Transaction on Evolutionary Algorithm and Scheduling Problem
ISSN: 2229-8711 Online Publication, June 2012
www.pcoglobal.com/gjto.htm
ET-S26/GJTO

GJO Global Journal of Technology & Optimization Volume 3, 2012

MANIPULATION OF TABU LIST TO HANDLE MACHINE BREAKDOWNS IN JOB SHOP SCHEDULING PROBLEMS

Erna Budhiarti Nababan[1], Opim Salim Sitompul[2], Salwani Abdullah[3]

[1]Department of Information Technology, Universitas Sumatera Utara, Indonesia
[2]Department of Mathematics, Universitas Sumatera Utara, Indonesia
[3]Department of Computer Science, Universiti Kebangsaan Malaysia
Email: ernabrn@usu.ac.id, opim@usu.ac.id, salwani@ftsm.ukm.my

Received September 2011, Revised November 2011, Accepted December 2011

Abstract
Machine breakdowns in a production schedule may occur on a random basis that make hard combinatorial problem of Job Shop Scheduling Problems (JSSP) becomes more complex. In this paper a new algorithm Fuzzy Tabu Priority List (FTPL) is proposed. Tabu search technique is applied to search optimal solution whereas FTPL is used to handle machine breakdowns. There are two tabu lists employed: one to keep moves during searching for optimal solution, another one is to keep broken machine if breakdown occurs. Period of how long the machine will be kept on the list is determined by fuzzy membership function. In order to avoid solution of being trapped into a local optimum Monte Carlo acceptance criterion is applied. Our techniques are tested to the benchmark data of JSSP available on the Operation Research library. From the experiment, we found that our algorithm is promising to help a decision maker to face the event of machine breakdowns.

Keywords: job shop scheduling problem, machine breakdowns, Fuzzy Tabu Priority List, Monte Carlo acceptance criterion

1. Introduction
Most manufacturing systems that operate in dynamic environments are subject to various disturbances [2, 4, 12, 14, 31]. Unexpected disruptions, such as machine breakdowns, material shortage, job arrivals or cancellations may occur during manufacturing processes which would then undergo delays of the various jobs at hand and would result in a deviation from the initial solution [9]. The disruption will produce uncertainty in the sequence of operation, i.e. the time taken to repair the broken machine. Moreover accumulation of these delays will significantly disturb the smooth progress of the whole schedule and may result in low productivity of the manufacturing system [7, 8]. Therefore, it is necessary for the original schedule to maintain its performance while reacting to production disruption in a timely manner, so that when disruptions occur the necessary changes in the schedule are minimal [4, 9]. Tabu Search is one of the most effective methods to find near-optimal solutions of combinatorial optimization for generating high quality solution for Job Shop Problem (JSP) [1, 33, 34]. It uses memory function that record the recent history of the search to prevent cycling back to previously visited solutions [5, 6]. The memory function helps the search in two ways: (i) avoid being trapped at local optima (ii) avoid being consumed excessive computation time. In this paper we will discuss the manipulation of the memory function on Tabu Search technique, i.e. tabu list, to handle machine-breakdowns. Instead of keeping move, tabu list will be used to keep broken machine for a period of its repair time. In order to determine the time limit for the elements remaining on memory, we apply fuzzy membership function. Block diagram of the overall process of scheduling and rescheduling is depicted in Figure 1.

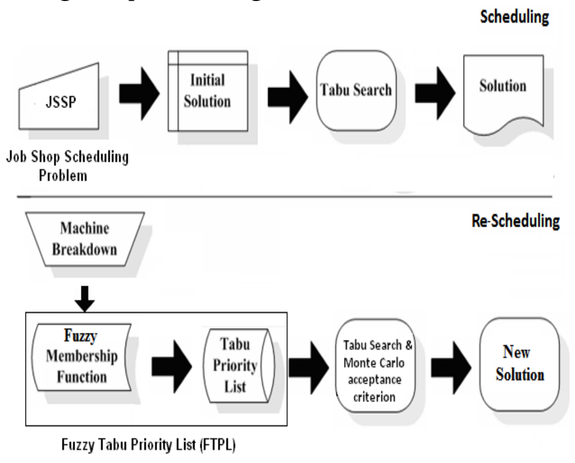

Scheduling
JSSP
Initial Solution
Tabu Search
Solution
Job Shop Scheduling Problem
Re-Scheduling
Machine Breakdown
Fuzzy Membership Function
Tabu Priority List
Tabu Search & Monte Carlo acceptance criterion
New Solution
Fuzzy Tabu Priority List (FTPL)

Figure 1: Block diagram of scheduling and rescheduling process

We divide our algorithm into two stages: scheduling stage, where there is no disruption, and rescheduling stage, where machine breakdowns occur.

2. Related works
Artificial Intelligence approaches encompass a rich collection of knowledge representation formalism for dealing with a wide variety of real-world problems. It provides richer and more flexible representations of real-world supporting efficient constraint-based reasoning mechanism as well as mixed initiative frameworks, which allow the human expertise to be in the loop.

Beside its ability to describe real-world problems at the same time artificial intelligence guarantees good and fast solutions [14, 25]. There are several capabilities of artificial intelligence that make this technology particularly suitable for scheduling. The Expert System approach in scheduling [6, 7, 26, 29] was to construct the schedule by interviewing one or more experts to acquire the rules which regulate their decision process [6]. The ability of expert systems could also be used to explain the reasoning process through back-traces and to handle levels of confidence and uncertainty in the area of production scheduling. In addition to constructing a schedule, expert systems could also be applied for rescheduling of job shops in which expert systems rule is used to set due-dates for newly arriving jobs [14]. However expert systems do not always succeed in generating competitive operational schedules [12]. Even the use of human expert knowledge (e.g. production rules) may lead to poor results in the face of an increasing problem size [11].The Neural Network approach for JSSP has been one of the actual approaches used in the production scheduling research since its parallel implementation is natural. Hopfield neural network was applied by [32] to solve JSSP. The JSSP is mapped to the corresponding neural network and a feasible solution for scheduling is related to the minimum of computational energy function. However, because large numbers of local extremes exist in the energy function, the stable state of a neural network frequently is local optima. They applied simulated annealing to avoid Hopfield neural networks converging to a local minimum. Simulated annealing makes neural networks converge to the minimum 0 of computational energy function and keeps the steady outputs of neural networks as a feasible solution for scheduling. [14] formulated an n job and m machines flow shop to model the fuzzy job due dates. A nonlinear membership function is used to represent the grade of satisfaction with the completion time of a job. A scheduling objective is to minimize completion time. They applied simulated annealing to get the initial solution and Tabu Search to obtain the feasible solution. From the experiment done they found that their algorithm give positive satisfaction grades for many test problems. Fuzzy dynamic scheduling algorithm (FDSA) for job shop scheduling problems was developed by [24], where fuzzy logic is used to combine conventional job shop scheduling rules to form aggregate heuristic rules. In their simulation experiment 20 jobs involving up to 15 machines were conducted. Their algorithm was compared to conventional priority rules, such as Short Processing Time (SPT), Early Due Date (EDD), First Come First Serve (FCFS) under FDSA for the performance measure of maximum and mean flow time, maximum and mean job lateness and the number of tardy jobs. Result shows that their algorithm performed well. [16] applied fuzzy technique for treating fuzzy information inherent in the problem and branch and bound method was used to obtain the optimal solutions for the given performance criteria. [18] applied linguistic value of fuzzy method for multi objective scheduling, which considered total flow time, maximum tardiness and number of tardy jobs, then applied Tabu Search to find the feasible solution. Fuzzy logic decision making algorithm is proposed to determine priorities of part types that are to be processed on a machine prior to scheduling. A new methodology for developing scheduling systems is proposed, which addresses two important issues: treating various types of uncertainty that exist in scheduling problems using fuzzy-logic based techniques, and considering multiple criteria which describe various performance measures of schedules. New fuzzy multi-criteria rescheduling methods proposed by [20]. It was used as response to various disruptions that can occur in a manufacturing environment. Processing time and due dates of the jobs are modeled by fuzzy sets and fuzzy If-Then rules were used to derive conclusions based on imprecise premises. Then Genetic Algorithm (GA) was used to search and measure the quality of schedules. A meta-heuristic, defined as:"…an iterative generation process which guides a subordinate heuristic." Whereas [11] defined it as:"… a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality". It can be applied to a large combinatorial problem [26], since it uses a high-level strategy that guides other heuristics in a search for feasible solutions. A popular technique that belongs to this family is Tabu Search (founder: Glover in 1986), which is a meta-heuristic method for guided local search. It is a local search technique but it incorporates a mechanism to help search escape from local optima [11, 17]. The basic idea of Tabu Search is the exploitation of the search space of all feasible scheduling solutions by a sequence of moves. A move from one schedule to another schedule is made by evaluating all candidates and choosing the best available. Each time a move is made, it is placed on a list called a tabu list. The tabu list keeps track of the most recently visited solutions and forbids move toward them. However, when considering a move, the move that is placed on the tabu list could not be chosen anymore, or tabu. Old moves are typically removed from the tabu list after some number of iterations. Tabu Search with simulation is applied to schedule production process through a set of machines [8]. Performance of the procedure was evaluated by comparison of the optimal solution for small problem instances, and to a good heuristic for larger problem. Combining Tabu Search with simulation provides a more accurate estimation of on-time performance and more realistic constraints on system operation. Compared to simple dispatching rules this method produces an improved schedule, however optimal solution were only found on their testing with small problems.

3. Tabu list

The tabu list, one of elements on tabu search technique, is used to keep the moves that are not allowed to be revisited in order to avoid returning back to the solutions obtained recently. It is not simply prohibit certain move reversals but compel such crossing and returns, offer an effective way to avoid the suboptimal entrapment of standard searches [11]. It keeps track of the solution attributes that have changed during the recent history. The elements stored in the tabu list are the attributes of moves which are refreshed each time a new basic processing order is found: the oldest elements are deleted and new ones added. The move maintained in the tabu list conforms to queue procedure whereby every time the reverse move is inserted at the end (tail) of the queue, all other entries that are currently in the queue are shifted forward one position. Afterward, the entry at the front (head) is removed from the queue. Then the tabu list is updated after each move to avoid cycling. Old moves are typically removed from the tabu list after some number of iterations. The tabu list is implemented as a queue in the first-in-first-out (FIFO) form, so the most recently inserted move into the list will be positioned in the tail of the list. Tabu list size is a tool used in guiding the search in the short time, given the determination of an effective set of attributes for defining tabu status. Other than that, the tabu list length determines the time limit for the elements remaining in memory. If the maximum length of the tabu list has been reached, the insertion of a new move will update the tabu list. Our algorithm of process inserting a new move into tabu list is described in the following steps:

```
Step 1: Create a new list.
        Set the information part to the move to be inserted.
        Set previous and next pointer to NULL
Step 2: If the tabu list is empty, set the head of the tabu
        list points to the new list. Goto Step 4.
Step 3: Set the next pointer of the new list points to the head of
        tabu list
        Set the previous pointer of the tabu list points to the new
        list
        Set the head of tabu list points to the new list
Step 4: Return the tabu list.
```

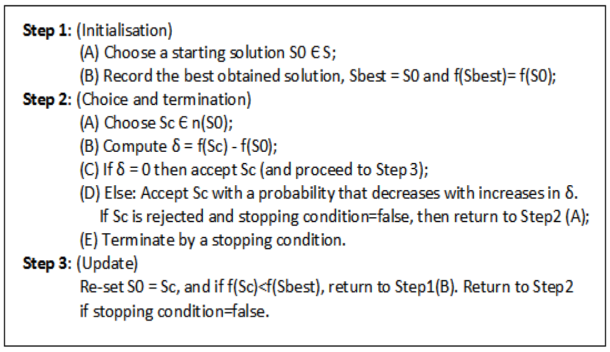
Whereas the algorithm of updating the tabu list is illustrated as follows:

```
Step 1: Find the list node of the tabu list.
        Set the next and the previous pointer to NULL and
        delate the last node.
Step 2: Create a new list.
        Set the information part to the move to be inserted.
        Set previous and next pointer to NULL.
Step 3: Set the next pointer of the new list point to the head of
        the tabu list.
        Set the previous pointer of the tabu listpoints to the new
        list.
        Set the head of tabu list points to the new list.
Step 4: Return the tabu list.
```

Tabu list size is a tool used in guiding the search in the short time, given the determination of an effective set of attributes for defining tabu status. Other than that the Tabu list length determines the time limit for the elements remaining on memory. If the maximum length of the tabu list has been reached, the insertion of a new move will update the Tabu list. Since the notion of the tabu list is a queue, then the insertion of the new node into the head of the list will cause the deletion of the oldest move from the list. If the length of list is too short cycling cannot be avoided, in the contrary a too long size creates too many restrictions and it has been observed that the mean value of the visited solution grows with the increase of the tabu list size.

Monte Carlo Acceptance Criterion

In order to avoid solution of being trapped into a local optimum, we apply Monte Carlo Acceptance Criterion δ, which was developed from basic Monte Carlo (MC) method [3]. The algorithm of MC method is depicted in Figure 2.



```
Step 1: (Initialisation)
        (A) Choose a starting solution S0 ∈ S;
        (B) Record the best obtained solution, Sbest = S0 and f(Sbest)= f(S0);
Step 2: (Choice and termination)
        (A) Choose Sc ∈ n(S0);
        (B) Compute δ = f(Sc) - f(S0);
        (C) If δ = 0 then accept Sc (and proceed to Step 3);
        (D) Else: Accept Sc with a probability that decreases with increases in δ.
            If Sc is rejected and stopping condition=false, then return to Step2 (A);
        (E) Terminate by a stopping condition.
Step 3: (Update)
        Re-set S0 = Sc, and if f(Sc)<f(Sbest), return to Step1(B). Return to Step2
        if stopping condition=false.
```

Figure 2.A basic Monte Carlo algorithm

In this algorithm, *S* is defined as a solution space, *f* as an objective function and *n* is a neighborhood structure. The Monte Carlo acceptance criterion, δ, always accepts an improved solution. Where as worse solutions will be accepted with a certain probability, which decreases as the solutions worsen, in order to escape local minima. Our TS algorithm to obtain feasible solution of $J_m||C_{max}$ is shown in the following steps:

```
Step 1: Generate an Initial Solution
        Store it as the current seed and the best solution
        Set tabu list to NULL, tabu list length to 0 and Iteration
        to 0
Step 2: If termination criterion is met, go to Step 9
Step 3: Generate Critical Path from the current seed
Step 4: Create critical blocks from the critical path
Step 5: If critical block is empty, go to Step 9
Step6: Generate neighbors of the current seed solution by
        neighborhood structure
6.1 Create neighbors from first block
6.2 Create neighbors from intermediate block
6.3 Create neighbors from last block
6.4 Save into the list of neighbors
Step 7: Repeat steps 7.1 – Step 7.2 until all neighbors in the
        neighbor list have
Be unvisited
7.1 Select a neighbor generated in Step 6, which is not tabu or
            satisfied a given aspiration criterion
        7.2 Calculate the current solution from the new neighbor.
            If the current solution is better than the best solution
                then store the Current Solution as the New
                    Solution and save the Best Solution from the
                    New Solution
              Else
                Calculate Monte Carlo acceptance criteriom, $\delta$, as
                    the difference between best solution and
                    current solution,
                    $\delta$ = New solution – Current solution
                    Generate a Random Number, RandNum, in the
                    range [0, 1]
                    If (RandNum< $e^{-\delta}$ )
                    Save the current solution as a New Solution
Step 8: Check tabu move
          8.1 If not tabu then go to step 8.2 else go to Step 8.3
          8.2 If tabu list length is less than maximum tabu list
              length,
              then insert move at the top of tabu list, otherwise
              update tabu list
          8.3 Increment Inter by 1 then go to Step 2.
Step 9: Output the best solution
```

The initial solution obtained from the Branch and Bound method is used as the current seed and as the best solution to be used by the tabu search method. The tabu search method iteratively modify the solution until the termination criteria is met. In this approach, the termination criteria are met whenever one of the following is found: the iterations have reached a maximum number given, the algorithm cannot create another critical block from the critical paths obtain so far (the critical block is empty), or the iterations have not obtained better solution from the previous solution. The tabu search approach uses the N6 neighborhood structure in order to generate the neighbors, which create the neighbors from the first, intermediate, and the last blocks of the critical blocks. The algorithm is then examined each of the generated neighbors by selecting neighbor that is not tabu or neighbor that is tabu but satisfy the given aspiration criteria. A current solution is then calculated from the new neighbor obtained and if the current solution is better (has a lower value) than the current best solution then the current solution is set as a new solution and save it as the best solution.

Otherwise, the algorithm is then calculates the difference (δ) between the best solution and the current solution, and generates a random number in the range of [0, 1]. If the random number generated is less than $e^{-\delta}$ then the current solution is saved as a new solution even if it is not better than the current solution. However, if the random number generated is not less than the Exponential Monte Carlo number, the algorithm does nothing and goes to Step 7 for another neighbor. If all the neighbors have been examined, the algorithm continues with the checking of the tabu move. If the move is not tabu and the length of the tabu list is less than the maximum length, then the move is inserted into the tabu list, but if the maximum length of the tabu list has been reached, then the tabu list is updated. In case if the move is tabu, the move is ignored and the iteration number is incremented by one. The iteration is restarted to Step 2.

4. Machine Breakdowns

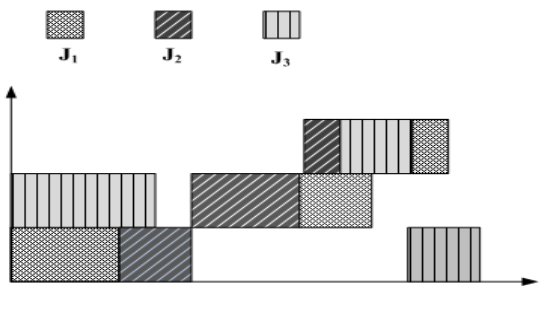
The triggering event such as machine breakdowns, occur randomly and we don't know in advance when the machine breakdowns will occur and which of the machines will be broken. During this time, the machine may be incapable of producing more parts because of blocking phenomena. In order to cope with the dynamic nature of a real world environment, we generate machine breakdown randomly with uniform random distribution and downtime is considered as deterministic. Some other parameters considered are:

i. time t at which the machine breakdown occurs
ii. machine k that is broken
iii. the length of time that the machine is unavailable or the time taken to repair the machine R_t
iv. the job's processing time remains the same before and after the breakdown.

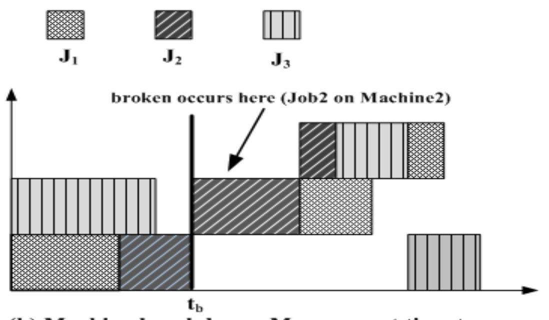
When a failure occurs a repair time is generated and the machine is kept unavailable during this time period. At the point of breakdown, an idle time equivalent to the time of breakdown is thereby inserted into the schedule and the operation on the machine is put on hold until the machine repair time is completed. The schedule is then subsequently repaired either by total rescheduling or partial rescheduling. After the machine is repaired, the unfinished operation usually has priority to be processed first. According to the information taken from the expert, the size of disruption for large problems, i.e when broken machines allowed during the production process is 8-10% of the total machines use. In this research, the repair time R_t of each machine is generated randomly between 8% to 10% of make span unit time [23]. At the point of breakdown we insert an idle time, which is equivalent to machine repair time, into the schedule. The operation on the machine is put on hold until the machine repair is completed. As was informed by the expert during interview, in manufacturing with large numbers of machines (*m* >10), it may happen that we have more than one machine broken down at the same time. The location of breakdowns and the repair time of each machine may vary since it occurs randomly.

5. Repair Mechanism

Schedule repair is a procedure for modifying the original predictive schedule to accommodate sudden temporal changes in the job shop [21, 22, 27, 30]. The repair mechanism is shown in Figure 3. Figure 3(a) illustrates the normal schedule and Figure 3(b) depicts the occurrence of machine M_2 breakdown at time t_b.
The goal of schedule recovery is to avoid wastage of time and resource in capturing the status of the job shop for rescheduling whenever a minor disruption occurs. Job shop disruptions are unintentional deviations from the planned operations of the schedule. To repair a variety of disruptions, the repair of any complex disruption can be realized through simple repair steps [23] which are identified as in the following point form:

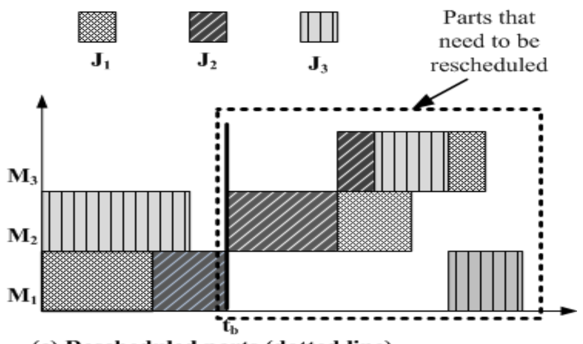


(a) Normal schedule

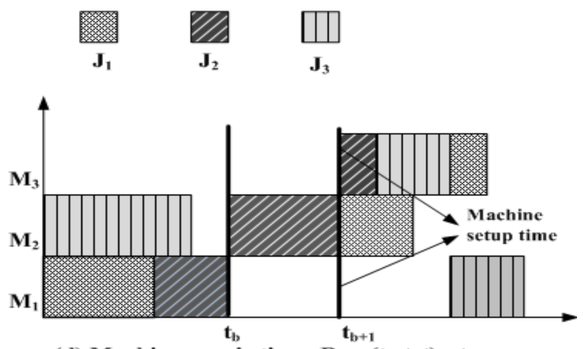


(b) Machine breakdown: M_2 occurs at time t_b

Parts that need to be rescheduled is depicted on Figure 3(c), and finally the machine setup time is shown in Figure 3(d).



(c) Rescheduled parts (dotted line)



(d) Machine repair time: $R_t = (t_b + t) - t_b$

Figure 3 Repair mechanism in occurrence of breakdown machine

i. Insert idle time: used for disruptions such as machine breakdown and absenteeism (unavailability of machine operator) where the affected machine is held idle for the duration of the disruption.

ii. Insert adjustment time: set-up delays cause changes to the start and end times of job operations. Consequently, the processing time of the job operation is adjusted by inserting an adjustment period, which is equivalent to the additional time anticipated for the operation to be processed by the machine.

iii. Insert operation: job operations are inserted in the schedule for disruptions such as the arrival of a new job, postponement of a job due to the unavailability of raw material and relaxation of due dates.

iv. Delete operation: involves the removal of a job operation from the original schedule if it is no longer required, e.g. cancellation of a job. In case of machine breakdown insert idle time repair action is needed to carry out the repair. At the point of breakdown we insert an idle time, which is equivalent to machine repair time, into the schedule. The operation on the machine is put on hold until the machine repair is completed. The schedule is then subsequently repaired using TS with complete regeneration mechanism. Most of the previous work done considered only one broken machine during the production process. As was informed by the expert during interview, in manufacturing with large numbers of machines (m >10), it may happen that we have more than one machine broken down at the same time. The location of breakdowns and the repair time of each machine may vary since it occurs randomly. Therefore, we apply fuzzy technique to 'manage' the repair time of each broken machine on the reactive stage. We first define the linguistic variable of fuzzy membership function which is machine repair time (R_t). Afterwards we categorize the variable into three linguistic values of fuzzy sets, which is: FAST, MEDIUM and SLOW. Subsequently, we generate a membership function of machine repair time. We refer to the work of [13] that generates membership functions automatically as shown in Figure 4. This method partitions a set of data into classes that can be used to derive membership functions. The procedure of the algorithm has several major steps, which is described in the following steps

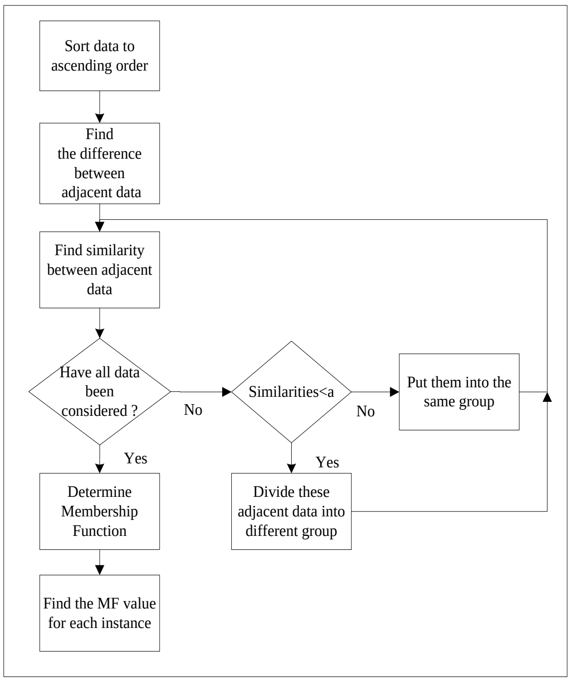

Sort data to ascending order
Find the difference between adjacent data
Find similarity between adjacent data
Have all data been considered ?
No
Similarities<a
No
Put them into the same group
Yes
Yes
Determine Membership Function
Divide these adjacent data into different group
Find the MF value for each instance

Figure 4 The procedure to generate membership function

Step 1. Given a data set, there are n training samples. The values for the parameter in question $\vec{X} = x_1, x_2, \cdots,$ are sorted into ascending order, denoted as $\vec{Y} = sort[\vec{X}] = y_1, y_2, \cdots,$ The values are sorted to find an association between adjacent values.

Step 2. The difference between adjacent values in the sorted data is determined. The difference obtained will provide a way to calculate the similarity between adjacent values. The difference for a set of training set data is: $diff_i = y_{i+1} - y_i$ for $i = 1, 2, 3, \dots, (n-1)$ where y_i and y_{i+1} are adjacent values in the sorted data.

Step 3. Find the similarities between adjacent values. Using(1) we find the similarities , between adjacent values and maps them into real numbers between 0 and 1,

$$s_i = \begin{cases} 1 - \dfrac{diff_i}{C * \sigma_s} & \text{for } diff_i \le C * \sigma_s, \\ 0 & \text{otherwise,} \end{cases} \qquad (1)$$

Where s_i represents the similarity between y'_1 and y'_{i+1} $diff_i$ is the difference between adjacent data, y'_1 and y'_{i+1}, $\sigma_s$is the standard deviation of $diff_i$, and C is the control parameter. The control parameter is used to determine the shape of the membership function. A larger C causes a great similarity.

Step 4. Cluster the training set instances according to its similarity. The data is grouped according to similarities. A threshold value,α, is used to cluster the instances, where the value of α determines the threshold for two adjacent data. If the similarity is greater than the threshold value, then the two adjacent values belong to the same class, otherwise the values are divided into different classes. It expresses as a formula:

$$\text{IF}\,(s_i > \alpha)\,\text{THEN}\ \ y_i, y_{i+1} \in \mathrm{C_i},$$
$$\text{ELSE}\ \ y_i \in \mathrm{C_i}, y_{i+1} \in \mathrm{C_{i+1}}$$

Where C_i and C_{i+1} denote two distinct classes for the same input or output parameter

Step 5. The membership function for each class is defined. Membership function of fuzzy sets can have a variety of shapes. Among the available fuzzy set shapes, we apply piecewise linear functions, i.e. triangular and trapezoidal shapes, which are simple to implement but provide an adequate representation of expert knowledge, and computationally efficient [19]. Triangular and trapezoid membership functions will be used for the remainder of the equations. After determining membership function, the R_t values are fuzzified against the appropriate linguistic fuzzy set. This process is called fuzzification, which is a process to determine the degree to which these inputs belong to each of the appropriate fuzzy sets. Every data belongs to a group or a cluster with degree of membership by [0, 1] intervals. Using fuzzy logic approach, a data's membership to a cluster is defined between 0 and 1 with a variety of different membership values. Consequently, a data can be belongs to more than one cluster at the same time. Schedule repair can be viewed as a constrained scheduling problem [23]. The constraints are built into the schedule repair algorithm in form of a set of "if-then" rules using expert's knowledge of the domain. There are cases that should be considered in forming If-Then rules, namely:

• If one machine belongs to the FAST category:

If Rt_1 is FAST and Rt_2 is MEDIUM Then Machine-1 is repaired first

If Rt_1 is FAST and Rt_2 is SLOW Then Machine-1 is repaired first

If Rt_1 is MEDIUM and Rt_2 is FAST Then Machine-2 is repaired first

If Rt_1 is SLOW and Rt_2 is FAST Then Machine-2 is repaired first

•If one machine belongs to the MEDIUM category:

If Rt_1 is SLOW and Rt_2 is MEDIUM Then Machine-2 is repaired first

If Rt_1 is MEDIUM and Rt_2 is SLOW Then Machine-1 is repaired first

Apart from the cases above, we should also consider the degree of membership in forming If-Then rules if the repair time of two (or more) broken machines R_{t2} is SLOW, degree of membership should be belongs to the same category, e.g. R_{t1} is SLOW and evaluated. The If-Then rules for such cases is described in the following, where x and y indicates degree of membership of machine-1 and machine-2 respectively.

- If both machines belong to the FAST or MEDIUM or SLOW category:

If Rt_1 FAST(x) and Rt_2 is FAST(y) And x>y Then Machine-1 is repaired first

If Rt_1 FAST(x) and Rt_2 is FAST(y) And x<y Then Machine-2 is repaired first

If Rt_1 MEDIUM (x) and Rt_2 is MEDIUM (y) And x>y Then Machine-1 is repaired first

If Rt_1 MEDIUM (x) and Rt_2 is MEDIUM (y) And x<y Then Machine-2 is repaired first

If Rt_1 SLOW (x) and Rt_2 is SLOW(y) And x>y Then Machine-2 is repaired first

If Rt_1 SLOW (x) and Rt_2 is SLOW(y) And x<y Then Machine-1 is repaired first

6. Managing tabu list

Our experimental study refers to the work of [10] where the Tabu list length is set to hold ten moves. It directs the search to explore the larger region. The principle of attaching the same size of Tabu list to all attributes, though it is not convincing, in many application it has produced good results [10, 28]. The implementation of Tabu list for 3 jobs 4 machines problem is shown in Table1.

For a given size Tabu list, when an element is added to the Tabu list, another (the oldest one) is removed. However, it may happen that a move (such as a move that improves the best solution already found) is Tabu. In order to avoid such move, an aspiration criterion is defined. In case of breakdown, the Tabu Search technique that was used in the scheduling process is modified to deal with the new condition. The modification is made in terms of the tabu list used during the scheduling process whereby in addition to the existing tabu list and the current schedule there is another tabu list added to manage the broken machine. In addition totabu list for keeping the move, another tabu list is used to keep the broken machine. The insertion of the broken machine into this tabu list is intended to prevent the broken machine to be included in the current schedule. In view of the fact that each machine has a different period of maintenance in case of breakdown, the length of the machine kept in the tabu list (the tabu tenure) is varied. In this case, the breakdown machine will be kept on the tabu list until its repairing period has been fulfilled. According to this variation, a fuzzy membership function is designed to classify the repairing period into three categories: fast, medium, and slow.

Table 1 Tabu list of 3 jobs 4 machines problem

Iteration	Neighbors	C_{max}	Move	Tabu list
1	(2,3)→(2,1)	24	(2,1)→(2,3)	(2,1)→(2,3)
2	(1,2)→(1,1)	24	(1,1)→(1,2)	(1,1)→(1,2),(2,1)→(2,3)
3	(1,1)→(1,2), (3,2)→(3,1)	24	(1,2)→(1,1)	(1,2)→(1,1),(1,1)→(1,2),(2,1)→(2,3)
4	(1,1)→(1,2)	24	(1,2)→(1,1)	(1,2)→(1,1),(1,1)→(1,2),(2,1)→(2,3)
5	(1,1)→(1,2), (3,2)→(3,1)	24	(1,2)→(1,1)	(1,2)→(1,1),(1,1)→(1,2),(2,1)→(2,3)
6	(1,2)→(1,1)	24	(1,1)→(1,2)	(1,2)→(1,1),(1,1)→(1,2),(2,1)→(2,3)
7	(1,1)→(1,2), (3,2)→(3,1)	24	(1,2)→(1,1)	(1,2)→(1,1),(1,1)→(1,2),(2,1)→(2,3)
8	(1,2)→(1,1)	24	(1,1)→(1,2)	(1,2)→(1,1),(1,1)→(1,2),(2,1)→(2,3)
…	…	…	…	…

Our design of managing the tabu list is called Fuzzy Tabu Priority List (FTPL) which conforms to the fuzzy membership function using a priority queue. With this design, every time a breakdown occurs a move is performed to the tabu list by inserting the broken machine into the queue. The tabu tenure of themachine is controlled by its membership value whereby whenever the membership interval of the broken machine has been fulfilled during a certain processing period, the machine is removed from the queue regardless of its order of insertion. By removing the newly repaired machine from the tabu list, it could be included again in the current schedule. Assume that we have two broken machines which occur at the same time. After obtaining the degree of membership of the repair time machine, Rt value, we have to decide which machine has to be repaired first. In order to see how the fuzzy technique is hybridized with Tabu Search, the procedure of FTPL is described in Figure 5. The first step is a preparation step where the parameters and variables required during the processing are initialized. The algorithm use an entry-condition loop to firstly check if the termination criteria are met, if this is so, then there are no further processing and the algorithm stop. Otherwise, the second step is to check if there is a machine breakdown, if none the algorithm will process the normal routine of machine scheduling beginning with Step 5. In case of machine breakdown, the scheduling process is handled as in the following. The processing of one broken machine will only need the addition of the length of breakdowntime to the machine processing time and the scheduling process will continue with the normal routine of machine scheduling in Step 5. However, if there are more than one machine breakdown, the fuzzy process will take place by setting up the fuzzy tenure into three categories: fast, medium, and slow. The algorithm then checks if the repair time of the broke machine which lies within the conjunction of two fuzzy sets, takes the minimum value then the processing of the broken machine will begin iteratively, otherwise no processing is needed and the algorithm go back to Step 2. In other word, stopping criteria of FTPL is met if there is no more broken machine found. The iterative process of handling the breakdown machines are performed by the downtime of two machines M_i and M_{i+1}. If the downtime of the two machines belongs to the same category of repair time R_t, then it further checks if the degree of membership of R_t M_i is less than R_tM_{i+1}, in which machine M_i is repaired first and if not, machine M_{i+1} will be inserted into Tabu Priority List. Alternatively, if the downtime of the two machines is not belongs to the same category, the algorithm checks If the degree of membership R_tMi is less than R_tM_{i+1}, and if this is true, then M_i is repaired first, otherwise M_{i+1} will be inserted into Tabu Priority List. After this checking, the repair time of M_i will be added to the processing time and the iterative process will be continued. The final processing of the algorithm is beginning from Step 5 by selecting a candidate schedule from schedule k. The algorithm firstly checks if the Tabu Priority List is empty, then proceed to Step 8. If the Tabu Priority List is not empty, then the algorithm will process each of the tabu machine k by selecting the next schedule S1+i ← S1, incrementing k by 1, and selecting a candidate schedule from schedule k. In Step 8, based on whether or not the move of candidate schedule (S1 ← S0) is tabu, the algorithm will select next schedule S1+i ← S1 and increment k and loop back to the beginning of the iteration loop, or do the following steps: set current schedule to candidate schedule S1+i ← S1, insert reverse move into Tabu List, and if aspiration criterion of candidate schedule is better, then set S0 ← Sc and Increment k and loop back to the beginning of the iteration loop.

```
Step 1. Initialization
        Get Initial Solution $S_1$
Set Tabu List to ∅
Set Tabu Priority List to ∅
Initialize number of iteration $k \leftarrow 1$
Set best solution to initial solution $S_0 \leftarrow S_1$
Step 2. Check termination criteria, if met Stop
Step 3. Check if there is machine breakdown, if none go to Step 5
Step 4. Process machine breakdown
        If broken machine = 1 add repair time $M_i$ to the processing time, go to Step 5
        Else
                Set fuzzy tenure to Fast, Medium or Slow
        If $R_t$ lies within the conjunction of two fuzzy sets take the minimum value
                For each machine breakdown $M_i$ Do
                        If downtime $M_i$=downtime $M_{i+1}$ check $R_t$ category of each machine
        If $R_t$ $M_i$ and $R_t$ $M_{i+1}$ belong to the same category check
        degree of membership
                                        If degree of membership of $R_tM_i < R_t M_{i+1}$,
                                        $M_i$ is repaired first
        Else
        Insert $M_{i+1}$ into Tabu Priority List
        End If
                        Add Repair time $M_i$ to its processing time.
                Next
        Go to Step 2
Step 5. Select candidate schedule from schedule k
Step 6. If Tabu Priority List is empty, go to Step 8
Step 7. While machine k is tabu, select next schedule $S_{1+i} \leftarrow S_1$
Increment k by 1, then select candidate schedule from schedule k
Step 8. If the move of candidate schedule ($S_1 \leftarrow S_0$) is tabu, go to Step 9 else go to Step 10
Step 9. Select next schedule $S_{1+i} \leftarrow S_1$, increment k, go to Step 2
Step 10. Set current schedule to candidate schedule $S_{1+i} \leftarrow S_1$
        Insert reverse move into Tabu List
        if aspiration criterion of candidate schedule is better, set S0 ← Sc, increment k
        Go to Step 2
```

Figure 5. The procedure of Fuzzy Tabu Priority List (FTPL)

7. Updating the Schedule: Total Rescheduling and Partial Rescheduling

After the broken machines are repaired, the disrupted schedule is updated where the feasible schedule before disruptions occur is now considered as an initial solution. Two rescheduling techniques i.e complete regeneration (total rescheduling) and right-shift rescheduling methods (RSR) are employed. In complete regeneration or total rescheduling procedure, the rescheduling is essentially the same as the scheduling procedure of the production planning stage. It leads to a new optimized predictive schedule and its quality, in term of its performance measures, such as cost, tardiness, etc. For complete regeneration we schedule all available schedules, including those that not affected by the disruption. We apply Tabu Search, i. e. FTPL for developing the complete regeneration whereas RSR is done without FTPL. RSR involves the global shifting of the job operations and expanding the schedule towards the right on the time axis. Right-shifting the schedule need only occur when the disruption overlaps with some scheduled processing time in the schedule. If a breakdown disrupts the processing of an operation, then the interrupted operation is right-shifted by the amount of the down time. Since this is done by adding a fixed increment of time of each operation of the current schedule, the right-shift algorithm is clearly polynomial; the computation only involves the addition of a time increment to each of the remaining operations. In other words, the operations on all the machines after the point of disruption are incremented (right shifted). We test the algorithm on benchmark data of JSSB available on Taillard's homepage.

8. Performance Measure

In order to see the potency of both methods, we carried out performance measure adapted from [23, 30]. The aim of this analysis is to determine which of these methods yields better results in terms of the two objectives considered, namely the efficiency and the stability. The measure of efficiency indicates the effectiveness of the repair in the schedule. It is defined as the percentage change in make span of the repaired schedule as compared to the original schedule. Efficiency, , is determied using (2)

$$\eta = \left\{1 - \frac{M_{new} - M_O}{M_O}\right\} x100 \qquad (2)$$

where

η = Efficiency

M_{new} = make span of the repaired schedule

M_O = make span of the original schedule

Stability is a cost function and is considered high if the normalized deviation is low. The normalized deviation, ξ is defined as (3):

$$\xi = \frac{\sum_{j=1}^{k}\sum_{i=1}^{p_j}\left|S^*_{ji} - S_{ji}\right|}{\sum_{j=1}^{k} p_j} \qquad (3)$$

where

ξ = normalized deviation

p_j = number of operations

k= number of jobs

S^*_{ji}= starting time of i^{th} operation of job j in repaired schedule

S_{ji}= starting time of i^{th}operation of job j in original schedule

9. Result and Discussion

The algorithm is applied to the problem with 15 and 20 machines which is taken from Taillard data sets with 15 and 20 machines. Table 2.The machine with SLOW repair time has a larger deviation than the machine with FAST and MEDIUM repair time. In some instances, i.e Ta14 and Ta72, the deviation (% change) is negative. The negative value illustrates that the new make span obtained is better that the initial schedule. This is due to complete regeneration reschedules all available schedules, including those are not affected by the disruption. The deviation (% change) obtained from the initial schedule if we employ right-shift rescheduling, where rescheduling process is done without FTPL, is shown in Table 3.

Based on the result shown in Table 3 the make span deviation (% change) on some instances i.e. Ta02, Ta14, Ta36, Ta54 and Ta72 using FTPL is better than without FTPL. However, on some instances we found the vice versa. Therefore we did performance measureon Tailard's instances, i.e. Ta (15x15) and Ta(20x20)

Table 2. New make span obtained using total rescheduling with FTPL on Tailard's Data Set

			Initial	Broken	Broken at	Downtime	Repair	Complete Regeneration	
No	Data Set	Size	Schedule	Machine	Operation	(unit time)	Time	New Makespan	%Change
1	Ta02	15x15	1265	1	(1,5)	206	FAST	1336	5.61
				5	(5,7)	837	MEDIUM		
2	Ta14	20x15	1562	3	(3,1)	471	MEDIUM	1548	-0.9
				8	(8,9)	985	FAST		
3	Ta36	30x15	1932	6	(5,20)	228	MEDIUM	2019	4.5
				2	(2,20)	1108	FAST		
4	Ta54	50x15	3133	11	(11,16)	1049	0.2 FAST	3224	2.9
				12	(12,16)	1154	0.3 FAST		
5	Ta21	20x20	1518	14	(14,14)	593	SLOW	1680	10.67
				2	(2,5)	1449	FAST		
6	Ta47	30x20	1970	2	(2,1)	494	FAST	2136	8.43
				9	(8,9)	1189	SLOW		
7	Ta70	50x20	3143	19	(19,36)	1133	MEDIUM	3307	5.22
				15	(15,11)	508	FAST		
8	Ta72	100x20	2000	4	(4,50)	360	0.1 FAST	1871	-6.45
				13	(13,44)	896	0.4 FAST		

with a 5% error bar, to see the efficiency and stability of both methods. The efficiency of the total rescheduling, where updating schedule is done using FTPL, and right-shift rescheduling (RSR), where updating schedule is done without FTPL, is illustrated in Figure 6 and Figure 7 respectivelty

From the illustration on Figure 6 and Figure 7, we found that the RSR method yields significantly better performance than the total rescheduling. The complexity of the problem does not make any difference on its performance. In some cases on the instances, the efficiency obtained is more than 100%. This is due to the fact that the new make span obtained in the rescheduling stage is better than its initial solution. The initial solution here is the feasible solution obtained before disruption occurred. Furthermore, in total reschdeuling the scheduleobtained after reschedule may totally different from the initial schedule. Total rescheduling constructs a complete schedule by rescheduling not only the affected operation (or jobs) but also those not affected.Stability is another performance measure to observe on these two rescheduling methods. Figure 8 and Figure 9 show the stability of RSR and total rescheduling on Ta instances: Ta(15x15) and Ta(20x20). Figure 8 shows that the stability of total rescheduling on Ta(15x15) instance is better than that of RSR.

Table 3. The deviation (% change) obtained without and with FTPL

			Initial	Without FTPL		With FTPL	
No	Data Set	Size	Schedule	New Makespan	%Change	New Makespan	%Change
1	Ta02	15x15	1265	1349	6.64	1336	5.61
2	Ta14	20x15	1562	1454	-6.91	1548	-0.9
3	Ta36	30x15	1932	2052	6.21	2019	4.5
4	Ta54	50x15	3133	3275	4.53	3224	2.9
5	Ta21	20x20	1518	1615	6.39	1680	10.67
6	Ta47	30x20	1970	2059	4.52	2136	8.43
7	Ta70	50x20	3143	3207	2.04	3307	5.22
8	Ta72	100x20	2000	2140	7	1871	-6.45

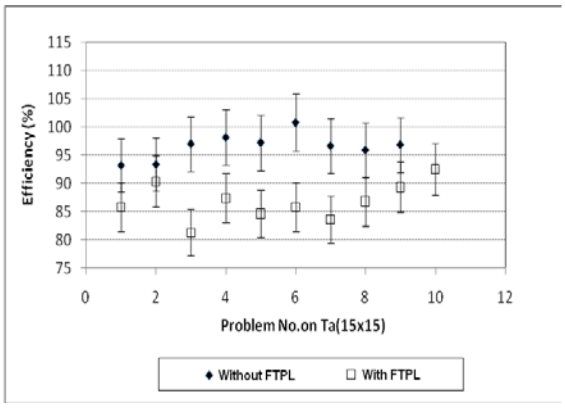


Figure 6.The efficiency of methods used during rescheduling on Ta(15x15)

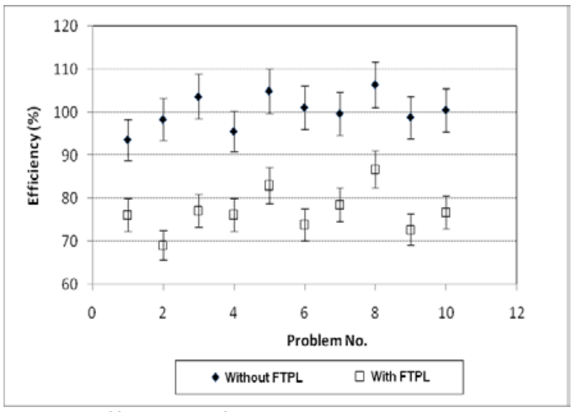


Figure 7. The efficiency of methods used during rescheduling on Ta(20x20)

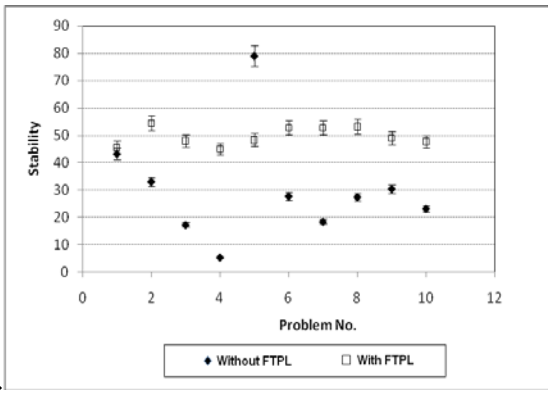


Figure 8. The stability of methods used during rescheduling on Ta(15x15)

However, on other instances, as is shown in Figure 9, RSR has better performance than total rescheduling on some number of experiments.

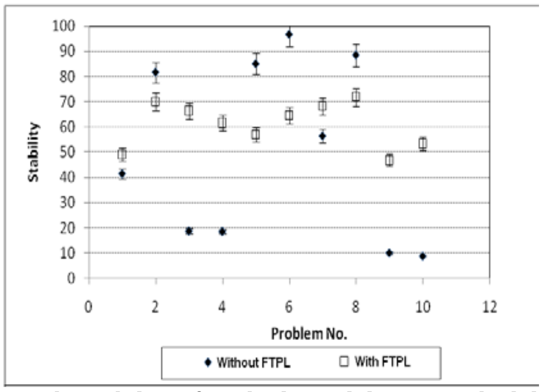


Figure 9. The stability of methods used during rescheduling on Ta(20x20)

The application of FTPL on rescheduling may be capable of maintaining solution with minimum make span though it required high computational effort. The main disadvantage of this procedure is that the stability cost might be extremely high because the resulting schedule can differ completely from the original schedule. On the other hand, rescheduling without FTPL requires the least computationalefforts but it cannot guarantee the solution quality. Therefore,selection of the rescheduling method tobe applied depends on which of the objectives considered is more important, the efficiency or the stability.The robustness of a repair mechanism should combine the maximization of both efficiency and stability.

10. Conclusion

The basic principle of the FTPL algorithm is to keep the broken machine for the amount of its repair time.
The broken machine will be removed from the list based on its priority, i.e. machine with fast repair time will remove first. Therefore, using this priority along with the use of If-then rule the algorithm can repair the schedule quickly. As for the method use for rescheduling, we find that complete regeneration needs more computational effort that leads to a new optimized predictive schedule which may differ considerably from the old (i.e. the initial schedule) or
sometimes better than the initial schedule. In our future research we will consider to apply other approach i.e Scatter Search to compare the result with ours.

References

[1] Abdullah, S., Aickelin, U., Burke. E., Mohamed Din, A. & Qu. R. 2007. Investigating Hybrid Metaheuristic Job shop scheduling. Lecture Notes in Computer Science 4828: 357-368.

[2] Alogoz, O. & Azizoglu, M.2003. Rescheduling of identical parallel machines under machine eligibility constraints. European Journal of Operational Research 149 (2003): 523–532.

[3] Ayob, M. & Kendall, G. 2003. A montecarlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. *The Proceedings of the International Conference on Intelligent Technologies (InTech'03)*, pp. 132-141.

[4] Chong, C.S., Sivakumar, A. I. & Gay, R. 2003. Simulation-based scheduling for dynamic discrete manufacturing.In Chick, S., Sanchez, P. J., Ferrin, D. &Morrice, D. J. (Eds).Proceeding of the 2003 Winter Simulation Conference, pp. 1456-1473.

[5] Finke, D. A., Medeiros, D. J. &Traband, M. T. 2002. Shop scheduling using Tabu search and simulation. In Yucesan, Chens, Snowdon&Charnes (eds) Proceeding of the 2002 Winter Simulation Conference, pp. 1013-1017

[6] Fox, M. S. 1994. ISIS: A Retrospective. In Zweben& Fox, S. (eds.) Intelligent Scheduling. San Fransisco: Morgan Kaufmann.

[7] Garg, I. 2008. Expert systems application in production scheduling.In Proceeding of 2008 IABR and TLC Conference, pp. 1-4.

[8] Gendreau, M. 2003. An introduction to Tabu Search. In Glover, F. &Kochenberger, G.A. (eds.). Handbook of Metaheuristics, pp. 37-54. New York. Springer.

[9] Gendreau, M. &Potvin, J-Y. 2005. Tabu Search. In Burke, E. K & Graham, K. (Eds.) Search Methodologies: Introductory Tutorials in Optimizations and Decision Support Techniques, pp. 165-186. New York: Springer.

[10] Geyik, F &Cedimoglu, I. H. 2004. The strategies and parameters of Tabu search for job shop scheduling. Journal of Intelligent Manufacturing 15:439-448.

[11] Glover,F.1989. Tabu Search. Part I. ORSA Journal of Computing, 1:190-206.

[12] Gomes, C. P. 2000. Artificial Intelligence and Operation Research: Challenges and opportunities in planning and scheduling. Journal of The Knowledge Engineering Review 15(1):1-10.

[13] Hong, T. P & Lee, C. Y. 1996. Induction of fuzzy rules and membership functions from training examples. Fuzzy Sets and Systems 84: 33-47.

[14] Ishibuchi, H., Yamamoto, M., Misaki, S. & Tanaka, H. 1994. Local search algorithms for flow-shop scheduling with fuzzy due dates. Int'l Journal of Production Economics 33: 53-66.

[15] Kunnathur, A.S, Sampath, S. &Sundaragharan, P.S. 1996. Dynamic rescheduling of a job shop: a simulation study. In Proceeding of 28th Conference of Winter Simulation, Charnes, J.M, Brunner, D.T, Swain, J.J (eds). California, pp. 1091-1098.

[16] Kuroda, M. & Wang, Z. 1996. Fuzzy job shop scheduling. Int'l Journal Production Economics 44: 45-51.

[17] Liao, C.J. & Chen, W.J. 2004. Scheduling under machine breakdown in a continuous process industry. Journal of Computer and Operations Research 31(3):415-428.

[18] Lee, H.T., Chen, S.H. & Kang, H.Y. 2002. Multi-criteria scheduling using fuzzy theory and Tabu search. International Journal of Production Research 40 (5): 1221-1234.

[19] Negnevitskyi. M. 2002. Artificial Intelligence: A Guide to Intelligent Systems. Essex: Addison-Wesley.

[20] Petrovic, S., Fayad, C. &Petrovic, D. 2005. Job Shop Scheduling with Lot-sizing and Batching in an Uncertain Real-world Environment, in: Kendall, G., Lei, L. &Pinedo, M. (Eds.) Proceedings of the 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), pp. 363-379.

[21] Pinedo, M. 2002. Scheduling: Theory, Algorithms, and Systems. 2nd ed. New Jersey, Prentice Hall.

[22] Pinedo. M. 2005. Planning and Scheduling in Manufacturing and Services. New York: Springer.

[23]Raheja, A.S. &Subramaniam, V. 2002. Reactive recovery of job shop schedule: a review. Journal of Advance Manufacturing Technology (2002)19: 756-763.

[24] Roy, U. & Zhang, X. 1996. A heuristic approach to n/m job shop fuzzy dynamic scheduling algorithms. Journal ofProduction Planning and Control 7(3): 299-311.

[25] Russell, S. &Norvig, P. 2002. Artificial Intelligence: A modern approach 2nd ed. New Jersey: Prentice Hall.

[26] Taillard, E. 2004, Metaheuristics for Hard Combintorial Optimization Problems, Applied Univ. of Western Switzerland.

[27] Sabuncuoglu, I. &Karabuk, S. 1999. Rescheduling frequency in an FMS with uncertain processing times and unreliable machines.Journal of Manufacturing Systems. 18(4): 268-283.

[28] Salhi, S. 2002. Defining tabu list size and aspiration criterion within tabu search methods.*Computers & Operations Research***29**:67-68

[29] Souyer, H. Kocamz, M. &Kazancoglu, Y. 2007. Scheduling through multiple parallel channels using an expert system.Production Planning & Control. 18(1): 35-43.

[30] Subramaniam, V., Raheja, A. S & Reddy, K.R.M. 2005. Reactive repair tool for job shop schedules. International Journal of Production Research 43(1): 1-23.

[31] Van de Vonder, S., Demeulesmeester, E. &Herroelen, W., 2007.A classification of predictive-reactive project scheduling procedures.Journal of Scheduling (2007) 10: 195-207.

[32] Wang, Wan-Liang, Xu ,Xin-Li, Wu. & Qi-Di. 2003. Hopfield Neural Networks Approach for Job Shop Scheduling Problems, Proceeding of the 2003 IEEE International Symposium on Intelligent Control, pp. 935-940.

[33] Watson, J. –P., Whitley, D. & Howe, A. E. 2005. A dynamic model of Tabu search for the job shop scheduling problem. In Kendall, G., Burke, E., Petrovic, S and Gendreau (Eds.) Proceeding of Multidisciplinary Scheduling: Theory and Applications, pp. 247-266.

[34] Zhang., C. Y., Li, P., Rao, Y. & Guan, Z. 2008. A very fast TS/SA algorithm for the job shop scheduling problem". Journal of Computers and Operations Research 35(2008): 282-294.