changes.

A simple explanation on how the process evolves is as follows: Generating the trial schedule, S_{k+1}^{trial} simply involves perturbing S_k. As discussed in Section 2.3, each schedule can be represented by a global operation sequence. To perturb the schedule, we must permute the corresponding sequence. We do this by randomly selecting a short subsequence and a new position in the sequence which remains. Then the subsequence is reversed and moved to the new position.

The resulting global sequence may not satisfy all batch precedence constraints. To resolve this potential violation, each offending operation is swapped

with the (unique) operation in the same batch which does not violate the batch precedence constraint. Finally, when all batch precedence constraints are satisfied, the start and finish times of all operations are computed using the rule that each operation begins as soon as its predecessors have completed (with an appropriate adjustment for setup time).

This heuristic method of generating trial schedules is derived, in an intuitive sense, from the actual practice of manual schedule generation. The idea of swapping operation, and then comparing the two resultant schedules, is consistent with the actual scheduling. There the scheduler is using his experience to evaluate and compare the two schedules, here the qualitative performance measure is being used for comparison. Hence, this adhoc procedure of *neighbourhood* definition provides good "trial" schedules.

The start and finish times are used to compute costs of tardiness, work-in-process, and finish goods storage. Based on the cost differential, a decision is made to accept or reject the trial state. Finally, after N such steps, we set $X_{k+1}^{trial} = S_N^k$. This method is shown to converge to the optimum in [34]. We stop the iterative procedure after a certain number of iterations or when cost stops improving, which results in a schedule which is good, though sub-optimal.

The initial schedule generation is done on the basis of addition of new batches to an empty schedule. The *operation instances* of the new batches are inserted in the schedule for minimum cost change. The complete algorithm is as follows:

```
algorithm INSERT_NEW_BATCH

    while (some operation in new batch is not in schedule)

       OP = next unscheduled operation in new batch;

       if (OP can be inserted without
                increasing the schedule cost)
          put OP in schedule at zero cost;
       else
          put OP in schedule at minimum cost;
       endif

    end_while

end_algorithm INSERT_NEW_BATCH
```

2.6 $\mathcal{ABES}$ Features

The descriptive language and factory model developed in the previous sections were implemented in a software package named $\mathcal{ABES}$ (*Annealing Based Ex-*

periment in Scheduling). This software was installed at the Texas Instruments PC board manufacturing facility in Johnson City, TN and was used to schedule the surface mount machines of the Custom Manufacturing Unit of that facility. The software, along with the underlying theoretical basis, is not specific to the Texas Instruments manufacturing facility. As mentioned earlier, the modeling and scheduling concepts hold true for any general flexible manufacturing unit. Hence, this software can be adapted, with minor modifications, to define and schedule different manufacturing systems.

The software package $\mathcal{ABES}$ is written in C and runs on a desktop computer. A menu driven user interface leads the user through a sequence of instructions for optimal schedule generation. Before a schedule is generated the principal process flow (*device*[1] *operations* $< O_k >_{k=1}^{N}$) for each device has to be defined in the system. The user can also define alternate process flows, which are combinations of the *available methods* for different operations in the principal process flow. A graphical interface allows the user to add devices, remove devices or to change device processing parameters. The concept of *method families* has also been implemented in this package. The user can define the *method family* structure along with device parameters in the initial setup. During optimization the swapping of operations does lead to grouping of like devices on the same machine.

Once all devices have been defined in the system the user can generate a schedule by adding batches (lots) for particular devices. The program generates an initial schedule, X_0, on the basis of the addition of new batches to the system. For each new batch the user is prompted for the following attributes, namely, earliest start time, due time and a priority level. The *operation instances* for the new batch are inserted in the machine-idle slots in the schedule for an earliest possible finish time. This along with the delayed concurrency leads to an efficient utilization of the machines.

The trial schedules $\{X_k^{trial}\}_{k=1}^{\infty}$ are generated either by swapping adjacent *operation instances* on a single machine or by selecting a particular batch at random and interchanging each *operation instance* of the batch with the succeeding *operation instance* on the same machine. At each iteration, the choice of method for trial schedule generation is random. A trial schedule is accepted or rejected with certain probability (as defined in Section 2.5). The acceptance probability is a function of the "annealing temperature". We use an adaptive temperature scheduling scheme in the implementation. The new temperature at each step is calculated as a function of the differential cost average over the previous iterations. This sequence of schedules constitute the so called "low level" Metropolis process. The trial schedules for the "high level" Metropolis process are generated after every 100 iterations of the low level process. The new schedule is accepted or rejected with a certain probability defined in terms

[1]Device is the nomenclature used, at the TI plant, to define a board. and is equivalent to the term "part" in our framework.

of the cost differential from the previous "high level" Metropolis iteration. Here also we use an adaptive temperature scheduling scheme, although with a different temperature variable. Thus effectively we have two temperature variables for the two level Metropolis process. Figure 2.4 shows an example run of the simulated annealing based optimization process. The sudden increase, and the subsequent drop, in the temperature is manually introduced, so that the process could escape out of a local optimum.

Each batch in the schedule has a cost associated to it. This batch cost is calculated by adding up the three cost components, namely Work in process (WIP), Tardiness (TARD) and the Inventory (INV) cost and are calculated using the following formulas:

(WIP) (FinishTime - StartTime)$*(\mathbf{a} + \mathbf{b}*$BatchQty)

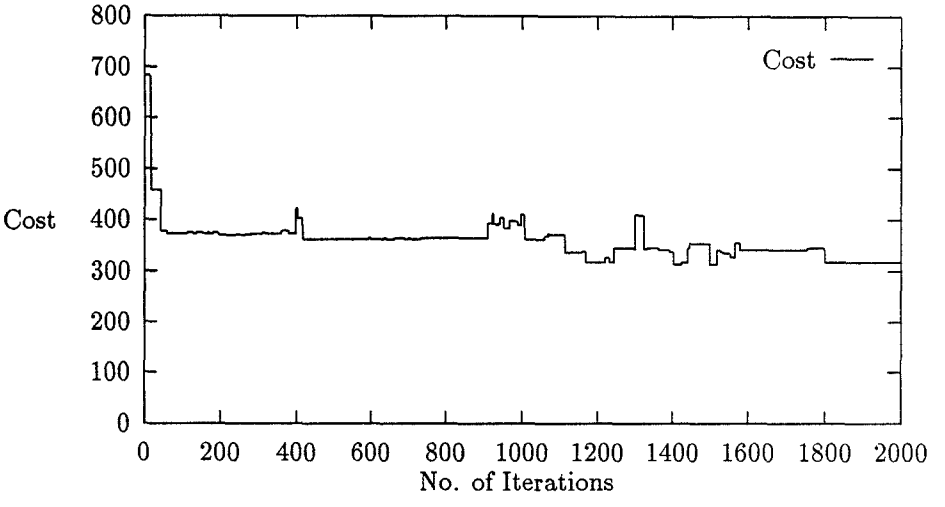
(TARD) $(\mathbf{a} + \mathbf{b}*\mathrm{T}+\mathbf{c}*\mathrm{T}^2) * (\mathbf{d} + \mathbf{e}*$BatchQty)
Here T = max(0, FinishTime - DueTime)

(INV) $(\mathbf{a} + \mathbf{b}*\mathrm{E}+\mathbf{c}*\mathrm{E}^2) * (\mathbf{d} + \mathbf{e}*$BatchQty)
Here E = max(0, DueTime - FinishTime)

The user has different priority levels he can assign to the new batches being added to the schedule. Each priority level is defined by the user by assigning different set of values to the parameters (a, b, c, d and e) for each of the cost components. The total schedule cost, defined as the sum of the costs associated to each batch in the schedule, is used as the optimization criterion. The user has the option to enable or disable the three components globally for the purpose of evaluation of the total cost.

Since this is not an online real-time scheduler, each generated schedule has an active time associated to it. This active time represents the time origin for operation scheduling. The user has the option of specifying the active time when generating a new schedule. Once an optimal schedule has been generated, the user can update the factory status by changing this active time and also update the operation status for the different machines. Also new batches can be added to the schedule as the active time changes. The optimization process is carried out after each such update.

Once an optimal schedule has been generated the user can view or print the schedule. Fig 2.5 shows a sample device schedule generated using $\mathcal{ABES}$. It displays the schedule for the various batches of the same device. Fig 2.6 displays the various batches scheduled on a particular machine. The user can view the machine utilization on a GANTT chart. Another graphical display depicts the Work-in-Process(WIP). A detailed description of $\mathcal{ABES}$ is available in [12]. In terms of the time involved for scheduling, it takes a trained user about 10 minutes to generate the initial schedule. For a typical work load of



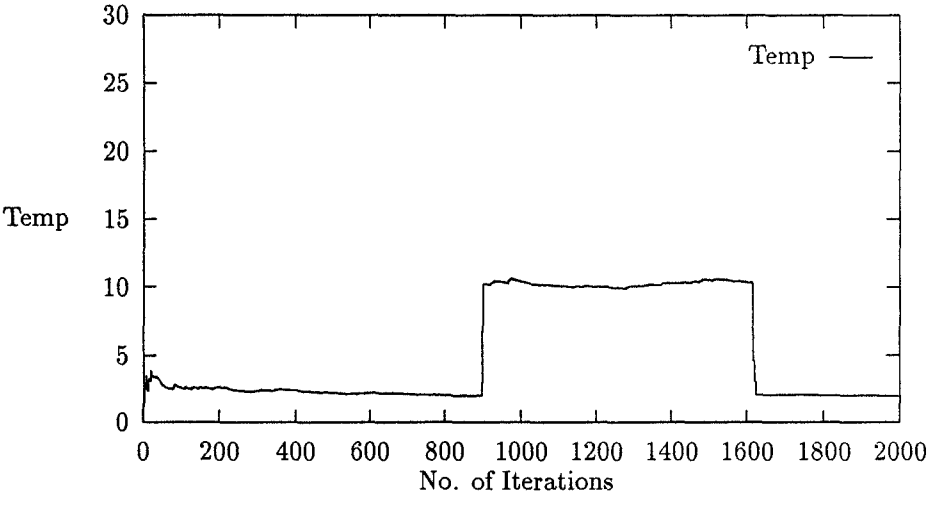


Figure 2.4: Optimization process

```
**************************************************************************
************************** PART  1  ******************************
**************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Batch ID: 001
 0380   SMT 4        24    800    18:00:00 (08/24)    20:40:00 (08/27)
                                                DUE:  23:00:00 (08/24)
Batch ID: 007
 0256      (parts)    0   1500                        18:00:00 (08/24)
 0257   SMT 1       150   1500    21:44:43 (08/31)    09:27:34 (09/04)
 0262   SMT 4        24   1500    22:00:00 (09/02)    00:00:00 (09/05)
                                                DUE:  23:00:00 (08/25)
Batch ID: 006
 0251      (parts)    0   1000                        10:00:00 (08/25)
 0252   SMT 1       150   1000    08:27:43 (08/28)    08:16:17 (08/29)
 0253   SMT 5        24   1000    04:11:28 (08/29)    13:31:28 (08/30)
                                                DUE:  23:00:00 (08/29)
```

Figure 2.5: Schedule by Device

about 30 batches, the optimization procedure takes about 4 minutes for 5000 iterations on a 486 33MHz PC. Therefore it takes about 15 minutes to generate a sub-optimal schedule. If the optimization procedure is stopped midway, the best schedule obtained till that point is displayed. Thus a sub-optimal schedule is available at each iteration and the user can stop the optimization at any point to get the best schedule found.

2.7 Concluding Remarks

In this chapter, we presented a deterministic manufacturing system model. The operation based manufacturing system model was used to represent the system schedule as a single sequence of operations. This representation was useful in the development of simulated annealing based optimization algorithm and also the subsequent implementation as a software package. The development of this software was helpful in better understanding of the manufacturing system modeling issues. This also provided insight into the control problems/issues in manufacturing systems. This led to the development of control algorithms which will be discussed in the following chapters.

Though the results, presented in this chapter, were based on TI assembly unit, the methodology (and the software) can be easily used for other manufacturing systems. The combination of heuristics and analytical techniques provided a reasonable compromise between optimal, intractable solution methods and sub-optimal, expert systems approach for solving this difficult problem. It is difficult to evaluate our schedules in "monetary" terms, the real evaluation

```
*************************************************************************
************************** MACH: SMT 2     ******************************
*************************************************************************
Operation Job  Setup  Batch  Batch    Start              Finish
   ID          Time    ID     Qty     Time               Time
*************************************************************************
  0314  Part 6  180     017     422    18:00:00 (08/24)   00:36:53 (08/27)
  0377  Part 2  120     029     254    02:36:53 (08/27)   06:17:37 (08/27)
  0392  Part 5  360     002      63    08:05:37 (08/27)   17:05:37 (08/27) L
        DUE:23:00:00 (08/25)
  0337  Part 4  240     022     200    21:05:37 (08/27)   02:00:51 (08/28)
  0374  Part 7  210     010    1000    05:30:51 (08/28)   14:50:51 (08/29) L
        DUE:23:00:00 (08/27)
  0341  Part 4  240     023      84    18:50:51 (08/29)   20:54:51 (08/29)
  0290  Part 6  180     012     800    23:54:51 (08/29)   08:17:43 (08/31)
  0345  Part 7  240     024     119    12:17:43 (08/31)   15:13:23 (08/31)
  0272  Part 3  210     009     800    08:49:03 (09/03)   11:29:03 (09/04) L
        DUE:23:00:00 (08/31)
```

Figure 2.6: Machine Schedule

is possible only by the actual user. The increase in throughput, reduced inventory levels, and reduced tardiness, as quantified by cost functional can be used as a good evaluation criteria.

CHAPTER
THREE

Hierarchical Reconfigurable Control

In the previous chapter, we developed a mathematical model for description of a general manufacturing system. This model was used to design an optimization based operations scheduling scheme for *planning* control of manufacturing systems. In this chapter, we introduce the inherent hierarchical structure in real manufacturing system planning. We present the reconfigurable control problem, and the different reasons for the choice of a hierarchical control structure as a solution methodology. We present the various control parameters of interest and propose a three level hierarchical scheme for reconfigurable control of manufacturing systems. In the end, we present a hybrid, i.e., with both continuous and batch mode processing, manufacturing system example. This will be our test case for implementation of the different algorithms and will be used to illustrate the associated control results.

3.1 Hierarchical Scheme as a candidate for Reconfigurable Control

Hierarchical schemes decompose a global problem[1] into sub-problems which can be either solved sequentially or in parallel. The essential idea of a hierarchical decomposition is as follows: The system behavior is described by a family of models. Each model is used for specification of the system behavior as viewed from a specific level of abstraction. The model at each level has an associated set of attributes, constraints and control parameters. The attributes are predefined on the basis of the desired model complexity at each level. The constraints of a specific level are determined on the basis of the attributes of adjacent levels. The effect of the control variables, associated to each level of abstraction, is confined to that particular level only. The control variables are assumed to be constant for all other levels, but can, and in general do, influence the attribute specifications for other levels.

The linkage between the different model levels is done by appropriate aggregation/disaggregation of the different system parameters. Due to the decomposition of the system model into detailed subsystem models, the lower level

[1]Here we are introducing a general hierarchical scheme which can be used for solving different decision making, optimization or control problems

models provide a better description of system behavior. The inter-subsystem interaction is taken into account by the higher level models. Hence, higher level models provide an aggregated, more general, global description of the system behavior and do not require detailed input information. This allows for reduced problem dimensionality and also allows for a longer "look ahead" time horizon for problem specification. The main advantage of using such hierarchical structure is reduced complexity, and hence, easier computational tractability for complete problem and solution specification.

To introduce the notion of hierarchical control schemes in manufacturing systems, let us consider a production management system for a steel company (Figure 3.1, [13]). The hierarchical planning uses different degrees of model complexity at the different levels. If we just take the aggregated version of our example, and divide the production management hierarchy into three levels (Figure 3.2): 1) Production Control, 2) Production Scheduling, and 3) Company Planning. At the topmost level, Level 3, based on long term goals, production rates for different product families are determined. The production schedule is defined on the basis of demand forecasts, and the expected production rates, over the time horizon h_3 ($\simeq$ years). These rates act as constraints for the subsequent model at Level 2. The production rate of a part family, from Level 3, is then used to calculate the specific production rates, for each constituent part in that particular part family, over the time horizon $h_2(\simeq$ months). The grouping of different parts into part families is the aggregation step required to define model m_3 from model m_2. If machines have been grouped to form machine cells, then the schedule at Level 3 also determines the operational schedule for the machine cells in Level 2. In a similar fashion, the production schedule at Level 2 is then used to schedule specific operations on the individual machines at the shop floor level, i.e. Level 1. The time horizon in this case is $h_1(\simeq$ hrs).

Keeping in mind this discussion on the inherent hierarchical structure in manufacturing system planning and the principles of hierarchical control schemes, let us redirect our attention to reconfigurable control in manufacturing systems. In the remainder of this section, we introduce the concept of *reconfigurable control* and the reasons for choice of hierarchical control scheme.

Flexible manufacturing systems are subject to various exogenous (parts unavailability, order arrivals, etc.) and endogenous (machine failures, operator unavailability, buffer blocking or starvations, etc.) random events. These random events can disrupt the normal operation of the manufacturing system for a certain period of time. The disturbance induced transient times are dependent on the corrective action time for that particular disturbance. For example, if a machine failure takes place then the time for the manufacturing system to return to normal operation is directly proportional to the repair time, which in turn is determined by the failure type. The basic premise of reconfigurable control of the manufacturing system, as the name suggests, is to **reconfigure the manufacturing system model such that the effect of the distur-**

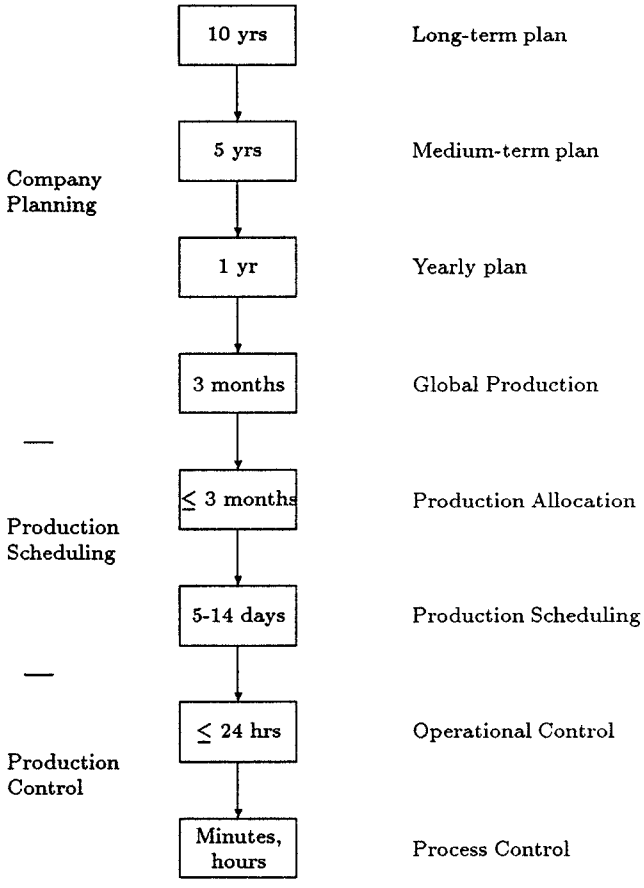


Figure 3.1: An example hierarchy for manufacturing systems

bance, quantified by some cost, is minimal. The reconfiguration can be in terms of size increase (addition of new machines/part types, operators) or involve the same set of entities/parameters but a configured differently. This will become more clear in the next section where we consider different reconfigurable parameters in greater detail.

Let us try to understand what we mean by minimization of the effect of the random disturbance. The feed-forward scheduling scheme presented earlier, assigns a machine to each operation in schedule and then sequences the operation (optimally) on the appropriate machines. The random disturbances/events cause the "actual" manufacturing system schedule to deviate from the predictive schedule generated by the scheduler. **The goal is to minimize this deviation.**

The various random disturbances/events affect the functioning of manufacturing system only at certain model levels. At the higher levels, the disturbances tend to average out, while at lower levels they get specified as constant parameters. To illustrate this point, let us consider our example of the steel factory.

Let us take the random event of a machine failure with an expected repair time of 10 hrs. Refering to figure 3.1, the company planners, with planning horizon of a few years, only take an aggregated (or averaged) machine failure data into account for long term target production definition. The actual production control (machine repair, processing of the present orders) is not of immediate concern at the different levels of company planning. At the production scheduling level, this random event again has a similar aggregated effect, i.e., the scheduler takes into account more accurate repair time data, for subsequent scheduling, but takes no immediate control action. For the operational control level, which gets the production schedule from the adjacent level, the immediate concern is to have continuous production and to meet order deadlines. Hence, the level at which the random event occurs, i.e. with the corresponding time horizon, is best suited for making the appropriate control decision. The lowest level of process control, just assumes that the machine is not available. It lets the operational level take the appropriate control action.

This feature of the absorption of random disturbance, as the model specification is aggregated from the lower level to a higher level, is another advantage of hierarchical control schemes. The ordering of random events, at different hierarchical levels, reduces the control definition complexity at the associated levels. Due to the natural decomposition of the reconfigurable control problem, on the lines of the natural hierarchical planning structure, into different control levels and the aforementioned advantages of hierarchical control schemes, we propose a hierarchical structure for the hierarchical reconfigurable control problem.

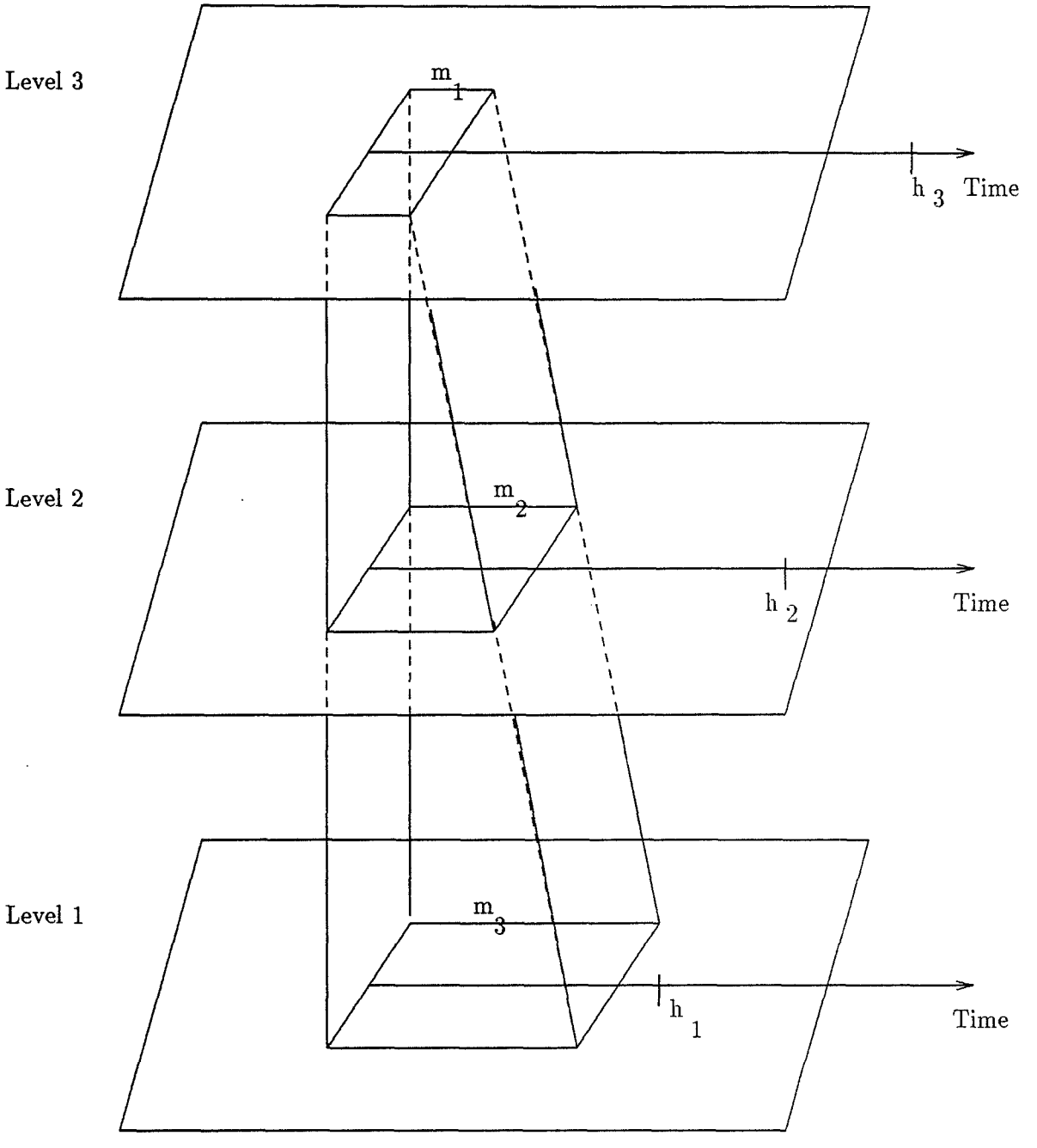


Figure 3.2: Hierarchical Production Management System

3.2 Reconfigurable Control Parameter Definition

Reconfigurable control of the manufacturing system entails reconfiguration of the manufacturing system parameters so that the effect of the random disturbance on the system functioning is minimized. On the basis of the model developed in Chapter 2, in this section, we identify the various reconfigurable control parameters. The corresponding levels for each control are defined using the concepts of time scale decomposition of the frequency of control application. The various control parameters which can be re-configured in a manufacturing system setup, along with their associated triggering events and reconfiguration scopes are discussed below. The presentation of the parameters is in decreasing order of the reconfiguration costs and in increasing order of frequency of control application.

Set of Machines: Depending on the machine utilization and the anticipated production rate over a time horizon, the controller can decide to add new machines to the manufacturing system. The replacement of old machines with new machines, reassignment of machines to different machine cells, etc., are some of the other reconfiguration possibilities. Reconfiguration of this type has a high associated cost and hence control of this control type should be less frequent.

Machine Buffer Allocation: This adds extra buffers on selected machines and can be directly translated to higher Work-In-Process for the manufacturing system. Higher Work-in-Process is helpful in meeting production deadlines, but at the cost of higher buffer costs and unfinished parts inventory. This type of control should be used infrequently due to the high reconfiguration costs.

Scheduling Cost Structure: As discussed in the previous chapter, each schedule has an associated cost. This cost is calculated on basis of some cost structure, which can be reconfigured in time. The change in the cost structure can be used to adapt to the evolution of long-term production policies and goals at the strategic level. In our framework, possible reconfiguration may incorporate changing the cost parameters, i.e. the weights in the weighted average cost.

Process Flows: If a machine fails, it may be possible to reconfigure the process flows of the various orders or batches of different part types which require processing at that particular machine. The essential idea is that the flow assignment for the batches can be changed. Since this reconfiguration does not involve any new installations, the reconfiguration cost involved is low, and this control option may be used frequently.

Operation Parameters: This reconfiguration, in general, will be the least expensive and hence most frequent. The machine processing parameters,

namely, processing time per part and setup time per batch can be changed[2] The desired production schedule, as determined by the controller, can be adhered to by proper adaptation of the above mentioned operation parameters.

3.2.1 Hierarchical Reconfigurable Control Levels

Based on the associated characteristics and reconfiguration costs, the reconfigurable (control) parameters can be grouped into three levels. The three level hierarchy, as mentioned earlier, is based on the time decomposition of the frequency of control. The associated time horizons for control specification are proportional to the corresponding time-periods. Figure 3.3 depicts the three levels along with the different attributes.

I. System Level Control: The system level control incorporates managerial decisions taken on the basis of long term plan development, anticipated production contracts, marketing strategies etc. The first three parameters discussed above, namely, machine set, buffer allocation, and cost structure, can be assigned to this level. The decisions at this (corporate planning) level, have high reconfiguration costs. The long term projected gains, based on the anticipated production plan over the next few months (or few years in some cases), must offset the present investment by some predefined margin. Thus the time horizon for system level control specification, i.e. manufacturing system configuration, is of the order of months.

II. Process Level Control: Based on the control specification at the system control level, the manufacturing system configuration is complete. The actual physical orders or batches, for different part types, are then used to schedule the different machines of the manufacturing system. The scheduler, human or computer, is used to assign process flows to each batch. This machine assignment for each operation is used to generate individual machine schedules. The process level reconfiguration incorporates the re-assignment of the scheduled operations, of the failed machine, on different machines. The reconfiguration cost, in this case, can be accounted in terms of the cost associated to the following factors: possible delays in the originally scheduled operations, reconfiguration of personnel, reconfiguration of buffers etc. In comparison to the system level reconfiguration cost, this cost is substantially less, and hence this type of control can be used more frequently. Depending on the MTBF (mean time between failures) for the different constituent machines, the control time horizon in this case can be of the order of few days or upto a week,

[2]For the TI assembly unit, the processing rate can be changed by changing the speed of the "placement" arm. The setup times can be reduced by assigning extra operators for setup.

III. Operation Level Control: Once individual operations have been scheduled on each machine, the start and finish times for each operation have been assigned. The cumulative effect of the deviation of the actual operation timings from the scheduled timings can be minimized by proper re-assignment of the processing parameters of each machine. We can continuously change the processing times, so that the actual operation finish time is as close to the scheduled finish time. This will ensure minimal effect on the subsequently scheduled operations on the same machine. The time horizon for this operation level control is of the order of a typical operation duration, i.e. few hours to a day. The associated reconfiguration costs are negligible in comparison to the other two levels.

In this chapter, we presented a hierarchical reconfigurable control scheme for manufacturing systems. The reconfiguration control parameters are based on the manufacturing system model presented in Chapter 2. The planning or feedforward scheduling control, discussed in the same chapter, plays an important role in this hierarchical control framework. The system level control involves strategic management at the corporate planning level and was not considered in this research. The emphasis in this work was on the micro level control which could be implemented in a shop floor of a real manufacturing system.

As mentioned earlier, reconfigurable control of the manufacturing system entails reconfiguration of the manufacturing system such that the effect of the random disturbance is minimized. The minimization of the effect of the disturbance was shown to be equivalent to the minimization of the deviation of the actual manufacturing system schedule, from the predictive schedule.

Let us investigate what this means in terms of the control parameters and their corresponding level definitions. At the Process Level, this reconfiguration control is reassignment of process flows (or equivalently reassignment of operations to a different machine) and resequencing of machine operations, so that the overall schedule deviation is minimized. Similarly, at the Operation Level, we can control the processing rates/times to minimize the overall schedule deviation. different cost functionals, As the deviation is measured in terms of different parameters at the two levels, a different cost functional has to be specified for each level. The hierarchical structure of the control problem allows us to tackle the problem at each individual level. Hence, the two problems may appear to be independent, but a strong inter-dependence exists due to the hierarchical decomposition of the global control problem.

3.3 Manufacturing System Model for Algorithm Application

In this section, we present a hybrid manufacturing system model, i.e., a model based on both continuous and batch mode processing. The manufacturing system model is again based on the Custom Manufacturing Unit at Texas Instru-

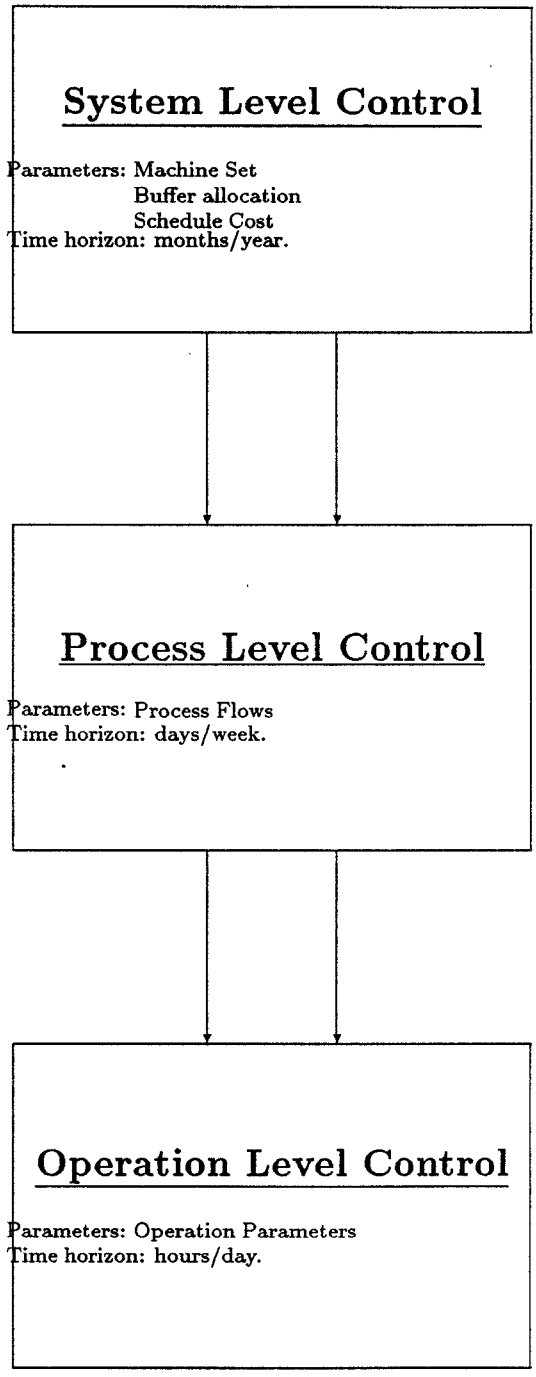


Figure 3.3: Hierarchical Reconfigurable Control Structure.

ments, Inc., Johnson City, but has been suitably modified to incorporate the complete process flow. This test model will be used to present results on actual implementation of the reconfigurable control algorithms.

Figure 3.4 depicts the complete hybrid manufacturing system model. The model consists of 7 machines and is configured to process 4 different part types. The 7 machines with their corresponding descriptive, functional titles are as follows:

M/C-1: IC Placement.

M/C-2: Axial Insert.

M/C-3: Axial Sequence Insert.

M/C-4: Radial Insert.

M/C-5: Manual Test Station.

M/C-6: Burn-In Station.

M/C-7: Final Test Station.

M/C-1 is models the surface mount machines discussed in Chapter 2, while the next three machines, namely, Axial Insert, Axial Sequence Insert, and Radial Insert, are part of the process flow as shown in Figure 2.1. The last three stations, namely Manual test, Burn-in and Final test, are part of the quality control unit. These stations have parallel work-stations where many parts can be processed at any time instant. The test sequences for different part types are very similar and hence these machines (in the case of Manual Test station, human operators) can be assumed to be operational in a continuous processing mode.

For our analysis we will assume that the insert machines, namely, machines M/C-1 through M/C-4, operate in a batch mode and the various operations are scheduled accordingly on these machines. On completion of the last insert type operation, the batches are just queued into the buffer of the first test station, namely M/C-5. The arrivals into this queue can then be assumed to be Poisson arrivals, if the processing times are exponentially distributed at the last insert operation. The various processing times and set-up times, at each of the machines, are adapted from actual data from the manufacturing unit.

Machines M/C-2 and M/C-3 are both Axial Insert machines, only difference being the mode of insertion of components. In the Axial Sequence Insert (M/C-3) the parts are pre-sequenced on a single part magazine and parts are sequentially accessed and inserted. In the case of M/C-2 the individual parts are available through a battery of part magazines and the part holder accesses each of the magazines for the respective parts. Hence, in principal these machines can

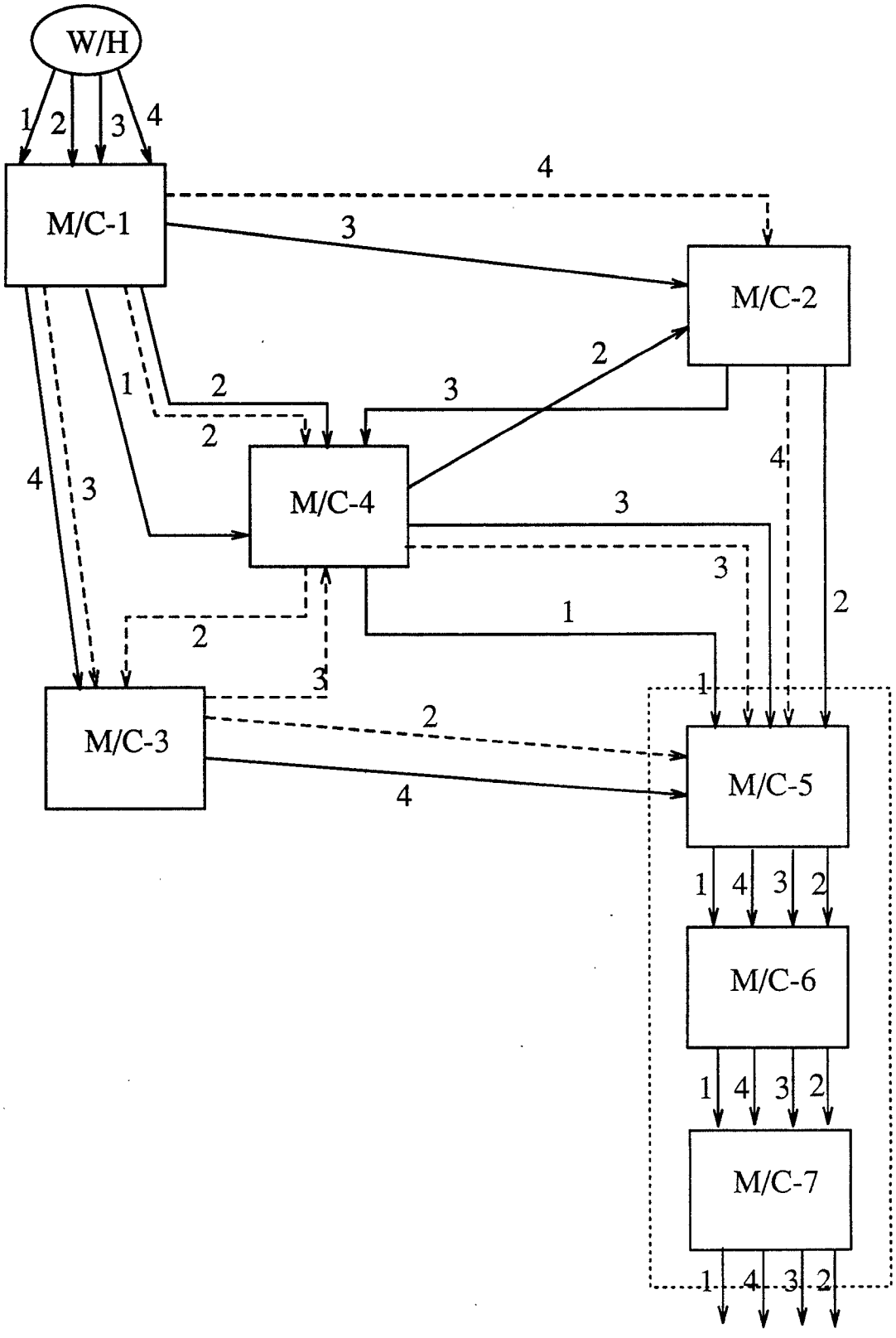


Figure 3.4: Hybrid Manufacturing System

be used interchangeably with minor reconfiguration. This allows for the specification of alternate flows for each of the part types which are processed through either of the machines. The alternate flows for each part types are depicted by dashed lines in Figure 3.4.

In Chapters 4 and 5, we present reconfigurable control algorithms for the Process Level and Operation Level control. Based on the development of our scheduling scheme, our Process Level control is only applicable for the manufacturing subsytem operating in batch mode, i.e. machines 1 through 4. The test stations, machines 5,6 and 7, are controlled only at the Operation Level. In chapter 5, we present a reactive operations scheduling scheme for reconfigurable process level control. For Operation Level control, we present in chapter 6 an algorithm each for continuous and batch mode processing. The manufacturing system model presented above will be used to illustrate the algorithm results.

CHAPTER
FOUR

Process Level Control

On the basis of the hierarchical reconfigurable control scheme presented in the previous chapter, the manufacturing system configuration for the Process Level, is defined by the System Level control. When new job orders or batches arrive in the system, the manufacturing system schedule is generated on the basis of: i) assignment of process flow to each batch, and ii) sequencing of operations on assigned machines. The reconfigurable control at the Process Level involves process flow re-assignment and the resequencing of operations on different machines. In this chapter, we present a reactive operations scheduling scheme for Process Level control of manufacturing systems. We use the predictive scheduler presented in Chapter 2, in conjunction with the manufacturing system model in the feed-forward path. The reconfiguration controller in the feedback path completes the control loop for online control of the manufacturing system. In section 4.1, we present details of the manufacturing system simulation model and present the basic reactive operations scheduling scheme. In section 4.2 we present the algorithm for reactive scheduling based process control. The two main components of the control algorithm, namely, machine re-assignment and operation re-sequencing are discussed in detail. To conclude the chapter, we present algorithm implementation results in section 4.3.

4.1 Reactive Scheduler

The block diagram for reactive scheduling scheme is shown in Figure 4.1. The desired optimal schedule S, generated by the scheduler, and the actual schedule from the model $\widehat{S}$, are compared by the controller. The deviation between the two schedules is measured on the basis of some cost criteria and a rule-based, threshold triggered scheme is used for rescheduling. At each rescheduling instant new jobs can be added to the schedule; this leads to on-line reactive scheduling of the manufacturing system. In chapter 2, we presented details of the predictive scheduler. In this section, we will present details of the remaining two components in the closed loop and describe the closed loop control algorithm.

We have used the software package QNAP (Queuing Network Analysis Package) to develop a manufacturing system simulation model. Schedule S is input to the manufacturing system simulation model. (In the online system, the sim-

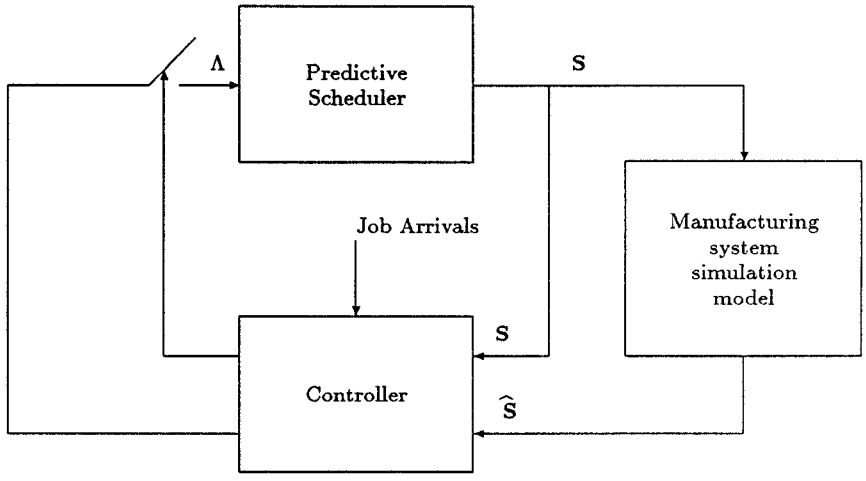


Figure 4.1: Block Diagram for Reactive Operations Scheduling

ulation model would be replaced by the actual factory.) The simulator has a central controller which keeps track of the batch operation precedence and also the machine ordering. The machines are modeled as single server queues and the batches are routed according to the pre-defined process flows. The simulator incorporates the various random manufacturing system features namely, short term/long term failures, random processing/setup times etc. The optimal schedules generated by the scheduler are used as an input to the simulator. The sequencing in the predictive schedule S is preserved in the simulation. However, the actual operation times, which differ from the scheduled time because of random processing and setup times, define the actual schedule $\widehat{S}$.

In our simulation model, we allow operation times to be random (exponentially distributed). The machines are modeled as single server queues. We allow the machines to be either single physical machines or virtual machines (sequence of physical machines processing parts in tandem). Each machine has three different modes of failure. Failure Mode 1 (FM1) allows for random repair time and is more frequent than FM2 or FM3. FM2 and FM3 represent more drastic

failures and have larger deterministic repair times. The difference between the three failure modes, is the severity of failure and the probability of occurrences. The severity of the failure can be translated into increase in repair times. For our simulation model, we let the repair times, for FM2 and FM3, to be 24 and 48 hrs respectively. The repair time for FM1 is a normal random variable, with mean 1 hr. The probability of failure for each mode is incremental with respect to number of operations processed since the previous failure. This is consistent with the machine failure probabilities in an actual manufacturing system.

As discussed in Chapter 2, each schedule has an *active time* associated to it. The rescheduling algorithm is based on a rule based, threshold triggered update of the active time and the manufacturing system status. In the remainder of this section, we will present the details of the different update rules and the rescheduling loop shown in Figure 4.1.

The predictive schedule S is used as an input to the manufacturing system simulation model. Since the batch order precedence is trivially maintained in the simulation, for notational convenience we omit the batch precedence ordering and use o_i^m to represent the ith operation to be performed on machine m. In effect, we are re-labeling our operations on each machine and will carry out the analysis on the basis of individual machine schedules, i.e the machine order precedence, instead of the complete manufacturing system schedule. As defined earlier, $t_s(o_i^m)$ and $t_f(o_i^m)$ respectively, denote the start and the finish times for the operation o_i^m in the predictive schedule S. Now the schedule for machine m can be defined as:

$$S_m \stackrel{\Delta}{=} < o_k^m >_{k=1}^{N_m}; \quad t_s(o_k^m) \; < t_f(o_k^m) \leq \; t_s(o_{k+1}^m)$$

Then the overall schedule can be written as

$$S = < \; S_m >_{m=1}^{M}$$

In the following we use t to represent "real time", that is, the position in the actual schedule, $\widehat{S}$. For each machine, there is a point in the predictive schedule corresponding to t in $\widehat{S}$. This point is the "active time" corresponding to real time t, which we write $\tau_m(t)$. Precisely, suppose operation o^m is in process at time t in the actual schedule, $\widehat{S}$, and suppose it is expected to complete at time $t + \delta$. Then the active time for machine m is given by

$$\tau_m(t) = t_f(o^m) - \delta.$$

Active times are used by the controller as indicated below.

Initially the active times on all machines are equal to the real time, say $\tau_m(t_0) = t = t_0$. Because of the various random features of the simulation, the two schedules differ in terms of the operation timings. To correlate the two schedules, at real time t ($t_0 \leq t$), we can define the active time vector $\Lambda(t) = [\tau_1(t), \tau_2(t) \ldots \tau_M(t)]^T$ ($\Lambda(0) = [t_0, t_0 \ldots t_0]$). $\Lambda(t)$ represents the present

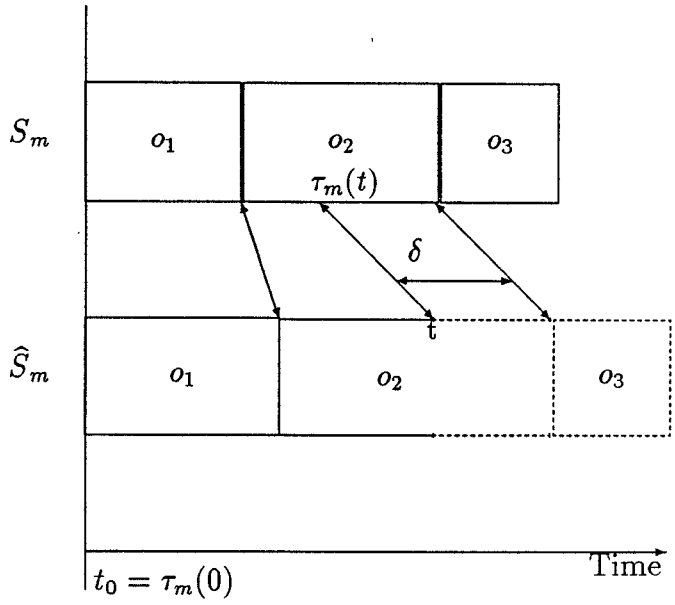


Figure 4.2: Active Times for a single machine

status of $\widehat{S}$ in S. For illustration, Fig 4.2 shows the two time scales for a single machine schedule.

For each operation o_m^i, we associate a cost $c_m^i = c(t_s(o_m^i), t_f(o_m^i), \widehat{t}_s(o_m^i), \widehat{t}_f(o_m^i))$. This cost represents the penalty involved for the actual operation times to differ from the scheduled times. Based on the individual operation costs, the net schedule cost can be calculated as

$$C_r(S, \widehat{S}, t_0, t) = \sum_{m=1}^{M} \sum_{\substack{i=1 \\ t_0 \leq \widehat{t}_f(o_m^i) \leq t}}^{N_m} c_m^i$$

The controller keeps track of the two schedules S and $\widehat{S}$ and at the end of the each operation updates the net schedule cost $C_r(S, \widehat{S}, t_0, t)$. The rescheduling loop is triggered under the following conditions:

- $C_r(S,\ \widehat{S},\ t_0, t) > C_T$. Here C_T is some predefined cost threshold.
- Failure Mode 2 on any machine
- Failure Mode 3 on any machine
- New Job arrivals

At each such rescheduling instant t, the controller polls in the current status $(\Lambda(t))$ of the manufacturing system schedule. This status is then input to the scheduler. If the rescheduling is necessary due to failure 2 or 3 on any machine,

an extra input in the form of the operation o_f^m along with its associated start and finish times is sent to the scheduler.

The Scheduler uses the above information to update the present schedule S. Past operations are discarded, and the current real time t becomes the new initial time $t_0 \leftarrow t$. Hence, $\tau_m(t) \leftarrow t$ also. The scheduler then allows the user to make changes (add/remove batches or operations) in the schedule S[1]. Next the simulated annealing based optimization is activated to optimize the remainder of the schedule. Since the manufacturing model is continuously simulating the schedule S, the optimization is not done for the immediately scheduled operations. Basically we do not reschedule the operations scheduled to start between t_0 and $t_0 + T$, for some disable period $T > 0$. This allows for on-line optimization of the current schedule while the simulation is still running. (We do not literally run the simulation and the controller concurrently; this is a consideration for online implementation.) Once an optimal schedule is obtained it is input to the simulation model. In case of failure mode rescheduling, the failure operation o_f^m is treated as an operation with fixed processing time. Since in this case $t_s(o_f^m) = t_0$ the operation is not rescheduled and it is simulated as a repair time. Taking the schedule shown in Fig 4.2, if a failure FM2 or FM3 occurs at time t on machine m, then the two schedules S and $\widehat{S}$ after rescheduling are depicted in Fig 4.3. This closed loop process is carried out continuously and leads to reactive operations scheduling for manufacturing system. The complete process can be represented in a flow chart as shown in Fig 4.4.

4.2 Reactive Scheduling based Process Control

The two level Metropolis process formulation of the optimization process for scheduling was discussed in Chapter 2. The embedded cost structure and the closed loop configuration of the reactive operations scheduling scheme gives us the ideal framework for the design and implementation of a central reconfiguration controller for the process level control. The rescheduling loop was triggered under the three distinct conditions, namely, cost threshold crossing, machine failures, and new job arrivals. We are interested in the reconfiguration of the process flows at each such re-scheduling loop activated due to the first two types of triggering modes. The addition of a new job arrival into the existing schedule, as mentioned earlier, is based on insertion of operation in the existing machine idle slots for a minimum cost increase.

Recalling the two level Metropolis process representation of the simulated annealing based scheduler, the *high level* process $< X_k >_{k=0}^{\infty}$ was generated on the basis of the *low level* process $< S_i^k >_{i=0}^{N}$. In case of the reactive scheduling based process control, the *high level* Metropolis process is modified to take into account the reconfiguration of process flows. The basic idea is to generate

[1]To maintain compatibility with the notation, the operations are re-numbered from 1 $\ldots N_m$ for each machine m

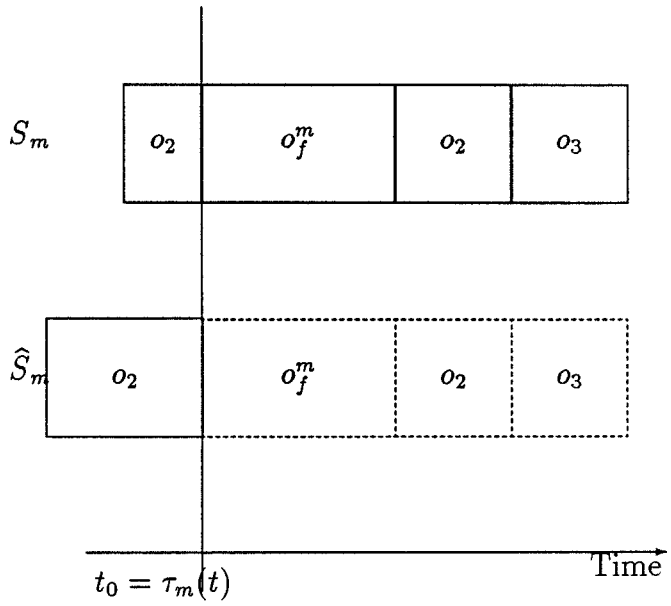


Figure 4.3: Schedule representation for machine failure induced rescheduling

the initial state S_0^{k+1} by randomly selecting a batch in the schedule X_{k+1} and changing its assigned flow. Since the new process is still Markov, the convergence results, discussed earlier, still hold. This modified two-level Metropolis process, then uses the built-in optimization process for optimal reconfiguration of process flows, and also optimal operation re-sequencing, in the manufacturing system.

The complete algorithm for the reactive scheduling based process control is as follows: At each rescheduling instant t, the time vector $\Lambda(t)$, which represents the manufacturing system status, is input to the controller. The additional information (if it is a failure triggered rescheduling) namely, dummy operations with their associated start and finish times, is also input to the scheduler. This manufacturing system status is then used to increment the active time, and hence the predictive schedule, by the scheduler. Using this updated schedule as the initial state, X_0, for the *high level* Metropolis process, the simulated annealing process is activated. The initial state S_0^k for the *low level* process, $< S_i^k >_{i=0}^N$, is generated using the following algorithms:

Initial state generation

algorithm $init\_state(S_{old})$
 $\star$ $b = Random\_Select(S_{old})$
  if $((\mathcal{F}_{part(b)} \setminus assign\_flow_{old}(b)) \neq \emptyset)$
   $assign\_flow_{new}(b) = f_i^{part(b)} \in (\mathcal{F}_{part(b)} \setminus assign\_flow_{old}(b))$

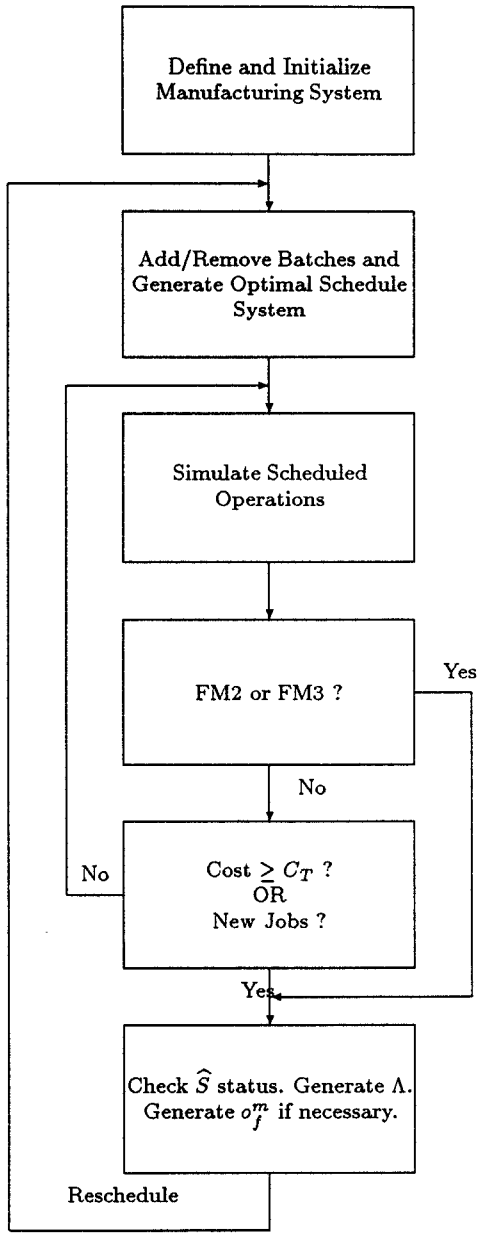


Figure 4.4: Rescheduling Algorithm Flow chart

else $b = Random\_Select(S_{old})$
go to $\star$
endif
return(S_{new})
end_algorithm

Random batch select

algorithm $Random\_Select(S)$
return($b_{random(1,|\mathcal{B}|)}$)
end_algorithm

Thus, using the initial schedule S_0^k, the *low level* Metropolis process is generated, for N steps, using the procedure discussed in Chapter 2. The trial schedule X_{k+1}, for the high level process is set equal to S_N^k. The high level temperature schedule is used to determine acceptance/rejection of the trial schedule. This process is continued for the two levels and since the scheduler keeps track of the best schedule generated in any time interval, a good sub-optimal reconfigured schedule is generated in finite number of iterations. This process is repeated at each re-scheduling instant and leads to on-line reconfigurable process control. The complete procedure is shown in an algorithmic form in Figure 4.5.

This algorithm relies on the low level Metropolis process to re-sequence operations on different machines, and on the high level Metropolis process for re-assignment of process flows. Hence both our process level control objectives are satisfied. In the remainder of this section, we present a heuristic subalgorithm, which can be inserted in the main process control algorithm. The idea was to use intuition from real manufacturing system control for re-assignment of operations on different machines. This addition to the above algorithm, is not necessary for our presentation, but fits in nicely with our objectives of process level control (see Figures 4.5 and 4.7). Hence, we present details of this scheme in the remainder of this section. The ad-hoc scheme, can be easily implemented in a centralized or a decentralized architecture.

As discussed in the previous section, the repair times for machines are represented as dummy operations on the individual machine schedules. The essential idea of this scheme is to reschedule the "displaced" operations, i.e., the operations which were scheduled on the failed machine during the machine repair time. It would be ideal if the displaced operations could be re-scheduled on alternate machines, with minimum (failure induced) delay for the operations scheduled in the future. The re-assignment of machines can be done using a centralized controller or in a decentralized manner by each of the failed machines.

In Figure 4.6, the first two sequences represent machine schedule for machine

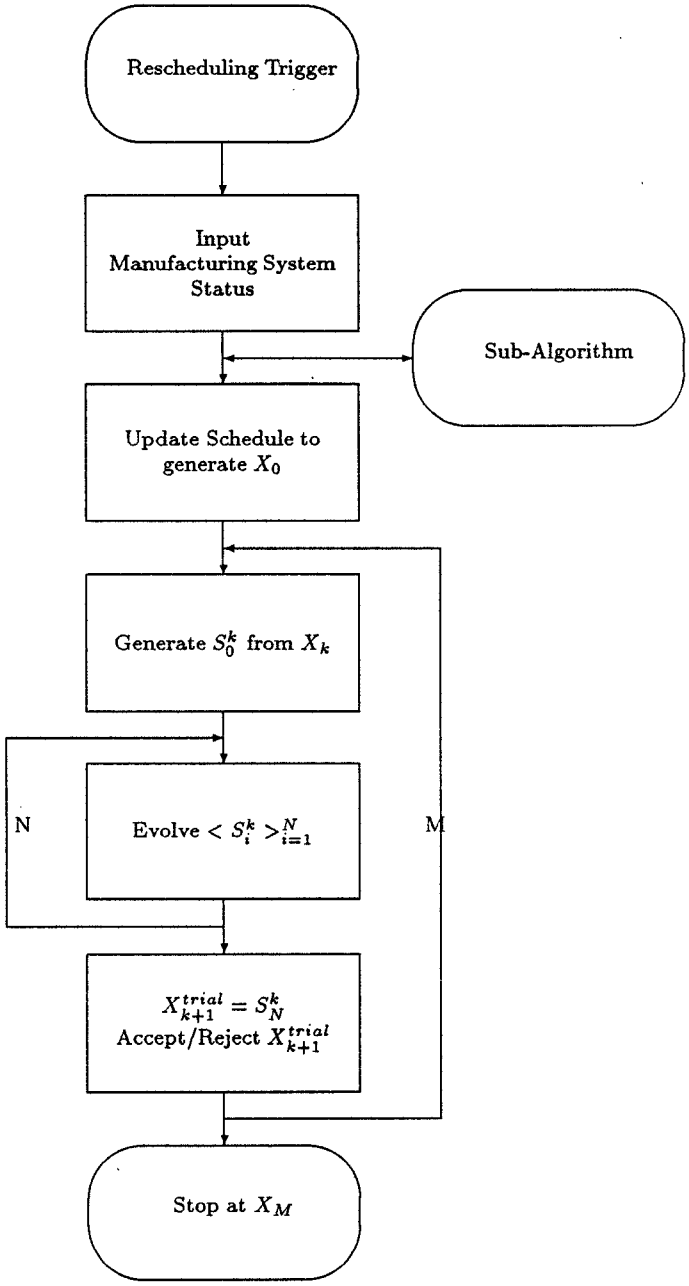


Figure 4.5: Flow Diagram for Rescheduling based Process Control

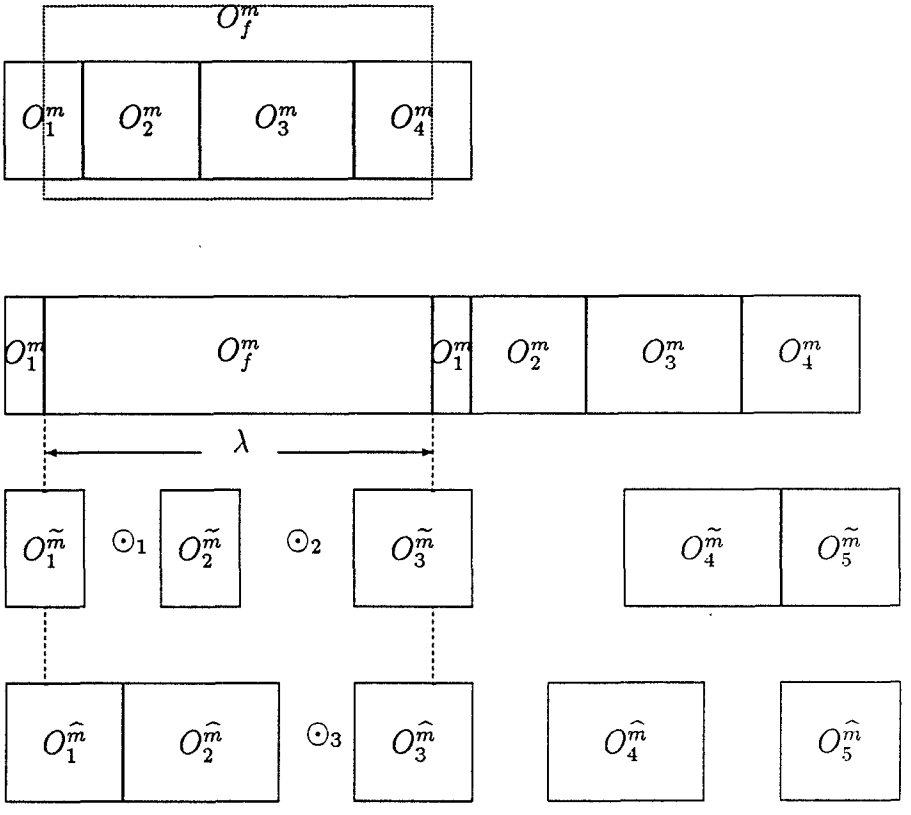


Figure 4.6: Machine Schedules for Process control sub-algorithm.

m, before and after the introduction of the failure/repair dummy operation, respectively. Since operations O_2^m, O_3^m and O_4^m were "displaced" due to the introduction of the operation O_f^m, these operations will be the candidates for machine re-assignment. The process control algorithm, at each failed machine, reassigns the "displaced" operations in the machine idle slots (denoted by $\odot$ in Figure 4.6) of all possible alternate machines. In our example, the operation O_1^m has already begun on machine m and hence is not re-assigned. Using the given set of process flows for $batch(O_2^m)$, let machine $\widetilde{m}$ be a possible alternate machine on which operation O_2^m can be performed. Then a trial schedule is generated by inserting the operation at $\odot_1$ position. The schedule costs for the original and the trial schedules are compared. The cost comparison of all such possible insertions (example: $\odot_2$ and $\odot_3$), and choice of the least cost schedule, leads to the minimum cost deviation reconfiguration. It might be possible, under certain conditions, that the original schedule is retained. A similar insertion procedure can be used for the remaining "displaced" operations, namely, O_3^m and O_4^m. Since the repair times are finite, the number of possible cost comparisons are finite and hence an exhaustive search procedure is used to find the minimum reconfigured cost schedule. The complete procedure for machine reassignment is shown in Figure 4.7.

Once the machine reassignment has been done, the failed machine can become a operation router, for the duration of the failure, and route the batches to the appropriate re-assigned machines. The insertion algorithm, discussed in Chapter 2, takes the batch and machine order precedence into account, hence the validity of the schedule is not a concern. The time window, λ, for identification of the "displaced operations" can be a variable parameter, but intuitively, it should be of the order of machine repair time.

4.3 Results

We used the predictive scheduler to generate a working schedule, for a given set of job orders (Figure 4.8), for the manufacturing system described in Chapter 3. Figures 4.9 and 4.10 represent the initial schedule, generated by the *insert algorithm* discussed in Chapter 2. The next step was to obtain an optimal while Figures 4.11 and 4.12 show the optimal schedule from the simulated annealing based optimization algorithm. On comparison of the two schedules, we note that the two particular batches of PCB 3, namely BATCH ID 011 and BATCH ID 004, and also BATCH ID 001 of part type PCB 4, have been scheduled using their alternate process flows in the optimized schedule. This alternate flow assignment was possible due to the *higher level* Metropolis process. Figure 4.15 shows the trace of the cost and the temperature versus number of iterations for the annealing based optimization process. The sharp increase/decrease in the temperature graph represents the artificially forced, manual increase/decrease in the temperature. The increase in the temperature allows the optimization

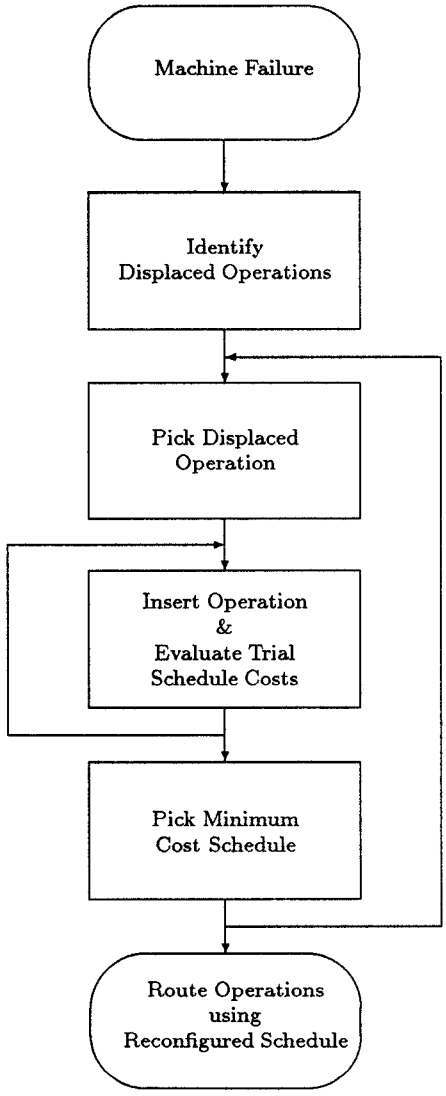


Figure 4.7: Flow Diagram for Process Control Sub-algorithm

Batch ID	Part Type	Batch Qty	Due Date
001	PCB 4	425	10/16/92
002	PCB 3	350	10/16/92
003	PCB 2	200	10/12/92
004	PCB 3	450	10/16/92
005	PCB 4	350	10/14/92
006	PCB 2	500	10/13/92
007	PCB 1	250	10/14/92
008	PCB 4	500	10/16/92
009	PCB 1	500	10/14/92
010	PCB 2	200	10/16/92
011	PCB 3	400	10/16/92
012	PCB 1	750	10/15/92

Figure 4.8: Job Orders for the Manufacturing System

process to accept higher cost schedules with a larger probability and hence allows the process "jump" out of the neighbourhood of a local optimal and hence converge to a global optimum. The corresponding cost trace during that period is, as expected, more "noisy". The schedule statistics for the initial and the optimized schedule are shown in Figures 4.13 and 4.14 respectively. The optimization process has reduced the number of tardy batches from 4 to 3 and also the average tardiness from 27 hrs/batch to 10.3 hrs/batch.

Once the initial optimal schedule was generated, the schedule was simulated using the QNAP2 based simulation model. The trace of the simulation for the first run (and also the subsequent second run after rescheduling) is shown in Figure 4.16. The simulation is stopped on failure type 2 on machine M/C-1. The simulation program then outputs a status report to the controller. The controller then updates the active time and also the status of different operations. The operation timings for the completed operations, for the first run are shown in Figure 4.18. The controller activates the optimization process again for an optimal schedule generation. The new schedule (shown in Figures 4.20 and 4.21) is now again input to the simulation program. The simulation trace and the completed operation timings for the second run are shown in Figures 4.16 and 4.19 respectively. The rescheduling cost (C_r) trace for the manufacturing system is shown in Figure 4.17. In both the simulation run, the rescheduling loop was triggered due to machine failures, but the machine operations could not be rescheduled, using the sub-algorithm, as no alternate machines were available. At each such rescheduling trigger, new batches/orders could be inserted into the updated schedule and the process can be continued indefinitely. This leads to online reconfigurable process control of the manufacturing system.

```
*************************************************************************
************************** PCB 01 ***************************************
*************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
-------------------------------------------------------------------------
-------------------------------------------------------------------------
Batch ID: 007
 0099      (parts)    0    250                        18:38:08 (10/12)
 0100    M/C-1       30    250    22:39:48 (10/12)    00:09:05 (10/13)
 0101    M/C-4       30    250    17:10:45 (10/13)    20:09:19 (10/13)
                                              DUE:    23:00:00 (10/14)
Batch ID: 012
 0074      (parts)    0    750                        18:38:08 (10/12)
 0075    M/C-1       30    750    05:32:11 (10/14)    10:00:02 (10/14)
 0076    M/C-4       30    750    20:02:11 (10/15)    04:57:54 (10/16)
                                              DUE:    23:00:00 (10/15)
Batch ID: 009
 0105      (parts)    0    500                        18:38:08 (10/12)
 0106    M/C-1       30    500    09:06:14 (10/13)    12:04:48 (10/13)
 0107    M/C-4       30    500    20:09:19 (10/13)    02:06:28 (10/14)
                                              DUE:    23:00:00 (10/16)


*************************************************************************
************************** PCB 02 ***************************************
*************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
-------------------------------------------------------------------------
-------------------------------------------------------------------------
Batch ID: 003
 0084      (parts)    0    200                        18:38:08 (10/12)
 0085    M/C-1       45    200    19:23:08 (10/12)    20:10:45 (10/12)
 0086    M/C-4       30    200    13:06:28 (10/13)    16:40:45 (10/13)
 0087    M/C-2       60    200    00:05:02 (10/14)    04:50:45 (10/14)
                                              DUE:    23:00:00 (10/12)
Batch ID: 006
 0095      (parts)    0    500                        18:38:08 (10/12)
 0096    M/C-1       45    500    20:10:45 (10/12)    22:09:48 (10/12)
 0097    M/C-4       30    500    04:10:45 (10/13)    13:06:28 (10/13)
 0098    M/C-2       60    500    12:10:45 (10/13)    00:05:02 (10/14)
                                              DUE:    23:00:00 (10/14)
Batch ID: 010
 0108      (parts)    0    200                        18:38:08 (10/12)
 0109    M/C-1       45    200    12:49:48 (10/13)    13:37:25 (10/13)
 0110    M/C-4       30    200    02:36:28 (10/14)    06:10:45 (10/14)
 0111    M/C-2       60    200    12:53:36 (10/15)    17:39:19 (10/15)
                                              DUE:    23:00:00 (10/16)
```

Figure 4.9: Initial Schedule (Part I)

```
**************************************************************************
************************** PCB 03  ***************************************
**************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Batch ID: 002
 0080      (parts)    0    350                        18:38:08 (10/12)
 0081    M/C-1      120    350   19:33:36 (10/14)     22:20:16 (10/14)
 0082    M/C-2       20    350   03:33:36 (10/15)     11:53:36 (10/15)
 0083    M/C-4       45    350   05:42:54 (10/16)     18:12:54 (10/16)
                                             DUE:     23:00:00 (10/13)
Batch ID: 011
 0112      (parts)    0    400                        18:38:08 (10/12)
 0113    M/C-1      120    400   15:37:25 (10/13)     18:47:54 (10/13)
 0114    M/C-2       20    400   05:10:45 (10/14)     14:42:11 (10/14)
 0115    M/C-4       45    400   13:10:45 (10/14)     03:27:54 (10/15)
                                             DUE:     23:00:00 (10/14)
Batch ID: 004
 0088      (parts)    0    450                        18:38:08 (10/12)
 0089    M/C-1      120    450   18:47:54 (10/13)     22:22:11 (10/13)
 0090    M/C-2       20    450   14:42:11 (10/14)     01:25:02 (10/15)
 0091    M/C-4       45    450   03:27:54 (10/15)     19:32:11 (10/15)
                                             DUE:     23:00:00 (10/15)


**************************************************************************
************************** PCB 04  ***************************************
**************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Batch ID: 005
 0092      (parts)    0    350                        18:38:08 (10/12)
 0093    M/C-1      150    350   00:52:11 (10/14)     05:02:11 (10/14)
 0094    M/C-3      120    350   08:52:11 (10/14)     21:22:11 (10/14)
                                             DUE:     23:00:00 (10/14)
Batch ID: 008
 0102      (parts)    0    500                        18:38:08 (10/12)
 0103    M/C-1      150    500   02:39:05 (10/13)     08:36:14 (10/13)
 0104    M/C-3      120    500   10:39:05 (10/13)     04:30:31 (10/14)
                                             DUE:     23:00:00 (10/15)
Batch ID: 001
 0077      (parts)    0    425                        18:38:08 (10/12)
 0078    M/C-1      150    425   12:30:02 (10/14)     17:33:36 (10/14)
 0079    M/C-3      120    425   21:22:11 (10/14)     12:32:54 (10/15)
                                             DUE:     23:00:00 (10/16)
**************************************************************************
**************************************************************************

Active Time of schedule: 18:38:08, Oct 12, 1992
```

Figure 4.10: Initial Schedule (Part II)

```
*************************************************************************
************************* PCB 01  ***************************************
*************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
-------------------------------------------------------------------------
-------------------------------------------------------------------------
Batch ID: 007
 0099      (parts)    0    250                        18:38:08 (10/12)
 0100    M/C-1       30    250    17:44:48 (10/13)    19:14:05 (10/13)
 0101    M/C-4       30    250    19:55:17 (10/14)    22:53:51 (10/14)
                                          DUE:       23:00:00 (10/14)
Batch ID: 000
 0074      (parts)    0    750                        18:38:08 (10/12)
 0075    M/C-1       30    750    08:21:14 (10/14)    12:49:05 (10/14)
 0076    M/C-4       30    750    22:53:51 (10/14)    07:49:34 (10/15)
                                          DUE:       23:00:00 (10/15)
Batch ID: 009
 0105      (parts)    0    500                        18:38:08 (10/12)
 0106    M/C-1       30    500    14:51:42 (10/14)    17:50:17 (10/14)
 0107    M/C-4       30    500    05:13:08 (10/16)    11:10:17 (10/16)
                                          DUE:       23:00:00 (10/16)


*************************************************************************
************************* PCB 02  ***************************************
*************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
-------------------------------------------------------------------------
-------------------------------------------------------------------------
Batch ID: 003
 0084      (parts)    0    200                        18:38:08 (10/12)
 0085    M/C-1       45    200    19:23:08 (10/12)    20:10:45 (10/12)
 0086    M/C-4       30    200    03:23:08 (10/13)    06:57:25 (10/13)
 0087    M/C-2       60    200    17:29:48 (10/13)    22:15:31 (10/13)
                                          DUE:       23:00:00 (10/12)
Batch ID: 006
 0095      (parts)    0    500                        18:38:08 (10/12)
 0096    M/C-1       45    500    20:10:45 (10/12)    22:09:48 (10/12)
 0097    M/C-4       30    500    06:57:25 (10/13)    15:53:08 (10/13)
 0098    M/C-2       60    500    22:15:31 (10/13)    10:09:48 (10/14)
                                          DUE:       23:00:00 (10/14)
Batch ID: 010
 0108      (parts)    0    200                        18:38:08 (10/12)
 0109    M/C-1       45    200    13:34:05 (10/14)    14:21:42 (10/14)
 0110    M/C-4       30    200    01:08:51 (10/16)    04:43:08 (10/16)
 0111    M/C-2       60    200    09:08:51 (10/16)    13:54:34 (10/16)
                                          DUE:       23:00:00 (10/16)
```

Figure 4.11: Optimized Schedule after 5000 iterations (Part I)

```
*************************************************************************
************************** PCB 03 ***************************************
*************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
-------------------------------------------------------------------------
-------------------------------------------------------------------------
Batch ID: 002
 0080      (parts)    0    350                        18:38:08 (10/12)
 0081    M/C-1      120    350   00:09:48 (10/13)     02:56:28 (10/13)
 0082    M/C-2       20    350   08:09:48 (10/13)     16:29:48 (10/13)
 0083    M/C-4       45    350   16:38:08 (10/13)     05:08:08 (10/14)
                                            DUE:     23:00:00 (10/13)
Batch ID: 011
 0112      (parts)    0    400                        18:38:08 (10/12)
 0113    M/C-1      120    400   02:56:28 (10/13)     06:06:57 (10/13)
 0114    M/C-3       20    400   10:56:28 (10/13)     20:27:54 (10/13)
 0115    M/C-4       45    400   05:08:08 (10/14)     19:25:17 (10/14)
                                            DUE:     23:00:00 (10/14)
Batch ID: 004
 0088      (parts)    0    450                        18:38:08 (10/12)
 0089    M/C-1      120    450   06:06:57 (10/13)     09:41:14 (10/13)
 0090    M/C-3       20    450   20:27:54 (10/13)     07:10:45 (10/14)
 0091    M/C-4       45    450   08:34:34 (10/15)     00:38:51 (10/16)
                                            DUE:     23:00:00 (10/15)


*************************************************************************
************************** PCB 04 ***************************************
*************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
-------------------------------------------------------------------------
-------------------------------------------------------------------------
Batch ID: 005
 0092      (parts)    0    350                        18:38:08 (10/12)
 0093    M/C-1      150    350   21:44:05 (10/13)     01:54:05 (10/14)
 0094    M/C-3      120    350   09:10:45 (10/14)     21:40:45 (10/14)
                                            DUE:     23:00:00 (10/14)
Batch ID: 008
 0102      (parts)    0    500                        18:38:08 (10/12)
 0103    M/C-1      150    500   01:54:05 (10/14)     07:51:14 (10/14)
 0104    M/C-3      120    500   21:40:45 (10/14)     15:32:11 (10/15)
                                            DUE:     23:00:00 (10/15)
Batch ID: 001
 0077      (parts)    0    425                        18:38:08 (10/12)
 0078    M/C-1      150    425   12:11:14 (10/13)     17:14:48 (10/13)
 0079    M/C-2      120    425   12:09:48 (10/14)     03:20:31 (10/15)
                                            DUE:     23:00:00 (10/16)
*************************************************************************
*************************************************************************

Active Time of schedule: 18:38:08, Oct 12, 1992
```

Figure 4.12: Optimized Schedule after 5000 iterations (Part II)

```
*********************************************************
*              SCHEDULE STATISTICS DISPLAY              *
*********************************************************
Schedule Active Time: 18:38:08, Oct 12, 1992

Number of batches...
           in schedule :     12
           tardy       :      4
Mean Tardiness          :   9.0 Hours

Average tardiness of tardy batches :     26.9 Hours

COST    WIP ==>             0.0
  Tardiness ==>           537.5
  Inventory ==>             0.0


*********************************************************
```

Figure 4.13: Initial Schedule Statistics

```
*********************************************************
*              SCHEDULE STATISTICS DISPLAY              *
*********************************************************
Schedule Active Time: 18:38:08, Oct 12, 1992

Number of batches...
           in schedule :     12
           tardy       :      3
Mean Tardiness          :   2.6 Hours

Average tardiness of tardy batches :     10.3 Hours

COST    WIP ==>             0.0
  Tardiness ==>           155.2
  Inventory ==>             0.0


*********************************************************
```

Figure 4.14: Optimal Schedule Statistics

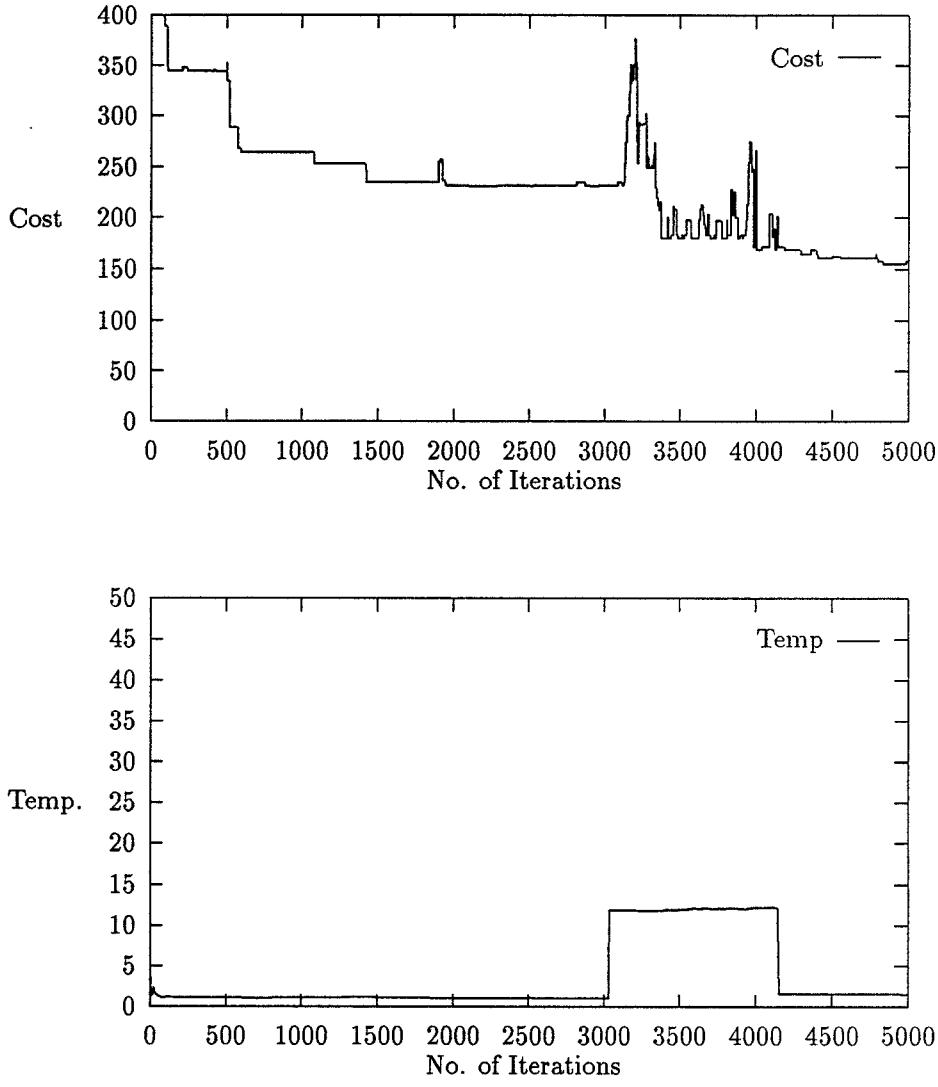


Figure 4.15: Cost and Temperature trace for Optimization (5000 iterations)

First Run

```
OP#  85 of Batch  3 done on M/C-1 at time  20: 7:28(10/12/92)
OP#  96 of Batch  6 done on M/C-1 at time  22:56: 0(10/12/92)
OP#  81 of Batch  2 done on M/C-1 at time   3:33:52(10/13/92)
OP#  86 of Batch  3 done on M/C-4 at time   6:52:48(10/13/92)
OP# 113 of Batch 11 done on M/C-1 at time   8:43:12(10/13/92)
FAILURE TYPE 2 on MACHINE M/C-1 at time   8:43:12(10/13/92)
 RESCHEDULE.....  HALTING SIMULATION
```

Second Run

```
OP#  97 of Batch  6 done on M/C-4 at time  16: 3:44(10/13/92)
OP#  82 of Batch  2 done on M/C-2 at time  17:53:36(10/13/92)
OP# 114 of Batch 11 done on M/C-3 at time  18:32:32(10/13/92)
OP#  87 of Batch  3 done on M/C-2 at time  23:37:36(10/13/92)
OP#  83 of Batch  2 done on M/C-4 at time   4:59:12(10/14/92)
OP# 100 of Batch  7 done on M/C-1 at time  10:44:48(10/14/92)
OP#  98 of Batch  6 done on M/C-2 at time  12:48:32(10/14/92)
OP#  93 of Batch  5 done on M/C-1 at time  17:57:20(10/14/92)
OP# 115 of Batch 11 done on M/C-4 at time  19:31:44(10/14/92)
OP# 101 of Batch  7 done on M/C-4 at time  23: 2:24(10/14/92)
FAILURE TYPE 3 on MACHINE M/C-4 at time  23: 2:24(10/14/92)
 RESCHEDULE.....  HALTING SIMULATION
```

Figure 4.16: Partial Simulation Run

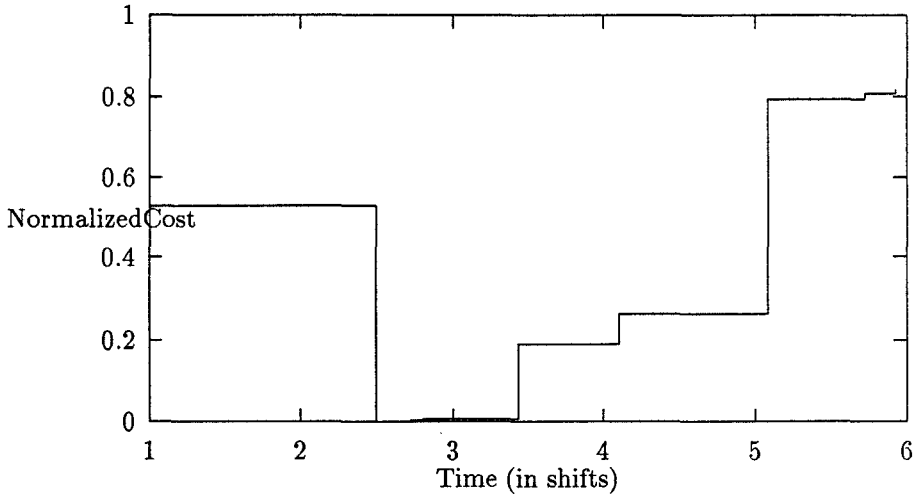


Figure 4.17: Rescheduling Trace

```
**********************************************************************
*   Batch No.  2         (QTY =  350)                                  *
*  mach   setup        start time      finish time    proc time(hr)*
**********************************************************************
*  M/C-1   120    0:56: 0(10/13/92)    3:33:52(10/13/92)    2.63   *
*  Sched times    0: 9:36(10/13/92)    2:56: 0(10/13/92)    2.78   *
**********************************************************************


**********************************************************************
*   Batch No.  3         (QTY =  200)                                  *
*  mach   setup        start time      finish time   proc time(hr)*
**********************************************************************
*  M/C-1    45   19:22:40(10/12/92)   20: 7:28(10/12/92)    0.75   *
*  Sched times   19:22:40(10/12/92)   20:10:40(10/12/92)    0.79   *
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  M/C-4    30    3:22:40(10/13/92)    6:52:48(10/13/92)    3.50   *
*  Sched times    3:22:40(10/13/92)    6:57: 4(10/13/92)    3.57   *
**********************************************************************


**********************************************************************
*   Batch No.  6         (QTY =  500)                                  *
*  mach   setup        start time      finish time    proc time(hr)*
**********************************************************************
*  M/C-1    45   20:52:16(10/12/92)   22:56: 0(10/12/92)    2.06   *
*  Sched times   20:10:40(10/12/92)   22: 9:36(10/12/92)    1.98   *
**********************************************************************


**********************************************************************
*   Batch No. 11         (QTY =  400)                                  *
*  mach   setup        start time      finish time    proc time(hr)*
**********************************************************************
*  M/C-1   120    5:33:52(10/13/92)    8:43:12(10/13/92)    3.15   *
*  Sched times    2:56: 0(10/13/92)    6: 6:56(10/13/92)    3.17   *
**********************************************************************
```

Figure 4.18: Schedule Timings for Completed Operations (First Run)

```
*****************************************************************
*   Batch No.  2         (QTY =  350)                             *
*  mach   setup        start time      finish time    proc time(hr)*
*****************************************************************
*  M/C-2    20    9: 2:56(10/13/92)   17:53:36(10/13/92)    8.84   *
*  Sched times    9: 3:28(10/13/92)   17:23:12(10/13/92)    8.33   *
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  M/C-4    45   17: 4: 0(10/13/92)    4:59:12(10/14/92)   11.92   *
*  Sched times   17:16:16(10/13/92)    5:46: 8(10/14/92)   12.50   *
*****************************************************************
*****************************************************************
*   Batch No.  3         (QTY =  200)                             *
*  mach   setup        start time      finish time    proc time(hr)*
*****************************************************************
*  M/C-2    60   18:53:52(10/13/92)   23:37:36(10/13/92)    4.73   *
*  Sched times   18:23:28(10/13/92)   23: 8:48(10/13/92)    4.76   *
*****************************************************************
*****************************************************************
*   Batch No.  5         (QTY =  350)                             *
*  mach   setup        start time      finish time    proc time(hr)*
*****************************************************************
*  M/C-1   150   13:14:40(10/14/92)   17:57:20(10/14/92)    4.71   *
*  Sched times   13:12:32(10/14/92)   17:22:40(10/14/92)    4.17   *
*****************************************************************
*****************************************************************
*   Batch No.  6         (QTY =  500)                             *
*  mach   setup        start time      finish time    proc time(hr)*
*****************************************************************
*  M/C-4     0    8:43:12(10/13/92)   16: 3:44(10/13/92)    7.34   *
*  Sched times    8:43:12(10/13/92)   16:31:28(10/13/92)    7.80   *
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  M/C-2    60    0:37:20(10/14/92)   12:48:32(10/14/92)   12.19   *
*  Sched times   23: 8:48(10/13/92)   11: 3:28(10/14/92)   11.90   *
*****************************************************************
*****************************************************************
*   Batch No.  7         (QTY =  250)                             *
*  mach   setup        start time      finish time    proc time(hr)*
*****************************************************************
*  M/C-1    30    9:13: 4(10/14/92)   10:44:48(10/14/92)    1.53   *
*  Sched times    9:13: 4(10/14/92)   10:42:40(10/14/92)    1.49   *
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  M/C-4    30   20: 1:36(10/14/92)   23: 2:24(10/14/92)    3.01   *
*  Sched times   20:33:36(10/14/92)   23:31:44(10/14/92)    2.98   *
*****************************************************************
*****************************************************************
*   Batch No. 11         (QTY =  400)                             *
*  mach   setup        start time      finish time    proc time(hr)*
*****************************************************************
*  M/C-3    20    9: 2:56(10/13/92)   18:32:32(10/13/92)    9.50   *
*  Sched times    9: 3:28(10/13/92)   18:34:40(10/13/92)    9.52   *
* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*  M/C-4    45    5:44: 0(10/14/92)   19:31:44(10/14/92)   13.79   *
*  Sched times    5:46: 8(10/14/92)   20: 3:12(10/14/92)   14.29   *
*****************************************************************
```

Figure 4.19: Schedule Timings for Completed Operations (Second Run)

```
**************************************************************************
************************** PCB 01  ***************************************
**************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Batch ID: 007
 0099      (parts)    0    250                        08:43:12 (10/13)
 0100   M/C-1        30    250    09:13:12 (10/14)    10:42:29 (10/14)
 0101   M/C-4        30    250    20:33:23 (10/14)    23:31:57 (10/14)
                                          DUE:       23:00:00 (10/14)
Batch ID: 000
 0074      (parts)    0    750                        08:43:12 (10/13)
 0075   M/C-1        30    750    23:49:38 (10/14)    04:17:29 (10/15)
 0076   M/C-4        30    750    07:49:38 (10/15)    16:45:21 (10/15)
                                          DUE:       23:00:00 (10/15)
Batch ID: 009
 0105      (parts)    0    500                        08:43:12 (10/13)
 0106   M/C-1        30    500    19:27:58 (10/15)    22:26:32 (10/15)
 0107   M/C-4        30    500    15:30:21 (10/16)    21:27:29 (10/16)
                                          DUE:       23:00:00 (10/16)


**************************************************************************
************************** PCB 02  ***************************************
**************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Batch ID: 003
 0087   M/C-2        60    200    18:23:12 (10/13)    23:08:55 (10/13)
                                          DUE:       23:00:00 (10/12)
Batch ID: 006
 0097   M/C-4         0    500    08:43:12 (10/13)    16:31:14 (10/13)
 0098   M/C-2        60    500    23:08:55 (10/13)    11:03:12 (10/14)
                                          DUE:       23:00:00 (10/14)
Batch ID: 010
 0108      (parts)    0    200                        08:43:12 (10/13)
 0109   M/C-1        45    200    10:36:46 (10/15)    11:24:23 (10/15)
 0110   M/C-4        30    200    18:36:46 (10/15)    22:11:03 (10/15)
 0111   M/C-2        60    200    02:36:46 (10/16)    07:22:29 (10/16)
                                          DUE:       23:00:00 (10/16)
```

Figure 4.20: Optimal Schedule after Rescheduling. (Part I)

```
**************************************************************************
************************** PCB 03 ***************************************
**************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Batch ID: 002
 0082   M/C-2        20    350    09:03:12 (10/13)    17:23:12 (10/13)
 0083   M/C-4        45    350    17:16:14 (10/13)    05:46:14 (10/14)
                                          DUE:       23:00:00 (10/13)
Batch ID: 011
 0114   M/C-3        20    400    09:03:12 (10/13)    18:34:38 (10/13)
 0115   M/C-4        45    400    05:46:14 (10/14)    20:03:23 (10/14)
                                          DUE:       23:00:00 (10/14)
Batch ID: 004
 0088      (parts)    0    450                        08:43:12 (10/13)
 0089   M/C-1       120    450    06:17:29 (10/15)    09:51:46 (10/15)
 0090   M/C-2        20    450    14:17:29 (10/15)    01:00:21 (10/16)
 0091   M/C-4        45    450    22:56:03 (10/15)    15:00:21 (10/16)
                                          DUE:       23:00:00 (10/15)


**************************************************************************
************************** PCB 04 ***************************************
**************************************************************************
Operation  Machine  Setup  Batch  Start                Finish
   ID        ID     Time    Qty   Time                 Time
--------------------------------------------------------------------------
--------------------------------------------------------------------------
Batch ID: 005
 0092      (parts)    0    350                        08:43:12 (10/13)
 0093   M/C-1       150    350    13:12:29 (10/14)    17:22:29 (10/14)
 0094   M/C-3       120    350    21:12:29 (10/14)    09:42:29 (10/15)
                                          DUE:       23:00:00 (10/14)
Batch ID: 008
 0102      (parts)    0    500                        08:43:12 (10/13)
 0103   M/C-1       150    500    17:22:29 (10/14)    23:19:38 (10/14)
 0104   M/C-3       120    500    09:42:29 (10/15)    03:33:55 (10/16)
                                          DUE:       23:00:00 (10/15)
Batch ID: 001
 0077      (parts)    0    425                        08:43:12 (10/13)
 0078   M/C-1       150    425    13:54:23 (10/15)    18:57:58 (10/15)
 0079   M/C-3       120    425    03:33:55 (10/16)    18:44:38 (10/16)
                                          DUE:       23:00:00 (10/16)
**************************************************************************
**************************************************************************

Active Time of schedule: 08:43:12, Oct 13, 1992
```

Figure 4.21: Optimal Schedule after Rescheduling. (Part II)

CHAPTER
FIVE

Operation Control

Based on the hierarchical reconfigurable control scheme presented in Chapter 3, the machine schedule for the Operation Level is defined at the Process Level. The process flow assignment and the operation sequencing on each machine, determines the start and finish times for each operation. This is for the batch mode case. This operation schedule also determines the arrival rates, for different part types, for the machines operating in the continuous mode. To illustrate this point, let us consider our example manufacturing system. The system schedule determines the start and finish times for different operations on machines M/C-1 through M/C-4. Based on the process flow definitions, the parts are routed to the first test station, M/C-5, through one of those machines. Hence, the arrival rates for the different part types, at Machine 5, can be determined on the basis of the corresponding "source" machine schedules.

Since, we have a hybrid manufacturing system, we will present algorithms for both, batch mode operation control (Section 5.1) and continuous mode operation control (Section 5.2). The reconfiguration in both cases involves optimal control of processing times, so that an appropriate cost function is minimized.

5.1 Batch Mode Operation Control

In this section, we consider the problem of rate control for a single operation on a failure prone machine. As discussed earlier, the Process Level control assigns different operations to machines and assigns start and finish times to each operation. The operation schedule plan is deterministic, i.e., it does not take into account various random factors, namely machine failures and random processing times per part. These random events cause the actual operation timings to differ from the scheduled times. This timing deviation, at a single machine, is propogated through the complete system schedule due to the machine and batch-order precedences. For example, a delay Δt in completing a particular operation, can cause a corresponding delay in the start of the next operations, both in the batch-order and machine-order operation sequences for that operation. Our objective is to control the actual (real time) processing rate of the machine, so that the operation is completed on schedule, or with the minimum deviation. This minimum deviation at the operation level ensures that the suc-

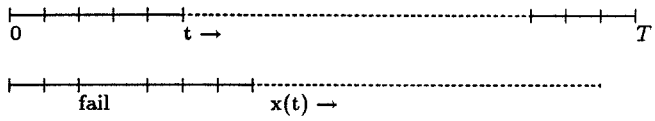


Figure 5.1: Real time versus Scheduled time for a single operation

ceeding operation timings are not adversely affected and the overall schedule deviation is minimized. We consider the "lateness," i.e. the absolute difference between the actual and the scheduled finish time, as a deviation measure. Hence the overall system schedule deviation is minimized, if each constituent operation deviation is minimized. Therefore, we consider the problem of rate control for a single operation.

The processing times are deterministic while the machine failures times are exponentially distributed. We assume that the machine can operate only at certain processing speeds. This assumption is again based on real manufacturing system operation. This problem is formulated as a dynamic programming problem. Due the finite number of discrete processing rates, this has an associated switching control. Hence, this is an impulse control problem.

In Section 5.1.1, we present the optimal control problem formulation. The associated dynamic programming, and the switching control, equations are derived in Section 5.1.2. The optimal impulse control specification, and the results from our example case, will be presented in Section 5.1.3. We will present the numerical computation details in the appendix.

5.1.1 Optimal Control Problem Formulation

Consider a single operation for a batch of N parts and with the corresponding scheduled operation time of T. The scheduled mean process time can be assumed to be T/N. The machine is characterized by K failure modes. For each failure mode k ($k = 1, \ldots, K$), the failure times are assumed to be exponentially distributed with parameter λ_k with an associated (deterministic) repair time $h_k(t)$. If we denote a schedule time instant by $t, t \in [0, T]$ then $x(t) \in \mathbb{R}^+$ denotes the "real time", that is, the the actual operation time instant. Under deterministic conditions with no failures and constant processing times of T/N, $x(t) = t$, $\forall t \in [0, T]$. If we introduce random failures or change processing times, then the above relationship does not hold. This is shown in Figure 5.1.

For exponentially distributed failure times (i.e. Poisson failures), which are mathematically represented by the Poisson process $N_{\lambda_k}(t)$ we can write:

$$x(t) = \frac{N}{T} \int_0^t \mu(s) ds + \sum_{k=1}^{K} \int_0^t h_k(s) dN_{\lambda_k}(s)$$

or writing it in a differential form

$$dx(t) = \frac{N}{T}\mu(t)dt + \sum_{k=1}^{K} h_k(t)dN_{\lambda_k}(t)$$

If we define $\Xi = \{\mu_1, \mu_2, \ldots, \mu_l\}$ as the set of possible process times for the machine, then the switching control $\mu(t)$ can be written as

$$\mu(t) = \sum_{i=0}^{\infty} \mu_i I_{t_i \leq t < t_{i+1}} \qquad \mu_i \in \Xi, \forall i$$

The above control $\mu(t)$ can also be represented as the sequence $< \mu_i,\ t_i >_{i=0}^{\infty}$, $\mu_i \in \Xi$ and $t_n \uparrow T$. Let Λ_{ad} denote the set of all such admissible controls.

The *cost-to-go* value function $J_i(x(t), t)$ can be written in terms of the three cost components:

1. Running cost $C_r(x, \mu)$
2. Switching cost $C_s(\mu_i, \mu_j)$
3. Terminal cost $C_f(x, T)$

$$J_i(x(t), t) = E_{\mu^i, x(t)}\left[\int_t^T C_r(x(s), \mu(s))ds + \sum_{\substack{j=1 \\ t \leq t_j < T}}^{\infty} C_s(\mu_{j-1}, \mu_j) + C_f(x(T), T)\right]$$

The control problem can now be defined as the choice of control $< \mu_i, t_i >_{i=0}^{\infty}$ $\in \Lambda_{ad}$, which minimizes the cost $J_i(x(0), 0)$.

5.1.2 Quasi-Variational Inequalities for Optimal Control

Using the above problem formulation, let us define

$$U_i(x, t) = \min_{\Lambda_{ad}} J_i(x, t) \qquad \text{where } x(t) = x;\ i = 1, \ldots, l$$

At any time instant "t", with $x(t) = x$ and $\mu(t) = \mu_i$, if we let the process to run freely for time interval δ, then using the dynamic programming principle, the optimal *cost-to-go* satisfies:

$$U_i(x, t) \leq E_{\mu^i, x(t)}\left[\int_t^{t+\delta} C_r(x(s), \mu(s))ds + U_i(x(t+\delta), t+\delta)\right] \qquad i = 1, \ldots, l$$

Expanding $U_i(x(t+\delta), t+\delta)$ and rearranging terms, we have:

$$\mathcal{L}U_i(x,t) \triangleq -\frac{\partial U_i(x,t)}{\partial t} - \frac{N}{T}\mu(t)\frac{\partial U_i(x,t)}{\partial x} - \sum_{k=1}^{K} \lambda_k \left[U_i(x+h_k(t),t) - U_i(x,t)\right] \leq C_r(x,\mu_i)$$

For the case of a switch in the control at time "t", we have another inequality:

$$U_i(x,t) \leq \min_{\substack{i \neq j \\ j=1\ldots l}} \left[U_j(x,t) + C_s(\mu_i,\mu_j)\right] \triangleq \mathcal{M}U_i(x,t) \qquad i = 1, \ldots, l$$

At the time instant "t", we must choose either to let the process to evolve freely or decide to switch control. In each case the corresponding inequality becomes an equation. Writing both inequalities together in a product form, which vanishes for all "t", we have:

$$(\mathcal{L}U_i(x,t) - C_r(x,\mu_i)).(U_i(x,t) - \mathcal{M}U_i(x,t)) = 0 \qquad i = 1, \ldots, l$$

The boundary conditions for the set of equations are determined from the terminal cost, i.e:

$$U_i(x,T) = C_f(x,T); \qquad \forall x \in \mathbb{R}^+, i = 1 \ldots l.$$

The existence and uniqueness of the the optimal *cost-to go* is discussed in [4].

Based on the operators $\mathcal{L}$ and $\mathcal{M}$ defined above, we can define the *continuation* sets $\mathcal{C}_i$ and *switch* sets $\mathcal{S}_i$ as:

$$\mathcal{C}_i = \{x,t | \mathcal{L}U_i(x,t) = C_r(x,\mu_i)\}$$

and

$$\mathcal{S}_i = \{x,t | U_i(x,t) = \mathcal{M}U_i(x,t); \mathcal{L}U_i(x,t) < C_r(x,\mu_i)\}$$

5.1.3 Optimal Impulse Control

These continuation and switch sets are used to determine the optimal impulse control from the optimal *cost-to-go*. For a given instant t, we observe the value of the process $x(t)$ and the control being applied $\mu(t) = \mu_i$. We let the process run freely if $(x(t),t) \in \mathcal{C}_i$. If $(x(t),t) \in \mathcal{S}_i$, we switch to control $\mu_{\widehat{i}}$, where $\widehat{i} = \arg\ \min_{j=1\ldots l;\ i \neq j} \left[U_j(x,t) + C_s(\mu_i,\mu_j)\right]$.

To compute the continuation and switch sets, we used the approach developed by Gonzalez and Rofman *et al* [20, 22, 21, 41] and Kabbaj *et al* [14]. We present the details of the procedure and the numerical computational aspects

in the Appendix A. We now proceed to present the results obtained by the application of this algorithm for control of a single operation.

We consider a batch size of 100 parts, with the nominal processing rate of 1 part per unit time. The reasons for this particular choice of the batch size and the associated processing rate will become clear a little later. We let the machine have 3 possible processing times, namely 0.5, 1.0 and 1.5 and a single failure mode (FM1 in our original simulation model). Using the numerical algorithm presented in Appendix A, the optimal cost functions were calculated and the corresponding continuation and switch sets were determined. Figures 5.2, 5.3 and 5.4 show the optimal cost function values and the associated continuation and switch sets for the three cases of $\mu_1 = 0.5$, $\mu_2 = 1.0$ and $\mu_3 = 1.5$ respectively.

Let us consider Figure 5.2: The top figure represents the function values of $U_1(x,t)$ for the region [0,150] $\times$ [0,100]. We used a 100 $\times$ 100 grid for the discretization. Based on the calculated values of $U_1(x,t)$, $U_2(x,t)$ and $U_3(x,t)$, we determine the associated continuation and switch sets. The region marked $\mathcal{R}_\infty$ represents the continuation set $\mathcal{C}_1$ for control μ_1, i.e. if $(x,t) \in \mathcal{C}_1$ then continue with $\mu(t) = \mu_1$. The other two regions, marked $\mathcal{R}_2$ and $\mathcal{R}_3$, represent the switch sets, associated to controls μ_2 and μ_3, respectively. That is, if (x,t) lies in region $\mathcal{R}_2$ then switch $\mu(t)$ from μ_1 to μ_2 and in a similar fashion, set $\mu(t) = \mu_3$ for (x,t) lying in the region $\mathcal{R}_3$. A similar argument holds for the definition of continuation and switch sets, corresponding to control μ_2 and μ_3, and are shown in Figures 5.3 and 5.4, respectively.

Figure 5.5 shows the actual run of a simulation for a machine with failures. We ran another simulation using 5 possible processing times The two figures correspond to the two different sets of possible processing times. The top figure is for $\Xi = \{0.5, 1.0, 1.5\}$ while the bottom figure is for the case of $\Xi = \{0.5, 0.75, 1.0, 1.25, 1.5\}$. The results shown in the figure can be interpreted in the following way: During the initial run, the optimal control process parts at the fastest rate and moves toward a "hedging point". This takes into consideration, the anticipated failures. The sample run for the "l = 3" case does not exhibit this hedging point, but in the "l=5" case the hedging point occurs at scheduled time = 33.0 (corresponding actual time = 22.0). This is the impulse control analogue for hedging point policy for the continuous control problem discussed by Gershwin *et al.*

We used a simple example, i.e. batch size 100, nominal processing time = 1 part/time and $\Xi = \{0.5, 1.0, 1.5\}$. The choice of 100 part batch with 1 part per unit time processing rate, was made so that the control problem becomes dimensionless. The reason is as follows:

Since the continuation and switching sets are numerically quite large, it might not be possible to implement this algorithm for all possible input parameters, i.e. for all possible processing times and batch sizes. The dimensionless characteristic is useful for using the same collection of switching and continuation sets for determining the optimal control policy. By proper aggregation (and dis-

aggregation) principles, larger (smaller) batch sizes can be controlled using the same continuation and switch sets. The only restriction is that the processing rate (or time) ratios remain the same. For example, let us consider a batch operation of 300 parts. Then the aggregation of 3 parts into a single *virtual part*, with the corresponding processing time being three times that of a single part, allows us to use the 100 part problem data for the above example. This allows for storing the data on a small memory chip and an on-board computer can access the proper switching curves for online control of an actual machine. For similar dimensionless definition of Ξ, the possible processing times are now processing time ratios (with respect to the nominal processing time).

This control might not be optimal. The tradeoff between recomputation of optimal control (and the associated increase in memory requirements) versus the sub-optimality of the existing solution can be used to justify the appropriate choice of the control strategy.

In this section, we presented a dynamic programming based impulse control algorithm for batch mode Operation Level control. To make our research complete, in the next section, we present continuous mode Operation control. Using these two algorithms, we can control almost any manufacturing systems at the Operation Level.

5.2 Continuous Mode Operation Control

If we recall our discussion on manufacturing systems in Chapter 1, continuous mode processing is used in cases where sets of physical machines can be grouped together to form "virtual machines" which process parts in tandem. In this section, we consider the problem of online control of the mean processing times for different parts at a virtual machine. In our test example, the test stations of the quality control unit can be grouped into a virtual machine. Keeping in view, this concept of virtual machine processing parts in tandem, we present online reconfigurable control algorithm for continuous mode Operation Control.

Our formulation of the control problem is partitioned into two sub-problems: (i) estimation of the derivative of the performance measure with respect to the mean processing times; and (ii) use of gradient approximation methods for finding the optimal values for the parameters. To estimate the gradient of the performance measure, we use *Perturbation Analysis* (PA) method. PA was developed for sensitivity analysis in linear continuous time systems. Ho, Cao, Gong and Suri [25, 26, 9, 19, 18, 48, 49] have extended the basic ideas and applied them to discrete event dynamic systems.

The basic idea in perturbation analysis is to estimate the gradient of a given objective function, with respect to a parameter, for a stochastic dynamical system using a *single sample path.* Although this involves extra computational overhead (in simulations), it is more efficient than other finite difference based estimation techniques which involve separate sample path generation for several

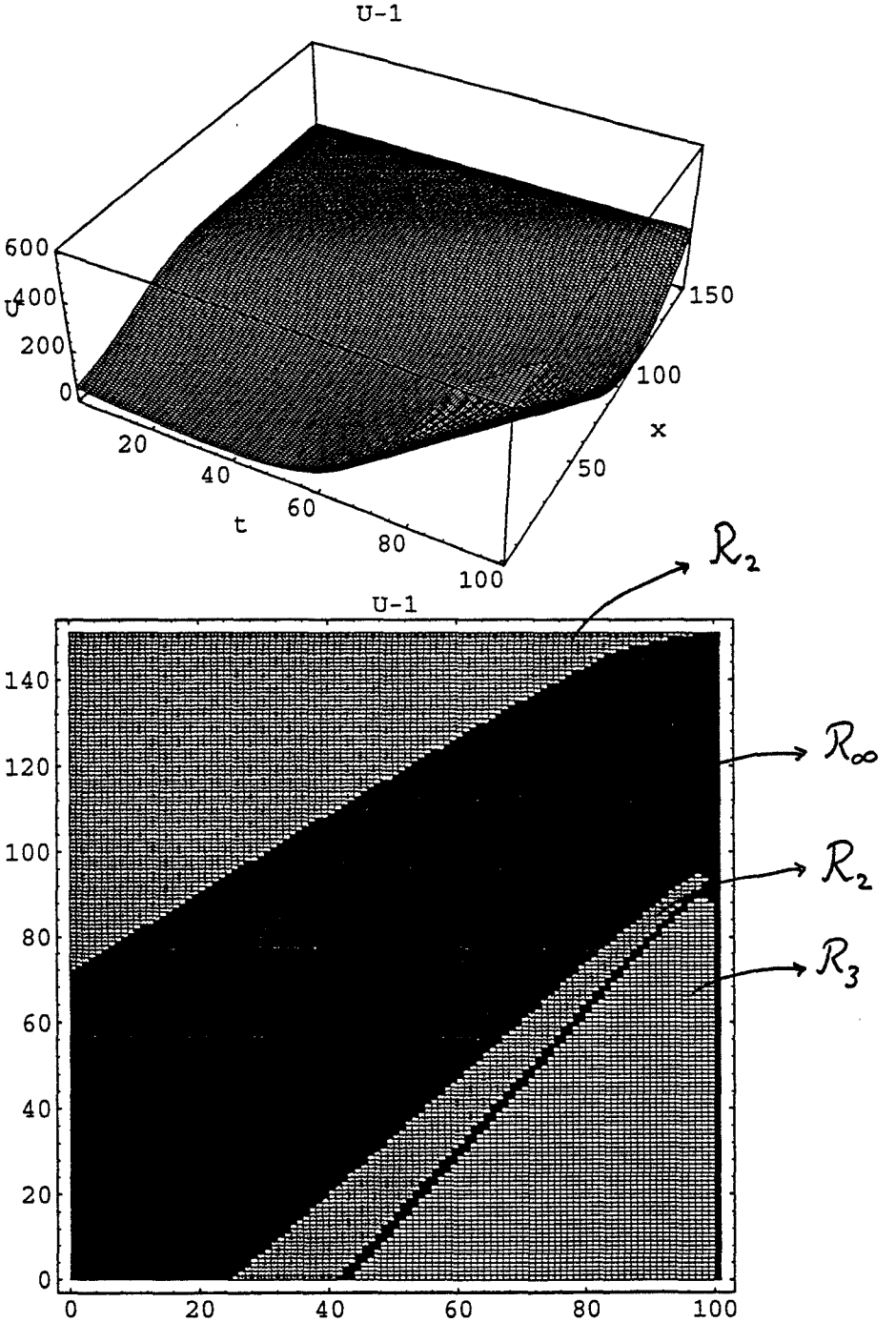


Figure 5.2: The optimal cost surface and the corresponding continuation and switch sets for U_1

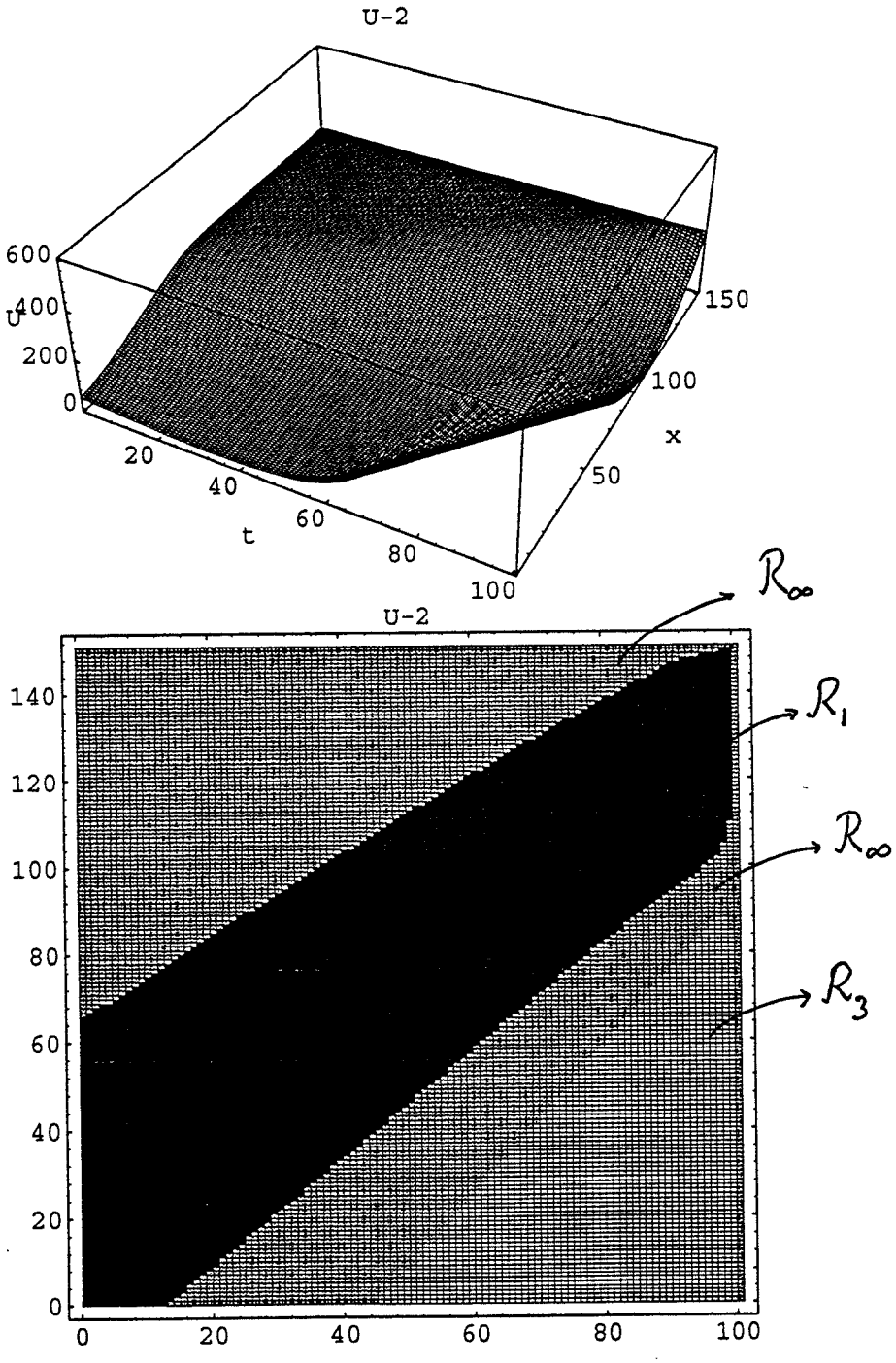


Figure 5.3: The optimal cost surface and the corresponding continuation and switch sets for U_2

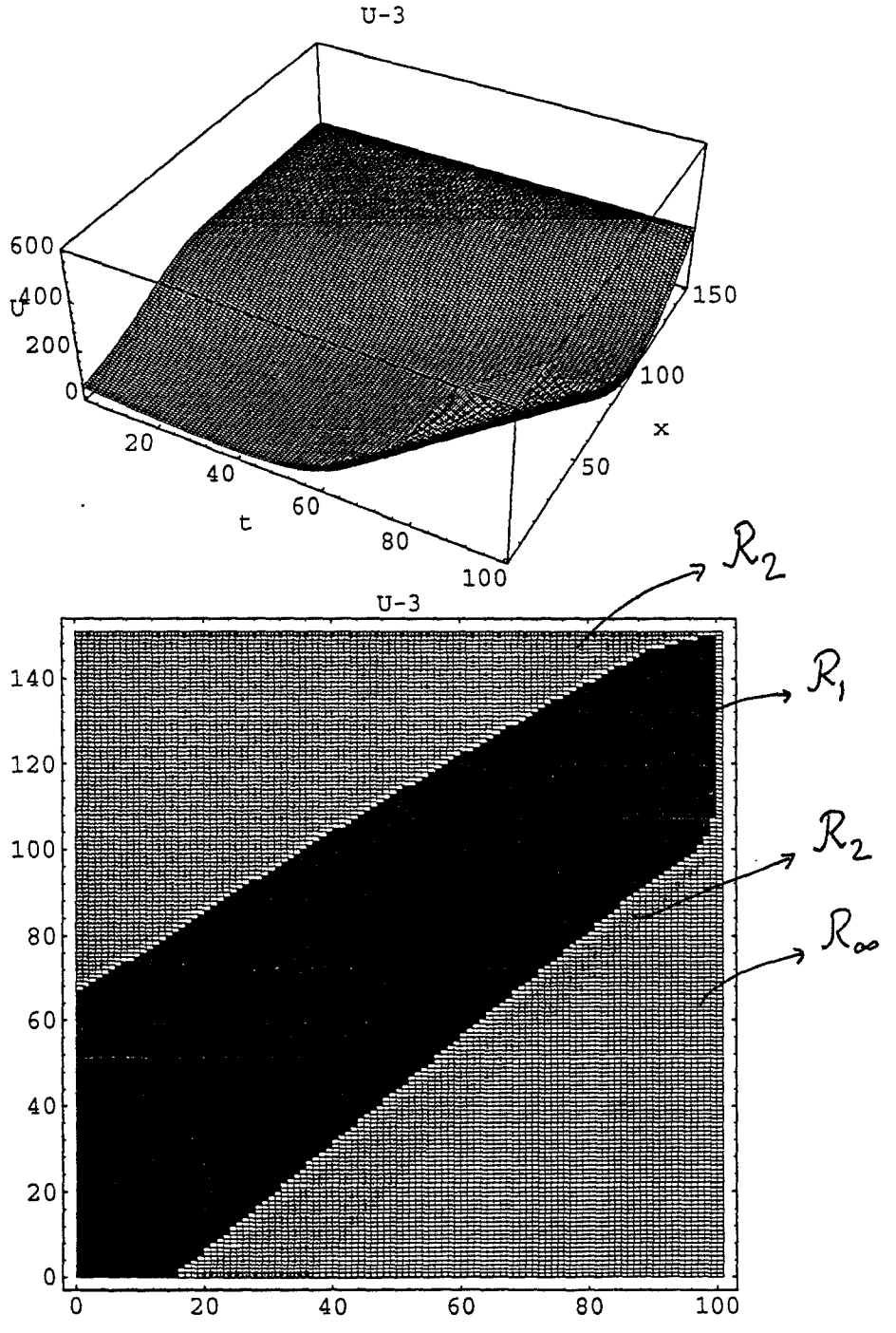


Figure 5.4: The optimal cost surface and the corresponding continuation and switch sets for U_3

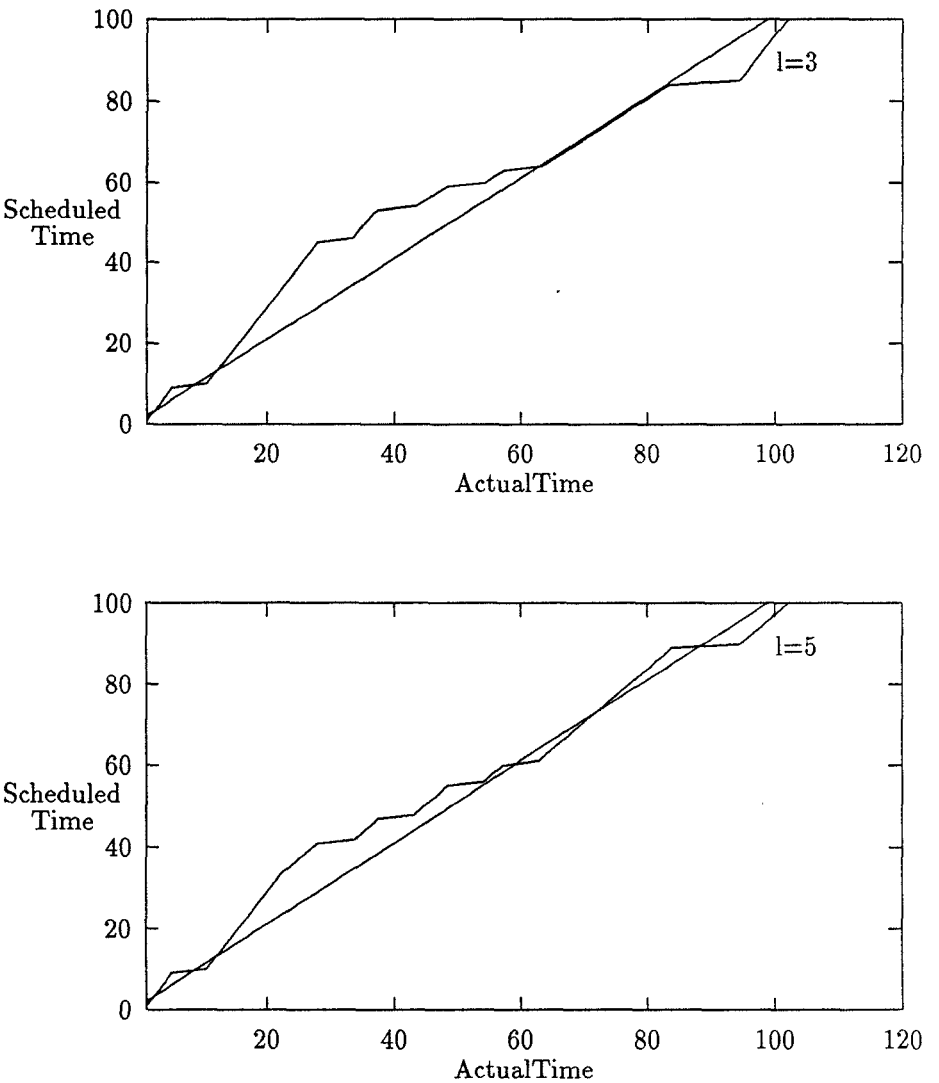


Figure 5.5: Actual Simulation Run for failure prone machine.

different values of the design parameter. PA has the added advantage that it permits gradient estimation with respect to several parameters using the same sample path. Suri and Leung [50] have used perturbation analysis for gradient estimation with respect to the processing and arrival rates for a single class G/G/1 queue. Fu and Hu [15] later extended the results to the single class GI/G/m queue and proved consistency of the gradient estimates. A contribution of our work is the optimization of multi-class queuing models.

The second step in the optimization is to use the gradient estimates for in stochastic optimization procedures. Two classical procedures used for stochastic optimization are the stochastic gradient, Robbins-Monro method, and the finite difference gradient estimation, Kiefer-Wolfowitz method. The Kiefer-Wolfowitz [29] procedure relies on gradient estimation using finite differences between a pair of sample paths, each generated using a different value of the design parameter. This involves dual sample path generation. Multiple parameter optimization requires separate dual path generation for each parameter. The Robbins-Monro [44] procedure, like perturbation analysis, uses a single sample path for stochastic optimization. The Robbins-Monro algorithm typically converges faster than the Keifer-Wolfowitz method. Hence, we use it in this work.

We develop the notation for the queuing system model and present an algorithm for derivative estimation using PA. We derive the optimization procedure based on the Robbins-Monro stochastic gradient algorithm. The validity of the scheme is also discussed.

5.2.1 Gradient Estimation Using Perturbation Analysis

If we consider a virtual machine with Poisson part arrivals, and exponentially distributed processing times, then global control problem for the virtual machine level can be reduced to local control at each physical machine. This is possible due to Burke's theorem [8]:

Theorem 1 (Burke) *The steady-state output of a queue with N channels in parallel, with Poisson arrival statistics, and with lengths chosen independently from an exponential distribution is itself Poisson.*

Therefore, if the part arrival in the virtual machine is Poisson, all constituent machines will also have Poisson part arrivals. If they also have exponentially distributed processing times, local optimization at the physical machine will lead to a global optimum for the whole virtual machine. Hence, we can focus on optimization of the mean processing time for an M/M/m queue (multi-class case).

For each machine we define:

m = Number of servers in the machine.
K = Number of different customer types the machine is capable of processing.
λ_i = The arrival rates for different customer types ($i = 1 \ldots K$)
θ_i = The mean processing times for different customer types ($i = 1 \ldots K$)
ρ_i = Machine workload due to customer type i. ($= \frac{\lambda_i \theta_i}{m}$)
λ = The net arrival rate of customers at the machine. ($= \sum_{i=1}^{K} \lambda_i$)
θ = The average processing time for customers at the machine. ($= \frac{\sum_{i=1}^{K} \lambda_i \theta_i}{\lambda}$)
ρ = $\frac{\lambda\theta}{m} = \sum_{i=1}^{K} \rho_i$ = average workload on the machine.

The objective function C used in our study is the total time through the virtual machine. It is calculated as the sum of the respective cost functions C_i for each part type being processed at the machine ($C = \sum_{i=1}^{K} C_i$). The individual cost components are defined in terms of the *average system time* T and the *mean processing time* θ_i. The optimization problem is:

$$\min_{0<\lambda\theta<m} \sum_{i=1}^{K} C_i(\theta_i, T)$$

To apply (stochastic) gradient optimization procedures, we need the gradient of C with respect to the individual mean processing times. The system time T can be written as the sum of actual processing time X and the queue delay D. Computation of the cost derivative can be reduced further using:

$$\begin{aligned}
\frac{dC}{d\theta_r} &= \sum_{i=1}^{K} \frac{dC_i(\theta_i, X, D)}{d\theta_r} \\
&= \sum_{\substack{i=1 \\ i\neq r}}^{K} \frac{dC_i(\theta_i, X, D)}{d\theta_r} + \frac{dC_r(\theta_r, X, D)}{d\theta_r} \\
&= \sum_{\substack{i=1 \\ i\neq r}}^{K} \frac{\partial C_i(\theta_i, X, D)}{\partial D}\frac{dD}{d\theta_r} + \frac{\partial C_r(\theta_r, X, D)}{\partial \theta_r} + \frac{\partial C_r(\theta_r, X, D)}{\partial X}\frac{dX}{d\theta_r} + \frac{\partial C_r(\theta_r, X, D)}{\partial D}\frac{dD}{d\theta_r} \\
&\qquad\qquad (\text{ As } \frac{d\theta_i}{d\theta_r} = \frac{dX}{d\theta_r} = 0 \quad \forall i \neq r) \\
&= \sum_{i=1}^{K} \frac{\partial C_i(\theta_i, X, D)}{\partial D}\frac{dD}{d\theta_r} + \frac{\partial C_r(\theta_r, X, D)}{\partial \theta_r} + \frac{\partial C_r(\theta_r, X, D)}{\partial X}\frac{dX}{d\theta_r}
\end{aligned}$$

The term $dX/d\theta$ represents the change in the actual processing time X due to the change in the mean processing time θ. For a smooth distribution function $F_\theta(X)$ for the processing time X, the derivative can be computed using the relation

$$\frac{dX}{d\theta} = -\left(\frac{\partial F_\theta(X)}{\partial \theta}\right) / \left(\frac{\partial F_\theta(X)}{\partial X}\right)$$

For the case of exponentially distributed service times, X with mean θ, the derivative is X/θ.

To understand the algorithm for derivative estimation, consider the simpler case of estimating $dD/d\theta_r$. Suppose we have a single server with multi-part arrivals. Assume that we only perturb the service parameter of a single part type (say part type 1). Under normal conditions, for the n^{th} part arrival in a specific busy period, let D_n and X_n be the queue delay and the processing time respectively. The new queue delay for the perturbed queue, i.e., with $\theta_1 \leftarrow \theta_1 + \Delta\theta_1$, will be D_n plus the sum of the change in the processing times for all the type 1 parts processed prior to the present part arrival, in the current busy period. As discussed earlier, for each type 1 part, the change in the processing time, due to a change in the parameter, is given by $(dX/d\theta_1)\,\Delta\theta_1$. If we define $type(i)$ as the type of the i^{th} part, the change in the queue delay for the n^{th} customer is:

$$\Delta D_n = \sum_{\substack{i=1 \\ type(i)=1}}^{n-1} \left.\frac{dX}{d\theta_1}\right|_{X_i} \Delta\theta_1$$

If there are N part arrivals in an iteration interval (different from a busy period), the derivative $dD/d\theta_1$ can be estimated using

$$\frac{dD}{d\theta_1} \simeq \frac{1}{N}\sum_{n=1}^{N}\frac{\Delta D_n}{\Delta\theta_1} = \frac{1}{N}\sum_{n=1}^{N}\sum_{\substack{i=1 \\ type(i)=1}}^{n-1}\left.\frac{dX}{d\theta_1}\right|_{X_i}$$

If we change the processing parameter for more than one part type (say for all part types), then the respective partial derivatives can be computed using

$$\frac{\partial D}{\partial \theta_r} \simeq \frac{1}{N}\sum_{n=1}^{N}\frac{\Delta D_n(r)}{\Delta\theta_r} = \frac{1}{N}\sum_{n=1}^{N}\sum_{\substack{i=1 \\ type(i)=r}}^{n-1}\left.\frac{dX}{d\theta_r}\right|_{X_i} \qquad r = 1 \ldots K$$

Here $\Delta D_n(r)$ is the partial contribution in the net change in queue delay due to the change in θ_r. This analysis is for a single server. For m servers, the net delay change tracking is done separately for all servers and the derivative estimation is done over the whole m server queue

$$\frac{\partial D}{\partial \theta_r} \simeq \frac{1}{N}\sum_{n=1}^{N}\frac{\Delta D_n^s(r)}{\Delta\theta_r} = \frac{1}{N}\sum_{n=1}^{N}\sum_{\substack{i=1 \\ type(i)=r \\ server(i)=s=server(n)}}^{n-1}\left.\frac{dX}{d\theta_r}\right|_{X_i} \qquad r = 1 \ldots K$$

Here $server(i)$ represents the server-id of the server processing the i^{th} part and N is the total number of parts processed by the m servers in the current iteration period.

Our problem is to estimate the cost derivatives $dC/d\theta_r$. For each part arrival in an iteration period, we calculate the change in the cost and take the ensemble average at the end of the iteration period. For the n^{th} arrival of type r, with X_n and D_n as the processing and the queue delay respectively, the overall cost change is given by

$$\Delta C_n = \sum_{i=1}^{K} \left.\frac{\partial C_i(\theta_i, X, D)}{\partial D}\right|_{X_n, D_n} \Delta D_n^s(r) + \left.\frac{\partial C_r(\theta_r, X, D)}{\partial \theta_r}\right|_{X_n, D_n} \Delta\theta_r$$

$$+ \left.\frac{\partial C_r(\theta_r, X, D)}{\partial X_n}\right|_{X_n, D_n} \left.\frac{dX}{d\theta_r}\right|_{X_n} \Delta\theta_r$$

This net cost change is split into the component changes $\Delta C_n^r \;\; r = 1 \ldots K$, each of which can be attributed to the respective changes in the individual processing parameters (θ_r). At the end of the iteration period, the average cost change is used as the derivative estimated.
Formalizing the procedure as an algorithm, we have:

INITIALIZE:

for ($server = 1 \ldots m$) do
  for ($type = 1 \ldots K$) do
    ($\Delta D^{server}(type) \leftarrow 0$)

for ($type = 1 \ldots K$) do
  ($\Delta C(type) \leftarrow 0; \Sigma_{type} \leftarrow 0$)

AT COMPLETION OF PART(n) PROCESSING:

($s \leftarrow server(n); r \leftarrow type(n)$)

($\Sigma_r \leftarrow \Sigma_r + 1$)

for ($type = 1 \ldots K$) do
  ($\Delta C(type) \leftarrow \Delta C(type) + \left.\frac{dC_{type}}{dD}\right|_{X_n, D_n} \Delta D^s(r)$)

($\Delta C(r) \leftarrow \Delta C(r) + \left.\frac{dC_r}{d\theta_r}\right|_{X_n, D_n} + \left.\frac{dC_r}{dX}\right|_{X_n, D_n} \left.\frac{dX}{d\theta_r}\right|_{X_n}$)

($\Delta D^s(r) \leftarrow \Delta D^s(r) + \left.\frac{dX}{d\theta_r}\right|_{X_n}$)

if (s idle) then

for ($type = 1 \ldots K$) do
($\Delta D^s(type) \leftarrow 0$)

AFTER N COMPLETIONS:

for ($type = 1 \ldots K$) do
($\frac{dC}{d\theta_{type}} \simeq \frac{1}{\Sigma_{type}} \Delta C(type)$)

5.2.2 Optimization Based on Stochastic Approximation

The Robbins-Monro procedure was developed initially as a root finding method for a random function of a single variable. This procedure can be used as an stochastic optimization method by applying it to find the root of the gradient function. The original Robbins-Monro algorithm, as extended to the multi-dimensional case by Blum [7], permits treatment of multi-dimensional stochastic optimization problems.

For a given cost functional $J(Y)$, the optimization problem reduces to finding the root of $\nabla_Y J(Y^*) = 0$. Here $Y = \{y_1, \ldots y_k\}$ is the vector of random parameters, over which the optimization is carried out. The optimization is done by recursive reassignment of the control parameter vector Y to the series of random vectors $\{Y_n\}_{n=0}^{\infty}$. The series $\{Y_n\}_{n=0}^{\infty}$ is generated using the recursive relation

$$Y^{n+1} = Y^n + A_n \widehat{\nabla}_Y J(Y^n).$$

Here $\widehat{\nabla}_Y J(Y^n)$ is the estimate of the noisy gradient and $A_n = \mathrm{diag}[a_n^1, \ldots a_n^k]$.

Blum [6, 7] proved that if we define the sequences $\{a_n^i\}_{n=0}^{\infty}$, $i = 1 \ldots k$ such that

$$\begin{aligned} \lim_{n\to\infty} a_n^i &= 0 && \forall i \\ \sum_{n=1}^{\infty} a_n^i &= \infty && \forall i \\ \sum_{n=1}^{\infty} (a_n^i)^2 &< \infty && \forall i \end{aligned}$$

and $\widehat{\nabla}_Y J(Y^n)$ is an unbiased estimate of $\nabla_Y J(Y)$, then the sequence $\{Y^n\}_{n=0}^{\infty}$ converges *a.s.* to Y^*.

5.2.3 Algorithm Implementation Results

To measure the performance of a virtual machine, we use component cost functions of the form:

$$C_i(\theta_i, X, D) = a_i(X + D) + \frac{b_i m}{\theta_i}$$

For a fixed number of servers (m) the cost function is convex, and for $0 < \rho < 1$, it has a unique minimum. The first term is proportional to the average system time ($X + D$) while the second term is proportional to the speed of each server ($1/\theta_i$). The tradeoff between costly, but faster servers, and longer system times creates a unique minimum for appropriate choices of the coefficients a_i and b_i. The average queue delay D is itself a function of $\{\theta_i\}_{i=1}^K$. The gradients can be computed for this cost functional by solving the following set of equations:

$$\frac{d\mathrm{E}[C]}{d\theta_r} = 0 = \sum_{i=1}^{K} a_i \frac{d\mathrm{E}[D]}{d\theta_r} + a_r \frac{d\mathrm{E}[X]}{d\theta_r} - \frac{b_r m}{\theta_r^2} \qquad r = 1 \ldots K$$

substitutingE$[X] = \theta_r$, we have

$$\frac{d\mathrm{E}[C]}{d\theta_r} = 0 = \sum_{i=1}^{K} a_i \frac{d\mathrm{E}[D]}{d\theta_r} + a_r - \frac{b_r m}{\theta_r^2} \qquad r = 1 \ldots K$$

For the multiple class, multi-server case we can define

$$\lambda = \sum_{i=1}^{K} \lambda_i, \qquad \theta = \frac{1}{\lambda} \sum_{i=1}^{K} \lambda_i \theta_i \qquad \text{and} \qquad \rho = \frac{1}{m} \sum_{i=1}^{K} \lambda_i \theta_i$$

From queuing theory, for a m server queue with intensity ρ and total arrival rate λ, the average queue delay is given by

$$\mathrm{E}[D] = P_0 \frac{(m\rho)^m}{m!(1-\rho)} \frac{\rho}{\lambda(1-\rho)}$$

where

$$P_0 = \left[\sum_{i=0}^{m-1} \frac{(m\rho)^i}{i!} + \frac{(m\rho)^m}{m!(1-\rho)} \right]^{-1}$$

The optimization problem can be reduced to solving the following set of equations for $\{\theta_r\}_{r=1}^K$

$$\frac{d\mathrm{E}[C]}{d\theta_r} = 0 = \sum_{i=1}^{K} a_i \frac{d\mathrm{E}[D]}{d\theta_r} + a_r - \frac{b_r m}{\theta_r^2} \qquad r = 1 \ldots K$$

$$\iff \qquad \frac{1}{\lambda_r} \left[\frac{b_r m}{\theta_r^2} - a_r \right] = constant = \frac{1}{\lambda} \sum_{i=1}^{K} a_i \frac{d\mathrm{E}[D]}{d\theta} \qquad r = 1 \ldots K$$

This allows us to compare theoretical values, calculated as shown above, to our simulation results.

The validity of our scheme also follows from the results of De Smit [47], Fu and Hu [15] and Blum [6, 7]. Specifically, De Smit [47] proved that the multiple class, multi-server queue with Poisson arrivals and exponentially distributed times is equivalent to the GI/H_K/m queue. Here H_K denotes the hyperexponentially distributed service times, with distribution function of the form:

$$F_\theta(X) = \begin{cases} \sum_{i=1}^{K} \frac{\lambda_i}{\lambda}(1 - e^{-\frac{X}{\theta_i}}) & X \geq 0 \\ 0 & X < 0 \end{cases}$$

Fu and Hu [15] proved consistency of the gradient estimates for a GI/G/m queue. Those results are directly applicable to the GI/H_K/m queue; and so, the gradient estimates are consistent. By an appropriate choice of the sequence a_n^i for all i, our optimization parameters $\{\theta_i^n\}_{n=0}^{\infty}, (i = 1 \ldots K)$ will converge to the optimum $\theta_i^*(i = 1 \ldots K)$. For our simulation we chose $a_n^i = a_0^i/n, \forall i$ which satisfies all the stochastic approximation conditions discussed earlier.

This algorithm was used to control the processing rates for the test stations in the quality control unit of our example manufacturing system. Based on the process flow specifications for the four parts, the arrival rates were calculated for the test stations. The processing times at the insert stations were assumed to be exponentially distributed, hence the arrival process is Poisson with the correspondingly calculated arrival rates. The simulation was run using the following set of arrival rates (in parts/hr):
$\lambda_1 = 120$,
$\lambda_2 = 60$,
$\lambda_3 = 40$,
and $\lambda_4 = 40$.
Discounting the initial transients in the plots, the processing times converge to a steady state value. Figures 5.6 through 5.11 show the trace of optimal processing times for the four parts on the three test machines. To check the online control aspect, we ran a second simulation which was perturbed by switching part arrival rates at time 400.0. The new arrival rates for the four part types were set to 40,40,120 and 60 (parts/hr) respectively. Figure 5.12 shows the trace of processing times for machine M/C-5. We observe that a change in arrival rates is accordingly offset by the changes in the processing times such that the operating cost is still minimized.

5.3 Conclusions

In this chapter, we presented operation level reconfigurable control algorithms. Since real manufacturing systems, in general, operate in a hybrid mode, i.e. with both continuous and batch processing modes, we presented algorithms for both

cases. For the batch mode operation, the overall schedule deviation was defined in terms of the deviation in timings of the individual operation schedule. Hence, we developed a dynamic programming based impulse control algorithm for online control of processing times for a single operation. For the continuous mode case, we developed a perturbation analysis based online processing rate control algorithm. We presented results for a single virtual machine. The algorithm can be applied for online control of multi-machine manufacturing systems operating in a continuous mode. As mentioned earlier, the emphasis in this thesis was on development of control algorithms which can be implemented on real flexible manufacturing systems. The two algorithms discussed in this chapter can be easily programmed into the existing (suitably modified, if necessary) numerical controllers. The extensions of these algorithms to other discrete event systems and the conclusions of this research will be presented in Chapter 6.

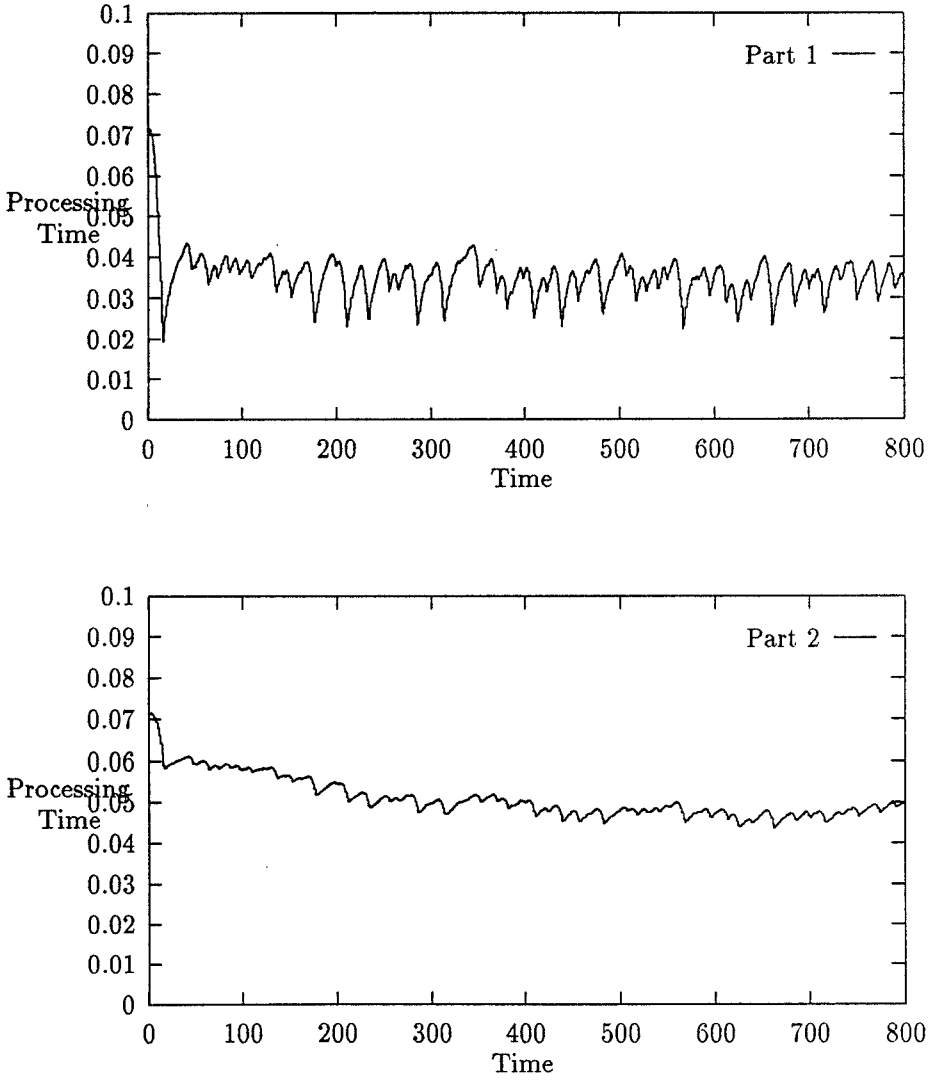
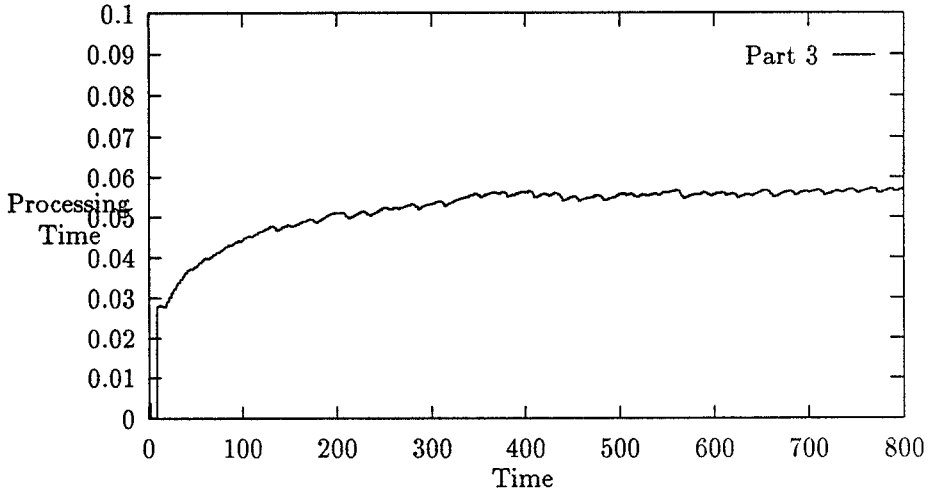


Figure 5.6: Actual Simulation Run (M/C-5).



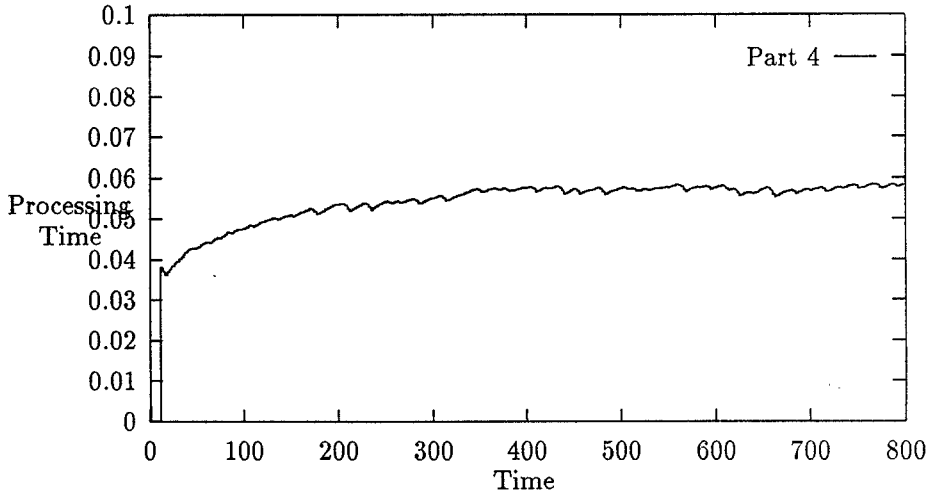


Figure 5.7: Actual Simulation Run (M/C-5).

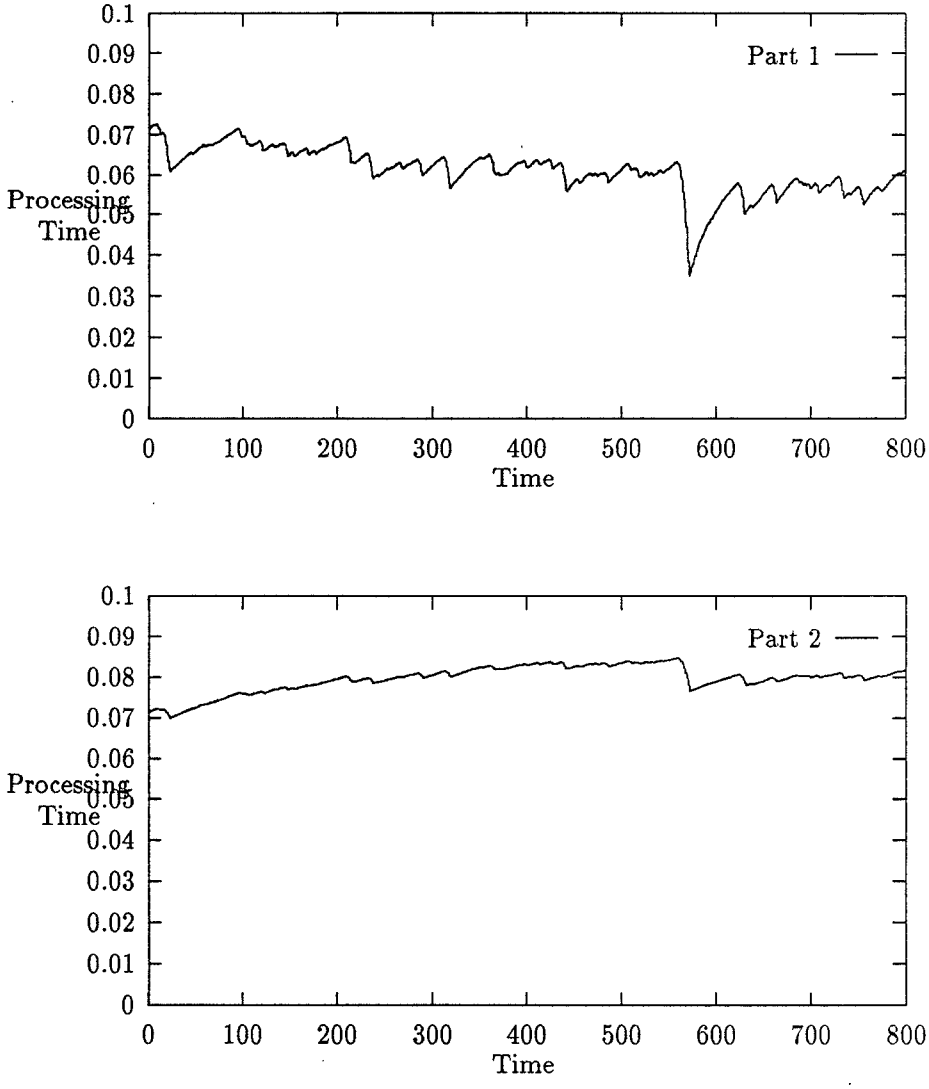
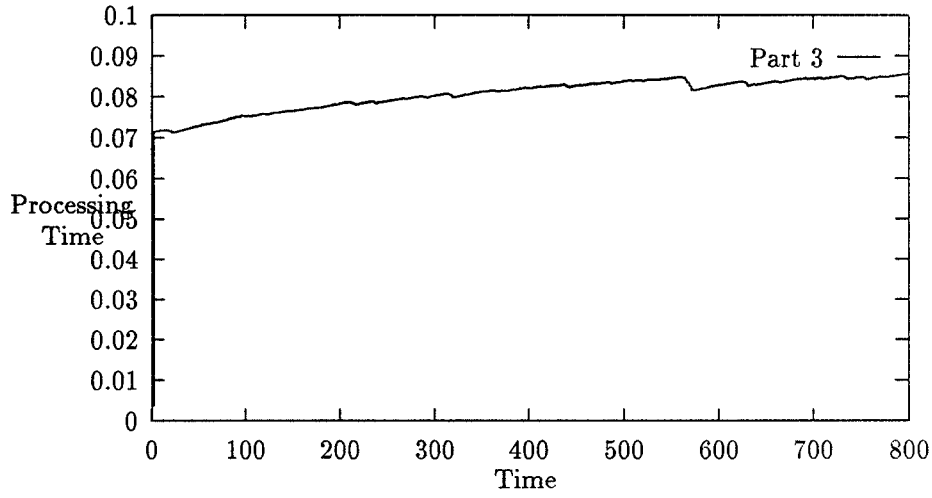


Figure 5.8: Actual Simulation Run (M/C-6).



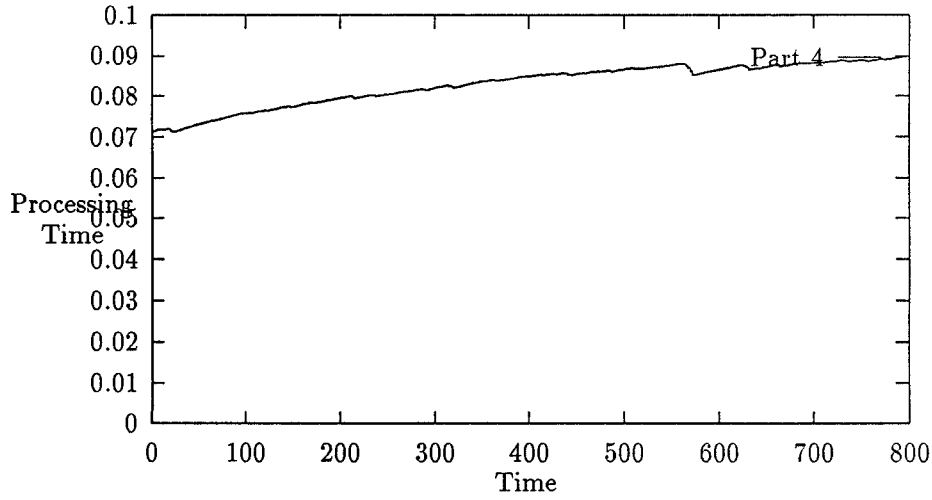


Figure 5.9: Actual Simulation Run (M/C-6).

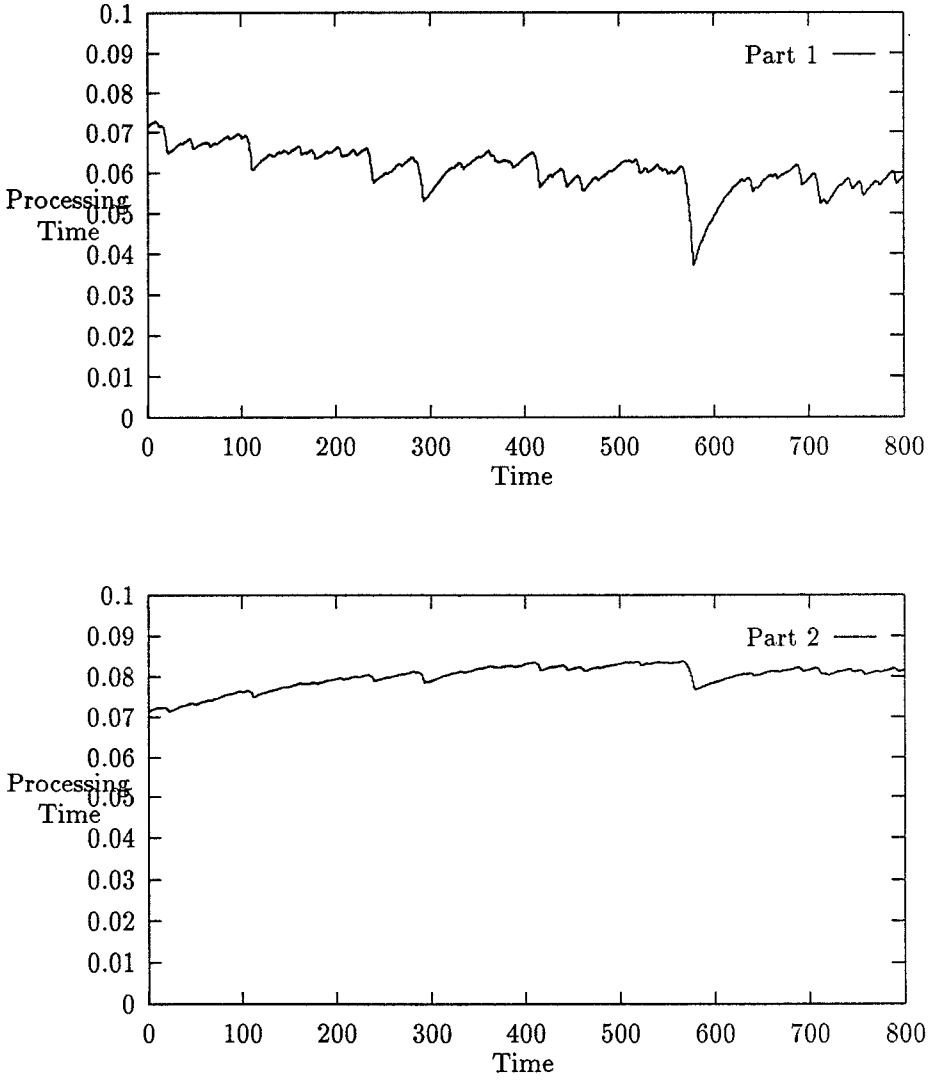
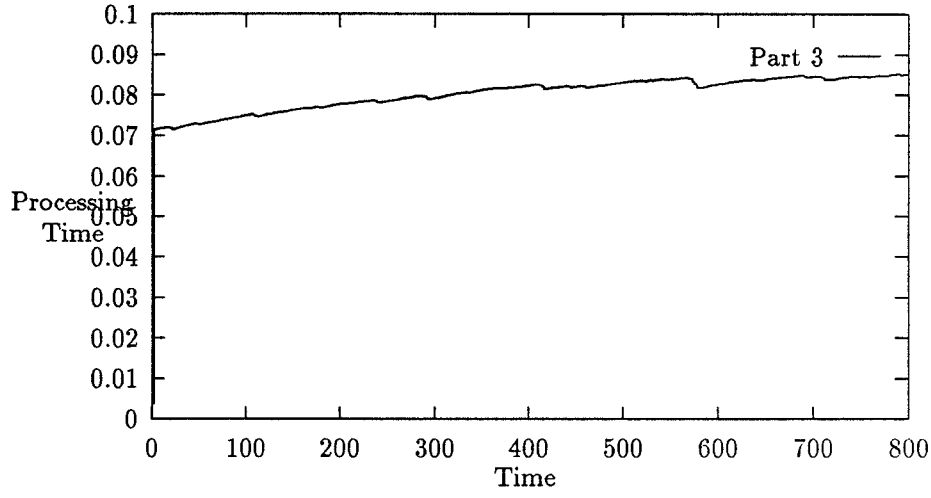


Figure 5.10: Actual Simulation Run (M/C-7).



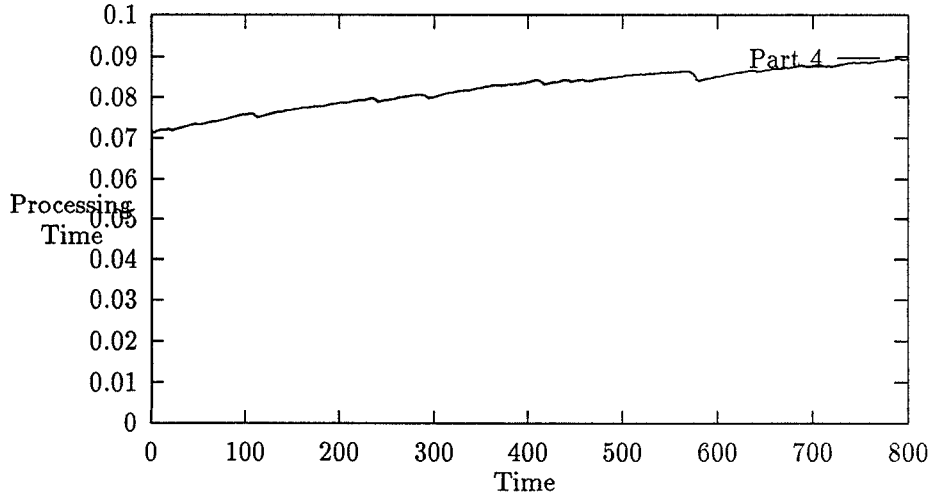


Figure 5.11: Actual Simulation Run (M/C-7).

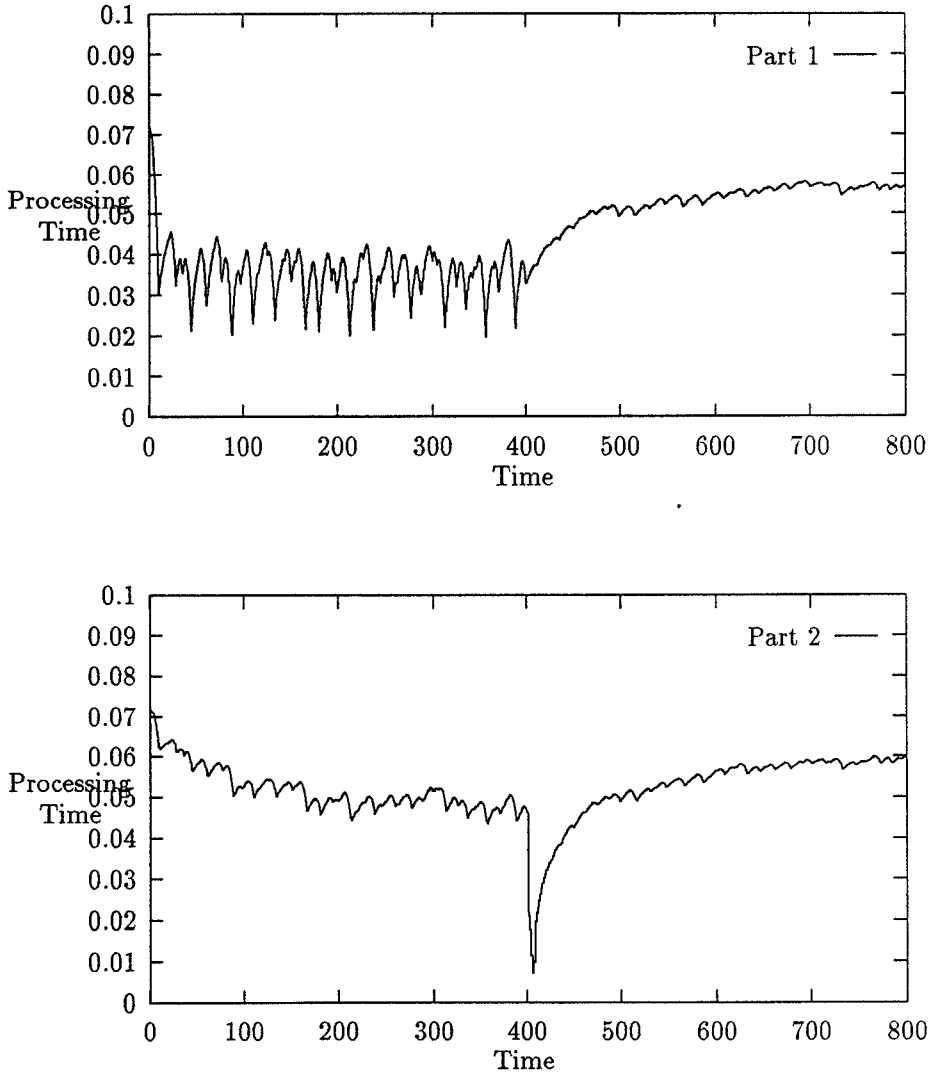


Figure 5.12: Actual Simulation Run for perturbed system (M/C-5).

CHAPTER
SIX

Conclusions

In this chapter, we summarize our research contribution and try to put our research into perspective in terms of practical implementation in a real manufacturing environment.

In Chapter 2, we presented a description of the TI manufacturing facility at Johnson City, TN. We presented a mathematical model for a general manufacturing system and developed an optimization based operations scheduling scheme. The scheme was implemented as a software package $\mathcal{ABES}$, which was used to schedule numerically controlled machines at an actual manufacturing unit. The experience gained during the design, development and installation, of $\mathcal{ABES}$ was useful in understanding the various issues involved in real manufacturing systems. This provided insight into the reconfigurable control problem and led to the development of online control algorithms.

The hierarchical reconfigurable control scheme, presented in Chapter 3, provides an ideal framework for utilizing the inherent hierarchical structure in manufacturing systems for online control. The identification of the different reconfiguration control parameters, with the associated frequency of control application, lead to the hierarchical decomposition of the reconfigurable control problem. We presented a hybrid manufacturing system model for application and testing of the different control algorithms developed in subsequent chapters.

The online reactive scheduling scheme, presented in Section 4.1, was an extension of the predictive scheduling scheme presented in Chapter 2. This was not implemented at the actual manufacturing facility, but the basic infrastructure exists for implementation of this algorithm. This scheme can be easily implemented in a modern manufacturing system. The idea is to use the built in microcontrollers to continuously update the operation status at the central controller/rescheduler. The continuously evolving actual schedule, and the random arrivals of jobs, can be incorporated by the rescheduler to generate schedules online. The simulated annealing based optimization algorithm can be suitable modified for real time optimization on a multi-processor platform. In conclusion, this reactive scheduling scheme can be very effectively implemented and used in a real manufacturing system and is one of the strong points of this research.

The process control algorithm, presented in Chapter 4, was based on heuristics developed in working with Texas Instruments Inc., Johnson City. The lower

level Metropolis process, in the reactive operations scheduling scheme, was used to re-sequence operations on different machines, while the higher level process was used for re-assignment of process flows. Since the process control algorithm is based on the same closed loop configuration, as the reactive operations scheduling scheme, it can be implemented easily using the same architecture, and be used for online reconfigurable process control of manufacturing systems.

The development of operation level control algorithms, presented in Chapter 5, is another important contribution of this thesis. The Poisson noise driven process formulation of the discrete part processing operation was used to develop the impulse control based algorithm for online control of processing times. The theoretical basis to solve this problem, and then using the obtained results to control the actual simulation, completes the "Real Problem $\Rightarrow$ Mathematical Model $\Rightarrow$ Analytical Solution $\Rightarrow$ Application" loop. The perturbation analysis based continuous control algorithm also provides a similar mix of theory and application. Since algorithms were presented for both modes of operation, i.e. continuous and batch, we have a framework for online operation level control of a whole spectrum of different manufacturing systems.

The algorithms developed here were implemented and tested models using models of an actual flexible manufacturing system. The formulation of the algorithms is quite general and can be applied, with suitable minor modifications, to other discrete event systems. In a communication network, the impulse control problem can be viewed as the optimal choice of different channels with different, but fixed, channel capacities. The perturbation analysis based continuous control can be used for optimizing processing rates for the case of Order Processing Systems. The reactive scheduling scheme can be suitably modified to take into account continuous processing and can be used for online scheduling/routing in communication networks, order processing systems, etc.

The concept of reconfiguration of system parameters can be extended to other aspects of manufacturing systems. Dynamic optimal allocation of machine buffers, optimal specification of schedule cost on the basis of changing production goals, etc., are some of the other reconfiguration attributes which can be studied. Use of the impulse control framework for part routing/re-routing warrants further investigation. The application of these algorithms for control of other discrete event systems will raise some interesting computational problems, which need to be investigated.

The impulse control problem didn't consider random processing times. Only random failures were taken into account. If we introduce Wiener processes in our mathematical formulation, then this problem can be posed as a optimal control problem for jump diffusion processes. The extension of the impulse control problem, from a single physical machine to a virtual machine will be an interesting problem for further research.

In this thesis, an attempt was made to understand the various modeling and control issues involved in a real world manufacturing system and to design

control algorithms. The on-site experience at the Texas Instruments, Inc., PC board assembly facility at Johnson City, TN., provided deep insight in this regard. The algorithms developed, for online control of manufacturing systems, are easily implementable using the existing technology, both in terms of hardware and software. Is the real world manufacturing environment ready for the investment, both in terms of time and money, and the ensuing change? This is the million dollar question.

A. Numerical Computation of Optimal Control

Using the framework presented by Gonzalez, Rofman *et al* [20, 22, 21, 41] and Kabbaj *et al* [14], the optimal *cost-to-go* value functions can be calculated as a maximal element of a set of functions $\mathcal{V}$, defined as below:
$\mathcal{V} = \{v_i(x,t) : \mathbb{R}^+ \times [0,T] \rightarrow \mathbb{R} \,|i = 1 \ldots l\}$, where

$$C_r(x,\mu_i) + \frac{\partial v_i(x,t)}{\partial t} + \frac{N}{T}\mu_i \frac{\partial v_i(x,t)}{\partial x} + \sum_{k=1}^{K} \lambda_k \left[v_i(x + h_k(t), t) - v_i(x,t)\right] \geq 0$$

$$v_i(x,t) \leq \min_{\substack{i \neq j \\ j=1\ldots l}} \left[v_j(x,t) + C_s(\mu_i,\mu_j)\right]$$

$$v_i(x,T) = C_f(x,T); \qquad \forall x \in \mathbb{R}^+$$

$U_i(x,t)$ is the maximum element of the set $\mathcal{V}$ under the following partial ordering on $\mathcal{V}$:

$$v^1 \leq v^2 \Longleftrightarrow v_i^1(x,t) \leq v_i^2(x,t), i = 1 \ldots l; \forall (x,t) \in \mathbb{R}^+ \times [0,T].$$

For the purpose of numerical computation, we restrict our state space $\mathbb{R}^+ \times [0,T]$ to $[0,X] \times [0,T]$. We further discretize our restricted domain using a $(P+1) \times (Q+1)$ grid Ω with grid points $(x_p, t_q); p = 0 \ldots P; q = 0 \ldots Q$ ($\Delta x = \frac{X}{P}; x_p = p.\Delta x; \Delta t = \frac{T}{Q}; t_q = q.\Delta t$).
Using the explicit finite difference approximation for the partial derivatives:
$\frac{\partial v_i}{\partial t}(x_p,t_q) = (v_i(x_p,t_{q+1}) - v_i(x_p,t_q))/\Delta t$, and
$\frac{\partial v_i}{\partial x}(x_p,t_q) = (v_i(x_{p+1},t_q) - v_i(x_p,t_q))/\Delta x$
along with the boundary condition $\frac{\partial v_i}{\partial x}(x_P,t_q) = 0$ $(q = 0 \ldots Q)$,
we have the following discretized version of the original problem:
Find the maximum element of the set $\mathcal{V}^d$, where

$$\mathcal{V}^d = \{v_i(x_p,t_q) : \Omega \rightarrow \mathbb{R} \,|i = 1 \ldots l\}$$

and

$$C_r(x_p,\mu_i) + \frac{v_i(x_{p+1},t_q) - v_i(x_p,t_q)}{\Delta x} + \frac{N}{T}\mu_i \frac{v_i(x_p,t_{q+1}) - v_i(x_p,t_q)}{\Delta t}$$

$$+\sum_{k=1}^{K}\lambda_k\left[v_i(x_{p+\lfloor\frac{h_k(t_q)}{\Delta x}\rfloor},t_q)-v_i(x_p,t_q)\right] \geq 0$$

$$v_i(x_p,t_q) \leq \min_{\substack{i\neq j\\ j=1\ldots l}}\left[v_j(x_p,t_q)+C_s(\mu_i,\mu_j)\right]$$

$$v_i(x_p,T) = C_f(x_p,T); \qquad p=0\ldots P.$$

with the partial ordering:

$$v^1 \leq v^2 \Longleftrightarrow v_i^1(x_p,t_q) \leq v_i^2(x_p,t_q),\ i=1\ldots l;\ \forall(x_p,t_q)\in\Omega.$$

Rearranging terms, we can rewrite the Dynamic programming inequality in the following form:

$$\begin{aligned} v_i(x_p,t_q) &\leq \Psi_i(x_p,t_q) \\ &\triangleq \frac{1}{\frac{1}{\Delta x}+\frac{N}{T}\mu_i+\sum_{k=1}^{K}\lambda_k}\left[C_r(x_p,\mu_i)+\frac{1}{\Delta x}v_i(x_{p+1},t_q)+\right. \\ &\quad \left.\frac{N}{T}\mu_i v_i(x_p,t_{q+1})+\sum_{k=1}^{K}\lambda_k v_i(x_{p+\lfloor\frac{h_k(t_q)}{\Delta x}\rfloor},t_q)\right] \end{aligned}$$

Combining this with switching inequality, we get:

$$v_i(x_p,t_q) \leq \min\left[\Psi_i(x_p,t_q),\ \min_{\substack{i\neq j\\ j=1\ldots l}}\left(v_j(x_p,t_q)+C_s(\mu_i,\mu_j)\right)\right]$$

Again using results from the references mentioned above, we have the solution as the fixed point of the operator $\mathcal{A}$

$$\bar{v}_i(x_p,t_q) = \mathcal{A}\bar{v}_i(x_p,t_q) \triangleq \min\left[\Psi_i(x_p,t_q),\ \min_{\substack{i\neq j\\ j=1\ldots l}}\bar{v}_j(x_p,t_q)+C_s(\mu_i,\mu_j)\right]$$

and the discretized solutions $v_i(x_p,t_q),\ \forall(x_p,t_q)\in\Omega, i=1\ldots l$, converge uniformly to $U_i(x_p,t_q)$, i.e.

$$\lim_{\Delta x,\Delta t\to 0}\max_{(x_p,t_q)\in\Omega}|v_i(x_p,t_q)-U_i(x_p,t_q)| = 0$$

The fixed point of the operator A can be found out using a recursive algorithm, which utilizes the special form of Ψ_i, and calculates $\bar{v}_i(x_p,t_q)$; ($p=0\ldots P$) using the previously calculated values of $\bar{v}_i(x_p,t_{q+1})$($p=0\ldots P$).

Initialize:

$$\hat{v}_i(x_p,t_q) = v_i(x_p,t_q) = 0;\ p=0\ldots P;\ q=0\ldots(Q-1);$$

$\widehat{v}_i(x_p, t_Q) = v_i(x_p, t_Q) = C_f(x_p, t_Q);\ p = 0 \ldots P;$ *(Note:* $t_Q = T$*).*

Begin Algorithm:

For q = (Q-1) to 0 do

$\star$ *For p = 0 to (P-1) do*

For i = 1 to I do

$v_i(x_p, t_q) \leftarrow \mathcal{A}v_i(x_p, t_q)$

if $(v_i(x_p, t_q) \neq \widehat{v}_i(x_p, t_q))\ \ (p = 0 \ldots P)$

then $\widehat{v}_i \leftarrow v_i(x_p, t_q)(p = 0 \ldots P);$ *go to* $\star$

$U_i(x_p, t_q) \leftarrow \widehat{v}_i(x_p, t_q);\ \ i = 1 \ldots l;\ \ \forall (x_p, t_q) \in \Omega$

End Algorithm.

The convergence of the above algorithm to the fixed point of the non-linear operator A is discussed in [20, 22, 21].

In this appendix, we presented the theoretical background for the impulse control algorithm presented in Chapter 5. We stated the methodology, without giving the proofs. The problem of serial multilevel production system control, presented in [14], has a general jump diffusion process model. Our formulation is a special case, and the results on maximal sets presented in [14] are directly applicable here. The discretized problem results are simple extensions of the results presented in [20, 22, 21]. Again, here the emphasis was to give an outline of the methodology used, without giving rigorous analytical proofs.

BIBLIOGRAPHY

[1] G. Murari A. Villa and G. Minucciani. Introduction: Decisional stuctures in automated manufacturing. In A. Villa and G. Murari, editors, *Proceedings of IFAC Workshop on Decisional Structures in Automated Manufacturing, Sept. 1989*, 1990.

[2] F. Baccelli, N. Bambos, and J. Walrand. Flow analysis of stochastic marked graphs. In *Proc. 28th Conf. on Decision and Control*, pages 1528–1531, 1989.

[3] F. Baccelli and A. M. Makowski. Queuing models for systems with synchronization constraints. *Proceedings of the IEEE*, 77(1):138–161, Jan. 1989.

[4] A. Bensoussan and J.-L. Lions. *Impulse Control and Quasi-Variational Inequalities.* Gauthier-Villars, 1984.

[5] T. Bielecki and P. R. Kumar. Optimality of zero-inventory policies for unreliable manufacturing systems. *Operations Research*, 36(4):532–541, Jul.-Aug. 1988.

[6] J. R. Blum. Approximation methods which converge with probability one. *Annals of Math. Stat.*, 25:182–386, 1954.

[7] J. R. Blum. Multidimensional stochastic approximation methods. *Annals of Math. Stat.*, 25:737–744, 1954.

[8] P. J. Burke. The output of a queuing system. *Operations Research*, 4:699–704, 1956.

[9] X. R. Cao and Y. C. Ho. Sensitivity analysis and optimization of throughput in a production line with blocking. *IEEE Transactions on Automatic Control*, 32(11):959–967, Nov. 1987.

[10] G. Cohen, D. Dubois, J. P. Quadrat, and M. Viot. A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing. *IEEE Transactions on Automatic Control*, AC-30(3):210–220, March 1985.

[11] G. Cohen, P. Moller, J. P. Quadrat, and M. Viot. Algebraic tools for the performance evaluation of discrete event systems. *Proceedings of the IEEE*, 77(1):39–58, Jan. 1989.

[12] J. S. Dhingra. Users guide to **ABES**. Technical report, Systems Research Center, University of Maryland, 1990. Unpublished.

[13] W. Findeisen *et al. Control and Corodination in Hierarchical Systems.* John Wilwy & Sons, 1980.

[14] J.-L. Menaldi F. Kabbaj and E. Rofman. Variational approach of serial multi-level production/inventory systems. Technical Report Rapport de Recherche 692, 1987.

[15] M. C. Fu and Jian-Qiang Hu. Consistency of infinitesimal perturbation analysis for the gi/g/m queue. *European Journal of Operational Research*, 54:121–139, 1991.

[16] S. B. Gershwin. Hierarchical flow control: A framework for scheduling and planning discrete events in manufacturing systems. *Proceedings of the IEEE*, 77(1):195–209, Jan. 1989.

[17] P. W. Glynn. A gsmp formalism for discrete event systems. *Proceedings of the IEEE*, 77(1):14–23, Jan. 1989.

[18] W. B. Gong. Smoothed perturbation analysis of markovian queuing networks. In *Proc. American Control Conference*, pages 456–461, 1988.

[19] W. B. Gong and Y. C. Ho. Smoothed (conditional) perturbation analysis of discrete event dynamical systems. *IEEE Transactions on Automatic Control*, 32(10):858–866, Oct. 1987.

[20] R. Gonzalez and E. Rofman. On deterministic control problems: An approximation procedure for the optimal cost i, ii. *SIAM Journal on Control and Optimization*, 23(2):242–266,267–285, 1985.

[21] R. Gonzalez and E. Rofman. Computational methods in scheduling optimization. In *Fermat Days 85: Mathematics for Optimization, J.B. Hiriart-Urruty Ed., Math. Studies 129.*, pages 135–156, 1986.

[22] R. Gonzalez and E. Rofman. On stochastic control problems: an algorithm for the value function and the optimal policy. In *13^{th} IFIP Conference on Septeur Modelling and Optimization*, 1987.

[23] I. Hatano, K. Yamagata, and H. Tamura. Modelling and on-line scheduling of flexible manufacturing systems using stochastic petri nets. *IEEE Transactions on Software Engineering*, 17(2):126–132, Feb. 1991.

[24] H. P. Hillon and J.-M. Proth. Performance evaluation of job-shop systems using event-graphs. *IEEE Transactions on Automatic Control*, 34(1):3–9, Jan. 1989.

[25] Y. C. Ho. Performance evaluation and perturbation analysis of discrete event dynamic systems. *IEEE Transactions on Automatic Control*, 32(7):563–572, Jul. 1987.

[26] Y. C. Ho. Extensions of infinitesimal perturbation analysis. *IEEE Transactions on Automatic Control*, 33(5):427–438, May 1988.

[27] Y. C. Ho. Dynamics of discrete event systems. *Proceedings of the IEEE*, 77(1):3–6, Jan. 1989.

[28] K. Inan and P. Varaiya. Finitely recursive process models for discrete event systems. *IEEE Transactions on Automatic Control*, 33(7):626–639, Jul. 1988.

[29] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Math. Stat.*, 23:462–466, 1952.

[30] J. Kimemia and S. B. Gershwin. An algorithm for the computer control of a flexible manufacturing system. *IEE Transactions*, 15(4):353–362, Dec. 1983.

[31] Y. T. Leung and R. Suri. Convergence of a single run optimization algorithm. In *Proc. American Control Conference*, pages 440–443, 1988.

[32] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1091, 1953.

[33] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli. Convergence and finite-time behavior of simulated annealing. In *Proc. 24th IEEE Conf. on Decision and Control*, pages 761–767, 1985.

[34] K. L. Musser, J. S. Dhingra, and G.L. Blankenship. Optimization based job shop scheduling. Technical Report TR-91-9, Systems Research Center, Univerisity of Maryland, 1991. to appear in IEEE Transactions on Automatic Control.

[35] Keith L. Musser. *A Distributed Algorithm for Scheduling in Discrete Manufacturing.* PhD thesis, University of Maryland, College Park, 1992.

[36] J. S. Ostroff and W. M. Wonham. A temporal logic approach to real time control. In *Proc. 24th Conf. on Decision and Control*, pages 656–657, 1985.

[37] J. S. Ostroff and W. M. Wonham. A framework for real-time discrete event control. *IEEE Transactions on Automatic Control*, 35(4):386–387, Apr. 1990.

[38] K. M. Passino and P. J. Antsaklis. Solutions to optimal control problems for discrete event systems. *preprint*, 1990.

[39] J. R. Perkins and P. R. Kumar. Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems. *IEEE Transactions on Automatic Control*, 34(2):139–148, Feb. 1989.

[40] E.H.L. Aarts P.J.M. van Laarhoven and J.K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):113–125, Jan.-Feb. 1992.

[41] E. Rofman R. Gonzalez and C. Sagastizabal. Optimal control of aborescent multilevel inventory production systems. Technical Report Rapport de Recherche 1066, 1990.

[42] P. J. G. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, Jan. 1987.

[43] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, Jan. 1989.

[44] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Math. Stat.*, 22:400–407, 1951.

[45] M. Sajkowski. Protocol verification using discrete event models. In *Discrete Event Systems: Models and Applications*, pages 100–114, 1987.

[46] A. Sharifnia. Production control of a manufacturing system with multiple machine states. *IEEE Transactions on Automatic Control*, 33(7):620–625, Jul. 1988.

[47] J. H. A. De Smit. The queue GI/M/s with customers of different types or the queue GI/H_m/s. *Advances in Applied Probability*, 15:392–419, 1983.

[48] R. Suri. Infinitesimal perturbation analysis for general discrete event systems. *Journal of the ACM*, 34(3):686–717, July 1987.

[49] R. Suri. Perturbation analysis: The state of the art and research issues explained via the g/g/1 queue. *Proceedings of the IEEE*, 77(1):114–137, Jan. 1989.

[50] R. Suri and Y. T. Leung. Single run optimization of discrete event simulations - an emperical study using the m/m/1 queue. *IIE Transactions*, 21:35–48, 1989.