Match-Up Scheduling with Multiple Resources, Release Dates and Disruptions

James C. Bean
John R. Birge
John Mittenthal
Charles Noon

Department of Industrial Engineering
University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 86-37

September 1986
Revised September 1988

MATCHUP SCHEDULING with MULTIPLE RESOURCES, RELEASE DATES and DISRUPTIONS

James C. Bean[†]
John R. Birge[‡]
Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109

John Mittenthal
Department of Decision Sciences and Engineering Systems
Rensselaer Polytechnic Institute
Troy, NY 12181

Charles E. Noon
Department of Management
College of Business Administration
University of Tennessee
Knoxville, TN 37996

September 12, 1988

ABSTRACT

This paper considers the rescheduling of operations with release dates and multiple resources when disruptions prevent the use of a preplanned schedule. The overall strategy is to follow the preschedule until a disruption occurs. After a disruption, part of the schedule is reconstructed to match up with the preschedule at some future time. Conditions are given for the optimality of this approach. A practical implementation is compared with the alternatives of preplanned static scheduling and myopic dynamic scheduling. A set of practical test problems demonstrates the advantages of the matchup approach. We also explore the solution of the matchup scheduling problem and show the advantages of an integer programming approach for allocating resources to jobs.

Keywords: scheduling, integer programming, unreliable machines.

[†] The work of James Bean was supported in part by NSF Grant ECS-8700836 to The University of Michigan.

[‡] The work of John Birge was supported in part by the Office of Naval Research under Grant ONR-N00014-86-K-0628 to The University of Michigan and by the National Research Council under a Research Associateship at the Naval Postgraduate School.

1. Introduction

Much of the research in scheduling considers environments with one or more of the following assumptions: a single required resource (a machine), identical resources, all jobs available at the same fixed time, or known and fixed future conditions (see, e.g., Graves [1981] and Rinnooy Kan[1976]). These conditions are seldom the case in actual production facilities. In this paper we consider a method for adapting a preplanned schedule (or *preschedule*) to a changing scheduling environment. The problem includes multiple resources with some degree of processing compatibility, varying times at which jobs are available and costs associated with finishing a job early or completing it late. This framework models two actual manufacturing facilities in the automobile industry that are used in testing the approach.

Few previous studies have considered the possibility for disruptions such as machine breakdowns. When disruptions are considered (see Glazebrook [1984], e.g.), the models have limiting conditions. Models without disruptions assume a common deadline (Root [1965]), identical processors (Dogramaci and Surkis [1979]), or unit processing times (Blazewicz [1979]). The objectives studied include minimizing maximum completion time (Bratley, et al [1975], Nichols, et al. [1978]) or maximum tardiness (Nunnikhoven and Emmons [1977]).

We model a general multiple resources system with disruptions and assume that the preschedule can be followed if no disruptions occur. The scheduling strategy in this paper follows this preschedule until a disruption occurs and then reschedules part of the preschedule to accommodate the disruption. We reschedule to match up with the preschedule at some time in the future, that is, reschedule so that the completed jobs and inventory positions are identical to what would have occurred in the preschedule. Were the problem formulated as a dynamic program, the state reached by the revised schedule is the same as that reached by the original schedule.

In the matchup method, existing schedules are revised at disruptions as in Chang, et al. [1984], Donath and Graves [1985] and Filip, et al. [1983]. Our matchup method differs in that it seeks to match up with the preschedule.

We use a set of real problems to demonstrate the effectiveness of this procedure compared with pushing back a preschedule and list processing, dynamic scheduling. We also present heuristic procedures for solving the matchup problem and compare them on the problem set.

The general matchup method is described in Section 2, where we show the optimality of the matchup approach under certain conditions. The implementation of this theory appears in Section 3. Experimental results appear in Section 4. Section 5 summarizes our results and conclusions.

2. Matchup Scheduling Strategy

In the extremes, scheduling systems may be run continuously, as in the dynamic algorithms of Baker and Kanet [1983] and Rachamadugu [1987], or very infrequently as in the static algorithms of Lageweg, et al. [1977] and Bratley, et al. The matchup method fits between these common approaches. It responds to disruptions as in a dynamic algorithm, yet considers future information as in a static algorithm. Under certain conditions the matchup approach leads to an optimal schedule following a disruption.

The theoretical basis for the matchup strategy is an extension of economic turnpike results (see McKenzie [1976]). In Bean and Birge [1985], it was shown that a scheduling model could fit the framework of McKenzie's model for the asymptotic stability of optimal solutions. We extend these results below by providing a general scheduling model that yields asymptotic stability under fairly general conditions.

2.1 Problem Definition

Most scheduling models assume a finite number of jobs. In most real problems, as jobs are completed, other jobs are introduced. Failure to recognize the ongoing nature of the problem constitutes a significant simplification. To model the indefinite time horizon of realistic scheduling problems we consider a discrete time infinite horizon optimization problem. While there may be an infinite number of jobs in the system over infinite time, at any finite time we assume a known upper bound.

Assumption 1: The number of jobs active in the system is uniformly bounded by n.

Since n may by large, this assumption has little effect on our ability to model real problems. For notational convenience, we assume that jobs appear in job channels, indexed $i = 1, 2, \ldots, n$. When a new job is introduced to the system, it flows into an empty job channel. By Assumption 1, n can be chosen large enough to make this possible. The fact that two jobs appear in the same channel has no consequence. At time $t = 0, 1, \ldots$, we are in state $x_t \in \Re^{2n}$, where x_t is a vector giving the inventory of each current job and current setups. For $i = 1, \ldots, n$, $x_t(i)$ gives the inventory, measured in periods of processing, of the job in channel i. Hence, addition to inventory is bounded above by one in any period. At each due date, the inventory position of the corresponding job channel is reduced by the amount due. If the job is not completed, the inventory position becomes negative and tardiness is charged until inventory is again positive.

For $i = 1, \ldots, n$, $0 \leq x_t(n+i) \leq 1$ gives the setup information corresponding to the job in channel i. For the job in channel i, the processing possible during period t will be limited by $x_t(n+i)$. If $x_t(n+i) = 1$, then a full period of processing is possible. A value of $x_t(n+i) < 1$ means that setup occurs during the period or that the job is setup on an inefficient machine. In either case, less than a full period of processing is possible. If a machine is not set up to produce job i in period t, $x_t(i)$ is constrained to be zero.

The sequence of states over time is $x \in \ell^\infty$, that is, a sequence of vectors in $\Re^{2n}$ where element values are uniformly bounded. The decision at time t is to choose the machine setups and amount of processing on each job in period t. This decision corresponds to choosing a state, x_{t+1}, to enter at time $t+1$. Each decision has cost $f_t(x_t, x_{t+1})$, defined below. Feasibility is enforced by infinite costs as in Rockafellar [1970]. That is, any infeasible sequence is assigned a cost of $+\infty$. The objective is to find

$$\inf_{x \in \ell^\infty} \sum_{t=0}^{\infty} f_t(x_t, x_{t+1}). \tag{P}$$

Under certain assumptions, we will show problem P to be a convex optimization problem. The objective may not, however, be finite for any $x \in \ell^\infty$. We avoid this difficulty by defining a policy $x^* = \{x_0, x_1^*, \ldots\}$ as *weakly optimal*, as in McKenzie, if x^*

is not overtaken by any other policy, that is, if there does not exist $x^1 = \{x_0, x_1^1, \ldots\}$ such that

$$\limsup_{\tau\to\infty} \sum_{t=0}^{\tau} (f_t(x_t^1, x_{t+1}^1) - f_t(x_t^*, x_{t+1}^*)) \leq -\epsilon, \tag{1}$$

for some $\epsilon > 0$. The following well known technique gives us a finite valued problem. By the construction of f_t, we know that the objective terms are bounded for each feasible (x_t, x_{t+1}). We can formulate an equivalent problem by subtracting the period t value of the optimal solution from all f_t in each period. This altered problem has the same optimal solutions as the original problem and is finitely valued for any weakly optimal policy. Therefore, we can assume without loss of generality that P has a finite optimum obtained by $x \in \ell^\infty$. For the remainder of the paper, the term optimal means weakly optimal in this transformed problem.

Define $F^t(x_t) = \inf \sum_{\tau=t}^{\infty} f_\tau(x_\tau, x_{\tau+1})$. The transformed problem has $F^t(x_t) < \infty$ for all t. Our problem can be restated as finding $F^0(x_0)$.

Due to the infinite horizon nature of the problem formulation, each job channel consists of an infinite sequence of requests involving processing requirements, $p(i,k)$, release dates, $r(i,k)$, early dates, $e(i,k)$, and due dates, $d(i,k)$, for channel $i = 1, 2, \ldots, n$ and request $k = 1, 2, \ldots$. We assume that $r(i,k) \leq e(i,k) \leq d(i,k)$. Release dates are considered hard constraints, that is, a job cannot be started before its release date.

While the theory below applies to a broader class of problems, we assume an objective which minimizes the weighted sum of earliness and tardiness. This specific objective function is common in real scheduling problems such as the automobile manufacturing problems that motivated this work. If a job is started before its early date, a penalty, $u(i) \geq 0$, is incurred for each unit of early processing. If a job is finished after its due date, a penalty, $w(i) \geq 0$, is incurred for each unit of late processing.

The availability of resources such as machines, tools and setup crews determines which state transitions are feasible. Let G_t be the set of all possible decisions, (x_t, x_{t+1}), that are feasible in resource usage and setup. For a transition, (x_t, x_{t+1}), let

$$\delta(x_t, x_{t+1}) = \begin{cases} 0 & \text{if } (x_t, x_{t+1}) \in G_t \\ \infty & \text{if not} \end{cases}.$$

The formulated objective function, f_t, has two components, $\tilde{f}_t$ and δ. The $\tilde{f}_t$ component contains job channel separable costs such as earliness and tardiness penalties and production feasibility ($x_{t+1} - x_t \le e$, a vector of ones). The objective function is then

$$f_t(x_t, x_{t+1}) = \left[\sum_{i=1}^{n} \tilde{f}_t^i(x_t(i), x_{t+1}(i)) \right] + \delta(x_t, x_{t+1}).$$

Note that, although we are not allowing explicit costs for setting up machines, to do so is a straightforward extension.

Below we construct $\tilde{f}_t^i$, and show, under certain conditions, that it is a piecewise linear, convex function of $x_t(i)$, for each feasible $x_{t+1}(i)$.

If the current time is a due time for some current job, $t = d(i,k)$, then we automatically deduct the necessary processing, $p(i,k)$, from the inventory position of job i. If job i is not completed, i.e., $x_t(i) < p(i,k)$, this will result in a negative inventory, or backlogged, position at time $t+1$. In this case, for one or more future periods, we will be charged a tardiness penalty. If $x_t(i) < 0$, we are charged penalty $-w(i)x_t(i) > 0$.

If $d(i,k)$ is the next due date for job i and $x_t(i) \ge p(i,k)$ then we have already completed the processing due at $d(i,k)$. If we are beyond the early time, $e(i,k)$, then there is no penalty. If the current $t < e(i,k)$, then we incur an earliness penalty for each unit of overproduction equal to $u(i)(x_t(i) - p(i,k))$.

These procedures for accounting tardiness and earliness are not standard. However, with appropriate selections of weights, they deliver the same values. Hence, we will retain these terms.

Feasible inventory transitions to the next period are

$$-p(i,k) \le x_{t+1}(i) - x_t(i) \le 1 - p(i,k)$$

if t is a due date; $0 \le x_{t+1}(i) - x_t(i) \le 1$ if t is not a due date and some production run in channel i is beyond its release date; and $x_{t+1}(i) - x_t(i) = 0$ if all jobs in channel i are inactive (completed or prerelease). Any transition from $x_t(i)$ to $x_{t+1}(i)$ that is not feasible by these rules leads to an infinite cost.

2.2 Convexity of Objective Function

In general we must assume that

Assumption 2: $f_t(x_t, x_{t+1})$ is convex in x_t and x_{t+1}.

In this subsection we show the convexity of the earliness plus tardiness objective. We may prove Assumption 2 valid in this special case if we have

Assumption 2$'$: Penalties and setups are charged continuously. Jobs can be preempted (resume).

That is, if 90% of the job is completed at the due date, it is shipped. Only the remaining 10% will be charged the tardiness penalty. This assumption is reasonable if the job consists of many parts. If, however, it consists of one large part, this assumption may invalidate our conclusions.

We also assume that setups are continuous. Realistically, setups are binary; a machine is setup for a job, or it is not. For convexity, we need $0 \leq x_t(n+i) \leq 1$. We interpret $x_t(n+i)$ as the maximum amount of production of the job in channel i, in period t. If $0 < x_t(n+i) < 1$, then the machine is set up midperiod, or on an inefficient machine.

Note that we will set $\delta(x_t, x_{t+1}) = \infty$ if $x_{t+1}(i) > x_{t+1}(n+i),\ i = 1, \ldots, n$, or if the number of setups on similar machines exceeds the available machines. Since δ is a function of both x_t and x_{t+1}, it can also be set to $+\infty$ if the set of new setups for period $t+1$ requires more than the available setup crews.

We also must allow jobs to be preempted, resuming where they were left off when processing continues. Since such preemption would lead to more setups, we let the indirect costs of additional setups deter the system from frequent preemption. As with the continuous penalty assumption, preempt resume is reasonable if the job is made up of many small parts. If the job is one large part, it may or may not make sense technologically to preempt.

The specific $\tilde{f}_t^i$ defined above is a piecewise linear, convex function of $x_t(i)$ for each feasible $x_{t+1}(i)$. Since the set of feasible $x_{t+1}(i)$ is convex, $\tilde{f}_t^i$ is a convex function. Since

δ is also convex, $f_t = \sum_i \tilde{f}_t^i + \delta$ is convex.

The fundamental assumption of our model is that the theoretical requirement of continuous penalties and setups does not fundamentally alter results from the discrete events that occur in practice. Similar assumptions have been used in a wide variety of operations research models, such as the work of Maimon and Gershwin [1988], on control theoretic approaches to loading and scheduling automated manufacturing systems. As scheduling systems become more flexible, Assumption 2 becomes more valid.

2.3 Supporting Prices

To obtain results on the stability of solutions, we first present conditions for a supporting price system. Recall that we can alter P to have finite optimal costs. For price development, we define the set of feasible states at time t, X_t, as all states which can be continued over the infinite horizon at finite cost.

The *attainable* states at t are

$$Y_t = \{x_t | f_{t-1}(x_{t-1}, x_t) < \infty \text{ for some } x_{t-1} \in X_{t-1}\}.$$

At time t, the states that the optimal path could pass through must be both attainable, and have a feasible continuation, that is, be a member of $X_t \cap Y_t$.

The next theorem expands on McKenzie's results for price supports by using specific properties of the objective functions, f_t.

Theorem 1: If x^* is optimal in (P) with initial state x_0 and any of the following hold,

(a) (interiority) x_0 is in the relative interior of X_0 ($x_0 \in ri(X_0)$) and the interior of $(X_t \cap Y_t)$ is nonempty ($\text{int}(X_t \cap Y_t) \neq \emptyset$) relative to the affine hull of $(X_t \cap Y_t)$ ($\text{aff}(X_t \cup Y_t)$); or,

(b) (slack time) for any $t, x_t \in X_t \cap Y_t$ there exists $\{x_{t+1}, x_{t+2}, \ldots\}$ and $T > t$ such that

$$F^t(x_t) = \inf \sum_{\tau=t}^{\infty} f_\tau(x_\tau, x_{\tau+1})$$

and the resources required for any set of parts are slack at T (current resource usage strictly less than capacity); or

(c) (asymptotic penalty free schedule) for any $t, x_t \in X_t \cap Y_t$, there exists $T > t, \{x'_{t+1}, x'_{t+2}, \ldots\}$ such that $f_t(x_t, x'_{t+1}) + \sum_{\tau=t+1}^{T-1} f_\tau(x'_\tau, x'_{\tau+1}) < \infty$, $f_\tau(x'_\tau, x'_{\tau+1}) = 0$ for all $\tau \geq T$; or

(d) (decreasing fixed matchup cost) there exists a feasible solution to P,

$$x' = \{x'_{t+1}, x'_{t+2}, \ldots\}$$

such that for any $t, x_t \in X_t \cap Y_t, T_K, K = 1, 2, \ldots; T_K < T_{K+1}$, such that

$$\sum_{\tau=t}^{T_K - 1} f_\tau(x_\tau, x_{\tau+1}) + f_{T_K}(x_{T_K}, x'_{T_K+1}) < \infty$$

and $\lim_{K \to \infty} f_{T_K}(x_{T_K}, x'_{T_K+1}) = 0$;

then there exists $p_t^*, t = 0, 1, \ldots$ such that

(i) $F^t(x_t^*) - p_t^* x_t^* \leq F^t(x_t) - p_t^* x_t$ for all $x_t \in X_t, t = 1, 2, \ldots$

(ii) $f_t(x_t^*, x_{t+1}^*) - p_t^* x_t^* + p_{t+1}^* x_{t+1}^* \leq f_t(x_t, x_{t+1}) - p_t^* x_t + p_{t+1}^* x_{t+1}$, for all (x_t, x_{t+1}) such that $f_t(x_t, x_{t+1}) < \infty$.

Proof: See Appendix.

2.4 Solution Convergence

The asymptotic stability of solutions follows again from the structure of the objective function. The following theorem states that optimal solutions from varying initial conditions asymptotically approach each other. That is, the tail of the optimal path is insensitive to the initial conditions. Let Z_t^* be the set of solutions (z_t, z_{t+1}) such that

$$f_t(x_t^*, x_{t+1}^*) - p_t^* x_t^* + p_{t+1}^* x_{t+1}^* = f_t(z_t, z_{t+1}) - p_t^* z_t + p_{t+1}^* z_{t+1}.$$

Then Z_t^* is the set of optimal states at time t. Our result is that all optimal solutions approach Z_t^* regardless of initial condition.

Theorem 2: Let x^* be optimal for (P) with initial condition x_0, price support p^* and facets Z_t^* defined as above. Let x' be optimal in (P) with initial condition x'_0 and price supports p'. Then, for any $\epsilon > 0$, there exists $T < \infty$, such that, for all $t \geq T$,

$$\inf_{(z_t, z_{t+1}) \in Z_t^*} ||(x'_t, x'_{t+1}) - (z_t, z_{t+1})|| < \epsilon. \tag{2}$$

Proof: See Appendix

2.5 Implementation of the Theory

Theorem 2 is an asymptotic result. However, if implemented decisions are discrete, Theorem 2 implies exact matchup. Two questions arise: how fast can this matchup take place, and what happens if additional disruptions occur in the system?

The answer to the first question depends on system parameters. For example, a system with well distributed ample slack resources can match up quickly. This observation leads to the important question of how the system and preschedule can be designed to make rescheduling easier.

The second question results in an assumption.

Assumption 3: Disruptions are spaced far enough apart to allow matchup between disruptions.

This situation occurs in the real test problems studied below. If this assumption is not valid, a preschedule will not generally be useful. Full dynamic scheduling is likely to be near optimal since the system is quite random. In production systems where disruptions occur too frequently to match up, however, scheduling is most likely not the primary problem in the facility.

In practice, determining matchup times, and then rescheduling from the disruption to the matchup time, must be accomplished heuristically. Even rescheduling a single machine subject to release dates to minimize total tardiness is unary NP hard (Graham, et. al. [1979]). We describe a matchup heuristic below and then test its performance against real problem data from the automotive industry. Our goal in this analysis is to see how well our discrete approximation of the theoretically optimal matchup policy performs in reality. The empirical data indicate that, despite the approximations necessary for practical application, the matchup strategy tends to outperform other scheduling procedures and achieves objective values that, although not always provably optimal, are close to lower bounds.

The heuristics are designed specifically for the automotive manufacturing problems studied. Instead of making decisions on each x_t to x_{t+1} transition in the heuristic, we follow the standard practice of scheduling complete production units and setups that are not divisible. For our problems, we define a *job* as the production of a fixed number of parts that all have a single due date and release date. Our heuristic assumes that jobs cannot be subdivided during processing. In our practical examples, a job is a single order (shipping quantity) of one part type.

Jobs are further consolidated into *lots* of similar part types that do not require setups between jobs. This allows some simplification of the resource allocation phase of our heuristic. In the test problems, the *lots* are the production quantities that did not require intermediate setups in the original preschedule.

In our practical examples, processing a job requires two resources, a *machine* and a *tool*. Each resource had finite availability. Tools and machines that can be used together to produce a job are called *compatible* with each other and the job.

The objective is to minimize weighted tardiness, summed over all jobs. This corresponds to the hypotheses in Theorems 1 and 2. It supports our search for an early matchup time.

For our heuristic, we assume that a preschedule has been constructed and implemented. The preschedule is assumed optimal for the theoretical problem. While the actual preschedule is constructed heuristically, it achieves a lower bound on tardiness in a majority of the practical problems.

When a disruption occurs, we seek to match up with the preschedule within a cost threshold. We accomplish this heuristically by fixing an initial matchup point $T = T1$ on the disrupted machine(s). We resequence jobs on those machines (without violating release dates or exceeding tool capacities) to minimize the tardiness costs ($COST$) before matchup. If the cost for this matchup exceeds the predetermined threshold, EPS, on any machine, then we increment the matchup point, T, by DT. We repeat the process until T exceeds a maximum value, T_{max}, for the initial resequencing. In this case, we group jobs into lots and reallocate lots across machines with T reset to $T1$. We then decompose the lots to jobs and resequence jobs on individual machines. This allocation and sequencing part of the heuristic is a common practice, similar to that in Dogramaci and Surkis. The full matchup heuristic algorithm is given below.

3. The Matchup Scheduling Algorithm (MUSA)

Step 0: For each disrupted machine, set a minimum matchup time, $T1$. Let $T = T1$. Go to Step 1.

Step 1: On each disrupted machine, resequence all jobs scheduled before T. Evaluate the schedule $COST$ on each disrupted machine. If $COST < EPS$ on all machines, STOP. Else, go to Step 2.

Step 2: Let $T = T + DT$. If $T > T_{\max}$, go to Step 3. Else, go to Step 1.

Step 3: Expand the set of machines to be rescheduled to include all machines that share job compatibilities with the current set of disrupted machines. Reallocate lots across these machines. Go to Step 1.

Note that MUSA implicitly assumes that matchup with the preschedule is eventually possible within an incremental tardiness cost of EPS on each machine. In practice, the algorithm runs with a time limit and stops after Step 1 if this time limit is exceeded.

The procedures used for the allocation and sequencing steps are described below and include the allocation of a finite set of tools. The following subsections describe the alternatives implemented for these procedures.

3.1 Single Machine Sequencing

In Step 1, MUSA assumes that all job/machine/tool assignments are fixed. The algorithm heuristically reschedules jobs on a single disrupted machine. This method must run quickly in practice to allow implementation of the matchup schedule before the next scheduling decision must be made. The overall objective is to find a sequence of start times to minimize tardiness costs without violating release date or tool availability constraints.

Several heuristic ordering rules were attempted in the implementation of this phase of MUSA. Six different rules were found to obtain optimal solutions in some subset of our test cases so these rules were selected for inclusion into Step 1 of MUSA. The heuristic calculates six feasible schedules based on these ordering rules and chooses the least cost schedule that is obtained.

The basic procedure is to consider machines in decreasing order of their utilization, to remove any tool assignments on the selected machines, and to update a time pointer from the current time through to T by scheduling the first eligible job according to the ordering rule. This search then includes all jobs that are not scheduled but have been released, can be processed on the given machine and have an available tool. The time pointer is then updated to the minimum of the end of the processing of the most recently scheduled job and the minimum feasible start time among unscheduled jobs. After an initial sequence is developed, the MUSA heuristic checks for any interchanges of jobs within the horizon that could lower the total tardiness.

Since these sequencing steps run quickly for all ordering rules, all six are investigated on each machine. The resulting lowest cost solution is chosen, tool assignments on that machine are fixed, and the algorithm proceeds to the next highest utilization machine.

The ordering rules in the heuristic implementation are 1) a shortest processing time ordering (SPT), 2) an earliest due date ordering (EDD), 3) a modified due date ordering (MDD), 4) a priority index ordering (API), 5) a ratio rule and 6) the ordering based upon the current sequence. The MDD rule was taken from the heuristic developed by Baker and Bertrand [1982]. The API rule was taken from the heuristic developed by Morton and Rachamadugu [1982]. The ratio rule is based on a comparison of the remaining processing time of a job to the length of time available until the job is due. The given sequence ordering rule pushes back the schedule in use at the disruption.

3.2 Multimachine Lot Reassignment

If the best single machine solution results in excessive tardiness costs ($COST > EPS$) and the maximum single machine matchup point T_{max} is exceeded, then MUSA proceeds to Step 3, multimachine lot reassignment to redistribute lot to machine assignments. The focus of the reassignment problem is to determine a feasible lot schedule across several machines. The multimachine reassignment uses lots in place of jobs for rescheduling to keep the problem manageable and reduce additional setups. We examined two approaches for reassignment: a multiple choice integer program (MCIP) formulation solved using the technique of Bean [1984], and a priority rule dynamic assignment heuristic.

The reassignment procedure can shift lots across machines to create a feasible multi-machine schedule without additional setups. The single machine resequencing algorithm is then reapplied in Step 1 to reduce costs further.

Integer Programming Approach

The decisions for each lot in this resource allocation phase of MUSA are the machines to which the lot is assigned and the time at which the lot starts processing. We assume that the lot uses a single tool (as in our practical problems). The resulting problem is still a difficult mixed integer program, so we further simplify the problem to a multiple choice, zero one program by only allowing a lot to start at only a finite number of times within its *window*, the time interval between its release date and its due date minus its cumulative processing time. Several choices for these possible start times were tested. Our implementation uses up to three possible placements, the beginning, center and end of the window for each lot.

This formulation leads to zero one decision variables, y_{ij}, where i corresponds to a lot and j corresponds to the machine and starting point of the lot. In our implementation this definition produces $\sum_{i=1}^{\mathcal{L}} 3\bar{M}_i$ binary variables variable for a problem with $\mathcal{L}$ lots and $\bar{M}_i$ compatible machines for lot i.

To ensure that lot i is scheduled in exactly one place, the logical constraint

$$\sum_{j=1}^{3\bar{M}_i} y_{ij} = 1 \tag{5}$$

is used. The model also adds machine utilization constraints to ensure that the scheduled processing time does not exceed the available processing time. These constraints are written

$$\sum_{i} \sum_{j(k)} p_{ik} y_{ij(k)} \leq H_k, \tag{6}$$

where p_{ik} is the processing time of lot i on machine k, the index $j(k)$ indicates a start on machine k, and H_k is the available processing time on machine k.

Additional constraints for feasibility of the lot to machine assignments are needed to prevent the assignment of more than one lot to one machine simultaneously and to prevent the use of a single tool on two machines simultaneously.

Given start times, S_{ij}, and finish times, F_{ij}, for placement j of lot i, constraints are created to avoid the assignment of substantially overlapping lots to the same machine. Some overlap is allowable for later resolution by single machine sequencing or overtime. In our implementation, the allowable overlap is denoted, $RELAX$. This relaxation helps compensate for modeling the continuous lot start/finish by discrete assignments. Constraints are added such that, if lot i in placement j and lot k in placement l use the same machine and, if $F_{ij} > (S_{kl} + RELAX)$, then

$$y_{ij} + y_{kl} \leq 1. \tag{7}$$

The final set of constraints prevents the simultaneous scheduling of two lots which use the same tool. Here, overlaps are not allowed, since this is not resolved by the single machine sequencing step. If lot i in position j and lot k in position l use the same tool and $F_{ij} > S_{kl}$ and $F_{kl} > S_{ij}$, then

$$y_{ij} + y_{kl} \leq 1. \tag{8}$$

Other considerations in the program include variable machine production rates and constraints on machine utilization to balance machine workload.

Constraints $(5), (6), (7)$, and (8) provide the basis for the multimachine reassignment program. With this foundation, the problem's objective can take on a number of forms. We tested two alternatives: minimize the number of lot to machine assignment changes, and maximize the sum of squared processing times scheduled. A weighted combination of these objectives produced the best results.

Priority Rule Approach

An alternative approach for reassigning lots to machines is a priority rule strategy. The alternative we describe is similar to the approach in Dogramaci and Surkis. This procedure is also used for lot reallocation to test the value of the MCIP solution in Step 3. The procedure follows the basic pattern of the single machine resequencing heuristic.

All compatible machines are considered and a time pointer is updated to the first time on any of these machines when some job may be scheduled. The highest ranking job according to the ordering rule is chosen from the set of unscheduled and released jobs that are compatible with that machine and do not violate previous tool commitments. Several ordering rules are used to develop alternative schedules. The alternative with lowest tardiness cost is again accepted for implementation.

A different set of rules from the single machine sequencing rules appeared optimal in our testing and is used in our implementation. These ordering rules are earliest due date (EDD) and modified due date (MDD), as in the single machine heuristic, plus least window slack (LWS) which ranks jobs according to their window lengths. A comparison of the performance of these ordering rules is given in the next section.

4. Experimental Results

The matchup scheduling algorithm (MUSA) including the alternatives for lot allocation and job sequencing was coded in FORTRAN and implemented on an IBM 4381 computer. The program was applied to a set of problems with real production data supplied by an automotive manufacturer. The data represented two facilities that the manufacturer considered representative of the types of plants in its organization. The first facility had a small number of machines producing a relatively large number of parts on each machine. The second facility consisted of many compatible machines which produce a relatively small number of parts on each machine.

In the implementation, we set $T1 = T_{max}$. The value for $T1$ is chosen so that the system has sufficient slack to absorb the disruption.

4.1 Test Problem Description

The problem set consisted of eight disruption scenarios from a facility with two machines, and five disruption scenarios from a facility with ten machines. Facility 1 included 58 lots (250 jobs) scheduled over two fully compatible machines with an average utilization of 76%. Facility 2 consisted of 25 lots (293 jobs) scheduled over ten partially compatible machines with an average utilization of 46%. The disruptions were chosen by the manufacturer to represent common difficulties which render a preschedule infeasible: machine breakdowns, tool unavailabilities, release or due date changes, and order quantity increases. The preschedules were developed by the manufacturer and contained some unresolvable tardiness before the disruptions were added. This was due to earlier disruptions that forced ready times plus processing times to be greater than due dates. This inherent tardiness was a lower bound on the total tardiness. Tables 1a and 1b identify the type of disruption and give the inherent tardiness lower bounds.

The solution horizons generally included about 70 percent of the lots and were chosen so that the cumulative idle time was at least twice the average lot processing time. In Facility 1, the earliest starting horizon that achieved this was 111 on Machine 1 and 125 on Machine 2. The longer horizon values implicitly consider the type of disruption to guarantee a feasible matchup point. The corresponding matchup values and utilization are given in Table 2. The matchup point is given as a pair, $(t1, t2)$, where $t1$ is the matchup time on Machine 1 and $t2$ is the matchup time on Machine 2. In Facility 2, the initial horizon chosen (150) with the first set of compatible machines allowed enough flexibility to match up with the preschedule without excessive additional tardiness costs. The number of compatible machines and the utilizations on these machines in each of the scenarios are given in Table 3.

4.2 Analysis of Test Results

The testing of MUSA using the real data of Section 4.1 had two goals.

Goal 1: to validate our theoretical model by observing whether the MUSA heuristic produced good results suggested by the theory. We measured this performance by obtaining lower bounds on tardiness values and comparing MUSA tardiness values with these lower bounds.

Goal 2: to determine whether MUSA outperformed other alternatives that are often used in scheduling practice. The alternatives we considered were simple static and dynamic approaches and the use of our reallocation and resequencing methods without attempting to match up.

The role of machine utilization and matchup horizon lengths can also be observed in Tables 2 and 3. In these tables, incremental tardiness is defined as tardiness in the new schedule less inherent tardiness. In general, the effectiveness of MUSA, as measured by incremental tardiness, decreases as machine utilizations and matchup horizons increase.

Furthermore, the incremental tardiness was less than .20 hours per job per problem. On this basis, we conclude that the MUSA heuristic produced nearly optimal schedules in these 13 real test problems.

To investigate Goal 2, four strategies for rescheduling were evaluated. The alternatives tested were:

Strategy 1 (pushback) gives a static solution as in standard deterministic scheduling. When a disruption occurs, the machine assignments and job sequences stay the same, only the job start and finish times are shifted to accommodate the disruption.

Strategy 2 (dynamic) gives a fully dynamic myopic solution using the Priority Rule Approach of Section 3.2. The same horizon, $T1$, is used.

Strategy 3 (total reschedule) represents full rescheduling over all known jobs. It uses the subroutines in MUSA but ignores preschedule assignments. That is, it does not seek to match up with job placements of the preschedule. For this strategy, the multimachine reassignment problem is modeled as an MCIP to solve the reallocation problem over horizon $T1$. The single machine resequencer then obtains further penalty reductions. Compared to Strategy 2, this Strategy attempts more optimization.

Strategy 4 (matchup) is an implementation of MUSA using the MCIP to reallocate lots with the preschedule job assignments included in the formulation. MUSA differs from Strategy 3 by seeking to maintain the preschedule assignments as far as possible.

The tardiness results for the four strategies are also displayed in Tables 1a and 1b. MUSA (Strategy 4) obtains the best results on ten of the 13 problems. On two that it does not, the difference between Strategy 4 and the best performance is negligible. These results again validate the use of the continuous processing and setup theory on practical problems that do not exactly fit these conditions.

The tardiness results for Strategy 1 illustrate the penalties incurred when machine compatibilities are not utilized during rescheduling. When the preschedule is pushed back to accommodate the disruption, only the jobs on the disrupted machine are affected. Although this strategy preserves the preschedule sequence and machine assignments, it is limited since jobs may not be offloaded to compatible machines.

In Strategy 2, among the three selection rules tested, EDD, MDD and LWS, the LWS rule had the lowest average tardiness, however, it did not significantly outperform the others. The tardiness comparison between Strategy 2 and Strategy 3 is mixed. The partial lookahead approach of Strategy 3 performed better on the Facility 1 problems but markedly worse on the Facility 2 problems. This is due to MCIP's difficulty, without the matchup objective, in finding a feasible schedule across the many machines of Facility 2.

As stated above, Strategy 4 yielded the least tardiness for both facilities among all strategies. By including the preschedule assignments in the MCIP formulation, the model switched job/machine assignments only as needed to correct for the disruption. The preschedule assignments then served as a good completion to the solution schedule.

Evaluation of rescheduling approaches must also consider machine assignment changes and computation times. Table 4 displays the number of lot/machine assignment changes for Strategies 2, 3 and 4. Since the simple preschedule pushback approach of Strategy 1 does not move lots across machines, no values are given. For the Facility 1 problems, Strategy 2 yielded a significantly higher number of lot changes than Strategies 3 or 4. This is to be expected since the reassignment heuristic does not attempt to maintain original assignments as the MCIP does. All three strategies had few lot/machine changes for the Facility 2 data since many lots had only one compatible machine.

Table 4 also displays the computation times for Strategies 2, 3 and 4. The time for Strategy 1 was negligible and, hence, omitted. The Strategy 2 heuristic has the advantage of being very fast when compared to the MCIP run times of Strategies 3 and 4. The latter strategies require the solution of MCIP's with up to 225 variables and several hundred constraints. As the number of lots increases, the CPU times for Strategies 1 and 2 can be expected to increase approximately linearly while those of Strategies 3 and 4 can be expected to increase more sharply.

Table 4 includes the final $RELAX$ values for Strategies 3 and 4. This represents the amount of job overlap that had to be allowed before a feasible integer solution could be found by the MCIP. The Strategy 3 $RELAX$ averages are higher than those of Strategy 4 indicating greater difficulty in finding a feasible solution. The matchup objective not only resulted in lower $RELAX$ values, but also lower average run time.

5. Conclusions

This paper presents a framework for scheduling production facilities when disruptions invalidate preplanned schedules. It is shown that, if disruptions are sufficiently spaced over time, the optimal rescheduling strategy is to match up with the preschedule. The theory, however, assumes that processing and setup times can vary continuously and that an optimal matchup schedule can be found. In practice, processing and setups are discrete events, and optima cannot be found in reasonable times.

To validate the use of the theoretical principles in practical problems, heuristics were designed to approximate optimal continuous matchup. These heuristics were implemented and tested on a practical set of test problems from an automobile manufacturer. The results supported the theory by showing that matchup scheduling costs were close to lower bounds. In addition, the results were significantly better than results from pure static and dynamic strategies that are often used in practice. The experiments also indicated the advantage of an MCIP integer programming solution for allocating production to machines when utilizations are high.

APPENDIX

Proof of Theorem 1: McKenzie [Lemma 1, 1976] proves the result given (a). This condition does not cover scheduling problems with $x_0 \in \partial X_0$, the boundary of X_0. Also, the interiority of $X_t \cap Y_t$ may be difficult to verify. Conditions (b), (c) and (d) are reasonable assumptions that may be more readily verified. We show that each implies (i) and use (i) and the structure of f_t to show (ii).

Condition (b) implies that there is sufficient slack in the schedule that, at some future time, all resources will be utilized under capacity. We wish to show that $F^t(x_t)$ is subdifferentiable at x_t.

Consider $x_t \in X_t \cap Y_t$. Let $\{x_\tau, \tau > t\}$ minimize $F^t(x_t)$ so that

$$F^t(x_t) = \sum_{\tau=t}^{\infty} f_\tau(x_\tau, x_{\tau+1}).$$

Let $x'_t = x_t + \gamma$ where γ is a vector of perturbations. To show subdifferentiability, we show that there does not exist a γ such that the one sided directional derivative of F^t with respect to γ is $-\infty$ (Theorem 23.3, Rockafellar [1970]). For this, is is sufficient to show that $F^t(x'_t) \geq F^t(x_t) - K||x'_t - x_t||$, where K is a constant. If $F^t(x'_t) \geq F^t(x_t)$, the result is trivial. If $F^t(x_t) \geq F^t(x'_t)$, note that $F^t(x_t)$ is bounded above by the cost of the following path. Carry out exactly the decisions used by $\{x'_\tau\}$ from t to T. Hence, $x_T = x'_T + \gamma$. By the slackness hypothesis, we can feasibly alter processing in x_T to match up with x'_{T+1}. The only subtlety here is job channels where $x_T(i) > x'_T(i)$. In this case, channel i must have a due date in the future. Match up in that channel at the greatest T. Follow the path $\{x'_\tau\}$ from here. The difference between the optimal path from x'_t and the path just described is limited to the first $T - t$ periods. The cost of this difference is bounded by $K \leq (T-t)\sum_{i=1}^{n} \max\{w_i, u_i\} < \infty$. Hence, $F^t(x'_t) \geq F^t(x_t) - K||x'_t - x_t||$ for this K, and F^t is subdifferentiable.

A similar argument is used if (c) holds by noting that only a finite number of costs are reduced in optimal paths from x'_t and x_t. This again implies subdifferentiability.

Condition (d) can be interpreted as a generalization of (c), in which a finite cost path is eventually obtainable from every feasible path at decreasing cost. Note that $\sum_{\tau=t}^{T_K-1} f_\tau(x_\tau, x_{\tau+1}) + f_{T_K}(x_{T_K}, x'_{T_K+1})$ has a finite number of terms, each with bounded slope, and is hence subdifferentiable. Hence, there exists p_t^K and path $\{x''_{t+1}, x''_{t+2}, \ldots x''_{T_K}\}$ such that

$$\begin{aligned}&\sum_{\tau=t}^{T_K-1} f_\tau(x_\tau, x_{\tau+1}) + f_{T_K}(x_{T_K}, x'_{T_K+1}) - p_t^K x_t \\ &\leq \sum_{\tau=t}^{T_K-1} f_\tau(x''_\tau, x''_{\tau+1}) + f_{T_K}(x''_{T_K}, x'_{T_K+1}) - p_t^K x''_t,\end{aligned} \tag{7}$$

for all $x''_t \in X_t$. Rewrite (7) as

$$\begin{aligned}p_t^K(x''_t - x_t) \leq \sum_{\tau=t}^{T_K-1} f_\tau(x''_\tau, x''_{\tau+1}) - \sum_{\tau=t}^{T_K-1} f_\tau(x_\tau, x_{\tau+1}) + \\ f_{T_K}(x''_{T_K}, x'_{T_K+1}) - f_{T_K}(x_{T_K}, x'_{T_K+1}).\end{aligned} \tag{8}$$

Note that $|F^t(x_t)| < \infty$ and $|F^t(x''_t)| < \infty$. Hence p_t^K has a limit point, p_t, and by (d)

$$p_t(x''_t - x_t) \leq F^t(x''_t) - F^t(x_t), \tag{9}$$

for all $x''_t \in X_t$. Hence, F^t is subdifferentiable at x_t.

To show conclusion (ii), consider the function, $g(x'_t)$ defined by

$$\begin{aligned}g(x'_t) &= f_{t-1}(x^*_{t-1}, x^*_t) + F^t(x^*_t) - p^*_{t-1}x^*_{t-1} \\ &\quad - f_{t-1}(x'_{t-1}, x'_t) + p^*_{t-1}x'_{t-1} \\ &\leq F^t(x'_t),\end{aligned} \tag{11}$$

for any (x'_{t-1}, x'_t) such that $f_{t-1}(x'_{t-1}, x'_t) < \infty$. Note that g is also subdifferentiable for $x'_t \in Y_t$. Hence, there exists p'_t such that

$$g(x'_t) - p'_t x'_t \leq g(x''_t) - p'_t x''_t \leq F^t(x''_t) - p'_t x''_t, \tag{12}$$

for all $x''_t \in X_t \cap Y_t$. Now, let $x''_t = x^*_t$, and $p'_t = p^*_t$ to obtain

$$f_{t-1}(x^*_{t-1}, x^*_t) - p^*_{t-1}x^*_{t-1} - f_{t-1}(x'_{t-1}, x'_t) + p^*_{t-1}x'_{t-1} - p^*_t x'_t \leq -p^*_t x^*_t, \tag{13}$$

for all (x'_{t-1}, x'_t) feasible, proving (ii).∎

Proof of Theorem 2: We want to show

$$\inf_{(z_t, z_{t+1}) \in Z_t^*} ||(x'_t, x'_{t+1}) - (z_t, z_{t+1})|| < \epsilon. \tag{14}$$

Let x' have supporting prices p'. Let $v_t(z_t^*) = (p_t^* - p'_t)(x'_t - z_t^*)$. By summing inequalities (ii), for all T,

$$p_{T+1}^*(x'_{T+1} - x_{T+1}^*) \geq \sum_{t=0}^{T}(f_t(x_t^*, x_{t+1}^*) - f_t(x'_t, x'_{t+1})) - p_0^*(x_0 - x'_0), \tag{15}$$

and

$$p'_{T+1}(x_{T+1}^* - x'_{T+1}) \geq \sum_{t=0}^{T}(f_t(x'_t, x'_{t+1}) - f_t(x_t^*, x_{t+1}^*)) - p'_0(x'_0 - x_0). \tag{16}$$

From the finiteness of $F^0(x'_0)$ and $F^0(x_0^*)$, both right hand sides in (15) and (16) are uniformly bounded from below for all T. Hence, we know

$$v_t(x_t^*) \geq K > -\infty, \tag{17}$$

for all t and x_t^*.

Now, if (14) does not hold, there exists $\epsilon > 0$ and a sequence of times, $\{t_j\} \to \infty$, with

$$\inf_{(z_{t_j}, z_{t_j+1}) \in Z_{t_j}^*} ||x'_{t_j} - z_{t_j}|| \geq \epsilon. \tag{18}$$

By boundedness, $Z_{t_j}^*$ is compact and the infimum in (18) is attained, for example, at z^*. Then, by the structure of f_t,

$$f_{t_j}(z_{t_j}^*, z_{t_j+1}^*) - p_{t_j}^* z_{t_j}^* + p_{t_j+1}^* z_{t_j+1}^*$$

$$\leq f_{t_j}(x'_{t_j}, x'_{t_j+1}) - p_{t_j}^* x'_{t_j} + p_{t_j+1}^* x'_{t_j+1} - \gamma ||(z_{t_j}^*, z_{t_j+1}^*) - (x'_{t_j}, x'_{t_j+1})|| \tag{19}$$

where $\gamma \geq \min(w(i), u(i)) > 0$. By definition of Z_t^*, for any (x_t^*, x_{t+1}^*)

$$f_t(x_t^*, x_{t+1}^*) - p_t^* x_t^* + p_{t+1}^* x_{t+1}^* = f_t(z_t^*, z_{t+1}^*) - p_t^* z_t^* + p_{t+1}^* z_{t+1}^*. \tag{20}$$

From (19) and (20), we have for any T, with $T_j = \max\{t_j : t_j \leq T\}$,

$$\sum_{t=0}^{T} f_t(x_t^*, x_{t+1}^*) - \sum_{t=0}^{T} f_t(x_t', x_{t+1}')$$

$$\leq (p_0^* - p_0')(x_0' - x_0^*) - (p_T^* - p_T')(x_T' - x_T^*) - \sum_{j=1}^{T_j} \gamma ||(z_{t_j}^*, z_{t_j+1}^*) - (x_{t_j}', x_{t_j+1}')||. \qquad (21)$$

By (17), if (14) does not hold, then the right-hand side of (21) approaches $-\infty$ as T approaches ∞ since the normed term exceeds ϵ infinitely often. This contradicts the finiteness of $F^0(x_0')$.∎

REFERENCES

Baker, K. R. and J. W. M. Bertrand [1982], "A Dynamic Priority Rule for Sequencing Against Due-Dates," **Journal of Operations Management**, 3, 37-42.

Baker, K. and J. Kanet [1983], "Job Shop Scheduling with Modified Due Dates," **Journal of Operations Management**, 4, 11-22.

Bean, J. C. [1984], "A Lagrangian Algorithm for the Multiple Choice Integer Program," **Operations Research**, 32, 1185-1193.

Bean, J. C. and J. R. Birge [1985], "Match-up Real-Time Scheduling," **Proceedings of the Real-Time Scheduling Conference**, National Bureau of Standards, January.

Blazewicz, J. [1979], "Deadline Scheduling of Tasks with Ready Times and Resourse Constraints," **Information Processing Letters**, 8, 60-63.

Bratley, P., M. Florian and P. Robillard [1975], "Scheduling with Earliest Start and Due Date Constraints on Multiple Machines," **Naval Research Logistics Quarterly**, 22, 165-173.

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

Dogramaci, A. and J. Surkis [1979], "Evaluation of a Heuristic for Scheduling Independent Jobs on Parallel Idnetical Processors," **Management Science**, 25, 1208-1216.

Donath, M. and R. J. Graves [1985], "Flexible Assembly Systems: Near Real-Time Scheduling of Multiple Products," Department of Industrial Engineering and Operations Research, University of Massachusetts, Technical Report.

Filip, F., G. Neagu, and D. Donciulescu [1983], "Job-Shop Scheduling Optimization in Real-Time Production Control," **Computers in Industry**, 4, 395-403.

Glazebrook, K. D. [1984], "Scheduling Stochastic Jobs on a Single Machine subject to Breakdowns," **Naval Research Logistics Quarterly**, 31, 251-264.

Graham, R. L., E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan [1979], "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey," **Annals of Discrete Mathematics**, 5, 287-326.

Graves, S. C. [1981], "A Review of Production Scheduling," **Operations Research**, 29, 646-675.

Lageweg, B. J., J. K. Lenstra and A. H. G. Rinnooy Kan [1977], "Job-Shop Scheduling by Implicit Enumeration," **Management Science**, 24, 441- 450.

Maimon, O. Z. and Gershwin, S. [1988], "Dynamic Scheduling and Routing for Flexible Manufacturing Systems That Have Unreliable Machines," **Operations Research**, 36, 279-292.

McKenzie, L. [1976], "Turnpike Theory," **Econometrica**, 44, 841-864.

Morton, T. F. and R. M. V. Rachamadugu [1982], "Myopic Heuristics for the Single Machine Weighted Tardiness Problem," Technical Report CMU-RI-TR-83-9, Carnegie-Mellon University.

Nichols, R. A., R. L. Bulfin and R. G. Parker [1978], "An Interactive Procedure for Minimizing Makespan on Parallel Processors," **International Journal of Production Research**, 16, 77-81.

Nunnikhoven, T. S. and H. Emmons [1977], "Scheduling on Parallel Machines to Minimize Two Criteria Related to Job Tardiness," **AIIE Transactions**, 9, 288-296.

Rachamadugu, R. [1987], "A Note on the Weighted Tardiness Problem," **Operations Research**, 35, 450-452.

Rinnooy Kan, A. H. G. [1976], **Machine Scheduling Problems: Classification Complexity and Computations**, Nijhoff: The Hague.

Rockafellar, R. T. [1970], **Convex Analysis**, Princeton University Press: Princeton, N. J.

Root, J. G. [1965], "Scheduling with Deadlines and Loss Functions on k Parallel Machines," **Management Science**, 11, 460-475.

PROBLEM			STRATEGY 1	STRATEGY 2				STRATEGY 3	STRATEGY 4
FACILITY INDEX	DISRUPTION	PRE-SCHEDULE INHERENT TARDINESS	STATIC PRE-SCHEDULE PUSHBACK	FULL DYNAMIC: PRIORITY RULE				MCIP RE-SCHEDULING	MUSA
				EDD	MDD	LWS	MINIMUM		
1.1	Both machines down	351	480	452	458	438	438	431	440
1.2	unavailable tool	355	382	376	376	376	376	385	375
1.3	one machine down	349	390	494	491	499	491	403	378
1.4	New lot release date	353	438	373	373	376	373	378	375
1.5	New job release dates	351	1218	375	375	375	375	371	381
1.6	New lot release date	293	368	404	404	404	404	314	308
1.7	New job due date	373	394	393	393	393	393	396	393
1.8	New job order amount	352	384	435	448	448	435	391	374
AVERAGE		347	507	412.75	414.75	413.62	410.63	383.60	378.00

Table 1a. Total tardiness for matchup strategies in Facility 1.

PROBLEM			STRATEGY 1	STRATEGY 2				STRATEGY 3	STRATEGY 4
FACILITY INDEX	DISRUPTION	PRE-SCHEDULE INHERENT TARDINESS	STATIC PRE-SCHEDULE PUSHBACK	FULL DYNAMIC: PRIORITY RULE				MCIP RE-SCHEDULING	MUSA
				EDD	MDD	LWS	MINIMUM		
2.1	One machine down	0	156	0	0	15	0	0	0
2.2	Two machines down	0	140	89	89	0	0	0	0
2.3	New job due date	0	167	4	4	4	4	112	31
2.4	Three machines down	20	320	216	185	184	184	600	139
2.5	New lot release date	0	387	9	9	9	9	9	9
AVERAGE		4	234	63.60	57.40	42.40	39.40	144.20	35.80

Table 1b. Total tardiness for matchup strategies in Facility 2.

Table 2. Matchup Points, Utilization and Incremental Tardiness on Facility 1

Problem	Matchup Point (hrs)	Utilization (%)	Incremental Tardiness
1.1	(144,150)	84.5	89
1.2	(127,125)	76.2	20
1.3	(111,125)	79.8	29
1.4	(127,125)	76.2	22
1.5	(127,125)	76.2	30
1.6	(111,125)	75.6	15
1.7	(127,125)	76.2	20
1.8	(111,125)	78.7	22

Table 3. Compatible Machines, Utilization and Incremental Tardiness on Facility 2

Problem	Number of Machines	Utilization (%)	Incremental Tardiness
2.1	4	43.5	0
2.2	5	42.2	0
2.3	4	56.4	31
2.4	6	51.4	119
2.5	3	41.6	9

PROBLEM	STRATEGY 2		STRATEGY 3			STRATEGY 4		
FACILITY INDEX	NO. JOB MOVES	CPU (SEC.)	NO. JOB MOVES	CPU (SEC.)	RELAX (HRS.)	NO. JOB MOVES	CPU (SEC.)	RELAX (HRS.)
1.1	18	2.2	4	166.8	4.0	3	128.5	2.0
1.2	10	1.5	3	48.4	2.0	1	46.4	0.0
1.3	14	1.5	3	80.3	4.0	0	45.9	0.0
1.4	14	1.5	4	51.4	2.0	1	70.2	2.0
1.5	19	1.5	3	44.5	2.0	4	63.5	2.0
1.6	15	1.5	2	51.4	2.0	0	43.9	0.0
1.7	10	1.5	3	53.0	2.0	0	52.5	0.0
1.8	16	1.5	1	63.0	4.0	0	46.3	0.0
AVERAGE	14.5	1.59	2.9	69.85	2.8	1.1	62.11	.75
2.1	4	1.0	2	6.8	0.0	3	11.4	0.0
2.2	1	1.3	1	8.4	0.0	2	13.8	0.0
2.3	1	1.3	2	21.3	4.0	2	15.0	0.0
2.4	1	2.8	7	70.8	4.0	2	43.2	2.0
2.5	1	.7	1	4.8	0.0	1	6.6	0.0
AVERAGE	1.6	1.42	2.6	22.42	1.6	2.0	18.0	.40

Table 4. Job Moves, CPU Times, and Final RELAX Values for Strategies 2, 3, and 4.