Minimizing Weighted Tardiness of Jobs with Stochastic Interruptions in Parallel Machines[*]

Manuel Laguna[a], Pilar Lino[b], Angeles Pérez[b],
Sacramento Quintanilla[b], and Vicente Valls[c]

a Graduate School of Business and Administration, Campus Box 419, University of Colorado, Boulder, CO 80309-0419, USA. Manuel.Laguna@Colorado.EDU.

b Dpto. de Economía Financiera y Matemática, Facultad de Económicas y Empresariales, Universitat de Valencia, Avda. de los Naranjos, s/n, Edificio Departamental Oriental, Valencia, Spain. Pilar.Lino@uv.es, Angeles.Perez.@uv.es, Maria.Quintanilla@uv.es.

c Dpto. de Estadística e Investigación Operativa, Facultad de Matemáticas, Universitat de Valencia, Dr. Moliner, 50, 46100 Burjassot, Valencia, Spain. Vicente.Valls@uv.es.

Latest Revision: September 29, 1998

Abstract — In this paper, we address the problem of minimizing expected total weighted tardiness of jobs that have stochastic interruptions and that are processed on a set of parallel machines. Our research generalizes the problem of scheduling parallel machines to minimize total weighted tardiness. The proposed solution method is based on the scatter search methodology and implements an innovative structured combination procedure. Extensive computational testing with more than 400 problem instances shows the merit of the proposed solution method.

[*] This research was partially supported by the Generalitat Valenciana under contract GV-3310/95.

1. Introduction

In this paper, we address the problem of scheduling parallel resources used to process a set of jobs, some of which may be interrupted by an uncertain amount of time. The resources may be machines in a production system, computers with specialized software packages (as those needed for engineering designs), highly specialized technicians, and so on and so forth. The set of jobs J consists of a subset of "deterministic" jobs (DJ) for which their duration is known and a subset of "stochastic" jobs (SJ) with uncertain duration. Hence, these sets are such that:

$$J = DJ \cup SJ \text{ and } DJ \cap SJ = \varnothing.$$

For a job $j \in SJ$, the initial processing time is considered to be known, however, the length of the interruption and the final processing time are uncertain. An example of such a situation may be a system in which some jobs are submitted to an approval process before they can be completed. The time to review and approve the work performed during the job's initial processing determines the length of the interruption, while the outcome of the approval process may or may not influence the length of the final processing. A purchasing activity, for instance, may require an initial round of negotiations, followed by an approval process. If the initial contract is approved, then the final processing may be shorter than when more negotiating is necessary. In a quality control environment, the interruption may be related to an off-line testing process. If the outcome of the testing is positive, the activity may terminate (resulting in a zero processing time for the final processing). The duration $d(j)$ of a job $j \in DJ$ is simply the processing time of such activity, while the duration of a job $j \in SJ$ is given by:

$$d(j) = t_1(j) + w(j) + t_2(j),$$

where $t_1(j)$ is the initial processing time, $w(j)$ is the waiting time during the interruption, and $t_2(j)$ is the processing time after the interruption. In settings where $w(j)$ represents the time needed for an approval process to take place, the final processing time, $t_2(j)$ may depend on the outcome of this secondary process. Also, there may be situations in which the value of $t_2(j)$ does not depend on the outcome of the interruption, for example, when the interruption corresponds to subcontracted work that has an uncertain lead-time. The final processing time, in this case, can be considered certain. We address the problem from the point of view that the final processing time may or may not be stochastic, and therefore, we develop a solution procedure capable of handling both situations.

Associated with each job j there is a due date $date(j)$ and a tardiness penalty $pen(j)$, which is assessed to the time elapsed between the due date and the job completion $comp(j)$. Although the problem may be generically stated in terms of resources, we will consider jobs and machines as in a production environment. The production facility consists of m parallel and identical machines. A job $j \in J$ can be processed on any machine. Also, if a job $j \in SJ$ is assigned to machine k, then such job must be processed entirely on this machine. In other words, the load on machine k due to job $j \in SJ$ is $t_1(j) + t_2(j)$.

The problem is to schedule a set of jobs J (with $n = |J|$) for processing on m parallel identical machines in order to minimize total weighted tardiness (WT). In mathematical terms, we seek a schedule that minimizes:

$$WT = \sum_{j \in J} pen(j) * \max(0, comp(j) - date(j)).$$

Since the duration of a job in *SJ* is uncertain, the scheduling problem is stochastic in nature. The problem is therefore a generalization of the problem of scheduling jobs on parallel machines to minimize total weighted tardiness recently revisited by Alidaee and Rosa (1997). Note that, to the best of our knowledge, the problem of scheduling jobs with stochastic interruptions has not been studied, with the exception of the work by Valls, et al. (1998) in the context of project scheduling.

Uncertainty in key data is typically handled via scenarios. Optimization techniques based on scenarios construct a so-called *deterministic equivalent* problem. These formulations capture the uncertain nature of the problem while allowing the use of techniques that are typical to deterministic settings (such as linear programming, for example).

2. Robust Optimization Formulation

Robust optimization (Mulvey, et al. 1995) belongs to a family of scenario-based optimization techniques. The main feature of robust optimization (RO) formulations is the flexibility to define the tradeoff between *model* robustness and *solution* robustness. A solution is robust with respect to the model (usually represented by a set of constraints), if it is feasible or “almost” feasible under many scenarios. Model robustness can be measured, for example, by the expected value of the infeasibility. Solution robustness, on the other hand, refers to being “close” to the optimal solution with respect to any given scenario. Measuring the deviation of the proposed solution to the scenario-optimum gives an idea of its robustness. The concept of model robustness can be illustrated in the setting of a capacity expansion problem with uncertain demand. The decision problem in this case is how and when to increase capacity, at the risk that under certain scenarios the system may not be able to satisfy demand. The tradeoff between the cost of increasing capacity and the penalty for not meeting demand can be modeled within the RO framework.

The use of scenarios as a tool for modeling uncertainty has the advantage of not requiring knowledge of the underlying probability distributions associated with the random variables. For example, it is not necessary to assume that the demand or the processing times must follow a Normal or Uniform distribution. The main disadvantage, however, is that the deterministic equivalent problem tends to be large, and therefore, an important body of research in the area of stochastic optimization concentrates on developing efficient methods to deal with large-scale systems. In the book by Kouvelis and Yu (1997), the reader can find additional information about robust optimization as applied to discrete optimization problems.

In our context, we assume that the uncertainty associated with the random variables $w(j)$ and $t_2(j)$ can be modeled as a set S of scenarios with known probabilities. A scenario $s \in S$ with probability p_s of occurring is then given by:

$$s = \{ (w^s(j),\ t_2^s(j))\ \forall\, j \in SJ \}.$$

A number of RO formulations can be obtained by a combination of solution and model robustness terms. Two examples of possible formulations follow.

Expected Weighted Tardiness

In an expected value formulation, we attempt to minimize the following function:

$$EWT = \sum_{S} p_s \sum_{j \in J} pen(j) * \max(0, comp(j,s) - date(j))$$

Where *comp*(*j*,*s*) is the completion time of job *j* under scenario *s*. Note that the completion time for all jobs (and not only those in *SJ*) depends on each scenario. This is due to the recourse of making certain adjustments to the schedule once the length of the interruption is known. We address this issue in more detail in the next section.

Mean-Variance

A mean-variance formulation attempts to minimize a non-linear objective function of the following form:

$$MV = EWT + \lambda \sum_{S} p_s \left(EWT - \sum_{j \in \mathbf{J}} pen(j) * Max(0, comp(j,s) - date(j)) \right)^2$$

In this formulation, λ reflects the risk attitude of the decision maker. We assume that risk is measured by the variability on the due date violation resulting from implementing a proposed schedule under the uncertainty modeled by the set of scenarios.

3. Job-Insertion as a Recourse

The stochastic version of the scheduling problem we are addressing, consists of two phases. In the first phase, jobs must be assigned to machines, and an initial processing sequence in each machine must be determined. The initial processing sequence includes all jobs $j \in DJ$ and the first part of those jobs with stochastic interruption. We will refer to the initial processing (or first part) of job $j \in SJ$ as job j_1 with processing time $t_1(j)$. Similarly, the final processing (or second part) of job $j \in SJ$ will be referred to as job j_2 with processing time $t_2(j)$.

Also in the first phase, a decision must be made to whether machine *k* assigned to job $j \in SJ$ will wait during the job interruption or not, i.e., we must establish the *interruption policy* for each job $j \in SJ$. If the policy is to make the machine wait during the interruption of job *j* (i.e., the machine is “locked” during the interruption), then the second part of the job begins processing immediately after the interruption. On the other hand, if the policy is to “free” the machine during the interruption of job *j*, then job j_2 may have to wait before processing can resume (because other job may be in process immediately after the interruption terminates). If the decision is to lock the machines after processing the first part of all jobs $j \in SJ$, then the sequences are exactly the same under each scenario (i.e., the problem has no recourse).

Let the interruption policy for a job $j \in SJ$ be given by *f*(*j*) in such a way that if the machine is freed during the interruption then *f*(*j*) = 1, while *f*(*j*) = 0 when the machine is locked. Then, for each job $j \in SJ$ with *f*(*j*) = 1 the two jobs j_1 and j_2 with processing times $t_1(j)$ and $t_2^s(j)$ must be separately scheduled on the same machine. As indicated, the processing time of job j_2 may depend on the scenario, but without exception this job cannot be scheduled for processing prior to the completion of j_1 plus the interruption time $w^s(j)$. When $f(j) = 0$, the processing time of job $j \in SJ$ under scenario s becomes $t_1(j) + w^s(j) + t_2{}^s(j)$, and the machine processing job j is considered to be busy during the entire period.

A recourse in this formulation can be defined as the insertion of job j_2 in the sequence for a given scenario s. Consider the following schedule representation:

$$\Pi_k^s = \{\pi_k^s(1), \ldots, \pi_k^s(n_k)\},$$

where $\pi_k^s(j) > 0$ is the position of job j in the sequence for machine k under scenario s, and n_k is the number of jobs assigned to machine k in the current solution. Note that the number of jobs assigned to machine k does not depend on the scenario, since the assignment is made during the first phase. If $SJ = \varnothing$ or $f(j) = 0 \ \forall \ j \in SJ$, then $\pi_k^s(j) = \pi_k(j) \forall s$.

A set of non-anticipativity constraints is necessary to enforce the decisions made during the first phase. In particular, suppose that jobs j_1 and j_2 (originated from a job $j \in SJ$ with $f(j) = 1$) have been assigned to machine k and must be scheduled under scenario s. Then, the following must be true for a schedule that does not anticipate the future:

$$\pi_k^s(j_2) > \pi_k^s(j_1) + 1 \qquad \text{if } \pi_k^s(j_1) < n_k - 1,$$

$$\pi_k^s(j_2) = n_k \qquad \text{if } \pi_k^s(j_1) = n_k - 1,$$

$$comp(j_2,s) \geq comp(j_1,s) + w^s(j) + t_2{}^s(j).$$

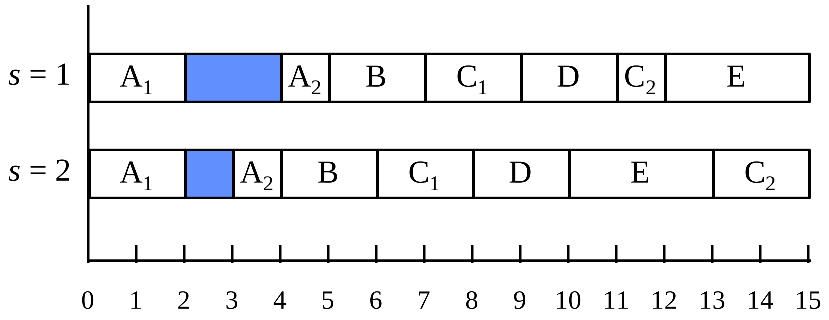
Also, the relative position of jobs $j \in DJ$ and jobs $j_1 \in SJ$ must be the same in each scenario. In other words, the only recourse is the insertion of a job j_2 after both its corresponding job j_1 and the interruption have finished. Finally, machine k should never be idle waiting for j_2 if $f(j) = 1$.

3.1 Illustrative Example

In order to illustrate the concept of non-anticipativity constraints and job-insertion as recourse, let us consider a facility with a single machine that must process the following jobs:

			$s = 1$		$s = 2$	
Job	t_1	*date*	t_2	w	t_2	w
A	2	5	1	2	1	1
B	2	7				
C	2	12	1	2	2	3
D	2	12				
E	3	15				

Then the following schedule is feasible if $f(\text{A}) = 0$ and $f(\text{C}) = 1$:



Note that all jobs are on time under scenario 1, while job C is three units of time tardy under scenario 2. Since the interruption $w^2(C)$ is three units of time long and the policy is to free the machine during the interruption, the machine must keep processing jobs as long as there are unprocessed jobs assigned to this machine. After processing job D during the interruption of C, the processing of job E is started at time 10 because the interruption has not yet finished and the decision maker has no way of knowing how much longer the interruption will last. Scheduling job E before C_2 causes the delay of job C. Note also that in this particular example, the best policy for A is $f(A) = 1$, since this job will be on time under both scenarios even if B is scheduled before A_2.

More complicated policies can be designed to introduce more flexibility into the decision making process. For example, the delay of job C in scenario 2 above could have been reduced if a policy of not scheduling more than one job during an interruption can be implemented. Then job D would have finished at time 10 and job C_2 would have started at time 11 (after an idle period of one time unit).

4. Scatter Search Procedure

We adapt the scatter search methodology (Glover, 1998) to the problem of parallel-machine scheduling of jobs with stochastic job interruptions. In this adaptation, we assume the following:

- A solution to the problem is fully characterized by the job partition. This means, that an auxiliary process determines the ordering of jobs according to the partition and selects an appropriate interruption policy.
- An initial set of reference solutions (or “population”) can be generated.
- A mechanism exists to create a new solution that is a “structured combination” of two or more reference (or “parent”) solutions.

In the following description we make use of random elements to induce search diversity and simplify the algorithmic design. However, systematic strategies to accomplish the same objectives may be part of a more refined version of the method.

4.1 Initial Set of Reference Points

The following procedure (referred to as *InitialSet*(x), where x is the desired size of the set) is used to generate and evaluate a set of solutions. The initial set of reference points **R** is the one generated by *InitialSet*($|\mathbf{R}|$).

1) Create a random permutation of all jobs in J.

2) Choose one job i at a time and assign it to the machine k with minimum load around $date(i)$, where the relative load of a machine processing jobs J_k is estimated as follows:

$$load(k) = \sum_{j \in J_k^{'}} Min\left\{\tilde{d}(j), date(j) - date(i) + \tilde{d}(i)\right\} + \sum_{j \in J_k^{''}} \left(date(i) - date(j) + \tilde{d}(j)\right)$$

where

$$J_k^{'} = \left\{ j \in J_k \middle| date(i) - \tilde{d}(i) \le date(j) \le date(i) \right\}$$
$$J_k^{''} = \left\{ j \in J_k \middle| date(i) < date(j) < date(i) + \tilde{d}(j) \right\}$$
$$\tilde{d}(j) = \begin{cases} d(j) & \text{if } j \in DJ \\ t_1(j) + \overline{w}(j) + \bar{t}_2(j) & \text{if } j \in SJ \end{cases}$$

and

$$\bar{t}_2(j) = \sum_s p_s t_2^{\,s}(j)$$
$$\overline{w}(j) = \sum_s p_s w^s(j)$$

3) Repeat (1) and (2) $|\mathbf{R}|$ times.

4) For each machine k, an ordering of the jobs $j \in DJ$ and jobs $j_1 \in SJ$ is found, an interruption policy f_k is determined and the insertion of jobs $j_2 \in SJ$ is performed.

   a) *Ordering* — The heuristic rules ATC, MDD and SLK (Vepsalainen and Morton, 1987) are applied twice using the following definitions for the processing time of jobs $j \in SJ$:

   $$d\ (j) = t_1(j)$$
   $$d\ (j) = t_1(j) + w^*(j) + t_2^*(j)$$

   where $w^*(j)$ and $t_2^*(j)$ are the most likely values for the interruption and the duration of the second part of a job $j \in SJ$. These definitions yield 6 potentially different orderings, to which the "interruption policy" and "insertion" procedures are applied.

   b) *Interruption Policy* — For each scenario s, the rule X-RM (Morton and Ramnath, 1995) is used to insert the jobs $j_2 \in SJ$. This rule is a dispatching heuristic for the single machine weighted tardiness problem that allows the insertion of idle times. The resulting sequence determines the "preferred" interruption policy under scenario s. Let us refer to this policy as $f_k^s(j)$. Since the policy may be different for different scenarios, an aggregate (unique) policy is found as follows:

$$f_k(j) = \operatorname{int}\left(\sum_s p_s f_k^s(j)\right)$$

c) *Insertion* — Once an initial ordering and an interruption policy has been determined, a modified X-RM rule is applied to insert jobs $j_2 \in SJ$. The modification to the rule is necessary in order to observe the interruption policies associated with each job.

d) *Evaluation* — The objective function value is evaluated for each of the different orderings associated with a machine (as determined in step 4(a)). The best solution for each machine is the one with the best objective function value (as determined after steps 4(b) and 4(c) are performed).

Note that 8 rules were initially considered for the ordering procedure in step 4(a). These rules are EDD, WEDD, SPT, WSPT, ATC, MDD, SLK and WSLK. After initial experimentation, it was determined that the quality of the solutions was not significantly affected when the set of rules was reduced to the three listed in step 4(a).

4.2 Structured Combinations

A structured combination is a mechanism by which new solutions are created from a subset of reference points. In the genetic algorithm jargon this is known as the crossover operation, and the subset of reference points (usually limited to two) is referred to as “parents”. Since the machines are considered identical, combining two or more solutions from the point of view of changing the job assignments is not particularly relevant. That is, swapping all the jobs assigned to machine 1 with those assigned to machine 2 has no effect on the objective function value. Therefore, we chose to define the structure of a solution as the *grouping* of jobs resulting from a given partition.

The structured combination in this context will be based on a voting scheme in which each chosen reference point contributes to the creation of a new point in a manner that is proportional to its weight. The weight of a reference point is a normalized measure of the corresponding solution quality.

4.2.1 Structured Job Assignment

An effective mechanism for creating structured combinations generates new points that inherit the structure embedded in the selected reference points. The reference points are selected according to their objective function value. Specifically, the probability of selecting a given point increases as its objective function value decreases (because our goal is to minimize). The mechanism that we are about to describe assigns jobs to machines based on identifying job groups within two reference points. The identification of these groups is considered the extraction of a “basic” structure embedded in the two reference points. This basic structure is preserved in any point generated from the given pair of reference points. Since the basic structure assigns only a subset of all jobs, the remaining jobs are assigned to machines following the deterministic process in step 2 of *InitialSet*(x).

The following procedure (referred to as *Structure*(x,y), where x and y are reference points) creates a structured combination of two reference points. Therefore, *Structure*(r_1,r_2) creates a new solution from the reference points r_1 and $r_2 \in \mathbf{R}$.

1) Solve the assignment problem:

$$\begin{aligned} Max \quad & \sum_k \sum_l c_{kl} x_{kl} \\ s.t. \quad & \sum_l x_{kl} = 1 \qquad \forall k \\ & x_{kl} = \{0,1\} \end{aligned}$$

   where $c_{kl} = \left| G_k(r_1) \cap G_l(r_2) \right|$, and $G_k(r)$ is the set of jobs assigned to machine k in reference point r. The solution to this problem extracts the basic structure of the reference points by matching each machine in r_1 with a corresponding machine in r_2.

2) For each $x^*_{kl} = 1$, assign jobs $G_k(r_1) \cap G_l(r_2)$ to machine k of the new solution, where x^* is the optimal solution of the assignment problem in step (1). Let all jobs assigned in this step be part of the set $A \subseteq J$.

3) Assign jobs $j \in J \setminus A$ to machines following step 2 of *InitialSet*(1).

4) Apply step 4 of *InitialSet*(1) to the job assignment obtained in the previous step.

4.2.2 Illustrative Example

To illustrate how the structured combination works, consider the job assignments in the following two reference points in a 3-machine 20-job problem:

	Machine		
Solution	*1*	*2*	*3*
r_1	1, 3, 6, 9, 12, 17, 18	2, 5, 8, 10, 11, 19, 20	4, 7, 13, 14, 15, 16
r_2	1, 4, 5, 10, 11, 20	2, 3, 6, 12, 13, 15	7, 8, 9, 14, 16, 17, 18, 19

The corresponding assignment problem is:

Maximize $\quad x_{11} + 3\,x_{12} + 3\,x_{13} + 4\,x_{21} + x_{22} + 2\,x_{23} + x_{31} + 2\,x_{32} + 3\,x_{33}$

subject to
$$\begin{aligned} x_{11} + x_{12} + x_{13} = 1 \\ x_{21} + x_{22} + x_{23} = 1 \\ x_{31} + x_{32} + x_{33} = 1 \end{aligned}$$

$$x_{ij} = \{0,1\} \qquad i, j = 1, 2, 3.$$

The optimal solution to this problem consists of $x_{12} = x_{21} = x_{33} = 1$, and therefore the basic structure of the solutions generated by these two reference points has the following job assignments:

Machine		
1	*2*	*3*
3, 6, 12	5, 10, 11, 20	7, 14, 16

The application of the first two steps of the procedure terminates with the assignment of jobs to machines that corresponds to the basic structure embedded in the two reference points. Note that these steps allow the identification of duplicated solutions in the population, that is, solutions for which the job assignments are the same. When this occurs, the procedure eliminates one of the duplicates and the current size of the set of reference points decreases by one. The application of step 3 completes the job assignments of the new solution. Finally, step 4 obtains a solution from the job assignment found with the structured combination procedure.

4.3 Tabu Search Elements

The scatter search method is designed to maintain a set of reference points. New reference points are added by applying the structured combination procedure described above. Tabu search (Glover and Laguna, 1997) rules are applied to avoid repeatedly choosing the same reference points to generate new ones. A short-term memory strategy is implemented to classify the selection of pairs of reference points as tabu. Specifically, after an iteration in which reference points r_1 and r_2 are used to construct a new solution, the selection of this particular pair is classified tabu for a number of iterations. The set of reference points changes as newly generated points are used to replace those members of **R** of inferior quality. A simple replacement strategy is employed. This strategy favors the newly created point as long as its objective function value is better than the worst solution currently in **R**. However, if the current size of **R** is less than a target value *RefSize*, then the new solution is always added to **R**.

A first-level diversification is achieved by the short-term memory function, since it forbids the selection of recently selected pairs. A second-level diversification consists of replacing each $r \in$ **R** as follows:

$$r \ = \mathit{InitialSet}(1)$$
$$r \ = \mathit{Structure}(r,r\ )$$
$$\mathbf{R} = \mathbf{R} \cup \{r\ \} \setminus \{r\}$$

In the actual implementation we check the objective function evaluation of r and r to determine whether the best solution found so far must be updated. If the size of the resulting **R** is less than a target value *RefSize*, then the procedure *InitalSet*(*RefSize* - |**R**|) is performed and the solutions are added to **R**.

4.4 Summary

We now summarize our implementation of scatter search. The procedure starts with the generation of an initial set of reference points **R**. The desired size of the set is *RefSize*, so **R** = *InitialSet*(*RefSize*). Then, the procedure sequentially applies an intensification phase followed by a diversification phase. The intensification phase is based on the short-term memory function described above. The diversification phase is based on the replacement procedure also described above. The procedure runs until *MainIter* iterations have been executed without improving the current best solution.

The intensification phase performs iterations in which two reference points r_1 and r_2 are chosen and *Structure*(r_1,r_2) is performed to find a new solution r. The new solution is added to **R** if it improves upon the worst solution currently in **R** (or if the current size of **R** is less than a target value). The pair (r_1,r_2) is recorded in a short term memory function for *TabuTenure* intensification steps. The best solution found is updated as appropriate as well as the number of intensification steps without improving. If *IntStep* intensification steps are performed without improving the current best solution, the procedure abandons the intensification phase and starts a diversification phase.

The diversification phase replaces the current population **R** as described in section 4.3. Therefore, this phase consists of |**R**| steps (and the possible application of *InitialSet*(*RefSize*-|**R**|) to make up for duplicated solutions).

5. Computational Experiments

In this section, we present the computational experiments performed with the goal of assessing the merit of the proposed solution method. However, before discussing our results, we first describe the procedure used to generate problem instances.

5.1 Problem Generation

The problem instances used to test our procedure were randomly generated as follows:

- Activity times and tardiness penalties are drawn from uniform distributions with parameters (5, 10) and (5, 20), respectively.
- The due date for job j is drawn from a uniform distribution with parameters $\left[t_1(j)+\max_s\{w^s(j)\}+\max_s\{t_2^s(j)\}, MaxDate\right]$, where

$$MaxDate = \frac{\sum_j \left(t_1(j) + \frac{\sum_s t_2^s(j)}{|S|} \right)}{\rho m}$$

- The generator uses the following parameters:

n	=	number of jobs
m	=	number of machines
$\%Stoch$	=	percentage of jobs with stochastic interruptions
ρ	=	expected machine utilization

We assume that the uncertainty in this problem is captured by a set of scenarios. These scenarios are generated by specifying three time estimates both for the interruption and for the second part of each stochastic job (i.e., for each $j \in SJ$). The estimates consist of the minimum time, the most likely time and the maximum time. The total number of scenarios is therefore equal to $9^*|SJ|$. (As noted in the Computational Results section, we use a sampling procedure to select scenarios to evaluate solutions during the search.)

5.2 Computational Results

Our computational experiments were designed with the main goal of comparing the best solution found by the proposed procedure with the best solution in the initial set of reference points. A secondary goal of our experiments is to analyze the robustness of the solutions obtained as the number of scenarios under consideration increases.

Tables 1, 2 and 3 summarize the results of our first set of experiments. The first two columns show the size of the instances in terms of the number of machines (i.e., *m*) and the number of jobs (i.e., *n*). The third column, labeled *%Stoch*, indicates the percentage of stochastic jobs in each instance. The forth column, *EWT*, consists of the expected weighted tardiness of the best solution found by the procedure. The fifth column, EWT_{NORM}, shows a normalized value of *EWT*. This value is obtained by dividing *EWT* by the number of jobs and the average penalty value (i.e., 12.5). The normalized value gives an indication of the average tardiness per job. The *Reduction* value in the sixth column is calculated as follows:

$$Reduction = \frac{EWT(Initial) - EWT}{EWT(Initial)} * 100 ,$$

where *EWT*(*Initial*) is the objective function value of the best solution in the initial set of reference points. The number of solutions generated during the process is reported in the seventh column, labeled *Solutions*. Finally, the average CPU seconds is reported in the eighth column, labeled *Time*. This time excludes input and output operations and corresponds to a personal computer with a Pentium 166 processor. All of these columns report average values over the 5 instances corresponding to each row. The following search parameters were used in all of the experiments:

- *RefSize* = 10
- *MainIter* = 5
- *IntStep* = 15
- *TabuTenure* = 7

Also, the results in Tables 1, 2 and 3 were found using a sample size of 30 scenarios. Considering all the combinations of values, the test set consists of 81 problem types. Since we generate 5 instances for each problem type, there are 405 instances in the entire set

— TABLE 1 —

The results reported in Table 1 show the effectiveness of our procedure, as is evident from the improvement over the best solution in the initial set of reference points. The average reduction on the expected weighted tardiness value is 94.22%. Furthermore, the small EWT_{NORM} values indicate that the solutions are either optimal or near optimal (where optimal solutions have *EWT* values equal to zero). In order to obtain these results, our procedure generates a relatively small number of solutions, especially when compared to the number of solutions typically generated by other population-based procedures. This allows the procedure to find high quality solutions within a reasonable amount of time, considering the difficulty of the problems and the realistic sizes of the instances tackled. We should also mention that we tested our procedure with $\rho = 0.6$ and found that these problem instances are easily solved by the dispatching rules used in the generation of the initial set of reference points.

— TABLES 2 and 3—

The results in Table 2 and 3 lead us to conclusions that are similar to those drawn from Table 1. However, as expected, these problems are more difficult, due to the increase in the expected machine utilization. That is, as the expected machine utilization increases, it is more difficult to find solutions with an expected weighted tardiness value of zero. The difficulty of these instances results in an increase on the values of *EWT*, EWT_{NORM}, *Solutions* and *Time*, and a decrease on the value of *Reduction*. The reduction, however, is still substantial, warranting the use of the proposed procedure.

In an attempt to improve upon the solutions reported in Tables 1, 2 and 3, we resolved all the problems increasing the value of *RefSize* to 20 solutions. We found that the results were only marginally better than those obtained with *RefSize* = 10, and the computational time increased significantly.

Since the results reported in Tables 1, 2 and 3 were obtained with a sample size of 30 scenarios, we performed a second experiment to assess the robustness of the method in relation to the number of scenarios used during the search. In this experiment, we employ the problem instances of size m =10 and increase the sample size to 50 and 70 scenarios. Tables 4, 5 and 6 show the results of this experiment for expected machine utilization equal to 0.7, 0.8 y 0.9, respectively.

— TABLES 4, 5 and 6 —

The *30 Scenarios* columns simply report the EWT_{NORM} values from Tables 1, 2 and 3, but averaging over the rows %*Stoch* = 25, 50 and 75. Under the *50 Scenarios* columns, we report the EWT_{NORM} values obtained using the 30-Scenario solution and evaluating it on a sample of size 50. That is, we take the solution found with 30 scenarios and we calculate EWT_{NORM} on set of 50 scenarios that includes the original 30. Note that the quality of the solutions found using 30 scenarios always decreases when evaluated over 50 scenarios. We then repeat the process for 70 scenarios. In this case, many of the solutions found using 50 scenarios turn out to be of a better quality than those found using 70 scenarios.

6. Conclusions

We have developed a procedure to approximately solve a generalization of the well-known weighted tardiness problem on parallel machines. The generalization consists of introducing a set of jobs with stochastic interruptions. This generalization allows the modeling of practical situations, where jobs cannot be completed in one installment, and the length of the interruption is not known with certainty.

Our procedure is an instance of scatter search. Scatter search is a population-based methodology that shares features with the so-called evolutionary methods. Our implementation balances search intensification and search diversification, while maintaining a set of solutions referred to as reference points. Our design includes the following innovative features:

- A generator of an initial set of reference points that is not completely random (as is typically the case in genetic algorithms). The generator exploits features that are specific to the problem instance being tackled and employs well-known dispatching rules to obtain high-quality solutions.

- A procedure to create structured combinations of solutions that solves an assignment problem to extract the structure of the reference points being combined.

- A diversification generator that combines solutions of high quality with newly generated solutions to induce diversity in the current set of reference points.

The results of our computational testing, with a total of 405 problem instances, indicate that the procedure is an effective solution method to the challenging stochastic scheduling problem addressed in our study.

7. References

Alidaee, B. and D. Rosa (1997) "Scheduling Parallel Machines to Minimize Total Weighted and Unweighted Tardiness," *Computers and Operations Research*, vol. 24, no. 8, pp. 775-788.

Glover, F. (1998) "A Template for Scatter Search and Path Relinking," in *Artificial Evolution: Lecture Notes in Computer Science* 1363, J. K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.), Springer, pp. 13-54.

Glover, F. and M. Laguna (1997) *Tabu Search*, Kluwer Academic Publishers, Boston.

Kouvelis, P. and G. Yu (1997) *Robust Discrete Optimization and Its Applications*, Kluwer Academic Publishers, Boston.

Morton, T. E. and P. Ramnath (1995) "Guided Forward Search in Tardiness Scheduling of Large One Machine Problems," In: *Intelligent Scheduling Systems*, D.E.Brown and W.T.Scherer (Eds.), Kluwer Academic Publishers.

Mulvey, J. M., R. J. Vanderbei, and S. A. Zenios (1995) "Robust Optimization for Large Scale Systems," Operations Research, vol. 43, no. 2, pp. 264-281.

Valls, V., M. Laguna, P. Lino, A. Pérez and S. Quintanilla (1998) "Project Scheduling with Stochastic Activity Interruptions," in *Recent Advances in Project Scheduling*, Jan Weglarz (Ed.), Kluwer Academic Publishers.

Vepsalainen, A. P. J. and T. E. Morton (1987) "Priority Rules for Job Shops with Weighted Tardiness Costs," *Management Science*, vol. 33, no. 8, pp. 1035-1047.

Table 1. Results for ρ = 0.7

m	n	%Stoch	EWT	EWT_{NORM}	Reduction	Solutions	Time
5	100	25	7.13	0.006	90.70%	10.00	373.40
		50	1.97	0.002	95.49%	10.00	360.91
		75	3.71	0.003	86.12%	10.00	396.72
	200	25	50.56	0.020	83.56%	645.80	1210.04
		50	1.81	0.001	97.15%	578.40	1367.49
		75	24.41	0.010	85.47%	411.20	1124.94
	300	25	0.00	0.000	100.00%	55.80	171.76
		50	0.19	0.000	99.62%	596.00	2793.65
		75	18.89	0.005	51.76%	362.60	2205.82
10	100	25	0.00	0.000	100.00%	282.20	117.95
		50	19.59	0.016	93.20%	284.60	158.08
		75	1.25	0.001	99.37%	410.20	278.37
	200	25	0.91	0.000	99.63%	492.40	606.64
		50	0.13	0.000	99.65%	187.00	283.16
		75	0.20	0.000	99.64%	351.60	657.44
	300	25	0.00	0.000	100.00%	48.20	88.73
		50	0.00	0.000	100.00%	41.20	88.87
		75	0.00	0.000	100.00%	30.80	68.04
15	100	25	32.42	0.026	85.58%	373.40	151.31
		50	22.08	0.018	88.91%	1102.00	593.25
		75	1.66	0.001	98.08%	391.40	254.28
	200	25	2.03	0.001	96.69%	343.20	369.04
		50	1.20	0.000	99.33%	330.80	456.09
		75	0.65	0.000	98.86%	178.80	287.92
	300	25	3.44	0.001	95.89%	411.00	820.09
		50	0.87	0.000	99.31%	773.80	1934.07
		75	0.00	0.000	100.00%	107.80	291.55
		Average	7.23	0.004	94.22%	326.67	648.50
		Minimum	0.00	0.000	51.76%	10.00	68.04
		Maximum	50.56	0.026	100.00%	1102.00	2793.65

Table 2. Results for ρ = 0.8

m	n	%Stoch	EWT	EWT_{NORM}	Reduction	Solutions	Time
5	100	25	8.77	0.007	95.41%	1104.40	633.35
		50	40.33	0.032	83.82%	1034.20	754.18
		75	526.21	0.421	80.08%	1254.00	1116.47
	200	25	149.49	0.060	87.79%	762.80	1431.11
		50	66.28	0.027	90.28%	715.40	1681.16
		75	0.48	0.000	98.55%	516.60	1422.87
	300	25	18.27	0.005	79.12%	833.60	3143.54
		50	4.03	0.001	99.30%	682.40	3151.68
		75	70.13	0.019	83.80%	692.40	3808.76
10	100	25	38.82	0.031	89.83%	569.80	251.57
		50	4.93	0.004	92.91%	698.40	402.56
		75	29.89	0.024	87.64%	984.20	697.28
	200	25	1.66	0.001	99.37%	175.20	213.76
		50	0.12	0.000	99.92%	402.80	638.25
		75	79.10	0.032	86.26%	555.40	1058.82
	300	25	96.28	0.026	89.16%	267.80	624.39
		50	44.77	0.012	89.01%	659.00	1981.13
		75	3.90	0.001	98.09%	344.00	1223.27
15	100	25	102.84	0.082	77.20%	865.20	360.32
		50	27.79	0.022	80.62%	748.80	408.22
		75	162.31	0.130	78.05%	1099.80	720.67
	200	25	139.19	0.056	81.16%	1043.40	1124.43
		50	29.15	0.012	87.80%	607.00	837.43
		75	16.79	0.007	90.08%	595.40	990.17
	300	25	25.17	0.007	69.05%	480.80	971.65
		50	23.15	0.006	89.55%	703.40	1790.16
		75	0.12	0.000	99.52%	221.20	632.09
		Average	63.33	0.038	88.27%	689.53	1187.75
		Minimum	0.12	0.000	69.05%	175.20	213.76
		Maximum	526.21	0.421	99.92%	1254.00	3808.76

Table 3 Results for ρ = 0.9

m	n	%Stoch	EWT	EWT_{NORM}	Reduction	Solutions	Time
5	100	25	18.75	0.015	96.82%	1107.80	618.85
		50	325.03	0.260	69.82%	1497.20	1096.11
		75	2190.54	1.752	45.92%	2124.20	1876.08
	200	25	178.85	0.072	84.70%	839.80	1509.56
		50	2337.69	0.935	61.58%	661.80	1565.03
		75	577.03	0.231	83.36%	977.00	2684.96
	300	25	2311.17	0.616	70.69%	916.80	3462.23
		50	73.19	0.020	91.53%	745.20	3409.85
		75	2876.06	0.767	47.48%	815.80	4485.19
10	100	25	274.93	0.220	63.83%	1109.00	487.60
		50	880.06	0.704	39.21%	742.80	438.38
		75	795.53	0.636	51.33%	1300.00	933.20
	200	25	1028.36	0.411	56.27%	1088.20	1371.54
		50	591.35	0.237	52.22%	777.60	1265.09
		75	319.64	0.128	75.58%	827.00	1592.98
	300	25	781.13	0.208	60.30%	886.40	2157.70
		50	139.47	0.037	74.94%	866.40	2694.37
		75	307.09	0.082	85.63%	1368.80	4978.02
15	100	25	904.89	0.724	25.44%	1010.40	422.21
		50	723.67	0.579	39.96%	1031.00	558.52
		75	718.15	0.575	41.31%	770.60	515.90
	200	25	823.13	0.329	49.82%	846.00	918.08
		50	1080.64	0.432	36.94%	712.20	977.31
		75	763.45	0.305	53.12%	1017.00	1718.47
	300	25	411.65	0.110	66.92%	739.80	1536.38
		50	1045.13	0.279	53.36%	869.60	2207.94
		75	1016.13	0.271	78.54%	930.20	2822.27
		Average	870.10	0.405	61.36%	984.39	1789.03
		Minimum	18.75	0.015	25.44%	661.80	422.21
		Maximum	2876.06	1.752	96.82%	2124.20	4978.02

Table 4. Robustness assessment for ρ = 0.7

	30 Scenarios	50 Scenarios		70 Scenarios		
			30-Scenario Solution		30-Scenario Solution	50-Scenario Solution
n	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}
100	0,0056	0,0043	0,0052	0,0028	0,0051	0,0043
200	0,0002	0,0001	0,0003	0,0001	0,0003	0,0001
300	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Average	0,0019	0,0015	0,0018	0,0010	0,0018	0,0015

Table 5. Robustness assessment for ρ = 0.8

	30 Scenarios	50 Scenarios		70 Scenarios		
			30-Scenario Solution		30-Scenario Solution	50-Scenario Solution
n	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}
100	0,0196	0,0183	0,0215	0,0291	0,0221	0,0184
200	0,0108	0,0102	0,0111	0,0145	0,0113	0,0106
300	0,0129	0,0095	0,0125	0,0136	0,0126	0,0096
Average	0,0144	0,0127	0,0150	0,0191	0,0153	0,0129

Table 6. Robustness assessment for ρ = 0.9

	30 Scenarios	50 Scenarios		70 Scenarios		
			30-Scenario Solution		30-Scenario Solution	50-Scenario Solution
n	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}	EWT_{NORM}
100	0,5201	0,4812	0,5162	0,4810	0,5146	0,4823
200	0,2586	0,2573	0,2636	0,2631	0,2613	0,2562
300	0,1091	0,1052	0,1089	0,1069	0,1086	0,1048
Average	0,2959	0,2812	0,2962	0,2837	0,2948	0,2811