2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Memetic Algorithm for solving Permutation Flow Shop problems with Known and Unknown Machine Breakdowns

Humyun F. Rahman, Ruhul A. Sarker, Daryl L. Essam, and Guijuan Chang

***Abstract*—The Permutation Flow Shop Scheduling Problem (PFSP) is considered to be one of the complex combinatorial optimization problems. For PFSPs, the schedule is produced under ideal conditions that usually ignore any type of process interruption. In practice, the production process is interrupted due to many different reasons, such as machine unavailability and breakdowns. In this paper, we propose a Genetic Algorithm (GA) based approach to deal with process interruptions at different points in time in Permutation Shop Floor scenarios. We have considered two types of process interruption events. The first one is predictive, where the interruption information is known well in advance, and the second one is reactive, where the interruption information is not known until the breakdown occurs. An extensive set of experiments has been carried out, which demonstrate the usefulness of the proposed approach.**

I. Introduction

THE Permutation Flow Shop Scheduling Problem (PFSP) is a challenging issue in the manufacturing industry. A classical PFSP consist of processing of the *n* jobs on *m* machines. Makespan minimization is a common measure of performance for PFSPs. It can be defined as the time difference between the starting of the first operation in the first machine and the ending of the last operation in the last machine. To solve this, first in 1954 Johnson [1] proposed an algorithm that solves two machine PFSPs, as well as special three machine PFSPs, optimally. However, the general PFSPs with more than two machines are *NP Hard* [2]. As a consequence, for large problems, the mathematical programming based methods, like integer programming, are unable to provide a good quality solution within a reasonable amount of time. This provides an opportunity to study the suitability of heuristics and meta-heuristics approaches for solving PFSPs. Nawaz *et al*. [3] proposed a simple, and one of the best, constructive heuristics for PFSPs [4]. However, it deviates up to 7% from the known optimal solutions for some well-known problems [5]. This algorithm is based on the concept that the jobs with longer operating times on all of the machines, should be placed as early as possible in the sequence. To achieve better quality solutions within a reasonable amount of time, researchers have applied other meta-heuristic such as ant colony algorithm [6], particle swarm optimization [7], simulated annealing [8], tabu search [9], Genetic Algorithms (GAs) [10], and Hybrid GAs [5, 11].

Almost all scheduling research in PFSPs has mainly focused on ideal conditions, assuming an uninterrupted production system. However, in practice, process interruptions are very common events on the shop floor. The addition of such interruptions make the PFSP more practical, but also complex. Production can be interrupted due to both preventive and breakdown maintenances of production equipment. These include: machine overhaul, machine failure, unavailability of raw materials, order rejection, sudden arrival of a new job, variation in processing time, and change in job priority [12, 13]. Machine unavailability due to preventive maintenance schedule is usually known in advance. So it can easily be incorporated within the Permutation Flow Shop scheduling as a dummy job. In case of sudden machine breakdown, the jobs scheduled in that machine cannot be processed until the machine is appropriately repaired. Such a machine breakdown would delay the completion time, for some jobs already scheduled, in order to satisfy precedence and capacity constraints. However, the delay in completion can be minimized by re-optimizing the remaining schedule. Even after re-optimizing the interrupted schedule, there may be some delays in completing some or all of the jobs in the sequence.

There are few studies that consider re-optimization of scheduling to deal with production interruptions. The planned process interruptions in scheduling, such as preventive machine maintenance, can be considered as an additional constraint while solving a scheduling problem in single machine and multi-machine environment. Sarker *et al.* [14] and Hasan *et al.* [15] proposed a GA based approach to solve job shop scheduling problems with the machine unavailability condition.

Production scheduling, with respect to sudden process interruption, is more challenging. The application of some dispatching rules, that appears to be the easiest approach, is widely used to minimize the delay after sudden machine breakdowns [16]. Abumaizar and Svestka [17] proposed a right shifting approach to repair the affected operations in a job shop schedule. Liu *et al.* [18] developed a tabu search based approach to solve job shop problems with machine breakdowns. Fahmy *et al*. [12] proposed an approach of inserting dummy jobs in the place of affected jobs, while the affected jobs were rescheduled later. The problems with such approaches are that they increase the computational complexity of the scheduling algorithm. Wu *et al*. [19] developed a GA, combined with a pair wise heuristic, to

Mr. Humyun F. Rahman, Associate Professor Ruhul A. Sarker, and Dr Daryl L. Essam are with the School of Engineering and Information Technology, the University of New South Wales, Canberra, Australia. (E-mail: md.rahman4@student.adfa.edu.au, r.sarker@adfa.edu.au, d.essam@adfa.edu.au).

Associate Professor Guijuan Chang (Lucy) is with the School of Science and Information Science, Qingdao Agricultural University, Shandong, China. (E-mail: lucycgj@163.com).

This work is supported by the University of New South Wales, Canberra, ACT 2600, Australia in the form of a Postgraduate Research Scholarship.

reschedule the scheduling problems after the disruptions. The problem with this approach is it increases machine idle times by uniformly shifting every operation. Sarker *et al*. [14] and Hasan *et al*. [15] developed a hybrid-GA based approach to solve job shop scheduling problems with sudden machine breakdowns. From the above discussions, some studies have been conducted on interruptions in scheduling, due to machine unavailability and breakdown, in the job shop environment.

In this paper, we have considered a Permutation Flow Shop environment, where production is disrupted due to machine unavailability and breakdowns. The process interruptions can be classified into three categories: the interruption is completely unknown, the possibilities that the machine will break in some point in time in the future, and that the information about the interruption is fully known at the beginning of schedule [20]. We have considered the first (machine breakdowns) and third types (machine unavailability) of interruptions. For machine unavailability, the schedule can be generated by considering a dummy job time slot equivalent to the corresponding machine's downtime. For machine breakdown, the information is only available after the real breakdown of some machine. In that case, re-optimizing the rest of the operations should help to minimize the delay in processing of some jobs. We have proposed a memetic algorithm to solve PFSPs with conditions of machine unavailability and breakdowns. To judge the effectiveness of the proposed algorithm, we have run a number of experiments with different interruption scenarios. The results showed that the revised schedule is able to minimize the delay if the interruptions occur in the early stages of the schedule.

The paper is organized as follows. After the introduction, a brief outline of PFSPs with process interruptions is given. The MA is discussed in section III. The experimental study and the effectiveness of the proposed algorithm are presented in section IV. Section V provides the conclusions of this research.

II. Problem Statement and Assumptions

The standard PFSPs definition, with necessary assumptions, is described in this section.

A. Problem Definition

The permutation flow shop problem consists of m machines and n jobs. Each machine processes the same sequence of jobs and each job has to follow the same order of machines. The processing time of the j^{th} job ($j = 1,2,\ldots..,\ n$) on the i^{th} machine ($i = 1,2,\ldots..,\ m$) is $p_{i,j}$, and is known. If the completion time of the j^{th} job on the i^{th} machine is $C(i, j)$ then makespan can be calculated as

$$\text{Makespan, } C_{max} = C(m, n) \tag{1}$$

So the objective is to find a job sequence, α*, from the set of all feasible job sequences A, so that

$$C_{max}(\alpha^*) \leq C_{max}(\alpha) \qquad \forall\ \alpha \in A \tag{2}$$

B. Assumptions

The following assumptions are made for PFSPs with process interruptions:

- The processing time for each operation in each machine is known.
- Operations are non-preemptive.
- Process may be interrupted due to machine unavailability and breakdowns.
- Set up costs, delivery, and transportation costs are negligible.
- Each machine can process only one job at a time.
- A machine cannot process another job until it finishes the current job.

The objective of the problem is to identify a sequence of jobs to minimize the makespan while satisfying all the constraints.

III. PFSPs With Machine Unavailability and Breakdowns

We have considered machine unavailability and breakdowns within the classical PFSPs. The most important aspect of preventive maintenance and breakdown, is that an overhauled or broken machine cannot process jobs until it achieves its full operating condition or it is being replaced by a new one. If a job is incomplete due to an overhauled or broken machine, the job has to wait for a certain period of time. In PFSPs, jobs are interrelated. If there is any delay in processing a job, all jobs in the right of the schedule will be affected. For convenience of analysis, in this paper, we have assumed that the interruption would occur only in the first or the last machine.

In case of machine breakdowns, we classify all the jobs in the sequence into two types: *a*) affected, and *b*) unaffected.

- *Affected jobs:* Any job or set of jobs that has not begun processing in the first machine when a breakdown starts are considered as affected jobs. It is noted that even if the breakdown occurs in the last machine, the list of affected jobs starts from the jobs that are waiting to be processed in the first machine after the interruption starts. The reason for this is that the affected jobs need to be rescheduled and in PFSP each machine has to follow the same processing order of jobs. Considering this precedence constraint, it is possible to determine the set of waiting jobs that can be rescheduled with a revised starting time in each machine. For example, assume 6 jobs are processed in a flow shop and the processing order of the jobs is 2-1-4-5-6-3. If a breakdown starts in the last machine while the job-5, 6, and 3 are waiting to be processed by the first machine, then these jobs are the affected jobs.

- *Unaffected jobs*: Any job that has completed

processing in the first machine before the breakdown starts are known as the unaffected jobs, or more simply, the jobs which are not affected in the job sequence are called the unaffected jobs. Following the above example jobs-2, 1, and 4 are the unaffected jobs in the sequence. Unaffected jobs do not need any rescheduling or reactive scheduling.

Abumaizar and Svestka [17] classified the interruptions into two modes: resume and repeat. When the tasks are preemptive, they can be resumed at the time when the interruption occurred. However, tasks in the repeat mode should be restarted whenever they are interrupted. In this study we have considered that tasks are non-preemptive. So we only consider repeat mode. In the following section, we describe the machine breakdown and machine unavailability events with examples.

A. Machine Breakdown:

Whenever a breakdown happens, there are some delays in completing the affected jobs. This effect can be minimized by rescheduling those jobs. In this research, we proposed and implemented a memetic algorithm (MA) to schedule and reschedule (after breakdown) the jobs in PFSPs.

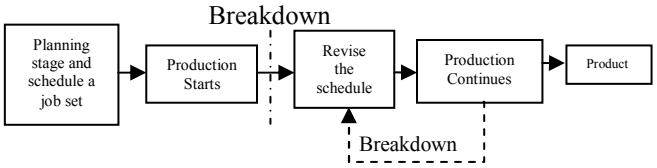

Fig. 1. Job processing flow diagram with machine breakdown

The job processing flow diagram with machine breakdown constraints is presented in Figure 1. At the beginning of a production plan, a set of jobs is scheduled by the MA. Whenever a machine breaks, the affected jobs have been identified and the jobs are rescheduled by re-runing the MA.

The Gantt chart in Figure 2 shows an example of a Flow Shop with a machine breakdown. In this example, we consider a10 job machine 5 machine flowshop with a single breakdown in the first machine (represented by a black rectangle). At the beginning of the planning horizon, the initial sequence obtained by the MA is 5-4-9-3-1-7-6-10-2-8. Assume that while processing job-9 (3rd job in the sequence), machine-1 breaks after 15 units of times from the start of the production. After 19 units of times (say downtime duration is 4) machine-1 returns to operating condition. As no-preemption is allowed, machine-1 repeats the processing of job-9. Meanwhile, jobs 3-1-7-6-10-2-8 are the affected jobs. With the traditional Right Shifting technique, all the affected jobs are shifted to the right and the makespan with this approach is 108 (Figure 2(a)). However if MA re-optimizes the affected jobs, then revised schedule is 5-4-9-1-6-7-3-10-2-8 (Figure 2(b)). After rescheduling, the makespan, C_{max}, is 106.

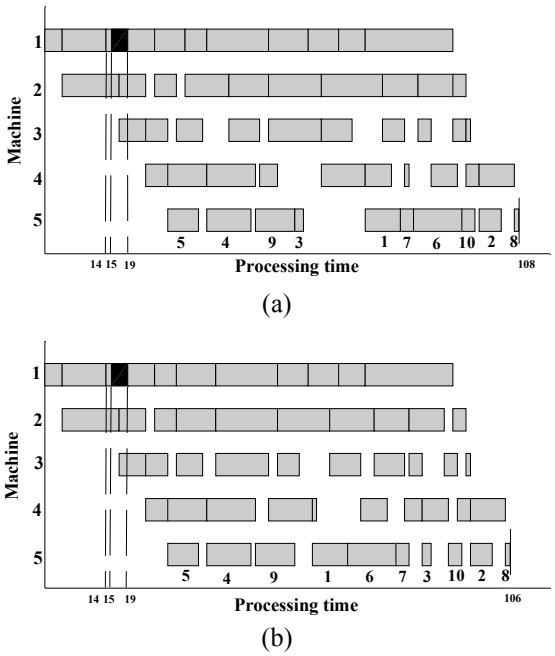

Fig. 2. Gantt chart of Permutation Flow Shop with machine breakdown

B. Machine Unavailability:

As indicated earlier, a machine's unavailability start time and duration is known at the beginning of a schedule. So there is no need to classify the jobs as affected and unaffected. We consider machine unavailability as a constraint which represents forbidden time periods that cannot be used to process any job.

Figure 3 represents a job processing flow diagram with the machine unavailability condition. At the beginning of production, the schedule generated by the MA considers each and every unavailability start time and duration.

A simple illustration of scheduling PFSPs with machine unavailability conditions is shown in Figure 4. Assume that the first machine is unavailable from 15 to 19 units of time. With respect to forbidden that time period in machine-1, a new schedule is generated by MA. The new schedule is 1-9-5-3-4-6-7-10-2-8 and the makespan, C_{max}, is 105.

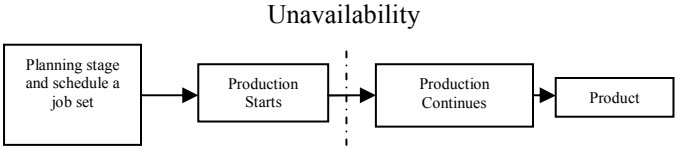

Fig. 3. Job processing flow diagram with machine unavailability

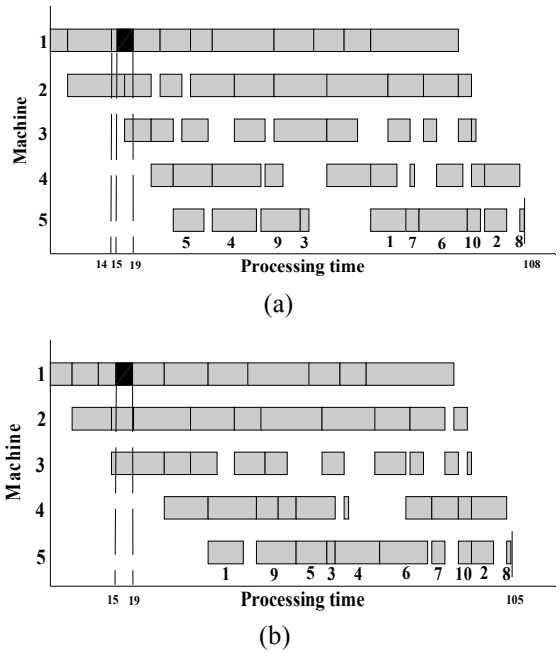

Fig. 4. Gantt chart of Permutation Flow Shop with machine unavailability

IV. The Proposed Algorithm

For solving PFSPs, GA is a popular technique to find optimal or near optimal solutions. According to the definition of the problem, the sequence of jobs used by each and every machine is given. In this case, the schedule can be evaluated by calculating the makespan.

A. Representation of Chromosome and Initial Generation

In GA, chromosomes can be represented by integer, binary or real numbers. In PFSPs, the most popular encoding is the sequence of jobs, where the job sequence represents the processing order of jobs by each machine. Traditionally, a GA starts with a random initial population, but in complex problem like PFSPs, a random initial population may not help to achieve quality schedules within a reasonable time [10, 11]. We have proposed a non-random initialization where certain sets of solutions in the initial population are generated by the NEH [3] and Johnson's algorithms [1]. In the first approach, we generate some individuals by randomly swapping two jobs from the NEH generated solution. In the second approach, the *m* machine (more than two machine problem) flow shop is divided into two machine PFSPs, and these sub-problems are solved by Johnson's algorithm. In summary, a certain set of solutions are generated from these techniques, and the rest of the solutions are generated randomly.

B. Selection approach and enhancement in population

The parents are selected by a traditional tournament selection technique. The parents are replaced by their offspring directly. In combinatorial optimization problems, the evaluation process usually generates many duplicate individuals in every generation, which can cause the algorithm to stagnate in local optima. To avoid this, in every generation duplicate individuals are directly replaced by the random ones. It has been observed that different job sequences may have the same fitness value [10], so the duplication is checked according to the same job sequence. After some generations, if there is no improvement in the solution, then a restart scheme has been applied [5, 10, 11]. If the solution becomes stuck for a certain number of generations, all the individual are splited into three sections: best 5%, middle 85% and 10% bottom. The middle set is then replaced by shift mutations of the best group, and the 10% members are replaced by new randomly generated individuals. An elite member from each generation is also saved and transferred directly to the next generation.

C. Crossover and Mutation

We have selected Similar Job Order Crossover (SJOX) [10] and shift mutation as the genetic operators as they have performed well in earlier experiments. In SJOX, the same jobs are in the same position for both parents, then they are inherited directly in the same position of both offspring. After that, all other jobs up to a random crossover cutting point from each of the parents are directly inserted into their offspring. Next, the missing jobs from each offspring are inserted from the other parent in the relative job sequence of that parent. In shift mutation, a job is chosen randomly from the job sequence. Next the selected job is inserted into a randomly selected position.

D. Local Search

Local search is used to achieve quality solutions within a reasonable amount of time. In this algorithm, a three stage local search has been applied. The local search process contains three steps: first, the selected individuals go through the Insertion Neighbourhood search, then through Gap Filling process and finally, through Job Shifting process.

- *Insertion Neighbourhood:* Insertion Neighbourhood act well in PFSPs [11, 21]. For it, each job is selected and placed in every possible position in the current job sequence, and if a better job sequence is found, then it replaces the current best.
- *Gap Filling technique:* In PFSPs, jobs in the first machine are compact, because no inter-job gap is allowed in the first machine. However due to precedence constraints, it is common to leave some gaps between the consecutive tasks processed on other machines. Preliminary experiments shows that the makespan of a job sequence can be improved if the inter-gap in the last couple of machines can be removed or reduced by inserting a job from the right of the gap's position. The procedure is: the total processing time of all jobs in the job sequence is computed, and the job with the minimum total processing time is directly inserted into the gap. If the movement minimizes the makespan, then the new job sequence has been selected. Else, insert the job with the second best total processing time. If one of the jobs has the least total processing time, then insert another job which has the second best total processing time. For each gap, trials are limited for up to three times.
- *Job Shifting:* The Makespan of a job sequence can be

improved if the job with (*a*) the longer operating time is at the last few machines and (*b*) the shorter operating times in the first few machines are placed as early as possible in the sequence. The procedure is: calculate the ratios (processing time in the last machine divided by the processing time in the first machine) for all jobs in the schedule. If the last three jobs (limited up to three bigger ratios) have the greater ratios in the schedule, then they will be placed as early as possible in the job sequence.

E. The Algorithm:

The proposed memetic algorithm (MA) can be described as follows. Assume that P is the total number of individuals in each generation and G is the maximum number of generations. $(C_{max})_p$ is the makespan of the p^{th} individual in a generation.

1. Generation Initialization, set $g = 1$
2. while $g < G$ (Repeat until the stopping criteria is met)
   a. set $p = 1$
   b. Repeat until $p > P$
      i. Selection: Select parents from the selection pool
      ii. Crossover: child 1, child 2 (Similar Job order Crossover)
      iii. Mutation: child 1, child 2 (Shift mutation)
      iv. Duplication scheme
      v. Local Search
         - Insertion neighbourhood
         - Gap Filling
         - Job Shifting
      vi. Evaluate the makespan, $(C_{max})_p$
      vii. Elitism strategy (select the elite member from the current generation)
      viii. Set $p = p + 1$

   [End of Loop 2 (b)]
   c. Restart Mechanism (Restart the generation, if the required condition is met)
   d. Set $g = g + 1$

[End of Loop 2]

3. Select the best sequence.

[End of the algorithm].

V. EXPERIMENTAL ANALYSIS AND RESULTS

We implemented MA to solve PFSPs with machine breakdowns and unavailability as interruptions. Recall that in Permutation Flow shop problems each job has to follow the same processing order of machines. So any job which is in process in the 1st machine or has already been processed by the 1st machine, cannot be revised or re-optimized. Due to this restriction, we divide the total processing time of the 1st machine into two equal segments and generate interruption events in those segments. Interruption events on the first machine are randomly generated in the first half of the segments and interruption events on the last machine are generated on the last half of the segments.

The algorithm was coded in C++ and ran on a personal computer under the windows operating system. To test the performance of the algorithm in a systematic manner, we have chosen 24 problems from Taillard's benchmark [22], where the first two problem instances have been selected from each problem group.

A. Parameters Selection

Interruption events are generated randomly. For an interruption event we choose the following parameters.

- *Number of breakdowns:* either single interruption in a single machine or two non-overlapping interruptions in two different machines (one interruption in the 1st machine and another in the last machine).
- *Choose a machine or machines:* in this study we consider that either the first machine, or the last machine, or both machine breakdown.
- *Start time and recovery duration:* Start time of an interruption is calculated by a Poission distribution and an Exponential distribution is used to identify the recovery duration, which is more similar to realistic interruption events [15].

Other parameters are selected on the basis of our earlier study [11]: population size is 100, the crossover rate is 90% and the mutation rate is 60%. The first individual in the initial generation is produced by the NEH algorithm [3], 40% of individuals are produced from the modified NEH algorithm, 10% of individuals are generated from the modification of Johnson's algorithm [1], and rest of the individuals are produced randomly . Tournament pool size is 5. Except the initial generation, the first 30 members from each generation go through the local search process. If there is no improvement in the fitness for 10 consecutive generations, the restart scheme attempts to escape from the local optima. For each problem instance, we ran the experiments for 10 times with different random seeds.

B. *Results and Analysis:*

To judge the performance of our approach, we compared our approach with the Right Shifting technique (RST) [14, 15]. We implemented both RST and MA based Revision scheduling (MA-RSV) independently with the same interruption scenarios. Moreover, to see how machine breakdown (MBD) differs from machine unavailability (MUN), we implemented the same downtime scenarios (downtime start time, duration and location) to both MBD and MUN.

Recall that, we consider three different interruption scenarios (for both MBD and MUN): i) single disruption in the first machine, ii) single disruption in the last machine, and iii) combination of both disruption (one in the first machine and another in the last machine).

TABLE I. COMPARISION OF RHT AND RSV WITH A MACHINE BREAKDOWN SCENARIO (SINGLE DISRUPTION IN THE FIRST MACHINE)

Problem	Problem Instance	Initial Makespan	Single Breakdown (1st machine)		Right Shifting		Rescheduling	
			Start Time	Breakdown Duration	Makespan	Computational Time (sec)	Makespan	Computational Time (sec)
20×5	Ta001	1278	45	358	1642	1	1642	3
20×10	Ta011	1582	51	408	1996	1	**1987**	8
20×20	Ta021	2297	40	309	2599	2	**2542**	12
50×5	Ta031	2724	66	565	3289	1	**3256**	13
50×10	Ta041	3025	53	434	3482	1	**3428**	30
50×20	Ta051	3893	56	460	4349	1	**4283**	40
100×5	Ta061	5493	43	333	5843	3	5843	45
100×10	Ta071	5770	58	486	6273	3	**6250**	89
100×20	Ta081	6268	229	2519	8836	3	**8798**	107
200×10	Ta091	10885	89	820	11816	5	**11735**	152
200×20	Ta101	11348	88	808	12185	8	**12131**	185
500×20	Ta111	26265	63	457	26786	10	**26712**	390

TABLE II. COMPARISION OF RHT AND RSV WITH A MACHINE UNAVAILABILITY SCENARIO (SINGLE DISRUPTION IN THE FIRST MACHINE)

Problem	Problem Instance	Initial Makespan	Single Unavailability (1st machine)		Right Shifting		Rescheduling	
			Start Time	Unavailability Duration	Makespan	Computational Time	Makespan	Computational Time
20×5	Ta001	1278	45	358	1642	1	**1278**	3
20×10	Ta011	1582	51	408	1996	1	**1622**	9
20×20	Ta021	2297	40	309	2599	1	**2311**	14
50×5	Ta031	2724	66	565	3323	1	**2768**	16
50×10	Ta041	3025	53	434	3483	1	**3077**	34
50×20	Ta051	3893	56	460	4347	1	**3960**	51
100×5	Ta061	5493	43	333	5861	3	**5493**	80
100×10	Ta071	5770	58	486	6274	3	**5832**	104
100×20	Ta081	6268	229	2519	8828	3	**8565**	120
200×10	Ta091	10885	89	820	11831	5	**11615**	140
200×20	Ta101	11348	88	808	12167	8	**12025**	200
500×20	Ta111	26253	63	457	26795	9	**26681**	380

Table I demonstrates the experimental results for both Right Shifting and Rescheduling approaches under single breakdown scenario in the first machine. The first two columns represent the problem size and problem instances respectively. In this table the comparative results for the first problem instance from each group has been presented. Next column presents the makespan value for the initial schedule. These makespan values are generated assuming that schedules are uninterrupted over the span of production. The column headed with Single Breakdown presents the relative first machine breakdown information. The two columns under the heading of Right Shifting show the makespan value obtained by repairing the disrupted schedule after a breakdown in the first machine and the computational times to evaluate the makespan of the schedule. The final two columns present the makespan values and the computational times required for by proposed algorithm after repairing the schedule under the same disruption scenario. From comparative analysis, it is clear that, for most of the cases (except the first instance *Ta* 001 and *Ta* 061), after breakdown, proposed rescheduling strategy shows superior performance (reduce the makespan) with respect to traditional Right Shifting techniques. Another observation is that, with respect to Right Shifting strategy, Rescheduling approach is bit more computationally expensive. The reason is, in Right Shifting approach, the initial schedule is repaired directly by shifting the affected jobs to the right of the schedule. On the other hand, in Rescheduling approach the affected jobs are re-optimized by the proposed MA. However, in contrast to quality of the final schedule after an interruption, the increase computational time may be acceptable.

The structure of Table II is same as the structure of Table I. The only difference is in Table II, we demonstrate the experimental results for machine unavailability or preventive maintenance scenario in first machine. The comparative analysis between Right Shifiting and MA based Rescheduling shows that Rescheduling or scheduling approach considering the machine unavailability constraints or dummy job, at the beginning of the schedule is more effective than repairing the schedule once the first machine is overhauled.

Effectiveness of RSV (for MBD or MUN) for each problem instance can be represented by the following equation-

$$\text{Average Percentage of Improvement, API} = \left(\sum_{i=1}^{r}\left(\frac{CRST - CRSV}{CRSV}\right)/r\right)\times 100 \quad (3)$$

**CRST*- the makespan value by *RST*; *CRST* - the makespan by *MA-RSV*,*r*-number of independent runs.

TABLE III. API COMPARISION OF RHT AND RSV WITH MACHINE BREAKDOWN SCENARIOS

Problem	Problem Instance group	Single Breakdown		Multiple Breakdown
		1st machine	Last machine	Combined
20×5	Ta001-002	0	0	0
20×10	Ta011-012	0.317649	0	0.317685
20×20	Ta021-022	0.560930	0	0.560938
50×5	Ta031-032	0.384430	0	0.55526
50×10	Ta041-042	1.208755	0	1.215738
50×20	Ta051-052	0.875744	0	0.902247
100×5	Ta061-062	0.164407	0	0.333729
100×10	Ta071-072	0.867567	0	0.226151
100×20	Ta081-082	0.381126	0	0.336008
200×10	Ta091-092	0.311720	0.073584	0.108946
200×20	Ta101-102	0.107240	0	0.250141
500×20	Ta111-112	0.634800	0	0.786134

Table III presents the simulation data to demonstrate the superiority of our approach under different machine breakdown scenarios. The table starts with a column of problem size (number of jobs × number of machines). The next column represents the problem instance group and each group contains the first two problem instances from Taillard's benchmark [22]. The column headed with 1st machine under the heading of Single Breakdown shows the average of API (average of 10 independent runs) of the single breakdown scenario in the first machine. The following column, headed last machine, represents the average API of the single breakdowns on the last machine. The final column shows the average of API for the two machine breakdown case. In case of early stage breakdown and two machine breakdown, our proposed algorithm clearly dominates the traditional RST.

From the experimental results we have observed that the location of breakdown within the early stage influences the makespan of the revised schedule.

TABLE IV. API COMPARISION OF RHT AND RSV WITH MACHINE UNAVAILABILITY SCENARIOS

Problem	Problem Instance group	Single Unavailability		Multiple Unavailability
		1st machine	Last machine	Combined
20×5	Ta001-002	11.51140	7.499140	16.22089
20×10	Ta011-012	23.34450	4.709435	23.07040
20×20	Ta021-022	24.16805	0	24.16805
50×5	Ta031-032	16.01200	2.669505	18.08260
50×10	Ta041-042	20.66265	2.633590	20.34855
50×20	Ta051-052	8.746920	0.016144	8.893040
100×5	Ta061-062	11.61425	2.124630	13.36810
100×10	Ta071-072	10.23710	2.320190	11.23655
100×20	Ta081-082	9.820530	0.595439	9.87410
200×10	Ta091-092	4.627278	0.241446	4.693970
200×20	Ta101-102	8.14650	0.146860	6.587410
500×20	Ta111-112	3.48120	0.214610	3.95740

Table IV presents the average API of MA-RSV (MUN) over RST with both single and multiple machine unavailability events. For a better understanding of the advantage of knowing disruption information in advance, we ran the experiments using the same interruption scenarios for both MA-RSV (MBD) and MA-RSV (MUN). The results show that when machine interruption information is known in advance, the effect is very high compared with machine breakdown. A sample comparison has been given in the Appendix.

VI. CONCLUSION

In almost all research on PFSPs, schedules were generated with the assumption that during the production, there will be no interruption. This paper has considered a PFSP with known and unknown process interruption. We have solved 24 benchmark problems and we use different distributions to generate interruption scenarios. From the experimental results, we have seen that the machine breakdown has more impact on a production schedule, than an unavailability event, as the information is known only after the breakdown. Another observation is that if a breakdown occurs at the early stage in the schedule, the schedule can be improved by rescheduling the effected tasks. As an initial attempt, we study the effect of machine related disruptions in first machine and in last machine. From this study it can be observed that the location of interruptions within the scheduling horizon, influence the quality of the revised schedule. In future, the proposed algorithms can be extended to study the effect of machine disruptions at any machine. Besides, some other aspects of disruptions, such as change in processing time, delay in machine set up, change in job priority and so on can also be introduced in conjunction with our proposed approach.

APPENDIX

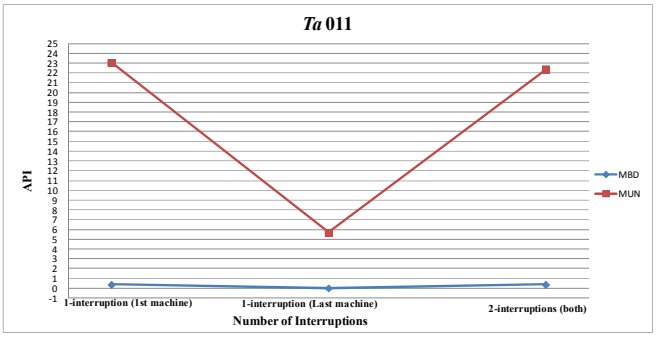

Fig. 5 (a). Comparing machine breakdown with respect to machine unavailability for problem *Ta* 011

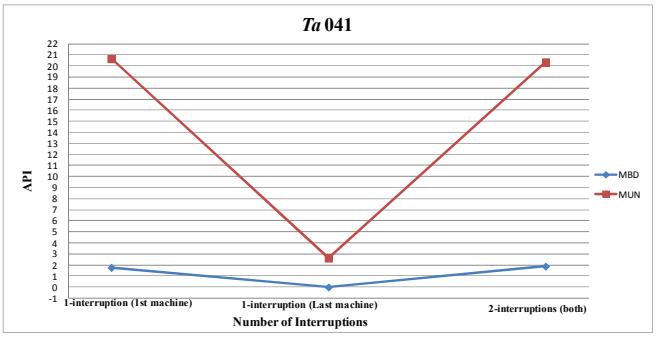

Fig. 5 (b). Comparing machine breakdown with respect to machine unavailability for problem *Ta* 041

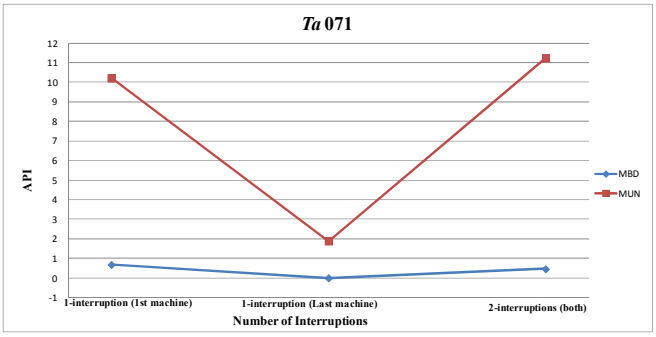

Fig. 5 (c). Comparing machine breakdown with respect to machine unavailability for problem *Ta* 071

The results of the comparative study between MA-RSV

(MBD) and MA-RSV (MUN) for three problem instances (*Ta* 011, *Ta* 041, and *Ta* 101) have been plotted in Figure 5. For each test problem, the comparison has been carried out considering single (either 1-interruption in the first machine or in the last machine) and double interruptions in both machines (first and last machine) with the same interruption information (star time and duration of interruptions). The only difference is, either interruption is known after it happens or known in advance. In both cases, MA-RSV (MBD) and MA-RSV (MUN) have been compared with Right Shifting heuristics and the corresponding API has been calculated. From the graph, it is clear that, at similar interruption scenario (like single interruption in the first machine with same start time and disruption durations) API under Machine Unavailability is much higher than API with Machine Breakdown scenario. So, it is clear that scheduling with known interruption information has greater scope of minimizing the effects of interruptions.

The graph also demonstrates that, with respect to scope of minimizing the effect of single breakdowns in the last machine, the rescheduling approach is more effective in minimizing the delay in single breakdown at the first machine or combination of two breakdowns. For Machine Unavailability, the similar pattern can be found.

REFERENCES

[1] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly,* vol. 1, pp. 61-68, 1954.

[2] M. R. Garey, D. S. Johnson, and R. Sethi, ""The complexity of flowshop and jobshop scheduling"," *Mathematics of Operations Research,* vol. 1, pp. 117-129, 1976.

[3] M. Nawaz, E. E. Enscore, and I. Ham, "A Heuristic Algorithm for the M-Machine, N-Job Flowshop Sequencing Problem," *Omega-International Journal of Management Science,* vol. 11, pp. 91-95, 1983.

[4] R. Ruiz and C. Maroto, "A comprehensive review and evaluation of permutation flowshop heuristics," *European Journal of Operational Research,* vol. 165, pp. 479-494, 2005.

[5] G. I. Zobolas, C. D. Tarantilis, and G. Ioannou, "Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm," *Computers & Operations Research,* vol. 36, pp. 1249-1267, Apr 2009.

[6] C. Rajendran and H. Ziegler, "Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs," *European Journal of Operational Research,* vol. 155, pp. 426-438, Jun 1 2004.

[7] M. F. Tasgetiren, Y. C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *European Journal of Operational Research,* vol. 177, pp. 1930-1947, Mar 16 2007.

[8] I. H. Osman and C. N. Potts, "Simulated Annealing for Permutation Flowshop Scheduling," *Omega-International Journal of Management Science,* vol. 17, pp. 551-557, 1989.

[9] J. Grabowski and M. Wodecki, "A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion," *Computers & Operations Research,* vol. 31, pp. 1891-1909, Sep 2004.

[10] R. Ruiz, C. Maroto, and J. Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problem," *Omega-International Journal of Management Science,* vol. 34, pp. 461-476, Oct 2006.

[11] H. F. Rahman, R. A. Sarker, and D. L. Essam, "A Memetic Algorithm for Permutation Flow Shop Problems," in *IEEE Congress on Evolutionary Computation*, Cancun, Maxico, 2013.

[12] S. A. Fahmy, S. Balakrishnan, and T. Y. ElMekkawy, "A generic deadlock-free reactive scheduling approach," *International Journal of Production Research,* vol. 47, pp. 5657-5676, 2009.

[13] V. Subramaniam, A. S. Raheja, and K. R. B. Reddy, "Reactive repair tool for job shop schedules," *International Journal of Production Research,* vol. 43, pp. 1-23, Jan 1 2005.

[14] R. Sarker, M. Omar, S. M. K. Hasan, and D. Essam, "Hybrid Evolutionary Algorithm for job scheduling under machine maintenance," *Applied Soft Computing,* vol. 13, pp. 1440-1447, Mar 2013.

[15] S. M. K. Hasan, R. Sarker, and D. Essam, "Genetic algorithm for job-shop scheduling with machine unavailability and breakdowns," *International Journal of Production Research,* vol. 49, pp. 4999-5015, 2011.

[16] J. H. Blackstone, D. T. Phillips, and G. L. Hogg, "A State-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations," *International Journal of Production Research,* vol. 20, pp. 27-45, 1982.

[17] R. J. Abumaizar and J. A. Svestka, "Rescheduling job shops under random disruptions," *International Journal of Production Research,* vol. 35, pp. 2065-2082, Jul 1997.

[18] S. Q. Liu, H. L. Ong, and K. M. Ng, "Metaheuristics for minimizing the makespan of the dynamic shop scheduling problem," *Advances in Engineering Software,* vol. 36, pp. 199-205, Mar 2005.

[19] S. D. Wu, R. H. Storer, and P. C. Chang, "One-Machine Rescheduling Heuristics with Efficiency and Stability as Criteria," *Computers & Operations Research,* vol. 20, pp. 1-14, Jan 1993.

[20] K. N. McKay, J. A. Buzacott, and F. R. Safayeni, "The scheduler's knowledge of uncertainty: The missing link," *Knowledge based production management systems,* pp. 171-189, 1989.

[21] E. Taillard, "Some Efficient Heuristic Methods for the Flow-Shop Sequencing Problem," *European Journal of Operational Research,* vol. 47, pp. 65-74, Jul 5 1990.

[22] E. Taillard, "Benchmarks for Basic Scheduling Problems," *European Journal of Operational Research,* vol. 64, pp. 278-285, Jan 22 1993.