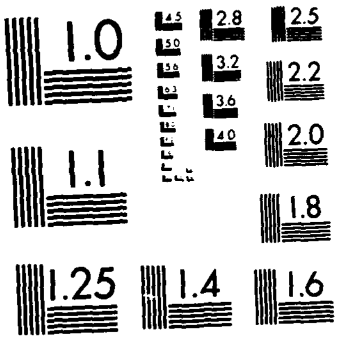
AD-A185 498 MATCH-UP SCHEDULING WITH MULTIPLE RESOURCES RELEASE DATES AND DISRUPTIONS (U) MICHIGAN UNIV ANN ARBOR DEPT OF INDUSTRIAL AND OPERATIONS ENG J C BEAN ET AL. 1/1

UNCLASSIFIED JUL 87 TR-86-37 N00014-86-K-0628 F/G 5/1 NL




MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(13)





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

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

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

Technical Report 86-37

**Department of Industrial and Operations Engineering**



The University of Michigan
College of Engineering
Ann Arbor, Michigan 48109-2117

13

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

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

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

Technical Report 86-37

September 1986
Revised July 1987

Draft Date: July 6, 1987

# Matchup Scheduling with Multiple Resources, Release Dates and Disruptions

James C. Bean[†]
John R. Birge[‡]
John Mittenthal
Charles E. Noon

Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, MI 48109

## 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 matchup 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.

## 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 true in actual production facilities. In this paper we consider a method for adapting a preplanned schedule to a changing scheduling environment. The problem includes multiple resources with some degree of processing compatibility, varying times

[†] The work of James Bean was supported in part by NSF Grants ECS-8409682 and 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

at which jobs are available and costs associated with not completing jobs by a due date. This situation 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 include using a common deadline (Root [1965]), identical processors (Dogramaci and Surkis [1979]), and 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 previously planned schedule until a disruption occurs and then reschedules part of the preschedule to accomodate the disruption. We reschedule to match up with the preschedule at some time in the future, that is, to 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.

The matchup method is similar to the approaches in Chang, et al. [1984], Donath and Graves [1985] and Filip, et al. [1983], in that existing schedules are revised at disruptions. Our matchup method differs from these approaches in that it seeks to matchup with the preschedule.

We use a set of real problems to demonstrate the effectiveness of this procedure in comparison with pushing back a preplanned schedule and fully 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 for sufficiently spaced disruptions. Techniques for sequencing jobs on one resource appear in Section 3. Methods for allocating jobs to the resources appear in Section 4. Section 5 describes the experiments conducted on our practical test set. Section 6 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 Morton and Rachamadugu [1983], 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 results under fairly general conditions.

Most scheduling models assume a finite number of jobs in the system. However, in most real problems, as jobs are processed 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. At time $t = 0, 1, \ldots,$ we are in state $x_t \in \Re^n$. In the scheduling framework a state includes information on the status of each job, e.g. , what fraction has been completed and what resources are assigned. The sequence of states over time is $x \in \ell^\infty$. The decision at time $t$ is to choose a state $x_{t+1}$ to enter at time $t+1$ with cost $f_t(x_t, x_{t+1})$. The objective is to find

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

where each $f_t$ is a proper convex function of $x_t$ and $x_{t+1}$.

We assume that each component $i$ of $x_t$ corresponds to production of part $i$. There are $R$ types of resources potentially required to produce each part. Each resource set $r$ has $M(r)$ groups, $G_1^r, \ldots, G_{M(r)}^r$ of identical resources. Part $i$ requires a resource in group $G_{g(i,r)}^r$ for $r = 1, \ldots, R$, for processing.

Each part $i$ also has an associated processing requirement $p(i)$, an earliness cost weight, $u(i) \geq 0$, and a lateness cost weight, $w(i) \geq 0$. There is also a sequence of release dates, $\{r(i,1), r(i,2), \ldots\}$, early dates, $\{e(i,1), e(i,2), \ldots\}$, and due dates, $\{d(i,1), d(i,2), \ldots\}$, such that $r(i,k) \leq e(i,k) < d(i,k)$ for each $k = 1, 2, \ldots$. The objective function $f$ has two components, $\tilde{f}_t$ and $\delta_r$. The $\tilde{f}_t$ component is separable in $i = 1, 2, \ldots, n$. It includes a lateness penalty if the $kth$ shipment of a part is not completed by the $kth$ due date. It includes an earliness penalty (holding cost) if the $kth$ shipment of a part is completed before the $kth$ early date. The $\delta_r$ component forces feasibility of processing with respect to the finite resources. The objective function is then

$$f_t(x_t, x_{t+1}) = \sum_{i=1}^{n} \tilde{f}_t^i(x_t(i), x_{t+1}(i)) + \sum_{r=1}^{R} \delta_r(x_t, x_{t+1}),$$

where $d(i, l-1) < t \leq d(i, l)$, and

$$\tilde{f}_t^i(x_t(i), x_{t+1}(i)) = \begin{cases} -w_i x_t(i), & \text{if } -1_{[t=d(i,l)]}p(i) \leq x_{t+1}(i) - x_t(i) \leq 1_{[t \geq r(i,l)]} - 1_{[t=d} \\ & x_t(i) < 0 \\ 1_{[t<e(i,l)]}u_i(x_t(i) - p(i)), & \text{if } -1_{[t=d(i,l)]}p(i) \leq x_{t+1}(i) - x_t(i) \leq 1_{[t \geq r(i,l)]} - 1_{[t=d} \\ & x_t(i) > p(i) \\ 0, & \text{if } -1_{[t=d(i,l)]}p(i) \leq x_{t+1}(i) - x_t(i) \leq 1_{[t \geq r(i,l)]} - 1_{[t=d} \\ & 0 \leq x_t(i) \leq p(i) \\ \infty, & \text{otherwise} \end{cases}$$

Note that $\tilde{f}_t^i$ is a piecewise, linear convex function (depending on $t$) of $x_t(i)$ for each feasible $x_{t+1}(i)$ and that the set of feasible $x_{t+1}(i)$ is convex. Hence $\tilde{f}_t^i$ is convex. Note also that $\tilde{f}_t^i$ forces a shipment on each due date (possibly resulting in negative inventory) so that feasible $x_t$ remain bounded. It represents incremental penalties for tardiness and earliness. The resource constraints are represented by

$$\delta_r(x_t, x_{t+1}) = \begin{cases} 0, & \text{if } \sum_{i=1}^{n} 1_{[g(i,r)=j]}(x_{t+1} - x_t) + p(i)1_{[t=d(i,l)]} \leq |G_j^r|, \ j = 1, \ldots, M(r) \\ 1, & \text{otherwise} \end{cases}.$$

These constraints limit the number of resources used of each type to the cardinality of those sets. Since each of these indicator functions is convex, $f_t$ is convex.

Problem $P$ is then 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, i.e., 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 $x \in \ell^\infty$.

To obtain results on the stability of solutions, we first present conditions for a supporting price system. For this 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

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

A resource group, $G_j^r$, is *idle* at $t$ if no processing occurs for parts requiring $G_j^r$, i.e., if $x_{t+1}(i) = x_t(i) - 1_{[t=d(i,l)]}p(i)$ for all $g(i,r) = j$.

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 $(P)$ and any of the following hold,

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

(b) (slack time) for any $t, x_t \in X_t \cap Y_t$ there exists $\{x_{t+1}, \ldots\}$, $F^t(x_t) = \inf \sum_{\tau=t}^{\infty} f_\tau(x_\tau, x_{\tau+1})$, $T > t$ such that the common resource groups required for any set of parts are idle at $T$; or

(c) (asymptotic penalty free schedule) for any $t, x_t \in X_t \cap Y_t$, there exists $T > t$, $\{x_t^1, x_{t+1}^1, \ldots\}$ such that $f_t(x_t, x_{t+1}^1) + \sum_{\tau=t}^{T} f_\tau(x_\tau^1, x_{\tau+1}^1) < \infty$, $f_\tau(x_\tau^1, x_{\tau+1}^1) = 0$ for all $\tau \geq T$; or

(d) (decreasing fixed matchup cost) there exists a feasible solution to $P$, $x' = \{x_t, x'_{t+1}, \ldots\}$ 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

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

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 $f_t(x_t, x_{t+1}) \leq \infty$.

**Proof**: See Appendix.

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 eventually optimal path is insensitive to the initial conditions.

Let $Z_t^*$ be the set of solutions $(z_t, z_{t+1})$ such that

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

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 $t \geq T$,

$$\inf_{(z_t, z_{t+1}) \in Z_t^*} ||(x_t', x_{t+1}') - (z_t, z_{t+1})|| \leq \epsilon. \quad (2)$$

**Proof**: See Appendix

Beyond the asymptotic approach to a turnpike solution, we can also show that matchup to a turnpike occurs as quickly as possible if the only objective is to minimize tardiness as defined above. This result further justifies our use of the matchup approach.

**Theorem 3**: Assume in Problem $(P)$ that $u_i = 0$ for all $i$ and that $x^*$ is an optimal solution given $x_0^*$. Let $x_0' \leq x_0^*$ be an alternative initial state. If there exists a feasible solution $\bar{x}$ such that $\bar{x}_0 = x_0'$ and $\bar{x}_t = x_t^*$ for some $t < \infty$, then there exists an optimal solution $x'$ with initial condition $x_0'$ such that $x_t' = x_t^*$.

**Proof**: See Appendix.

The impact of this result is that the optimal schedule beginning at the disruption state matches up with an optimal preschedule as soon as possible with finite cost. Two questions arise: how fast can this matching up take place, and what happens if additional disruptions occur in the system?

The answer to the first is dependent on system parameters. For example, a system with well distributed ample slack resources can matchup quickly. Note that 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 that disruptions are spaced far enough apart that match up is possible between disruptions. This situation occurs in the real test problems studied below. If this assumption is not valid, the system is so disrupted that no preschedule will be useful and full dynamic scheduling is likely near optimal.

As an implementation issue, we wish to use this matchup philosophy to reschedule a system upon disruption. We need to determine the matchup time and the optimal schedule from the disruption time to the matchup time. In practice, neither is a simple task. Below we describe a heuristic born of the matchup philosophy and test it against real problems from the automotive industry.

We define a *job* as production unit (e.g., part) with a single due date and release date that cannot be subdivided for processing. A *lot* is a collection of jobs from the same part type that does not require setups between jobs. A *tool* is a resource required in addition to a machine for processing a job.

Assume that a preschedule has been constructed and implemented. When a disruption occurs we seek to reschedule jobs on machines and tools to minimize total weighted tardiness such that no release dates are violated.

Since this problem is NP hard (even rescheduling a single machine subject to release dates to minimize total tardiness is unary NP hard, Graham, et. al. [1979]), we use a heuristic algorithm. We first attempt to match up on individual machines. If this is unsuccessful we reallocate lots across machines and resequence jobs on individual machines. This allocation and sequencing step is similar to that in Dogramaci and Surkis. In the algorithm given below note that $T1, T_{\max}, DT$ and $EPS$ are user defined parameters.

**Matchup Scheduling Algorithm (MUSA)**

*Step 0* : For each disrupted machine, set a minimum match-up 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 0.

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

## 3. Single Machine Sequencing

In Step 1, MUSA heuristically reschedules jobs on a single disrupted machine. This method must run quickly due to the real time environment. The goal of the procedure is to minimize the total cost for late processing of the jobs where each job is started no earlier than its release date. To increase scheduling flexibility, lots are broken into jobs. The overall objective is to find a sequence of start times to minimize the sum of setup and tardiness costs without violating release date or tool availability constraints.

The heuristic calculates six feasible schedules based on different ordering rules and chooses that with least cost. The ordering rules 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 given 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. 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 yields a pushback of the schedule in use when a disruption occurs.

For details on the implementation of the single machine sequencing heuristic see Bean, Birge, Mittenthal and Noon [1986].

## 4. Multimachine Lot Reassignment

If the best single machine solution results in excessive overtime or premium freight charges, the multimachine lot reassignment program is used 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 for rescheduling in order to keep the problem size 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 multimachine schedule without additional setups. The single machine resequencing algorithm can then be applied to order jobs to achieve further penalty reductions.

### 4.1 Integer Programming Approach

The multiple choice, zero one program has decision variables $x_{ij}$, which equal 1 if lot i is assigned to machine $j$ and equal 0 otherwise. If lot $i$ can be run on some subset of the machines, $M_i$, with cardinality $n$, then $n$ variables are created for lot $i$. To ensure that lot $i$ is scheduled on exactly one machine, the logical constraint

$$\sum_{j \in M_i} x_{ij} = 1 \tag{3}$$

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

$$\sum_{i} p_{ij} x_{ij} \leq H_j, \tag{4}$$

where $p_{ij}$ is the processing time of lot $i$ on machine $j$ and $H_j$ is the available processing time on machine $j$.

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. A lot's *window* is the interval between its release date and due date. Ideally, we would like to allow any placement within this window. For the integer programming approach, however, the resulting formulation is too complicated to solve in real time. To simplify this formulation placements within the window were discretized. One or more possible placements are determined for each lot to machine combination. The formulation is then altered so that $x_{ij} = 1$ if lot $i$ is placed in position $j$ (which has an associated machine). This increases the number of variables, but the structure of the MCIP is retained allowing efficient solution. One approach allows three placements, justified left, centered, or justified right. The other fixes placements to avoid conflicts. For details see Bean, Birge, Mittenthal and Noon.

Given start times, $S_i$, and finish times, $F_i$, for all potential lot placements, constraints are created to avoid the assignment of substantially overlapping lots to the same machine. Some overlap is allowable as it can be resolved later by single machine sequencing or overtime. 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 on machine $k$, if $F_i > (S_j + RELAX)$ then

$$x_{ik} + x_{jk} \leq 1, \tag{5}$$

The final set of constraints prevents the simultaneous scheduling of two lots which use the same tool. If lot $i$ on machine $j$ and lot $k$ on machine $l$ use the same tool and $F_i > S_k$ and $F_k > S_i$

then

$$x_{ij} + x_{kl} \leq 1. \tag{6}$$

Constraints (3), (4), (5), and (6) 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.

Other considerations incorporated in the program include constraints on machine utilization to balance machine workload and variable machine production rates. Other possible additions include constraints on manpower requirements for discrete intervals to inhibit manpower imbalance, and adherence to precedence requirements.

### 4.2 Priority Rule Approach

An alternative approach to reassigning lots to machines via the MCIP solution is to use a priority rule strategy. The alternative we describe is similar to the approach in Dogramaci and Surkis and to that of the single machine resequencing heuristic. Both develop initial schedules based on a number of different rules and then select one. The heuristic develops an initial lot sequence over the machine group and submits them one at a time for resequencing to the single machine heuristic. A description of this heuristic algorithm is given in Bean, Birge, Mittenthal, and Noon.

## 5. 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 from an automotive manufacturer. The goal of the comparisons on this data set was to determine the value of the matchup approach relative to the simple static and dynamic approaches, to evaluate the use of the MCIP allocation scheme relative to priority rules and to determine the relative values of the individual sequencing rules.

The problem set consisted of eight disruption scenarios from a facility with two machines, and five disruption scenarios from a facility with ten machines. Data for 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 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 gives the inherent tardiness lower bounds.

Parameters were chosen to reflect operating conditions in the facilities. 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. The MCIP formulation initially allows no overlap ($RELAX = 0$) among lot/machine assignments. If no feasible integer solution can be found, $RELAX$ is successively increased until enough constraints are relaxed that a solution can be found.

Four strategies for rescheduling were evaluated to illustrate the nature of the matchup problem. Strategy 1 gives a static solution. 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 gives a fully dynamic myopic solution using the myopic priority rule heuristic. For this strategy the reassignment heuristic was used to reschedule lots according to one of three selection rules: EDD, MDD and LWS. The last rule orders lots based on least slack time in its window.

Strategy 3 considers only partial look ahead. For this strategy, the multimachine reassignment problem was modeled as an MCIP. The preschedule lot/machine assignments and starting times, however, were excluded from the formulation. Rather than dynamically scheduling forward through time, as in the priority rule heuristic, this approach solves the problem over the matchup horizon and provides some degree of "look ahead" when considering the impact of scheduling a lot. After the MCIP determines the multimachine lot schedule, the single machine resequencer is used to obtain further penalty reductions.

Strategy 4 represents full matchup rescheduling using the MCIP with the preschedule job assignments included in the formulation. This strategy displays the strength of combining the look ahead aspects of the MCIP with the option of returning to the preschedule when it is advantageous to do so. After the MCIP determines the multimachine lot schedule, the single machine resequencer is used as in the previous strategy.

The tardiness results for the four strategies are displayed in Tables 1a and 1b. The values represent total job tardy hours across all machines. 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 look ahead approach performed better on the Facility 1 problems but markedly worse on the Facility 2 problems. This is due to MCIP's difficulty, without the preschedule assignments, in finding a feasible schedule across the many machines of Facility 2.

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.

The appropriateness of a rescheduling strategy must also consider machine assignment changes and computation times. Table 2 displays the number of job/machine assignment changes for Strategies 2, 3 and 4. Since the simple preschedule pushback approach of Strategy 1 does not move jobs across machines, no values are given. For the Facility 1 problems, Strategy 2 yielded a significantly higher number of job 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 job/machine changes for the facility 2 data since many jobs had only one compatible machine.

Table 2 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 2 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 direction from prescheduling assignments not only resulted in lower $RELAX$ values, but also lower average run time.

The next set of comparisons evaluates the matchup scheduling using the MCIP solution relative to the priority rule reassignment. The priority rule heuristic was run with each of its five rules. The resulting schedules were then evaluated with respect to lot tardiness and the minimum tardy schedule was chosen. The single machine resequencer was applied to the chosen schedule to evaluate and improve the job tardiness. This approach was compared to Strategy 4 which used the MCIP for lot reassignment. Table 3 displays the results of the comparison.

For the heuristic selection rule choice, the results show a fairly even distribution among the five rules tested and, on several problems, identical lot schedules among some of the rules. In terms of mean tardiness, the MCIP approach displays a slight edge over the heuristic. This difference arises from several problems in which the heuristic performed quite poorly compared to the MCIP approach because of the myopic versus look ahead characteristics of the two approaches. The average number of machine changes also favors the MCIP, especially among the Facility 1 problems. The heuristic, however, displays much lower CPU times than the MCIP.

## 6. 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, that the optimal rescheduling strategy is to match up with the preschedule. Heuristics are designed to implement this philosophy for problems with multiple resources, setups and release dates. The methods were tested on a practical set of test problems from an automobile manufacturer. The results showed that the matchup scheduling approach provided significantly better results than the pure static and dynamic strategies. The experiments also indicated that the MCIP integer programming solution to the allocation problem may be useful when utilizations are high.

## REFERENCES


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

Bean, J. C. and J. R. Birge, J. Mittenthal, C. Noon [1985], "Matchup Scheduling with Multiple Resources, Release Dates and Disruptions," Technical Report 86-37, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan 48109.

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, pp. 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 Disrete Mathematics**, 5, 287-326.

Graves, S. C. [1981], "A Review of Production Scheduling," **Operations Research**, 29, 4, 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.

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

Morton, T.F., and R. M.V. Rachamadugu [1983], "Myopic Heuristics for the Single Machine Weighted Tardiness Problem," Technical Report CMU-RI-TR-83-9, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

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.

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

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

## APPENDIX

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

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

(b) (slack time) for any $t, x_t \in X_t \cap Y_t$ there exists $\{x_{t+1}, \ldots\}$, $F^t(x_t) = \inf \sum_{\tau=t}^{\infty} f_\tau(x_\tau, x_{\tau+1})$, $T$ $t$ such that the common resource groups required for any set of parts are idle at $T$; or

(c) (asymptotic penalty free schedule) for any $t, x_t \in X_t \cap Y_t$, there exists $T > t, \{x_t^1, x_{t+1}^1, \ldots\}$ such that $f_t(x_t, x_{t+1}^1) + \sum_{\tau=t}^{T} f_\tau(x_\tau^1, x_{\tau+1}^1) < \infty$, $f_\tau(x_\tau^1, x_{\tau+1}^1) = 0$ for all $\tau \geq T$; or

(d) (decreasing fixed matchup cost) there exists a feasible solution to $P$, $x' = \{x_t, x'_{t+1}, \ldots\}$ 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

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

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 $f_t(x_t, x_{t+1}) \leq \infty$.

**Proof**: McKenzie [Lemma 1, 1976] proves the result given (a). This condition may not hold, however, in the scheduling context for $x_0 \in \partial X_0$ and the interiority of $X_t \cap Y_t$ may, indeed, be difficult to verify. The other conditions 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 for all resources for common parts to be free eventually. This condition should hold for all but heavily loaded systems. Note that the objective in $(P)$ is completely separable among sets of parts sharing common resources so we can assume without loss of generality that all parts have a common resource and all resources are eventually simultaneously idle. We wish to show that $F^t(x_t)$ is subdifferentiable at $x_t$.

Consider $x_t \in X_t \cap Y_t, F^t(x_t) = \sum_{\tau=t}^{\infty} f_\tau(x_\tau, x_{\tau+1})$ and $x'_t = x_t + \delta$. The idleness property implies that there exists a feasible schedule, $x''_\tau$, such that $x'' = x'_t$ and $x''_\tau = x_\tau, \tau \geq T$. We claim there exists an optimal schedule from $x'_t$ among $x''_\tau, \tau \geq T$, i.e., there exists $\{x'_{t+1}, \ldots\}$ such that $F^t(x'_t) = \sum_{\tau=t}^{\infty} f_\tau(x'_\tau, x'_{\tau+1}), x'_\tau = x_\tau, \tau \geq T$, for all $||\delta|| < \epsilon$ and some $\epsilon > 0$. Suppose not, then there exists some processing after $T$ in $x_t$ that can be moved before $T$ to reduce cost or processing before $T$ in $x_t$ that can be moved after $T$ to reduce cost. In either case, the same change in processing can be moved to $T$ with reduced cost by the structure of $f_t$. This implies that $\{x_\tau, \tau \geq t\}$ is not optimal, a contradiction.

From the above, only a finite number of states $x_\tau$ are changed in an optimal path from $x_t$ to an optimal path from $x'_t$. Since $f_t$ has bounded slope, there exists $K \leq (T-t)\sum_{i=1}^{n} \max\{w_i, u_i\} < \infty$ such that $F_t(x'_t) \geq F_t(x_t) - K||x'_t - x_t||$. Hence, $F_t$ is subdifferentiable at $x_t$, proving (i).

A similar argument is used if (c) holds by noting that only a finite number of costs are reduced in an optimal path from $x'_t$ from an optimal path from $x_t$. 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 $\sum_{\tau=t}^{T_K-1} f(x_\tau, x_{\tau+1}) + f_{T_K}(x_{T_K}, x'_{T_K+1})$ has a finite number of terms with bounded slope and is hence subdifferentiable. Hence, there exists $p_t^K$ such that

$$\sum_{\tau=t}^{T_K-1} f(x_\tau, x_{\tau+1}) + f_{T_K}(x_{T_K}, x'_{T_K+1}) - p_t^K x_t$$

$$\leq \sum_{\tau=t}^{T_K-1} f(x''_\tau, x''_{\tau+1}) + f_{T_K}(x''_{T_K}, x'_{T_K+1}) - p_t^K x''_t, \tag{7}$$

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

$$p_t^K(x''_t - x_t) \leq \sum_{\tau=t}^{T_K-1} f(x''_\tau, x''_{\tau+1}) - \sum_{\tau=t}^{T_K-1} f(x_\tau, x_{\tau+1}) +$$

$$f_{T_K}(x''_{T_K}, x'_{T_K+1}) - f_{T_K}(x_{T_K}, x'_{T_K+1}). \tag{8}$$

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

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

for all $x''_\tau \in X_t$.

Note that

$$\begin{aligned} F^{t-1}(x_{t-1}) &= \inf_{x'_t}\{f_{t-1}(x_{t-1}, x'_t) + F^t(x'_t)\} \\ &= f_{t-1}(x_{t-1}, x_t) + F^t(x_t). \end{aligned} \tag{10}$$

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 $f_{t-1}(x'_{t-1}, x'_t) < \infty$ and 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).■

**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 $t > T$,

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

**Proof**: Let $x'$ have supporting prices $p'$. From (ii) of Theorem 1, we have for any $z^*_t \in Z^*_t$,

$$\begin{aligned} p^*_t(x'_t - z^*_t) - p^*_{t+1}(x'_{t+1} - z^*_{t+1}) &\leq f_t(x'_t, x'_{t+1}) - f_t(z^*_t, z^*_{t+1}) \\ &\leq p'_t(x'_t - z^*_t) - p'_{t+1}(x'_{t+1} - z^*_{t+1}). \end{aligned} \tag{15}$$

Let $v_t(z^*_t) = (p^*_t - p'_t)(x'_t - z^*_t)$ . Inequality (15) implies that

$$v_t(z^*_{t+1}) \leq v_{t+1}(z^*_t) \tag{16}$$

for all $t$ and any $(z^*_t, z^*_{t+1}) \in Z^*_t$. By summing inequalities (ii), for all $T$,

$$p^*_{T+1}(x'_{T+1} - x^*_{T+1}) \geq \sum_{t=0}^{T} (f_t(x^*_t, x^*_{t+1}) - f_t(x'_t, x'_{t+1})) - p^*_0(x_0 - x'_0), \tag{17}$$

and

$$p'_{T+1}(x^*_{T+1} - x'_{T+1}) \geq \sum_{t=0}^{T} (f_t(x'_t, x'_{t+1}) - f_t(x^*_t, x^*_{t+1})) - p'_0(x'_0 - x_0). \tag{18}$$

From the finiteness of $F^0(x')$and $F^0(x^*)$, both right hand sides in (17) and (18) are uniformly bounded for all $T$. Hence, we can assume

$$v_t(x_t^*) \geq K > -\infty, \tag{19}$$

for all $t$ and $x_t^*$.

Now, if (14) does not hold, then, for all $T$, there exists some $T' > T$ such that

$$\inf_{(z_t, z_{t+1}) \in Z_t^*} ||x_t' - z_t|| \geq \epsilon. \tag{20}$$

By boundedness, let the infinium in (18) be attained at $z^*$. Then, by the structure of $f_t, t = T'$.

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

$$\leq f_t(x_t', x_{t+1}') - p_t^* x_t' + p_{t+1}^* x_{t+1}' - \gamma||(z_t^*, z_{t+1}^*) - (x_t', x_{t+1}')|| \tag{21}$$

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

$$f_t(x_t^*, x_{t+1}^*) - p_t^* x_t^* + p_{t+1}^* x_{t+1}^* = f_t(z_t^*, z_{t+1}^*) - p_t^* z_t^* + p_{t+1}^* z_{t+1}^*. \tag{22}$$

From (21) and (22), we have for any $T$,

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

$$\leq (p_0^* - p_0')(x_0' - x_0^*) - (p_T^* - p_T')(x_T' - x_T^*) - \sum_{t=0}^{T} \gamma||(z_t^*, z_{t+1}^*) - (x_t', x_{t+1}')||. \tag{23}$$

By (19), if (14) does not hold, then the right-hand side of (23) approaches $-\infty$ as $T$ approaches $\infty$. This contradicts the finiteness of $F^0(x')$.■

**Theorem 3**: Assume in Problem ($P$) that $u_i = 0$ for all $i$ and that $x^*$ is an optimal solution given $x_0^*$. Let $x_0' \leq x_0^*$ be an alternative initial state. If there exists a feasible solution $\bar{x}$ such that $\bar{x}_0 = x_0'$ and $\bar{x}_t = x_t^*$ for some $t < \infty$, then there exists an optimal solution $x'$ with initial condition $x_0'$ such that $x_t' = x_t^*$.

**Proof**: First construct a feasible solution $\bar{x}$ such that

(i) $\bar{x}_r(i) = x_r'(i)$ if $x_r'(i) \leq x_r^*(i)$,

(ii) $\bar{x}_r(j) - \bar{x}_{r-1}(j) \geq x_r^*(j) - x_{r-1}^*(j)$ for all $j$ sharing resources with $i$ if $x_r'(i) > x_r^*(i)$ and $x_r'(i) - x_{r-1}'(i) > 0$.

A solution satisfying (i) and (ii) satisfies $\bar{x}_\tau \leq x^*_\tau, \tau = 1, 2, \ldots, t$, since no processing is made beyond $x^*_\tau(i)$ for each $i$. The feasibility of a path from $x'_0$ to $x^*_t$ only requires $\sum_{\tau=1}^n (x_0(i) - x'_0(i))$ available capacity units on the resources required by those parts. This processing can be completed anywhere in $[0, t]$ such that $x_0 \geq x'_0$ implies $r(i) \leq 0$ for all $i$ such that $x_0(i) \leq x'_0(i)$ and no other resource constraints are present. Therefore, $\bar{x}$ satisfying (i) and (ii) can be extended to satisfy $\bar{x}_t = x^*_t$ and $\bar{x}_\tau = x^*_\tau, \tau = 0, 1, \ldots, t$.

Suppose $\bar{x} \neq x'$ for any $x'$ optimal given $x'_0$. Then, there exists a sequence $\Delta x = (\Delta x_0 = 0, \Delta x_1, \Delta x_2, \ldots)$ such that $x' = \bar{x} + \Delta x$ and $F(\bar{x} + \Delta x) < F(\bar{x})$. We will show that this implies that there exists a sequence $\Delta x^*$ such that $F(x^* + \Delta x^*) < F(x^*)$. To show this, let $F(x') = \sum_{i=1}^n F_i(x'(i))$, where $F_i$ is the contribution to the overall objective value from part $i$. Suppose $F_i(x'(i)) < F_i(\bar{x}(i))$. Then there exists $\tau$ such that $\Delta x_\tau(i) - \Delta x_{\tau-1}(i) > 0$. For each such $\tau$, note that $\bar{x}_\tau(i) = x^*_\tau(i)$, so $x^*_\tau(i) + \Delta x_\tau(i) = \bar{x}_\tau(i) + \Delta x_\tau(i)$ and

$$F_i(\bar{x}(i) + \Delta x(i)) - F_i(\bar{x}(i)) = F_i(x^*(i) + \Delta x^*(i)) - F_i(x^*(i)), \tag{24}$$

where $\Delta x^*(i) = \Delta x(i)$.

In order for $\bar{x}(i) + \Delta x(i)$ to be feasible, however, there must exist $\Delta x_\tau(j) - \Delta x_{\tau-1}(j) \leq 0$ for parts $j$ requiring common resources with $i$. For each such $j$, we can define $\Delta x^*$ recursively from $\tau = 1$ by

$$\Delta x^*_\tau(j) - \Delta x^*_{\tau-1}(j) = \min\{x^*_\tau(j) - x^*_{\tau-1}(j), \Delta x^*_\tau(j) - \Delta x^*_{\tau-1}(j)\}. \tag{25}$$

By (ii), this ensures the feasibility of $x^* + \Delta x(i) + \sum_{j \in J(i)} \Delta x^*(j)$, where $J(i) = \{$set of parts sharing resources with $i\}$. For each $j \in J(i)$, note that

$$F_j(\bar{x}(j)) - F(\bar{x}(j) + \Delta x(j)) = \sum_{\{\tau | \bar{x}_\tau(j) < 0\}} w_j \Delta x_\tau(j)$$

$$+ \sum_{\{\tau | \bar{x}_\tau(j) \geq 0, \bar{x}_\tau(j) + \Delta x_\tau(j) < 0\}} w_j (\bar{x}_\tau(j) + \Delta x_\tau(j)) + \sum_{\{\tau | \bar{x}_\tau(j) \geq 0, \bar{x}_\tau(j) + \Delta x_\tau(j) \geq 0\}} 0. \tag{26}$$

Since $\bar{x}_\tau \leq x^*_\tau$ and $\Delta x^*_\tau \leq \Delta x_\tau$, from (26),

$$F_j(\bar{x}(j)) - F_j(\bar{x}(j) + \Delta x(j)) \leq F_j(x^*(j)) - F_j(x^*(j) + \Delta x^*(j)). \tag{27}$$

Construct $\Delta x^*$ using this procedure for each $i$ such that $F_i(x'(i)) < F_i(\bar{x}(i))$. It follows from (24), (26), and (27) that

$$F(x^* + \Delta x^*) - F(x^*)$$

$$= \sum_{I=\{i|F_i(x'(i))<F_i(\bar{x}(i))\}} F_i(x^*(i)+\Delta x^*(i)) - F_i(x^*(i)) + \sum_{\{j\in J(i),i\in I\}} F_j(x^*(j)+\Delta x^*(j)) - F_j(x^*(j))$$

$$\leq \sum_I F_i(x'(j)) - F_i(\bar{x}(i)) + \sum_{\{j\in J(i),i\in I\}} F_j(x'(j)) - F_j(\bar{x}(j)) < 0, \qquad (28)$$

but this contradicts the optimality of $x^*$. Hence, $\bar{x} = x'$ for some optimal $x'$.∎

| PROBLEM | | | STRATEGY 1 | STRATEGY 2 | | | | STRATEGY 3 | STRATEGY 4 |
|---|---|---|---|---|---|---|---|---|---|
| FACILITY INDEX | DISRUPTION | PRE-SCHEDULE INHERENT TARDINESS | PRE-SCHEDULE PUSHBACK | MYOPIC PRIORITY RULE REASSIGNMENT | | | | MCIP W/O PRE-SCHEDULE ASSIGNMENT | MCIP WITH PRE-SCHEDULE ASSIGNMENT |
| | | | | 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 match-up strategies in Facility 1.

| PROBLEM | | | STRATEGY 1 | STRATEGY 2 | | | | STRATEGY 3 | STRATEGY 4 |
|---|---|---|---|---|---|---|---|---|---|
| FACILITY INDEX | DISRUPTION | PRE-SCHEDULE INHERENT TARDINESS | PRE-SCHEDULE PUSHBACK | MYOPIC HEURISTIC REASSIGNMENT | | | | MCIP W/O PRE-SCHEDULE ASSIGNMENT | MCIP WITH PRE-SCHEDULE ASSIGNMENT |
| | | | | 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 match-up strategies in Facility 2.

| 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 | 52.2 | 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 | [illegible] | 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 2. Job Moves, CPU Times, and Final RELAX Values for Strategies 2, 3, and 4

| PROBLEM | Priority Rule Reassignment Methods with Single Machine Resequencing | | | | | MCIP with Single Machine Resequencing | | |
|---|---|---|---|---|---|---|---|---|
| FACILITY INDEX | Method Used | Tardy (Hrs.) | Jobs Moved | CPU (Sec.) | Tying Selection Methods | Tardy (Hrs.) | Jobs Moved | CPU (Sec.) |
| 1.1 | MDD | 451 | 19 | 71.7 | | 440 | 3 | 128.5 |
| 1.2 | LPT | 374 | 10 | 17.4 | EDD, MDD, LWS | 375 | 1 | 46.4 |
| 1.3 | MDD | 494 | 10 | 13.8 | | 378 | 0 | 45.9 |
| 1.4 | EDD | 373 | 12 | 18.4 | MDD | 375 | 1 | 70.2 |
| 1.5 | LPT | 374 | 18 | 23.1 | | 380 | 4 | 63.5 |
| 1.6 | LWS | 367 | 20 | 13.0 | | 308 | 0 | 43.9 |
| 1.7 | LPT | 392 | 10 | 17.6 | EDD, MDD, LWS | 393 | 0 | 52.2 |
| 1.8 | EDD | 431 | 15 | 36.0 | | 374 | 0 | 46.3 |
| AVERAGE | | 407.0 | 14.25 | 26.4 | | 378.0 | 1.13 | 62.1 |
| 2.1 | EDD | 0 | 5 | 5.8 | MDD | 0 | 3 | 11.4 |
| 2.2 | LWS | 0 | 2 | 5.5 | | 0 | 2 | 13.8 |
| 2.3 | EDD | 3 | 2 | 2.6 | MDD, LWS | 31 | 2 | 15.0 |
| 2.4 | EDD | 184 | 1 | 5.3 | MDD, LWS | 139 | 2 | 43.2 |
| 2.5 | LPT | 9 | 1 | 4.1 | EDD, MDD, LWS | 9 | 1 | 6.6 |
| AVERAGE | | 39.2 | 2.2 | 4.7 | | 35.8 | 2.0 | 18.0 |

Table 3. Priority Rule Heuristic vs. MCIP for Multi-Machine Reassignment

END

DATE

FILMED

DEC.

1987