REAL-TIME ADAPTIVE SCHEDULING IN FLEXIBLE MANUFACTURING SYSTEMS

John R. Birge
Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, MI 48109

August, 1985

Technical Report 85-26

Real-Time Adaptive Scheduling in Flexible Manufacturing Systems

John R. Birge
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109

Abstract: An efficient scheduling method for flexible manufacturing systems must be able to respond to system disruptions. These disruptions may involve resource unavailability and demand changes. We consider an approach to respond to such disruptions in real time. The goal of the response is to minimize costs. We place particular emphasis on costs due to lateness.

1. Introduction

Flexible manufacturing systems (FMS) include automated material handling capabilities and computer-controlled machines that can perform various tasks. The variety of paths that a part can take through this network makes the analysis of such systems complex. This complexity is compounded by the possibilities for machine and other resource failures that dynamically change the system state. System flexibility, however, allows for quick response to changes in the system state. In this paper, we describe a general method for responding to these state changes.

In our analysis, we concentrate on the dynamic scheduling of parts on machines. We assume that this process is one part of a hierarchical decision process, as in Kimemia and Gershwin [1983], Hildebrandt and Suri [1980] and Morton and Smunt [1985]. The higher layers of control have decided on system layout, aggregate production and an initial schedule for all parts up to a certain horizon to follow. This pre-schedule is assumed to consider inventory, set-up and handling costs. The dynamic scheduler attempts to return from a disruption state to the pre-schedule in real-time and to minimize costs which emphasize tardiness. We, therefore, assume that batches of finished parts have definite shipping dates and that lateness costs are quantifiable (for example, increased freight and expedition charges).

This viewpoint contrasts with other scheduling analyses, such as Gershwin, Akella, and Choong [1984] and Hildebrandt and Suri, that use control techniques to achieve generally high production rates and low in-process inventories. It is also distinct from planning and loading analyses as in Stecke [1983] and Wittrock [1984]. We consider each part as a member of a shipping batch of finished parts or assemblies. This position is consistent with Morton and Smunt and the methods in Morton, Rachamadugu, and Vepsalainen [1984]. Our approach is, however, to allow certain parts of the pre-schedule to remain fixed while others are changed. This general idea also appears in Chang, Sullivan, and Bagchi [1984].

The basic model and its justification are given in Section 2. A method for calculating costs for the scheduling algorithm's use is presented in Section 3. Section 4 discusses the definition of problem sets on which to apply the scheduling algorithms. Section 5 describes some implementation questions, and Section 6 presents conclusions.

2. Match-up Scheduling

The basic model assumes a pre-schedule that would be followed in the absence of disruptions. In implementing this pre-schedule, a disruption occurs. The following procedure is then invoked.

Match-Up Real-Time Scheduling Algorithm (MURTSA)

1. Determine the effects of the disruption and define new internal cost parameters.

2. Determine a portion of the pre-schedule to release.

3. Reschedule the released operations to minimize costs.

Some iteration may occur among the three steps as different schedules are assessed. Each of the steps is discussed below. Note that "disruption" can be defined as any change in system state so that MURTSA can be constantly operating and optimizing over some subset of parts and resources.

Economic turnpike theory provides a theoretical justification for MURTSA (see Bean and Birge [1985]). If the pre-schedule is optimal for an infinite horizon, the schedule has sufficient slack to recover from disruptions, and disruptions are well-spaced, then it is optimal to follow the MURTSA objective of returning to the pre-schedule in order to minimize a variety of costs, including tardiness. The time for this match-up is not specified, but bounds may be obtained for fixed times to match up.

These theoretical conditions are not always present in practice but MURTSA is flexible and allows for a range of responses depending on the problem definition in Step 2 and the cost definition in Step 1. The next sections describe these steps and possible implementations.

3. Calculating Internal Costs

The costs for lateness are assumed here to dominate other costs in determining an optimal return to the pre-schedule. We assume also that tardiness costs only occur at shipping for finished parts. Since many parts may be combined into a single assembly for shipping and since the FMS allows for parts to interact in many ways, the lateness of a single part or part-batch may affect several shipping quantities. We present a project network for determining these effects.

For simplicity, we assume that a set of parts is processed together as a <u>tray</u> or <u>transfer batch</u>. The transfer batch may consist of different part-types, but it is assigned to a specific shipment or shipments that have due dates. Tardiness costs are incurred if the shipment is sent after its due date. These costs may vary according to the amount of tardiness as alternative methods of shipping become necessary.

The goal of Step 1 of MURTSA is to determine the costs that would be incurred if an operation of a transfer batch were delayed. This definition assumes that some portion of the schedule is fixed and that costs may be incurred on all shipments. The costs are calculated by determining a vector for every operation that represents its earliest possible start time and its latest possible start time before a shipment is delayed. For operation i of transfer batch j, this vector is

$$(e_{ij}, \ell_{ij}^{1}, \ldots, \ell_{ij}^{K}), \qquad (1)$$

where e_{ij} is the earliest time i can start, ℓ_{ij}^{k} is the latest time i can start before shipment k is delayed, and K shipments may be affected by ij. (In practice, a pointer indicating the shipment indices of the K lowest ℓ_{ij}^{k} values would be stored.)

The vector in (1) is defined by a project network as illustrated in Figure 1. We assume transfer batches T1, T2, and T3 are used in shipments S1 and S2. T2 is scheduled before T3 on machine A, and T1 is scheduled before T2 on machine B. T2 is processed on B after processing on A and some material handling. The solid arcs correspond to processing on machines A and B and to material handling (MH). Arc lengths are processing times. The dotted arcs correspond to scheduling precedences. Early times are calculated by proceeding forward in the network, and late times are processed by backward recursion. The due dates for S1 and S2 are both assumed to be 5.

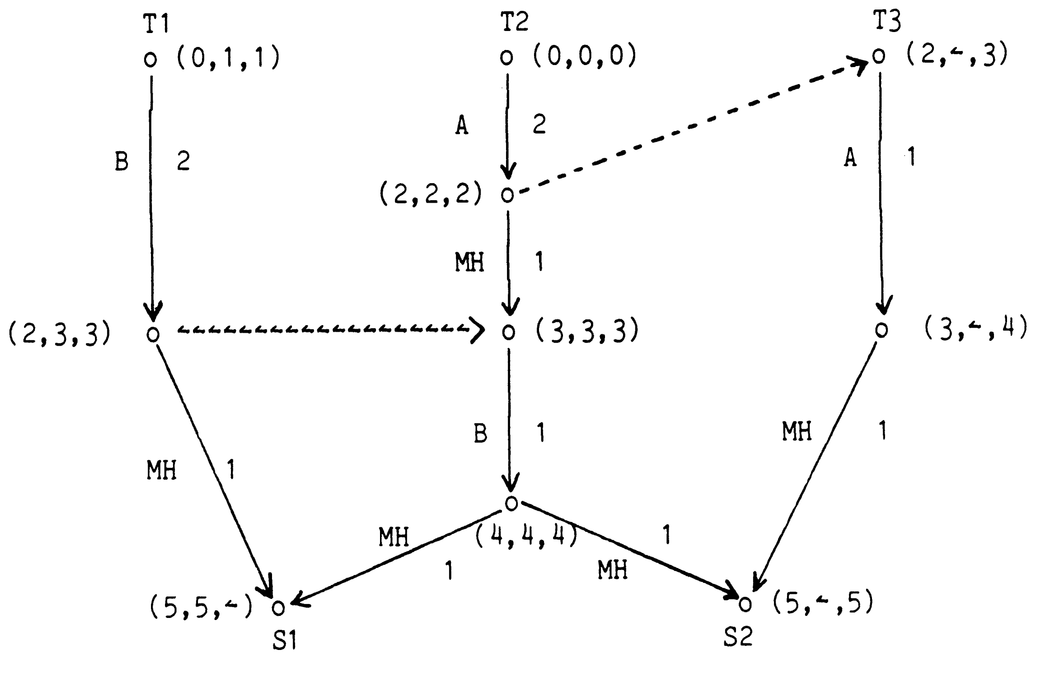


Figure 1. Cost Network.

Note that T2 in Figure 1 is on the critical path for both S1 and S2. Any delay in processing T2 results in both lateness costs for S1 and S2. This information would be used to schedule T2 in the event of a disruption to minimize the tardiness effect.

In using the project network definition after a disruption, the e_{ij} values may pass certain ℓ_{ij}^{k} values. In this case, some part of the schedule must be released in order to allow for tardiness avoidance. Releasing part of the schedule is equivalent to eliminating dotted arcs for schedule precedences. After the elimination of these arcs in Step 2 in MURTSA, new values of e_{ij} and ℓ_{ij}^{k} are calculated. These values are in turn used in determining a cost structure used by the detailed scheduling algorithms.

The tardiness cost structure is not necessarily a piecewise linear function of an operation's completion time. The possibility for machine failures may increase costs before the deterministic levels are passed. Expected tardiness costs should then be the objective function. These functions become nonzero for each operation sometime before the first operation due date or late time. They rise until they become linear after all due dates are passed (see Figure 2). Expected tardiness may be approximated

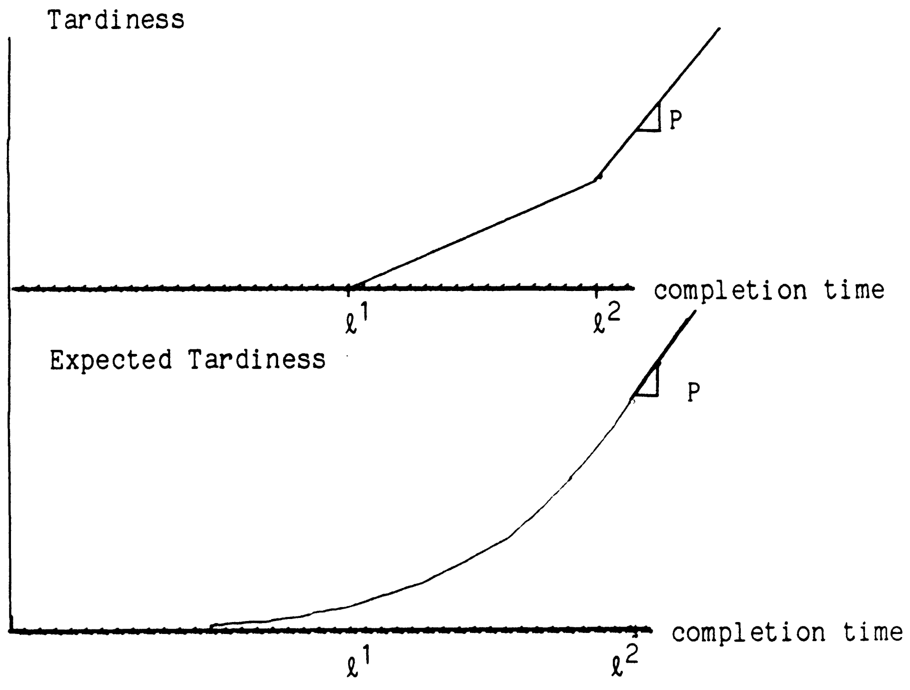


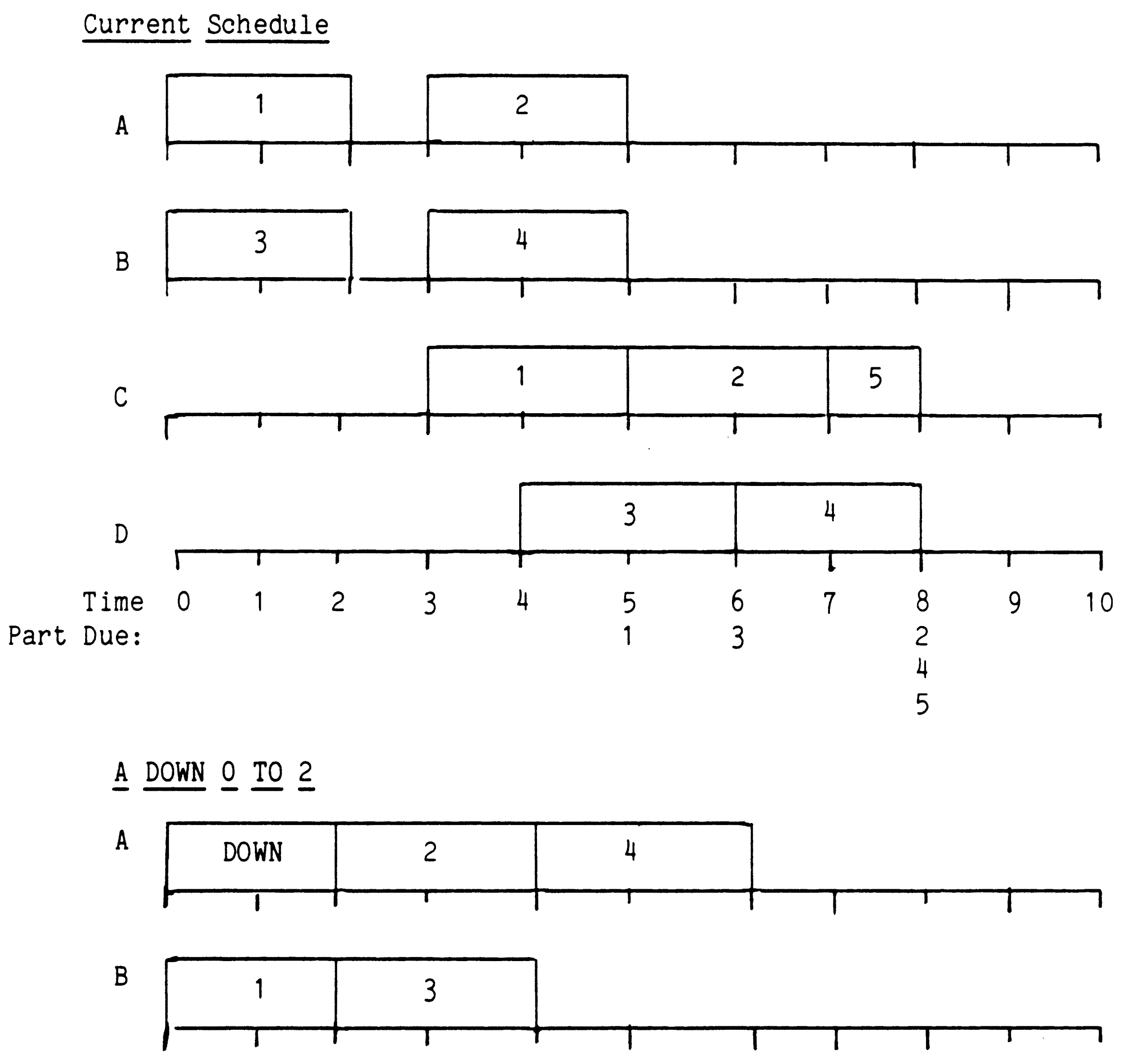
Figure 2. Tardiness and expected tardiness costs.

by other piecewise linear functions, or it may incorporated explicitly into a scheduling algorithm.

4. Defining Problem Sizes and Characteristics

Step 2 of MURTSA involves the definition of a set of operations and resources to be rescheduled. This set represents the portion of the pre-schedule which is released. The object in defining the new problem set is to allow for a smooth transition back to the pre-schedule turnpike. The problem should be large enough to allow for sufficient adjustments to reduce disruption effects, but it cannot be too large for the scheduling algorithms. Alternatives range from using dispatch rules that schedule one operation at all machines to a single machine scheduling algorithm to schedule all operations at a single machine. Some intermediate problem level between these extremes is sought that combines good solutions without excessive computational burden.

Possible methods for selecting a problem set of operations are based on including additional machine pools to the pool where the disruption occurred. Machine pools are defined here as groups of identical machines with identical setup requirements for all alternative tools. One method for finding additional pools to include in the problem set is to use information from the bill-of-material. If parts generally flow from one pool to another, then we may consider the preceeding or succeeding pool for inclusion with the disrupted pool. An example illustrating the use of this technique appears in Figure 3. Suppose Machines A and B form a pool and Machines C and D form a separate pool. Parts 1, 2, 3, and 4 must be processed on the AB pool followed by the CD pool. Part 5 is only processed on the CD pool. Due dates and processing times are given on the figure. Note that a two-unit downtime for A can be accommodated by the AB pool alone. A four-unit downtime for A requires inclusions of the CD pool.



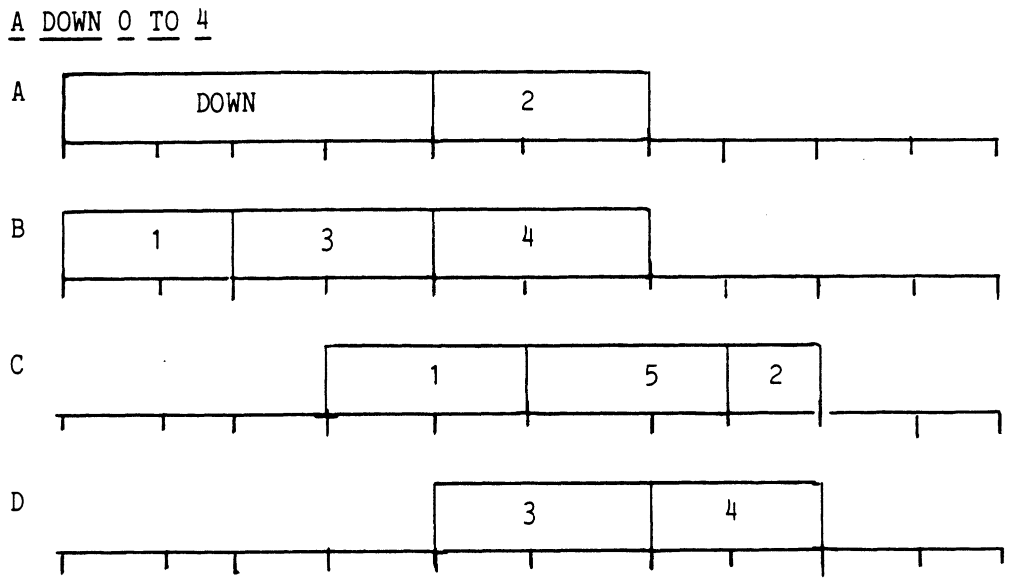


Figure 3. Succeeding Level Scheduling.

Other choices for pools to include in the problem set can involve information about the utilizations of the pools, their reliability, the slack times of operations on the pools, and measures of the criticality of the pool. Preliminary investigations have indicated that a good strategy is to choose the pool that would have the most operation tardiness if no schedule adjustments were made. This measure seems to be a good indicator of a pool's criticality. The tardiness can also be found easily using the project network described above.

The other decision in problem set definition concerns the length of the horizon and the number of operations to include in the rescheduling effort. This horizon should again be long enough to allow for match-up, but short enough to allow for adequate computation. One procedure for finding an initial horizon is to subtract processing times from usable machine time and to increase the horizon until that difference is positive for the processing times of all operations released from the pre-schedule. The usable machine time in this case does not include downtime or necessary idle time. A match-up is possible in this horizon if pre-emption is allowed. If pre-emption is not allowed, then some shifting of scheduled times beyond the horizon may be necessary. In either case, the cost of the match-up can be assessed and evaluated to determine whether to increase the horizon.

5. Implementation

The steps of MURTSA should be executed interactively so that information from the schedule-maker in Step 3 can determine whether additional problem sets and cost structures need to be defined. The operation of MURTSA can be continual with every problem set solution being improved until another disruption occurs. In this mode, the next scheduled operation always remains fixed in the MURTSA schedule and is not admitted to the problem set definition. When an operation is initiated, then the subsequent operation from the current best schedule found by MURTSA is fixed and MURTSA continues to solve within its current problem set. This process allows the scheduling algorithm to take advantage of real-time information and to use the capacity of the host processor.

The use of an incumbent solution that can be partially implemented at any system state change suggests methods which improve from feasible solutions. These methods may include heuristic orderings using dominance

relationships, simple interchanges to find local optima, branch-and-bound procedures that find feasible solutions quickly, and simulated annealing (see, e.g., Kirkpatrick, Gelatt, and Vecchi [1983]). This last approach may be especially useful in MURTSA because it can investigate numerous local optima and take advantage of all processing time. It may even be modified to weight changes in various parts of the pre-schedule to reflect criticality and to allow for a wider range of solutions.

6. Conclusions

Real-time adaptive scheduling of an FMS requires a flexible approach that takes advantage of the system's flexibility and considers inherent uncertainties in system status. A general approach as part of a hierarchical decision process was presented. The basic algorithm MURTSA; attempts to guide a schedule back onto a turnpike path that would be followed under ideal conditions. The implementation of this approach requires objective and problem definition. Some indications were given for defining these characteristics and for implementing the overall procedure.

References

Bean, J.C. and J.R. Birge [1985], "Match-Up Real-Time Scheduling", Department of Industrial and Operations Engineering, The University of Michigan, Technical Report 85-22.

Chang, Y-L., R.S. Sullivan, and U. Bagchi [1984], "Experimental Investigation of Quasi-Realtime Scheduling in Flexible Manufacturing Systems", in: Proceedings of the First ORSA/TIMS Special Interest Conference on Flexible Manufacturing Systems, pp. 307-312.

Gershwin, S.B., R. Akella, and Y. Choong [1984], "Short Term Production Scheduling of an Automated Manufacturing Facility", Laboratory for Information and Decision Sciences, Massachusetts Institute of Technology, Report LIDS-FR-1356.

Hildebrandt, R.R. and R. Suri [1980], "Methodology and Multi-Level Algorithm Structure for Scheduling and Real-Time Control of Flexible Manufacturing Systems", Proceedings of the 3rd International Symposium on Large Engineering Systems, pp. 239-244.

Kimemia, J. and S.B. Gershwin [1983]," An Algorithm for the Computer Control of a Flexible Manufacturing System", IIE Transactions 15, pp. 353-362.

Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi [1983], "Optimization by Simulated Annealing", Science 220, pp. 671-680.

Morton, T.E., R.M. Rachamadugu, and A. Vepsalainen [1984], "Accurate Myopic Heuristics for Tardiness Scheduling", GSIA Working Paper 36-83-84, Carnegie-Mellan University.

Morton, T.E. and T.L. Smunt [1984], "A Planning and Scheduling System for Flexible Manufacturing", in: Proceedings of the First ORSA/TIMS Special Interest Conference on Flexible Manufacturing Systems, pp. 313-326.

Stecke, K. [1983], "Nonlinear Integer Production Planning Problems", Management Science 29, pp. 273-288.

Wittrock, R. [1984], "Scheduling Algorithms for Flexible Flow Lines", IBM Research Report RC 10899, IBM Thomas J. Watson Research Center, Yorktown Heights, NY.