RESCHEDULING UNRELATED PARALEL MACHINES WITH TOTAL FLOW TIME AND TOTAL DISRUPTION COST CRITERIA

M Özlen[1] and M Azizoğlu[2*]

[1] *School of Mathematical and Geospatial Sciences, RMIT University, Melbourne, Australia; and*
[2]*Department of Industrial Engineering, Middle East Technical University, Ankara, Turkey*

In this paper, we consider a rescheduling problem where a set of jobs has already been assigned to unrelated parallel machines. When a disruption occurs on one of the machines, the affected jobs are rescheduled, considering the efficiency and the schedule deviation measures. The efficiency measure is the total flow time, and the schedule deviation measure is the total disruption cost caused by the differences between the initial and current schedules. We provide polynomial-time solution methods to the following hierarchical optimization problems: minimizing total disruption cost among the minimum total flow time schedules and minimizing total flow time among the minimum total disruption cost schedules. We propose exponential-time algorithms to generate all efficient solutions and to minimize a specified function of the measures. Our extensive computational tests on large size problem instances have revealed that our optimization algorithm finds the best solution by generating only a small portion of all efficient solutions.

Keywords: Rescheduling; Unrelated Parallel Machines; Efficient Schedules

[1] *Corresponding Author*
Mailing Address: Department of Industrial Engineering,
Middle East Technical University, Ankara 06531, Turkey
Phone: +90 312 210 22 81
Fax: +90 312 210 12 68
E-Mail: meral@ie.metu.edu.tr

INTRODUCTION

The majority of the scheduling literature considers a manufacturing environment with no disruptions. However in manufacturing practice, the environment is very often subject to disruptions that make the initial scheduling plan inefficient, or even infeasible and necessitate rescheduling. Common disruptions are machine breakdowns, hence subsequent repairs, new order arrivals, order cancellations, changes in order specifications like priorities, release times, and due dates, and shortages of resources like materials, labor, tools and equipments.

We consider a parallel machine environment where the machines are subject to disruptions and where the jobs are initially scheduled so as to minimize the total flow time, i.e., the total time the jobs spent in the system. Flow time gives a direct indication of the work-in-process inventory levels, hence its minimization is an important concern of many manufacturers. Two generalizations of the total flow time objective are the total weighted flow time and total tardiness objectives. The total weighted flow time objective assumes different priorities for the jobs whereas total tardiness objective considers different due dates for the jobs.

We assume that the parallel machine machines are unrelated in the sense that the speeds of the machines vary by the tasks they process. The unrelated parallel machines fit well to the manufacturing environments where some machines are specialized for a particular class of jobs and hence process them at higher speed, whereas some other machines can process another class of jobs quicker. The uniform parallel machines are the special cases of the unrelated parallel machines where the speeds of the machines vary, however the speed of a particular machine does not vary among tasks, i.e., a machine processes all jobs at the same speed. The identical parallel machines are the special cases of the uniform parallel machines where all machine speeds are identical. The unrelated parallel machines form the most general, hence the hardest, class of parallel machines.

We assume that the customer promises are given and the resource allocations are made according to the initial minimum total flow time schedule. During the execution of the initial scheduling plan, a disruption blocks the machines for a specified length of time, no matter which jobs are processed. Thereafter, considering the effect of the disruption, the manufacturer still aims to minimize the total flow time of the jobs that have not yet started. The current minimum flow time schedule, i.e., the schedule formed after the disruption, may have different machine allocations than the initial schedule. It may be desirable to keep these deviations at minimum level, in particular when the preparations such as machine setups like tool loadings and resource allocations like labor assignments are made according to the initial scheduling plan and any

change to the plan adversely affects the preparations. As mentioned in Clausen et al. (2001), such rescheduling problems fall within the scope of the disruption management. Clausen et al. (2001) reports on the recent developments on the disruption management problems from an operational point of view and discusses some applications in shipbuilding, airlines and telecommunications.

We use the total disruption cost as a schedule deviation measure. We call a job disrupted, if it is assigned to different machines in the initial and current schedules. The disruption costs are incurred according to the machines on which the jobs are assigned in the current schedule. The total disruption cost is an important measure, particularly in flexible manufacturing systems and supply chains. As stated in Olumolade and Norrie (1996), the setup costs are incurred when the tools and pallets are allocated in advance according to the initial job assignments. Hence retooling the machines and reallocating the pallets due to the changes in their assigned jobs may require additional time and may bring additional disruption costs. For each job, these disruption costs may depend on the machines they are assigned to in the current schedule. The transportation costs may be incurred while transferring the jobs and their required materials/tools from the initial machine to the current machine. Hence the locations of the initial and current machines in the shop floor may affect the disruption costs. Moreover the jobs may have different setup requirements on different machines, for example some machines may already be equipped with a subset of the tools, fixtures, labor or equipment required by the disrupted jobs and some others may not be.

We consider the trade-off between the efficiency of the current schedule, measured by the total flow time and the schedule deviation measured by the total disruption cost caused by the differences between the initial and current machine allocations. Hence we can classify our disruption management, i.e., rescheduling, problem as a multi-criteria problem where both efficiency and schedule deviation measures are explicitly included in the model.

An important contribution of our paper is the design of an optimization algorithm for finding an optimal solution for a general non-decreasing function of the total flow time and total disruption cost criteria. The models that we are considering are integer programs which suggest that any optimization procedure will run into computational troubles at some point as the problem size increases. There is, however, the practical question concerning the sizes of the problem that are solvable in reasonable amount of time. Our computational results suggest that the answer to this question for our algorithm is 80 jobs and 12 machines. Hence our algorithm can serve as useful tool to solve large sized practical problems.

The rest of the paper is organized as follows. First, we review the rescheduling literature that is pertinent to our work. Next, we introduce the basic definitions, notation, and define our problems. After that, we present the optimization algorithms for each problem discussed. The results of our experiments are presented next. And finally, we present our conclusions and discuss some possible directions for future research.

LITERATURE REVIEW

The literature on rescheduling studies is of relatively recent origin. For a thorough review of the area we refer readers to Aytug et al. (2005) and Vieira et al. (2003). Wu et al. (1993), Daniels and Kouvelis (1995), Ünal et al. (1997), O'Donovan et al. (1999), Hall and Potts (2004) and Qi et al. (2006) study some rescheduling problems on single machines. Wu et al. (1993) consider minimizing the makespan and the deviation of the job start times between the initial and current schedules. Daniels and Kouvelis (1995) develop schedules that are robust to the future disruptions and processing time variability. Ünal et al. (1997) consider the insertion problem of the new jobs with part-type dependent setup times. They aim to minimize the total weighted flow time and makespan while preserving the relative sequence of the initial jobs and incurring no additional setups. O'Donovan et al. (1999) apply a predictive approach to the total tardiness problem where the processing times are affected by the machine failures. Hall and Potts (2004) consider inserting the new jobs in a schedule with small disruption of the old jobs. They consider the maximum lateness and total flow time as efficiency measures and the sequence deviation and total completion time deviation between the initial and current schedules as schedule deviation measures. They either provide efficient algorithms or show that such algorithms are unlikely to exist. Recently Qi et al. (2006) give an overview of the disruption management in machine scheduling and concentrate on the single machine problems for which the shortest processing time rule is optimal for their efficiency measures.

There are a number of rescheduling studies in parallel identical machine environments. The most note-worthy of these studies are due to Church and Uzsoy (1992), Bean et al. (1991), Leung and Pinedo (2004), Alagöz and Azizoğlu (2003), Azizoğlu and Alagöz (2005), Curry and Peters (2005) and Özlen and Azizoğlu (2008). Church and Uzsoy (1992) consider single machine and parallel identical machine environments to minimize maximum lateness and the number of times rescheduling is done. They provide a simulation study to test the efficiencies of some strategies like periodic, event-driven and continuous rescheduling. Bean et al. (1991) propose an algorithm that enables the current schedule to match-up with the initial schedule, at

some future time. Leung and Pinedo (2004) consider an environment where the machines are subject to breakdowns and subsequent repairs, and the jobs are dependent and have deadlines. They aim to minimize the total flow time, makespan and maximum lateness. Alagöz and Azizoğlu (2003) and Azizoğlu and Alagöz (2005) address the trade-off between the total flow time and number of disrupted jobs. Azizoğlu and Alagöz (2005) develop a polynomial time algorithm to generate all non-dominated solutions, whereas Alagöz and Azizoğlu (2003) consider eligibility constraints and propose approximation and optimization algorithms. Recently Curry and Peters (2005) consider the total disruption cost as a schedule deviation measure and total tardiness as an efficiency measure. They propose a simulation study to test the performance of some heuristic procedures and rescheduling strategies. Ozlen and Azizoğlu (2008) develop a branch and bound algorithm to generate all efficient solutions with respect to the total flow time and total disruption cost criteria.

A number of rescheduling studies consider multi-stage environments, the most noteworthy of which are Aktürk and Görgülü (1999) and Li and Shaw (1996) on flow shops and Raheja and Subramaniam (2002), Mason et al. (2004) and Abumaizar and Svetska (1997) on job shops. Recently, Lim and Xu (2009) study a rescheduling problem on automated assembly lines.

In this study we consider a rescheduling problem on unrelated parallel machine environments that addresses the trade-off between the total flow time and the total disruption cost or the number of disrupted jobs criteria. To the best of our knowledge there is a unique rescheduling study for unrelated parallel machine environments which is due to Ozlen and Azizoğlu (2008) who generate all efficient solutions with respect to the total flow time and the total disruption cost criteria whereas we optimize any nondecreasing function of these two criteria.

PROBLEM DEFINITION

We consider a manufacturing environment with m unrelated parallel machines. We assume all jobs are available at time zero, and each should be assigned to one of the machines, and processed without interruption. Each job i is characterized by an integer processing time p_{ij} time units on machine j.

We assume that the initial schedule is known. There are disruptions of specified time units on some of the machines after executing the initial schedule for DT time units. At time DT, the jobs that are being processed on disrupted machines, and the jobs that start on or after DT on non-disrupted machines are to be rescheduled. The job that is being processed on any disrupted

machine at time *DT* requires all its processing to be done again on its new machine. We assume there are *n* such jobs. Once we take the reference starting point from time zero to *DT*, our rescheduling problem reduces to scheduling *n* jobs, available at time zero, on *m* unrelated parallel machines where machine *j* becomes available at time a_j. Accordingly a_j is either the time at which the machine is unavailable due to the disruption or the completion time of the job in process at time *DT*. We assume *DT*, and a_j are integers.

The scheduling cost, that defines our efficiency measure, is the total flow time, *F*. The total flow time is the total time the jobs spent in the system and therefore is the direct indication of total work-in-process inventory levels. As we assume all zero ready times, the total flow time and total completion time are equivalent measures. If we let C_i denote the completion time of job *i* in the current schedule, the total flow time, $F = \sum_{i=1}^{n} C_i$.

We call a job disrupted if it is assigned to different machines in the initial and current schedules by different machines. Our schedule deviation measures are the number of disrupted jobs and total cost incurred over all disrupted jobs, i.e., total disruption cost.

We let $R_i = \begin{cases} 1 & \text{if job } i \text{ is disrupted} \\ 0 & \text{otherwise} \end{cases}$.

The number of disrupted jobs, *ND*, is $\sum_{i=1}^{n} R_i$.

We let wn_{ij} be the integer disruption cost due to reassigning job *i* to machine *j*. In other words, wn_{ij} is the disruption cost incurred when job *i* is reassigned to machine *j* due to the additional set-up, adjustment, tooling, material/labor shifting, etc.. We assume wn_{ij} is zero if job *i* is assigned to machine *j* in the initial schedule, i.e., if it is not disrupted.

The total disruption cost, *WND*, is $\sum_i wn_{iM_i}$ where M_i is the current machine of job i.

A schedule *S* is said to be efficient with respect to *F* and *ND (WND)* if there exists no schedule *S′* with $F(S') \le F(S)$ and $ND(S') \le ND(S)$ ($WND(S') \le WND(S)$) with at least one strict inequality.

The standard classification scheme for scheduling problems use three-field representation $\alpha \mid \beta \mid \gamma$ where α is the machine environment, β specifies the constraints or special characteristics of the problem and γ is the objective function (see Lawler et al. (1989)). We consider unrelated parallel machines and hence set α = *R*. When the parallel machines are identical, i.e., $p_{i\,j} = p_i$ for all *i* and *j*, we set α = *P*. We have initial machine available times denoted by a_j in the β field. Moreover, we use the following constraints:

β : $F = F^*$: total flow time should be kept at its minimum value

β : $WND = WND^*$: total disruption cost is kept at its minimum value

β : $ND = ND^*$: total number of disrupted jobs is kept at its minimum value

β : $F \leq k$: total flow time can be at most k

β : $WND \leq k$: total disruption cost is at most k

β : $ND \leq k$: total number of disrupted jobs is at most k

We consider *F*, *WND*, *ND* as efficiency and schedule deviation measures, hence we set

γ : *F, WND* : generating a set of efficient schedules with respect to *F* and *WND*

γ : *F, ND* : generating a set of efficient schedules with respect to *F* and *ND*

γ : f (*F, WND*) : finding an optimal schedule of a specified function of *F* and *WND*

γ : f (*F, ND*) : finding an optimal schedule of a specified function of *F* and *ND*

SOLUTION PROCEDURES

In this section, we provide solution procedures to the unrelated parallel machine problems we discussed in the previous section.

The $R \mid a_j \mid F$ problem

Kaspi and Montreuil (1988) show that the $P \mid a_j \mid F$ problem can be solved in polynomial time by assigning the shortest available job to the earliest available machine. Lee and Liman (1992) and Mosheiov (1994) study the more general case of the $P \mid a_j \mid F$ problem where the machines are unavailable at arbitrary, but not necessarily initial, times. Lee (1996) and Lee and Chen (2000) consider the weighted version of the identical parallel machine flow time problem with arbitrary machine unavailable times. The arbitrary machine unavailable times on unrelated parallel machines are considered in several studies including Suresh and Chandhuri (1996), Yaghubian et al. (1999), Logendran and Subur (2004), and Logendran et al. (2005).

A special case of the $R \mid a_j \mid F$ problem where $p_{ij} = p_i$ or ∞ for all *i* and *j*, is formulated as an assignment problem in Alagöz and Azizoğlu (2003). We now extend this formulation to the arbitrary p_{ij} case.

Our decision variable is defined as

$$X_{ikj} : \begin{cases} 1 & \text{if job } i \text{ is the } k^{th} \text{last job on machine } j \\ 0 & \text{otherwise} \end{cases}$$

The objective function requires the minimization of the total flow time values, i.e.,

$$\text{Min} \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m}(kp_{ij}+a_j)X_{ikj} \qquad (1)$$

kp_{ij} is the contribution of the processing time of job i to the total flow time if sequenced at the k^{th} position from last on machine j and a_j is the start time of the first job on machine j.

The constraint sets are as stated below:

$$\sum_{k=1}^{n}\sum_{j=1}^{m}X_{ikj}=1 \qquad \forall\, i \qquad (2)$$

$$\sum_{i=1}^{n}X_{ikj}\le 1 \qquad \forall\, j,k \qquad (3)$$

$$X_{ikj}\in\{0,1\} \qquad \forall\, i,j,k \qquad (4)$$

Constraint sets (2) and (3) ensure that each job is scheduled exactly once and each position of each machine is occupied by at most one job. Constraint set (4) requires binary assignments. This is a weighted bipartite matching problem, so that the integrality constraints can be replaced by nonnegativity constraints without altering the feasible set (see Lawler (1989)).

The $R|a_j|ND$ and $R|a_j|WND$ problems

The $R|a_j|ND$ and $R|a_j|WND$ problems can be formulated as assignment models with the constraint sets (2), (3) and (4) and the following objective function

$$\text{Min} \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m}wn_{ij}X_{ikj} \qquad (5)$$

The objective function expressed in (5) requires the minimization of the total disruption cost. It reduces to minimizing the number of disrupted jobs when all wn_{ij} values are either 0 or 1.

The optimal solutions to the $R|a_j|ND$ and $R|a_j|WND$ problems can be found by applying the right-shift strategy to the initial schedule. The right-shift strategy shifts all jobs on disrupted machines, a_j time units to the right on the time-axis, while keeping the other job assignments same. The resulting schedule is optimal for both problems, as it has no disrupted jobs and zero total disruption cost, that is, as the *ND* and *WND* values are at their minimum

possible values of zero. The *F* values that solve $R|a_j|ND$ and $R|a_j|WND$ problems, i.e., *F* values of the right-shift schedules, give upper bounds on the *F* values of all efficient schedules.

The $R|a_j, F = F^*|WND$ and $R|a_j, F = F^*|ND$ problems

The $R|a_j, F = F^*|ND$ and $R|a_j, F = F^*|WND$ problems are singly-constrained assignment models. The additional single constraint to the assignment model (defined by the constraint sets (2), (3) and (4)) is expressed as follows:

$$F = F^* \equiv \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m}(kp_{ij} + a_j)X_{ikj} = F^*$$

The objective function of the $R|a_j, F = F^*|WND$ problem is to minimize the total disruption cost, i.e., Min $\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m} wn_{ij}X_{ikj}$ (expression (5)). The function reduces to *ND* when all wn_{ij} values are either 0 or 1.

Note that, *F**, i.e., the *F* value that solves the $R|a_j|F$ problem, gives a lower bound on the *F* values of all efficient solutions. However the resulting schedule may not be efficient as there may exist alternate optimal schedules to the $R|a_j|F$ problem having smaller *WND* values. Among the alternate optimal schedules to the total flow time problem, the one that has the smallest *WND* value, hence the efficient schedule requires an exact solution of the $R|a_j, F = F^*|WND$ problem. In place of incorporating $F = F^*$ to our assignment model, we can modify the objective function as $F + \varepsilon_{WND}\, WND$, for a sufficiently small value of $\varepsilon_{WND} > 0$. Theorem 1 states this result formally and defines a range for ε_{WND}.

Theorem 1. The $R|a_j, F = F^*|WND$ and $R|a_j|F + \varepsilon_{WND}WND$ problems are equivalent when $\varepsilon_{WND} < \dfrac{1}{\sum_{i=1}^{n} Max_j\{wn_{ij}\}}$.

Proof. ε_{WND} should be set small enough so that the total flow time value should not increase even for the largest possible value of the total disruption cost. That is $F^* + \varepsilon_{WND}WND_{UB} < F^* + 1$

should hold. This follows $\varepsilon_{WND} WND_{UB} < 1$, i.e, $\varepsilon_{WND} < \frac{1}{WND_{UB}}$. The maximum possible value of the total disruption cost is $\sum_{i=1}^{n} Max_j\{wn_{ij}\}$. Hence $\varepsilon_{WND} < \frac{1}{\sum_{i=1}^{n} Max_j\{wn_{ij}\}}$ should hold. ■

When wn_{ij} = 0 or 1 for all i and j, i.e, the number of disrupted jobs is of concern, then $\varepsilon_{WND} = \varepsilon_{ND} < \frac{1}{n}$, as $Max_j\{wn_{ij}\}=1$ for all i. In our experiments, we use $\varepsilon_{ND} = \frac{1}{n+1}$ and $\varepsilon_{WND} = \frac{1}{\sum_{i=1}^{n} Max_j\{wn_{ij}\}+1}$ for the $R|a_j|F+\varepsilon_{ND}ND$ and $R|a_j|F+\varepsilon_{WND}WND$ problems respectively.

The $R|a_j, ND = ND^*|F$ and $R|a_j, WND = WND^*|F$ problems

The $R|a_j, ND = ND^*|F$ and $R|a_j, WND = WND^*|F$ problems are both singly-constrained assignment problems. *ND** and *WND** are the ND and WND values that solve the $R|a_j|ND$ and $R|a_j|WND$ problems respectively.

The additional single constraint to the assignment model (defined by constraint sets (2), (3) and (4)) is expressed as follows:

$$WND = WND^* \equiv \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m} wn_{ij} X_{ikj} = WND^*$$

The above constraint reduces to $ND=ND^*$ when all wn_{ij} values are either 0 or 1.

The objective function of the problems is to minimize the total flow time, i.e., Min $\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m}(kp_{ij}+a_j)X_{ikj}$ (expression (1)).

The right-shift schedule solves the $R|a_j, ND = ND^*|F$ problem, as it produces an *ND* value of zero and all other schedules produce at least one disrupted job, i.e., $ND \geq 1$.

When wn_{ij} is zero, even when machine j is not the initial machine of job i, there can be a schedule, other than the right-shift schedule, having a *WND* value of zero and an *F* value that is smaller than that of the right-shift schedule. The efficient schedule having smallest *F* value, among the ones having zero *WND* value, can be found by solving the $R|a_j, WND = WND^*|F$ problem. Instead of treating *WND=WND** constraint, one can modify the objective function as

$WND + \varepsilon_F F$ for a sufficiently small value of ε_F. Theorem 2 states this result formally and defines a range for ε_F.

Theorem 2. The $R|a_j, WND = WND^*|F$ and $R|a_j|WND + \varepsilon_F F$ problems are equivalent when $\varepsilon_F < \frac{1}{F_{UB}}$ where F_{UB} is an upper bound on the F values of all efficient solutions.

Proof. ε_F should be set small enough so that $WND^* + \varepsilon_F F_{UB} < WND^* + 1$ should hold. That is the total disruption cost should not increase even for the largest possible value of the total flow time. Hence $\varepsilon_F F_{UB} < 1$, i.e, $\varepsilon_F < \frac{1}{F_{UB}}$. ■

In our experiments we use $\varepsilon_F = \frac{1}{F_{UB}+1}$ where F_{UB} is the F value that solves the $R|a_j|WND$ problem.

The constrained optimization problems

The $R|a_j, WND \le k|F + \varepsilon_{WND} WND$, $R|a_j, F \le k|WND + \varepsilon_F F$, $R|a_j, ND \le k|F + \varepsilon_{ND} ND$ and $R|a_j, F \le k|ND + \varepsilon_F F$ are all singly-constrained assignment problems. The additional single constraints to the assignment model (defined by constraint sets (2), (3) and (4)) are expressed below:

$$WND \le k \equiv \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m} wn_{ij} X_{ikj} \le k$$

$WND \le k$ reduces to $ND \le k$ when all wn_{ij} values are either 0 or 1.

$$F \le k \equiv \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m} (kp_{ij} + a_j) X_{ikj} \le k$$

The objective functions of the $R|a_j, WND \le k|F + \varepsilon_{WND} WND$, $R|a_j, F \le k|WND + \varepsilon_F F$, $R|a_j, ND \le k|F + \varepsilon_{ND} ND$ and $R|a_j, F \le k|ND + \varepsilon_F F$ problems are minimizing $\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m} (kp_{ij} + a_j) X_{ikj} + \varepsilon_{WND} \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m} wn_{ij} X_{ikj}$, $\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m} wn_{ij} X_{ikj} + \varepsilon_F \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m} (kp_{ij} + a_j) X_{ikj}$,

$\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m}(kp_{ij}+a_j)X_{ikj}+\varepsilon_{ND}\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m}wn_{ij}X_{ikj}$ and $\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m}wn_{ij}X_{ikj}+\varepsilon_{F}\sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m}(kp_{ij}+a_j)X_{ikj}$

respectively.

When the coefficients of the additional constraints are all 0 or 1, the singly-constrained assignment problem is an open problem (See Aggarwal (1985)), so is the $R|a_j, ND \le k|F+\varepsilon_{ND}ND$ problem. For arbitrary coefficients, the singly-constrained assignment problem is NP-Hard so are the $R|a_j, F \le k|ND+\varepsilon_F F$, $R|a_j, WND \le k|F+\varepsilon_{WND}WND$ and $R|a_j, F \le k|WND+\varepsilon_F F$ problems.

We generate the efficient schedules through the Procedure 1 below by varying *k* between WND_{LB} and WND_{UB}.

Procedure 1. Finding All Efficient Schedules

Step 0. Solve the $R|a_j|F+\frac{1}{\sum_{i=1}^{n}Max_j\{wn_{ij}\}+1}WND$ problem and form a right-shift schedule.

WND_{LB} = *WND* value of the right-shift schedule = 0

WND_{UB} = *WND* value that solves the $R|a_j|F+\frac{1}{\sum_{i=1}^{n}Max_j\{wn_{ij}\}+1}WND$ problem

Let $k = WND_{UB} - 1$

Step 1. Solve the $R|a_j, WND \le k|F+\frac{1}{\sum_{i=1}^{n}Max_j\{wn_{ij}\}+1}WND$ problem.

Let (F', WND') be the solution.

Step 2. If $WND' = WND_{LB}$ then STOP.

$k = WND' - 1$

Go to Step 1

Alternately, we could solve the $R|a_j, F \le k|WND+\varepsilon_F F$ problem and vary *k* between F_{LB} and F_{UB}.

Note that each step of Procedure 1 generates an efficient solution. The $R|a_j|F,WND$ problem has at most $Min\{F_{UB}-F_{LB}+1, WND_{UB}-WND_{LB}+1\}$, i.e., pseudo-polynomial number of efficient solutions. The $R|a_j|F,ND$ problem has at most *n* + 1, i.e., polynomial number of

efficient solutions, as $ND_{UB} \le n$ and $ND_{LB} = 0$ follows $Min\{F_{UB} - F_{LB} + 1, ND_{UB} - ND_{LB} + 1\} \le n + 1$. Hence the algorithm iterates pseudo-polynomial and polynomial number of times for *WND* and *ND* measures, respectively. Each iteration requires a solution of a singly-constrained assignment problem for which no polynomial algorithm exists for *ND* and cannot exist for *WND* measures.

The $R|a_j|f(F,WND)$ and $R|a_j|f(F,WND)$ problems

In this section, we address the problem of finding an optimal solution for a specified general non-decreasing function of *F* and *WND*.

When, the function, *f*, is a linear function of *F* and *WND* then one can use an assignment model (defined by the constraint sets (2), (3) and (4)) and the following objective function,

$$\text{Min } w_1F + w_2WND \equiv \text{ Min } w_1 \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m}(kp_{ij} + a_j)X_{ikj} + w_2 \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m} wn_{ij}X_{ikj}$$

and find an optimal solution in polynomial time.

When *f* is non-linear, finding an optimal solution to our assignment model would not be possible by available mathematical programming solvers. For non-linear *f*, one can generate all efficient solutions and select the one that minimizes the objective function value. However such an approach may not be time-efficient as each generation requires a solution of a singly-constrained assignment problem in exponential time. To overcome this difficulty, we develop an optimization algorithm that implicitly generates all efficient solutions. We start with the two extreme efficient schedules whose *F* and *WND* values provide upper and lower limits on *F* and *WND* values of all efficient schedules. In each iteration, we generate an efficient schedule by setting an upper limit on the *F* and *WND* values of any schedule that can improve the best known solution, namely f_{BEST}. By setting these limits, we eliminate some efficient schedules that cannot lead to an optimal solution. Kondakci et al. (1996) use the idea of putting upper limits on one criterion for their bicriteria single machine scheduling problem.

Moreover, we set lower limits on the *F* and *WND* values by solving the LP relaxations of the singly-constrained assignment problem. If the *f* value found by setting the lower limits is no smaller than f_{BEST}, then we terminate by recording the optimality of the best known schedule.

To find an initial f_{BEST}, we solve an assignment problem having an objective function of $Min \dfrac{w}{F_{UB} - F_{LB}} F + \dfrac{(1-w)}{WND_{UB} - WND_{LB}} WND$ for w=0.1, 0.5, 0.9 and get at most three distinct schedules. We also consider the *f* values of the two extreme efficient schedules. The smallest *f* value among five resulting schedules is used as an initial f_{BEST}. We update f_{BEST} whenever a

feasible schedule with smaller *f* value is reached. Procedure 2, below, is the stepwise description of our approach.

Procedure 2. Finding an Optimal Solution

Step 0. Solve the $R|a_j|F+\frac{1}{\sum_{i=1}^{n} Max_j\{wn_{ij}\}+1}WND$ problem and form a right-shift schedule.

Let F_{LB} = F value that solves the $R|a_j|F+\frac{1}{\sum_{i=1}^{n} Max_j\{wn_{ij}\}+1}WND$ problem.

F_{UB} = F value of the right-shift schedule

WND_{LB} = WND value of the right-shift schedule, i.e., zero

WND_{UB} = WND value that solves the $R|a_j|F+\frac{1}{\sum_{i=1}^{n} Max_j\{wn_{ij}\}+1}WND$ problem

Solve the $R|a_j|\frac{w}{F_{UB}-F_{LB}}F+\frac{(1-w)}{WND_{UB}-WND_{LB}}WND$ problem with w=0.1, 0.5, 0.9.

Let the solution be (F_w, WND_w) for a specified w value.

f_{BEST} = Min{$f(F_{LB},WND_{UB})$, $f(F_{UB},WND_{LB})$, $f(F_{0.1},WND_{0.1})$, $f(F_{0.5},WND_{0.5})$, $f(F_{0.9},WND_{0.9})$}

Step 1. If $f(F_{LB}, WND_{LB}) \geq f_{BEST}$ then STOP.

Find WND_a that solves $f(F_{LB}, WND_a) = f_{BEST}$.

$WND_{UB} = \lceil WRJ_a \rceil - 1$

If $WND_{UB} \leq WND_{LB}$ then STOP.

Solve the LP Relaxation of the $R|a_j, WND \leq WND_{UB}|F+\varepsilon_{WND}WND$ problem.

Let (F', WND') be the solution.

$F_{LB} = \lceil F' \rceil$

If $f(F_{LB}, WND_{LB}) \geq f_{BEST}$ then STOP.

If the resulting solution is integer then

f_{BEST} = Min {f_{BEST} , $f(F', WND')$}

$WND_{UB} = WND'$-1

If $WND_{UB} \leq WND_{LB}$ then STOP

Go to Step 3

Step 2. Find F_a value that solves $f(F_a, WND_{LB}) = f_{BEST.}$.

$F_{UB} = \lceil F_a \rceil - 1$

If $F_{UB} \leq F_{LB}$ then STOP.

Solve the LP Relaxation of the $R|a_j, F \le F_{UB}|WND + \varepsilon_F F$ problem.

Let (F', WND') be the solution.

$WND_{LB} = \lceil WND' \rceil$

If $f(F_{LB}, WND_{LB}) \ge f_{BEST}$ then STOP.

If the resulting solution is integer then

f_{BEST} = Min {f_{BEST} , $f(F', WND')$}

$F_{UB} = F'$-1

If $F_{UB} \le F_{LB}$ then STOP

Go to Step 1

Step 3. Solve the $R|a_j, WND \le WND_{UB}|F + \varepsilon_{WND} WND$ problem.

Let (F', WND') be the solution.

$F_{LB} = F'$ + 1

$WND_{UB} = WND'$ -1

If $F_{UB} \le F_{LB}$ or $WND_{UB} \le WND_{LB}$ then STOP.

f_{BEST} = Min {f_{BEST} , $f(F', WND')$}

Solve the $R|a_j, F \le F_{UB}|WND + \varepsilon_F F$ problem.

Let (F', WND') be the solution.

$WND_{LB} = WND'$+1

$F_{UB} = F'$-1

If $F_{UB} \le F_{LB}$ or $WND_{UB} \le WND_{LB}$ then STOP.

f_{BEST} = Min {f_{BEST} , $f(F', WND')$}

Go to Step 1

Note that we assume a single disruption or multiple disruptions occurring at the same time. When another disruption occurs, we have a new rescheduling problem and can use the same procedures. Hence each disruption can be handled in turn.

COMPUTATIONAL EXPERIENCE

We conducted a computational experiment to assess the efficiency of our algorithms. We generate random problem instances having n = 40, 60 and 80 jobs and m = 4, 8 and 12 machines. The job processing times, p_{ij} values, are drawn from two discrete uniform distributions between [1,100] and [50,100]. The disruption costs, wn_{ij} values, are drawn from two discrete uniform

distributions between [1,60] and [30,60]. The U[1,60] distribution has lower wn_{ij} values, hence the number of efficient solutions is likely to be lower and the range of the efficient solutions is likely to be narrower. We assume a disruption of duration, *D*, on the first machine. *D* is set to three levels: Long (*L*), Medium (*M*) and Short (*S*). For level *L*, *D* is set to the completion time of the last job on the disrupted machine in the initial schedule. Level *M* is set to the half of the duration of level *L*. Level *S* is set to the half of the duration of level *M*.

Our algorithm is applicable to all non-decreasing and non-linear functions of *F* and *WND*. To test the performance of our algorithm we adapted the following two non-linear, non-decreasing functions of *F* and *WND* from Kondakci et al. (1996). Note that the *F* and *WND* values may have significantly different magnitude, hence we scaled between [0,1].

$$f_1 = w\left(\frac{F - F_{LB}}{F_{UB} - F_{LB}}\right)^2 + (1-w)\left(\frac{WND - WND_{LB}}{WND_{UB} - WND_{LB}}\right)^2$$

$$f_2 = w\left(\frac{F - F_{LB}}{F_{UB} - F_{LB}}\right)^8 + (1-w)\left(\frac{WND - WND_{LB}}{WND_{UB} - WND_{LB}}\right)^8$$

We refer to f_1 and f_2 as quadratic and quasi-chebyschev functions respectively. For each function, we set *w* to 0.1, 0.5, and 0.9. Ten problem instances are generated for each combination of the parameter values. We conducted the experiment on a PC with Intel Pentium 4 2.8 Ghz processor and 1 GB of RAM running under Linux, specifically Fedora Core 4, operating system. We implemented our optimization algorithm in C, compiled with GCC 4.0.1 and utilized Borland C++BuilderX 1.0 as the development environment. We solved our integer and linear programming models using CPLEX 8.1.1.

Tables 1, 2 and 3 report the average number of the efficient solutions and average percentage of the efficient solutions searched by our algorithm when n = 40, 60 and 80 respectively. The tables also include the average CPU times in seconds for finding all efficient solutions using Procedure 1 and the average CPU time spent by our optimization algorithm, Procedure 2.

INSERT TABLES 1, 2, and 3 HERE

Table 4 reports the same statistics for the number of the disrupted jobs problem, i.e., the disruption cost is set to 1 on all machines, except the initial machine where it is zero. The table includes the results when n = 60 and 80 jobs and $D = L$ and M.

INSERT TABLE 4 HERE

It can be observed from the tables that the number of the efficient solutions increases as n increases. An increase in the difficulty of attaining the efficient solutions is much more pronounced than an increase in the number of the efficient solutions. This is due to the fact that as n increases the number of the integer variables used in our models increases considerably, which in turn increases the solution times of the integer programs.

For fixed n, the average number of efficient solutions and the CPU times of attaining these solutions are not always proportional. For example, when n = 60, m = 12, p_{ij} ~ U[50,100], wn_{ij} ~ U[30, 60] and the disruption duration is long, 39.6 efficient solutions are obtained in 154.8 CPU seconds on average. For the same n, when m = 8, p_{ij} ~ U[50,100], wn_{ij} ~ U[1, 60] and the disruption duration is long, more efficient solutions is obtained in smaller CPU times (44.9 solutions are generated in 84.1 CPU seconds) on average. This may be due two reasons: first, as m increases the number of the integer variables increases, thereby increasing the solution times of the integer programs. Second, some integer solutions are obtained very quickly in particular when the optimal LP solutions return many integer variables.

For all problem combinations, when the disruption duration is longer, there are more efficient solutions. This is because the jobs on the disrupted machine in the initial schedule are likely to be placed on the non-disrupted machines in the current schedule, thereby increasing the number of disruption choices. For m = 4, p_{ij} ~ U[1,60] and wn_{ij} ~ U[1, 60] , n= 40 combination, there are 28.5, 11.5 and 4.3 efficient solutions when the disruption durations are at levels *L*, *M* and *S*, respectively. For the same combination, but with 60 jobs, there are 60.3, 23.2 and 8.7 efficient solutions, and with 80 jobs, there are 91.3, 28.3 and 10.7 efficient solutions, when the disruption durations are set to *L*, *M* and *S* respectively. The average number of the efficient solutions generated increase, but the percentages of the efficient solutions decrease, when the disruption durations get longer. For example when the average number of efficient solutions is 91.3, 9.1 % (hence 8.3 efficient solutions) is generated and when there are 10.7 efficient solutions on average, 51.5 % (hence 5.5 efficient solutions) is generated.

Moreover when p_{ij} and wn_{ij} values are higher, i.e., p_{ij} ~ U[50,100] and wn_{ij} ~ U[30, 60], the number of efficient solutions is higher, as the range of the efficient solutions is wider. When the disruption costs are either 0 or 1, the integer programs are solved very quickly and in many cases the optimal LP relaxations return all integer values. As can be observed from Table 4, where wn_{ij} values are either 0 or 1, the average number of efficient solutions is smaller, and on average the CPU times of attaining those solutions are much lower.

In general, for all parameter combinations and both objective function types, the optimization algorithm finds the optimal schedule by generating only a small percentage of all efficient solutions. The higher percentages are associated to the cases with smaller number of efficient solutions. Hence, for those instances, the number of efficient solutions generated is also very small. Note that when n=80, the percentages are lower as the number of efficient solutions is higher.

The results on all tables reveal that our algorithm solves all instances in much smaller CPU time than that of spent in generating all efficient solutions.

CONCLUSIONS

In this study, we have addressed a rescheduling problem on unrelated parallel machines. We consider the trade-off between the efficiency of the current schedule, measured by the total flow time and the schedule deviation measured by the total disruption cost caused by the differences between the initial and current machine allocations. Hence we can classify our disruption management, i.e., rescheduling, problem as a multi-criteria problem where both efficiency and schedule deviation measures are explicitly included in the models.

We provide polynomial-time solution methods to the following hierarchical optimization problems: minimizing total disruption cost among the schedules with minimum total flow time and minimizing total flow time among the schedules with minimum total disruption cost. We propose an exponential-time algorithm for finding an optimal solution for a general non-decreasing function of the total flow time and total disruption cost criteria. Our computational runs on large sized problem instances revealed that our optimization algorithm generates only a small percentage of all efficient solutions and solves the instances in much smaller CPU time than that of spent in generating all efficient solutions. We could solve problems with up to 80 jobs and 12 machines in reasonable times. We hope our algorithm will serve as a useful tool for the practitioners who need to solve large sized problem instances and who are interested in reducing the schedule performance while keeping the assignments not too far from the initial plans.

The models we have studied represent a growth area in the rescheduling literature. Our results can be extended to the multi-stage scheduling problems that reside on unrelated parallel machines in each stage. Another interesting research extension might be addressing more general efficiency measures, like weighted flow time and total tardiness.

REFERENCES

Abumaizar R J and Svestka J A (1997). Rescheduling job shops under random disruptions, *Int J Prod Res* **35**: 2065-2082.

Aggarwal V (1985). A lagrangean-relaxation method for the constrained assignment problem. *Comp Oper Res* **12**: 97-106.

Aktürk M S, Görgülü E (1999). Match-up scheduling under a machine breakdown. *Euro J Oper Res* **112**: 81-97.

Alagöz O and Azizoğlu M (2003). Rescheduling of identical parallel machines under machine eligibility constraints. *Euro J Oper Res* **149**: 523-532.

Aytug H, Lawley M A, McKay K, Mohan S and Uzsoy R (2005). Executing production schedules in the face of uncertainties: a review and some future directions. *Euro J Oper Res* **161**: 86-110.

Azizoğlu M and Alagöz O (2005). Parallel machine rescheduling with machine disruptions. *IIE Trans* **37**: 1113-1118.

Bean J C, Birge J R, Mittenthal J, and Noon C E (1999). Matchup scheduling with multiple resources, release dates and disruptions. *Oper Res* **39**: 470-483.

Church L K and Uzsoy R (1992). Analysis of periodic and event-driven rescheduling policies in dynamic shops. *Int J Comp Integ M* **5**: 153-163.

Clausen J, Hansen J, Larsen J and Larsen A (2001). Disruption management. *ORMS Today* **28**: 40-43.

Curry J and Peters B (2005). Rescheduling parallel machines with stepwise increasing tardiness and machine assignment stability objectives. *Int J Prod Res* **43**: 3231-3246.

Daniels R L and Kouvelis P (1995). Robust scheduling to hedge against processing time uncertainty in single stage production. *Manage Sci* **41**: 363-376.

Hall N G, and Potts C N (2004). Rescheduling for new orders. *Oper Res* **52**: 440-453.

Kaspi M and Montreuil B (1988). *On the scheduling of identical parallel processes with arbitrary initial processor available time*, Research Report 88-12, School of Industrial Engineering, Purdue University.

Kondakci S, Azizoglu M and Koksalan M (1996) Note: Bicriteria scheduling for minimizing flow time and maximum tardiness. *Nav Res Logist* **43**: 929-936.

Lawler E L, Lenstra J K, Rinnooy Kan A H G and Shmoys D B (1989). *Sequencing and scheduling: algorithms and complexity*, Reports BS-R8909, Centre for Mathematics and Computers Science, Amsterdam.

Lee C Y (1996). Machine scheduling with an availability constraint. *J Glob Opt* **9**: 395-416.

Lee C Y and Chen Z L (2000). Scheduling jobs and maintenance activities on parallel machines. *Nav Res Logist* **47**: 929-936.

Lee C Y, and Liman S D (1992) Single-machine flow-time scheduling with scheduled maintenance. *Acta Inform* **29:** 375-382.

Leung J Y-T, and Pinedo M (2004). A note on scheduling parallel machines subject to breakdown and repair. *Nav Res Logist* **51**: 60-71.

Li E., and Shaw W (1996). Flow-time performance of modified-scheduling heuristics in a dynamic rescheduling environment, *Comp Ind Eng* **31**: 213 - 216.

Lim A., and Xu Z (2009). Searching optimal resequencing and feature assignment on an automated assembly line, *J Oper Res Soc* **60**: 361 - 371.

Logendran R, McDonell B and Smucker B (2005). Unrelated parallel machine scheduling with sequence-dependent set-ups, *IIE Ann Conf Exp*.

Logendran R, and Subur F (2004). Unrelated parallel machine scheduling with job splitting. *IIE Trans* **36:** 359-372.

Mason S J, Jin S and Wessels C M (2004). Rescheduling strategies for minimizing total weighted tardiness in complex job shops. *Int J Prod Res* **42**: 613-628.

Mosheiov G (1994). Minimizing the sum of job completion times on capacitated parallel machines. *Math Comp Mod* **20:** 91-99.

O'Donovan R, Uzsoy R and McKay K N (1999). Predictable scheduling of a single machine with breakdowns and sensitive jobs. *Int J Prod Res* **18**: 4217-4233.

Olumolade M O and Norrie D H (1996). Reactive scheduling system for cellular manufacturing with failure-prone machines. *Int J Comput Integr Manufact* **9**: 131-144.

Qi X T, Bard J R and Yu G (2006). Disruption management for machine scheduling: the case of SPT schedules. *Int J Prod Econ* **103**: 166-184.

Ozlen M and Azizoğlu M (2008). Generating all efficient solutions of a rescheduling problem on unrelated parallel machines. *Int J Prod Res*, forthcoming.

Raheja A S and Subramaniam V (2002). Reactive recovery of job shop schedules – A review. *Int J Adv Man Tech* **19**: 756-763.

Suresh V and Chaudhuri D (1996). Scheduling of unrelated parallel machines when machine availability is specified. *Prod Plan Cont* **7**: 393-400.

Ünal A T, Uzsoy R and Kıran A S (1997). Rescheduling on a single machine with part-type dependent setup times and deadlines. *Annals Oper Res* **70**: 93-113.

Vieira G E, Herrmann J W and Edward L (2003). Rescheduling manufacturing systems: a framework of strategies, Policies and Methods. *J Sch* **6**: 39-62.

Wu S D, Storer R H and Chang P C (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. *Comp Oper Res* **20**: 1-14.

Yaghubian A R, Hodgson T J, Joines J A, Culbreth C T and Huang J C (1999). Dry kiln scheduling in furniture production. *IIE Trans* **31**: 733-738.

Table 1. Results for Total Disruption Cost, n=40 (average of 10 instances)

m	p_{ij}	wn_{ij}	*D*	Number of Efficient Solutions	Procedure 1	OPTIMIZATION ALGORITHM (Procedure 2)											
						Quadratic Function						Quasi-Chebychev Function					
						w =0.1		*w*= 0.5		*w* =0.9		*w* =0.1		*w*= 0.5		*w* =0.9	
					CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time
4	U[1,100]	U[1,60]	L	28.5	8	20.2	1.8	18.1	1.7	21.7	2.1	20.7	1.9	18.7	1.7	21.9	2.7
			M	11.5	1.9	44.9	0.9	36.8	0.8	48.5	1.3	40.8	0.7	44.2	1	46.4	1
			S	4.3	0.6	62.3	0.6	60.4	0.5	62.9	0.5	59.5	0.4	62.9	0.6	62.9	0.5
		U[30,60]	L	32.6	18.1	21.0	3.7	16.0	2.2	16.9	3.5	25.3	5.2	21.9	4.1	16.0	2.8
			M	10.3	1.8	40.4	1.5	30.2	0.6	38.2	0.9	50.4	1.2	40.8	0.7	36.0	0.6
			S	4.4	0.7	46.3	0.5	55.8	0.6	56.6	0.6	51.7	0.6	53.3	0.6	55.3	0.7
	U[50,100]	U[1,60]	L	42.7	20.4	12.2	1.7	15.6	1.9	12.8	3	13.2	2.2	14.3	2.7	15.7	3.1
			M	18.5	4.5	23.7	0.9	25.3	1.1	23.6	1.1	29.4	1.2	23.5	1.1	24.6	1.1
			S	8.2	1.4	48.1	0.4	52.3	0.6	60.0	0.8	54.1	0.7	53.6	0.7	55.9	0.8
		U[30,60]	L	53.5	31.2	12.2	4.9	16.7	5.8	19.4	8.9	20.2	6.5	20.0	7.6	14.8	4.3
			M	17.3	4.4	23.0	1.3	28.1	1.4	29.2	1.8	29.6	2.3	30.7	1.8	25.0	1
			S	6.4	1.1	55.2	0.6	58.9	0.6	63.3	0.7	55.9	0.6	60.0	0.6	58.9	0.6
8	U[1,100]	U[1,60]	L	12.8	4.4	29.6	1.2	38.9	1.8	36.5	1.6	31.8	1.4	37.4	1.8	38.6	1.8
			M	5.4	1.3	50.7	0.9	57.0	1	58.7	1	61.3	1	62.4	1	69.3	1
			S	2.9	0.7	56.1	0.9	50.7	0.9	50.7	0.9	50.7	0.8	50.7	0.9	50.7	0.9
		U[30,60]	L	9.9	3.7	50.6	1.9	48.2	1.4	48.0	1.6	53.1	1.8	53.3	1.9	49.3	1.4
			M	4	0.9	71.7	1	77.0	1	76.7	0.9	77.3	1	76.7	0.9	80.0	0.9
			S	1.8	0.4	26.7	0.7	26.7	0.9	26.7	0.8	26.7	0.8	26.7	0.9	26.7	0.8
	U[50,100]	U[1,60]	L	26.8	13.4	14.5	1.3	14.4	1.9	17.6	2.6	16.0	1.8	16.7	2.9	18.7	2.8
			M	13.3	4.9	28.8	1.4	32.0	1.9	24.2	1.7	30.1	1.5	27.8	1.5	33.2	1.9
			S	4.9	1.8	59.9	1	62.9	1.2	64.4	1.4	62.4	1.1	62.9	1.2	62.9	1.2
		U[30,60]	L	36.7	25.2	12.8	2.6	22.4	5	12.9	2.5	24.2	7.2	24.0	6.3	19.8	4.6
			M	12.4	6.2	28.9	1.3	30.1	2.1	29.2	1.6	30.4	1.5	33.7	2.3	37.2	2.7
			S	5	2.2	39.0	1	51.0	1.6	47.1	1.4	44.1	1.3	51.0	1.8	47.6	1.4
12	U[1,100]	U[1,60]	L	9.9	5.6	51.7	2	46.9	2	48.5	1.7	50.3	2.5	49.5	2.4	47.7	1.8
			M	5.8	2.4	62.2	1.5	61.5	1.3	71.8	1.6	68.2	1.7	66.0	1.6	68.4	1.5
			S	2.5	0.8	38.8	1.2	43.5	1.5	40.2	1.2	40.2	1.3	43.5	1.6	43.5	1.5
		U[30,60]	L	6.8	3.7	40.9	1.7	46.2	1.8	48.3	1.7	50.6	1.8	47.3	1.8	50.7	1.6
			M	3.6	1.3	72.1	1.2	83.3	1.6	80.0	1.3	83.3	1.5	89.2	1.6	89.2	1.6
			S	2	0.8	18.7	1.1	24.7	2.1	20.7	1.2	22.7	1.9	24.7	1.8	22.7	1.9
	U[50,100]	U[1,60]	L	19.6	11.1	25.8	3	21.0	2.1	17.1	1.6	24.4	3.1	22.2	2.7	22.6	2.7
			M	6.7	2.3	47.3	1.4	57.1	2	46.6	1.4	56.6	2	57.2	2.3	60.9	1.8
			S	1.7	0.3	12.5	1.2	20.8	2.7	15.0	1.1	15.0	1.7	20.8	2.6	20.8	2.2
		U[30,60]	L	20.6	21.1	25.3	1.8	31.2	2.6	25.6	2.5	29.2	3	27.6	3.7	28.0	3.3
			M	8.1	6.3	43.4	1.3	61.6	2	47.7	3.3	56.6	2.3	59.8	2.6	53.3	1.7
			S	2.7	0.9	15.0	1.8	19.6	2.8	15.0	1.4	20.8	3.6	19.6	3	17.9	2.3

Table 2. Results for Total Disruption Cost, n=60 (average of 10 instances)

m	p_{ij}	wn_{ij}	*D*	Number of Efficient Solutions	Procedure 1	OPTIMIZATION ALGORITHM (Procedure 2)											
						Quadratic Function						Quasi-Chebychev Function					
						w =0.1		*w*= 0.5		*w* =0.9		*w* =0.1		*w*= 0.5		*w* =0.9	
					CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time
4	U[1,100]	U[1,60]	L	60.3	138	13.4	17.9	14.8	39.6	14.6	15.7	16.3	44.6	14.6	41.3	15.2	33.1
			M	23.2	18.5	19.2	5.4	21.3	5.2	19.3	4.5	27.0	7.1	21.6	6	20.6	6.5
			S	8.7	3.4	49.1	1.4	56.3	2.6	57.5	2.5	55.5	2.1	57.4	2.8	55.3	1.7
		U[30,60]	L	80.5	1952.22	17.5	95.8	13.8	142.3	14.1	50.3	16.9	735.3	14.7	105.1	12.7	133.1
			M	25.8	34.7	27.1	13.3	24.4	11.3	17.6	7.1	28.1	15	24.9	12.7	19.3	7.5
			S	8.1	4.6	42.4	1.6	43.6	2.2	46.5	1.9	46.1	2.3	43.5	2	49.3	3.2
	U[50,100]	U[1,60]	L	75.7	506.2	5.4	3.4	5.6	3.2	7.7	7.4	8.3	11.4	7.6	7.4	6.6	13.8
			M	30.8	30.9	12.6	2.8	14.0	3	12.1	2.6	14.8	4.3	14.4	5.4	16.3	5.2
			S	13.9	6.5	47.2	1.3	48.2	1.9	49.7	1.6	51.1	2.9	50.0	1.9	50.0	1.8
		U[30,60]	L	126.8	1506.1	8.9	39.4	11.1	43.9	16.9	108.2	13.6	60.1	12.7	59.4	9.5	48.3
			M	41.1	68.8	16.7	13.3	18.7	18.5	22.2	20.1	24.2	26.6	18.7	19.2	18.8	13.9
			S	12.6	9.7	46.0	1.9	47.3	3.9	41.9	1.3	48.3	4.4	51.8	5.2	50.2	3.2
8	U[1,100]	U[1,60]	L	26.8	37.3	24.4	8.7	19.0	5.4	20.6	7.5	26.4	9	22.0	6.7	20.3	8.4
			M	9	8.9	44.4	2.8	45.3	2.8	38.4	3.3	46.9	2.8	47.2	3.5	42.2	3.5
			S	2.8	1.6	34.7	2.1	34.7	2.3	37.2	2.1	34.7	2.1	34.7	2.2	35.8	2.6
		U[30,60]	L	23.4	37.8	22.6	12.4	20.4	12.5	19.5	5.2	29.4	16.2	24.1	12.4	23.8	10.9
			M	8.5	7.5	39.8	5.7	49.4	3.8	35.3	2.6	53.6	5.1	51.4	3.8	43.2	3.7
			S	3.8	3.8	46.7	2.3	52.4	3.2	51.0	2.1	51.0	2.6	52.4	3.3	52.4	2.9
	U[50,100]	U[1,60]	L	44.9	84.1	8.7	4.7	10.2	9.7	9.4	5.9	9.3	6.9	14.4	17.1	16.0	16
			M	20.2	30.8	19.0	3.5	21.7	7.3	16.7	5.5	20.4	5.7	24.4	9.7	24.1	10.1
			S	9.8	9.1	55.5	3.8	55.6	4.4	50.5	2.7	59.3	5.1	58.9	4.5	57.7	3.9
		U[30,60]	L	65.5	208.9	12.3	31.9	17.6	37.9	12.8	27.9	20.9	48.7	17.1	37.1	15.8	37
			M	28.6	80.1	15.9	6	26.5	16.8	16.1	10.7	24.9	15.1	25.7	21.2	25.1	15.5
			S	11.7	24.5	44.1	3.7	40.4	3.7	42.6	6.5	41.4	4.9	41.8	5.1	43.4	9.1
12	U[1,100]	U[1,60]	L	16.3	33.7	23.3	5.9	26.6	5.8	27.5	7.6	28.8	7.4	29.1	6.9	30.5	11.4
			M	4.9	5.4	58.3	3.5	64.4	4.2	70.4	4.2	62.0	3.8	62.7	3.7	67.6	6.1
			S	2.8	3.2	45.3	3	57.2	5.3	50.7	3.4	55.2	4.5	57.2	5.6	53.2	5.2
		U[30,60]	L	11	21.5	36.2	9.3	29.4	4.9	35.8	5.3	45.8	7.5	44.7	6.3	37.2	6.2
			M	4	4	64.1	4.3	67.5	3.9	77.4	5.7	72.5	5.4	73.3	3.9	76.7	3.8
			S	2	1.7	36.0	2.8	36.0	3.5	36.0	3	36.0	3.2	36.0	3.4	36.0	3.5
	U[50,100]	U[1,60]	L	32.7	82.7	10.5	4.9	13.6	7.9	13.6	13	15.7	9	14.8	11.1	16.4	14.4
			M	13.4	28.3	28.6	5.6	27.5	5.2	25.9	7.8	29.4	7.2	29.9	6.8	29.8	7.4
			S	4.3	8	50.4	4	58.8	6.7	58.8	6	56.3	6.6	60.2	6.6	57.9	4.9
		U[30,60]	L	39.6	154.8	12.0	10.8	15.1	15.4	18.8	26.5	23.4	33.3	17.6	20.4	13.3	13.1
			M	15.1	49.4	23.7	5.5	26.0	12.2	39.8	15.7	27.8	9.9	26.5	8.9	34.6	21.4
			S	5.9	14.3	39.5	4.3	62.2	13.5	41.1	8.8	60.6	9.4	57.3	13.8	56.1	9.5

Table 3. Results for Total Disruption Cost, n=80 (average of 10 instances)

m	p_{ij}	wn_{ij}	D	Number of Efficient Solutions	Procedure 1	OPTIMIZATION ALGORITHM (Procedure 2)											
						Quadratic Function						Quasi-Chebychev Function					
						w=0.1		w=0.5		w=0.9		w=0.1		w=0.5		w=0.9	
					CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time
4	U[1,100]	U[1,60]	L	91.3	3795.2	9.1	76.8	8.5	193.6	12.3	226.7	8.6	274.1	10.3	268.1	12.9	551.8
			M	28.3	136.3	15.0	7.9	18.6	13	22.2	24.2	21.0	14.6	20.8	16.5	19.5	14.7
			S	10.7	11.1	51.5	5	51.7	5.5	48.7	4.1	49.8	4.2	53.2	5.6	58.4	9
		U[30,60]	L	119.8	10952.8	15.5	548.4	8.8	1439.8	10.5	116.5	12.5	398	8.9	1061.9	6.9	216
			M	39.5	1008.5	19.5	23.5	20.9	408.3	19.3	20.2	27.8	609.5	20.8	607.4	18.2	482.4
			S	13	36.2	37.4	4.5	36.6	4.3	40.1	5.1	41.5	7.5	39.7	7.4	41.2	5.7
	U[50,100]	U[1,60]	L	121.2	3428.7	4.3	13.2	5.0	47.3	7.7	82.4	6.5	276.6	6.1	100	7.5	136.1
			M	50.5	488.5	8.6	7.4	12.6	20.9	9.5	163	11.2	16.4	13.3	32.7	13.0	60.6
			S	19.5	30.2	43.1	3.5	44.0	4.3	42.1	4.1	45.9	5.3	43.9	5.9	48.4	8.8
		U[30,60]	L	216	14591.4	8.7	533.9	7.2	900.7	11.5	434	10.0	1223.7	7.6	1104.2	6.6	2297.9
			M	65.4	694.4	14.3	40.7	15.0	71.5	15.9	53.9	18.2	85.3	16.7	105.8	14.3	81.9
			S	19.2	54.2	34.8	5	40.7	12.2	35.6	7.8	38.1	7.9	39.6	12	46.0	16.9
8	U[1,100]	U[1,60]	L	35.3	117.3	20.3	30.2	16.0	16.1	16.3	22.6	21.5	26.5	18.6	21.9	18.5	23
			M	12.8	26.0	38.5	13	31.3	7.3	30.9	7.2	31.8	8.6	32.8	8.4	34.9	7.8
			S	6.6	13.5	61.3	6.1	63.2	6.4	64.0	8	63.9	6.7	67.1	8.8	61.1	6.9
		U[30,60]	L	39.2	221.2	25.0	58.1	18.5	37.2	15.7	29.9	25.5	57	19.8	40	19.6	47.9
			M	15.1	42.8	36.5	22.4	33.6	17.9	28.4	9.1	44.4	17.4	43.7	25.2	34.2	15
			S	6.2	15.8	55.1	6.7	51.4	7.5	47.6	4.9	57.1	7.4	55.8	10	53.9	6.7
	U[50,100]	U[1,60]	L	67.2	389.3	6.7	13.1	9.1	24.8	6.5	18.7	11.0	32.1	8.7	30.1	9.5	39.7
			M	31.8	140.2	13.4	8.8	11.5	7	13.0	16.4	14.6	15.9	13.3	12.8	16.3	23.1
			S	14.7	44.6	43.0	8.8	42.0	9.7	39.1	7.6	42.9	8.4	45.0	12.7	45.6	10.5
		U[30,60]	L	92.8	941.2	10.4	80	13.3	135.9	9.8	70	16.2	124.3	15.7	159.3	12.7	126.8
			M	30.6	235.8	13.1	12.4	18.8	31	17.0	31.3	22.5	50.4	23.4	50.2	22.5	54.4
			S	10.4	48.7	46.6	9.3	44.0	13.8	40.8	7.2	43.1	12.3	46.3	14.1	50.2	15.4
12	U[1,100]	U[1,60]	L	27.7	134.7	20.6	21.1	25.1	44.3	20.6	19.5	22.8	33.8	26.1	48.5	27.1	43.6
			M	7.9	24.3	49.2	12.2	46.7	9.6	53.4	11.2	55.0	7.9	55.3	11.8	56.2	11.9
			S	3.7	13.2	49.8	7.2	52.2	7.8	52.2	6.4	48.8	8.4	53.8	9.6	53.8	10.5
		U[30,60]	L	25.9	153.1	28.2	54.6	22.8	27.2	20.2	20.3	33.6	33.7	27.4	32	22.9	25.8
			M	11.2	56.5	37.7	19.7	42.1	16.2	29.4	7.5	50.1	19.2	52.3	19.1	52.3	16.1
			S	4.7	21.9	49.6	10.7	38.9	6.8	38.9	11.8	45.9	10	40.9	6.9	40.3	6.9
	U[50,100]	U[1,60]	L	63.8	488.5	9.3	22.8	7.8	27.2	5.3	13.9	9.7	36.9	8.5	36.2	10.7	48.2
			M	25.8	158.8	20.1	18.1	14.3	11.8	17.3	27.9	20.1	20.6	16.2	16.9	17.0	19.2
			S	10.4	52.7	54.3	11.9	54.7	12.4	55.9	14.5	55.7	11.8	55.5	13.2	58.7	17.7
		U[30,60]	L	65.2	958.0	8.1	46.7	14.8	107.7	10.7	64.3	18.6	144.1	13.1	110	15.3	131
			M	24.7	274.1	17.5	27.4	25.5	52.2	16.7	23.8	24.5	38.1	29.1	71.7	23.2	56.7
			S	10	95.2	40.4	11.6	45.9	17.4	43.5	16.6	42.4	12.3	47.2	21.7	49.7	28.3

Table 4. Results for the Number of Disrupted Jobs (average of 10 instances)

n	*m*	p_{ij}	*D*	Number of Efficient Solutions	Procedure 1	OPTIMIZATION ALGORITHM (Procedure 2)											
						Quadratic Function						Quasi-Chebychev Function					
						w =0.1		*w*= 0.5		*w* =0.9		*w* =0.1		*w*= 0.5		*w* =0.9	
					CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time	%ES generated	CPU time
60	4	U[1,100]	L	11.3	2.7	38.5	1	33.9	1	35.2	1	31.2	1	29.3	0.9	26.9	0.9
			M	7	1.6	52.2	0.9	50.0	0.9	45.4	0.9	47.1	0.9	44.3	0.9	44.3	0.8
		U[50,100]	L	13.1	3.6	33.6	1.1	31.4	1.1	34.7	1.1	31.4	1.1	26.1	1	27.6	1.1
			M	7.5	1.9	46.6	1	49.1	1	45.7	1	41.7	1	41.7	1	40.5	1
	8	U[1,100]	L	8.9	4.2	40.1	1.6	46.8	1.7	39.1	2	41.8	1.6	38.8	1.6	37.1	1.6
			M	5.4	2.4	59.6	1.6	65.5	1.6	61.2	1.8	65.5	1.5	64.6	1.5	61.2	1.6
		U[50,100]	L	8.9	4.4	44.5	1.9	41.0	1.9	40.2	1.9	36.4	1.9	35.4	1.8	34.5	1.8
			M	6.3	3.1	48.0	1.7	57.3	1.8	51.4	1.7	48.0	1.6	48.0	1.6	51.4	1.7
	12	U[1,100]	L	7.1	4.8	44.3	2.6	51.3	2.7	48.7	2.6	45.6	2.5	46.6	2.6	46.6	2.6
			M	3.6	1.7	68.7	2.3	86.5	2.5	86.2	2.5	66.7	2.3	71.2	2.3	74.5	2.4
		U[50,100]	L	9	7.6	34.3	2.8	36.5	2.9	34.3	2.8	34.3	2.8	34.3	2.7	34.3	2.6
			M	5.3	4.2	56.2	2.9	59.5	3	61.2	2.9	54.2	2.6	56.2	2.6	59.5	2.7
80	4	U[1,100]	L	14.3	7.6	35.0	2.1	33.6	2.2	31.2	2.2	31.7	2.1	26.8	2.1	23.9	2.1
			M	7.7	3.7	44.9	2	45.5	2	50.7	2	41.3	1.9	41.3	1.9	40.5	1.9
		U[50,100]	L	17.3	10.8	32.5	2.3	27.3	2.4	29.2	2.5	30.1	2.3	24.9	2.3	22.6	2.3
			M	10.1	5.9	40.2	2.2	40.2	2.2	34.2	2.2	37.0	2.2	34.0	2.2	30.2	2.1
	8	U[1,100]	L	9.8	9.4	42.7	3.6	40.4	3.6	39.0	3.6	38.8	3.5	35.2	3.4	34.0	3.5
			M	6.1	5.2	51.6	3.3	54.6	3.4	57.4	3.4	53.0	3.3	53.0	3.2	53.0	3.2
		U[50,100]	L	12.7	15.9	35.9	4.4	32.8	4.4	29.5	4.5	32.0	4.3	26.2	4.3	25.6	4.3
			M	7.6	8.9	45.8	4	49.4	4.1	42.4	4.1	41.7	3.8	40.8	4	40.8	3.9
	12	U[1,100]	L	8	11.6	42.1	5	48.1	5.2	47.0	5.4	43.5	4.9	43.5	4.9	41.8	4.9
			M	4.6	5.4	56.7	4.6	79.6	5.1	77.5	5	60.4	4.5	60.4	4.5	67.1	4.6
		U[50,100]	L	11.2	20.8	28.4	5.9	33.9	6.1	30.8	6.1	29.9	6	28.9	5.9	27.4	5.7
			M	7.1	13	46.0	5.7	51.1	5.8	44.9	5.9	44.9	5.6	44.9	5.4	44.9	5.4

Table Captions

Table 1. Results for Total Disruption Cost, n=40 (average of 10 instances)

Table 2. Results for Total Disruption Cost, n=60 (average of 10 instances)

Table 3. Results for Total Disruption Cost, n=80 (average of 10 instances)

Table 4. Results for the Number of Disrupted Jobs, n=40 (average of 10 instances)