Applied Mathematics and Sciences: An International Journal (MathSJ ), Vol. 1, No. 1, May 2014

# AN EFFICIENT HEURISTIC ALGORITHM FOR FLEXIBLE JOB SHOP SCHEDULING WITH MAINTENANCE CONSTRAINTS

Mohsen Ziaee*

Department of Industrial Engineering, University of Bojnord, 94531-55111 Bojnord, Iran

## *Abstract*

*This paper deals with the flexible job shop scheduling problem with the preventive maintenance constraints where the objectives are to minimize the overall completion time (makespan), the total workload of machines and the workload of the most loaded machine. A fast heuristic algorithm based on a constructive procedure is developed to solve the problem in very short time. The algorithm is tested on the benchmark instances from the literature in order to evaluate its performance. Computational results show that, the proposed heuristic method is computationally efficient and promising for practical problems.*

## *Keywords*

*Scheduling, Multi-Objective Flexible Job Shop, Preventive Maintenance, Heuristic, Local Search.*

## 1.Introduction

The job shop scheduling problem (JSP) is one of the most popular scheduling problems and has attracted many researchers due to both its practical importance and its complexity [1]. In the $n \times m$ classical JSP, a set of $n$ jobs have to be processed on a group of $m$ machines, where the processing of each job $i$ consists of $J_i$ operations performed on these machines. Each job has a processing order on the machines which is fixed and known in advance, i.e., each operation has to be performed on a given machine. The processing times of all operations are also fixed and known. Each machine can process at most one operation at a time, and the operations are processed on the machines without interruption [2,3]. A typical performance indicator for the JSP is the makespan, i.e., the time needed to complete all the jobs.

The flexible job shop scheduling problem (FJSP) is a generalization of the classical JSP, in which each operation is allowed to be processed by any among set of candidate machines, instead of a given machine; and thus, the scheduling problem is to choose for each operation, a machine and a starting time at which the operation must be processed. The FJSP is more difficult than the classical JSP because it contains an additional problem which is to determine the job routes, or to assign the operations to the machines. This problem is known to be strongly NP-hard even if each job has at most three operations and there are two machines [4].

The FJSP with PM constraints is to assign each operation to an appropriate machine out of a set

of machines capable of executing it, and to sequence the job operations and PM operations on each machine, in order to minimize one or more criteria. This paper presents a solution method to solve this problem in order to minimize the following three objectives:

(1) The maximum completion time of the machines, i.e. the makespan ($C_{max}$).
(2) The total workload of the machines, which represents the total working time of all machines ($W_T$). This objective is of interest if machines differ with respect to the efficiency.
(3) The maximal machine workload, i.e., the maximum working time spent on any machine ($W_{max}$). This objective is used to prevent assignment of too much work to a single machine and to keep the balance of work distribution over the machines.

In this article, the weighted sum of the above three objectives is taken as the objective function. This approach, i.e. the weighted sum, in dealing with the multi-objective optimization, is easy for decision makers to understand, convenient for developers to implement, and available to modify the weights for satisfying the requirements of decision makers. In this study, the PM periods are also included in the above three criteria, since the PM periods are mandatory and the scheduler has to schedule them along with the jobs to be processed. The objective function of the studied problem is therefore computed as (1):

$$\begin{aligned} &\textit{Minimize } Obj = W_1 \times C_{max} + W_2 \times W_T + W_3 \times W_{max} \\ &\text{such that: } W_1 + W_2 + W_3 = 1; \quad W_1, W_2, W_3 \geq 0. \end{aligned} \tag{1}$$

$W_1$, $W_2$, and $W_3$ represent the weight coefficients of the three objectives.

The FJSP with PM constraints is strongly NP-hard, since the problem without PM scheduling is already strongly NP-hard [4]. Therefore, as it can be seen in the literature review section, approximate algorithms, mainly metaheuristics, have been used to solve the problem.

In this study, a simple and easily extendable heuristic algorithm based on a constructive procedure is presented to solve the FJSP with PM constraints (section 3). The main purpose is to produce reasonable and applicable schedules very quickly. It can also be used to improve the quality of the initial feasible solution of metaheuristics applied to solve the problem, since the choice of a good initial solution is an important aspect of the performance of algorithms in terms of computing time and solution quality [5,6,7]. In order to evaluate the performance of the proposed heuristic, it is implemented using several benchmark problems and the results of the computational experiments are presented (section 4). The results show that our novel method can obtain good solutions in very short time. Concluding remarks are given in the last section.

Assumptions considered in this paper are as follows:

(1) There is only one PM operation on each machine during the planning horizon. Each PM operation has to be performed within a predefined time window, and its duration is determined in advance.
(2) Jobs are independent of each other.
(3) Machines are independent of each other.
(4) Setup and transportation times are negligible.
(5) Preemption is not allowed, i.e. a started operation (either operation of a job or a PM operation) can not be interrupted during its processing.
(6) Each machine can process at most one operation (either job operation or PM operation) at the same time.
(7) An operation can not be performed by more than one machine at the same time.
(8) All jobs have equal priorities.

(9) Machines never break down and are available throughout the scheduling period.
(10) All jobs are available at time zero.

The notations used throughout the paper are as follows:

***n***: number of jobs,
***m*:** number of machines,
***i,z:*** index of jobs; $i,z=1,\ldots,n$,
$J_i$**:** number of operations of job $i$,
***maxJ:*** maximum number of operations per job (i.e., $maxJ=\max_i J_i$ ),
***j:*** index of operations; $j=1,\ldots, J_i$,
***k,y*:** index of machines; $k,y=1,\ldots,m$,
$t_{ijy}$: processing time of operation $j$ of job $i$ on machine $y$,
$C_{ij}$**:** completion time of operation $j$ of job $i$,
$W_T$**:** total workload of machines (the workload of each machine is defined by the sum of processing times of all operations assigned to it),
$W_{max}$**:** critical machine workload, i.e. the workload of the most loaded machine,
$dur_y$**:** duration of the preventive maintenance task of machine $y$,
$(tE_y , tL_y)$**:** time window associated with the PM task on machine $y$, where $tE_y$ is the earliest possible completion time, and $tL_y$ is the latest possible completion time.

## 2.Literature Review

The vast majority of papers dealing with scheduling problems assume that the machines are continuously available for processing during the whole planning horizon. However, this assumption may not be true in real industrial settings, since the machines may become unavailable during the planning period for many reasons such as unforeseen breakdowns (stochastic unavailability [8]), or due to the use of the equipments for planned activities such as preventive maintenance (PM) tasks that conflict with scheduling decisions (i.e. deterministic unavailability, in which the periods of unavailability are known in advance).

Recently, many researchers began to study the scheduling problems with PM constraints, because PM is considered as a common reason for machine unavailability. PM is used by manufacturing industries for following reasons: PM can reduce the probability of unforeseen breakdowns; in manufacturing systems that use PM, if a machine breaks down, then its unavailability time interval is expected to be significantly reduced; and therefore PM can restore the reliability of machines and improve the machine utilization ratio. This paper deals with the FJSP with PM constraints and assumes that the starting time of PM operations is not known in advance and must be determined during the production scheduling process. However, each PM operation has to be executed within a given time window. These constraints are referred to as non-fixed availability constraints in the literature. In fixed availability constraints, each PM operation is started at a fixed time point which is predetermined by maintenance planning system. We also suppose that there is only one PM operation on each machine during the planning horizon. This assumption brings the problem closer to the real manufacturing situations [9].

Scheduling problems under the machine availability constraints have recently been investigated in numerous papers. Schmidt [10] presents a survey of existing methods for solving deterministic scheduling problems with availability constraints, as well as complexity results; and most recent survey of scheduling with deterministic machine availability constraints can be seen in reference

[11]. However, for the FJSP, there are only a few papers that deal with the FJSP under resource constraints. Dalfard and Mohammadi [12] propose two metaheuristics (a hybrid genetic algorithm and a simulated annealing algorithm) for the multi-objective FJSP with parallel machines, maintenance costs, jobs due dates and jobs release times to minimize the mean tardiness, the makespan and the mean flowtime. As there is no similar work in the literature, they compare the solutions of these metaheuristic methods with those obtained by solving a mathematical model using the software LINGO. Gao et al. [9] consider the FJSP with non-fixed availability constraints where each machine is subject to an arbitrary number of PM tasks, and present a hybrid genetic algorithm (hGA) to solve the problem with a multiobjective function including makespan, total machine workload and the workload of the most loaded machine. Chan et al. [13] develop a heuristic based on genetic algorithm (GA), namely iterative GA (IGA), for solving the bicriteria FJSP under resource constraints. The objectives considered are the minimization of makespan and machine idle cost. Zribi et al. [14] investigate the multi-purpose machine (MPM) job shop scheduling problem with fixed machine availability constraints, and apply a GA to solve the problem. Rajkumar et al. [15] present a greedy randomized adaptive search procedure (GRASP) to solve the FJSP under non-fixed availability constraints and with the same objectives as those considered by Gao et al. [9]. Moradi et al. [16] solve the FJSP with PM under two objectives: the minimization of makespan for the production part and the minimization of system unavailability for the maintenance part. Vilcot and Billaut [17] study a general job shop scheduling problem with multiple constraints, coming from printing and boarding industry. They present an algorithm based on tabu search and GA for minimizing the makespan and the maximum lateness. Chan et al. [18] consider the distributed flexible manufacturing system (FMS) scheduling problem subject to machine maintenance constraints, in which the maintenance time is related to the machine age. The presented method also covers the FJSP with maintenance activities. Wang and Yu [19] present a heuristic based on the filtered beam search (FBS) algorithm to solve the FJSP with the non-fixed and fixed machine availability constraints due to the PM.

## 3.Proposed Heuristic Approach

In this section, we present a heuristic method to solve the problem. This approach is motivated by the idea of developing a constructive heuristic that considers simultaneously several factors affecting the solution quality and intelligently balances their effects in the process of schedule generation, and the observation that it could lead to good results in some preliminary computational experiments on a wide range of difficult scheduling problems. This algorithm has a simple structure, is easy to implement, and requires very little computational effort which makes it preferable over other more complex and time-consuming approaches. Some notations that will be used in the algorithm are defined as follows:

***$A_{ij}$:*** set of machines which are capable to execute operation $j$ of job $i$,
***$N_{ij}$:*** number of members of the set $A_{ij}$,
***$s'_{ij}$:*** mean processing time of operation $j$ of job $i$ over the machines belonging to the set $A_{ij}$ (i.e., $s'_{ij} = (\sum_{y \in A_{ij}} t_{ijy}) / N_{ij}$ ),

***$sj_i$:*** total mean processing time of job $i$ (i.e., $sj_i = \sum_{j=1}^{J_i} s'_{ij}$ ),

***sk$_y$:*** total weighted processing time on machine $y$ which is calculated as: $sk_y = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ if\ y \in A_{ij}}}^{J_i} \frac{s'_{ij}}{N_{ij}}$,

***M:*** a large number.

An outline of the proposed heuristic algorithm is given in Fig. 1 and the pseudocode of the algorithm is shown in Fig. 2. Some other notations used in these two figures will be defined later.

---

until all operations of all jobs are scheduled, repeat
{

- For all $i$, $j$, $k$ (such that: 1. $j \leq J_i$, 2. $j$=1 or ($j$-1)th operation of job $i$ is already scheduled, and 3. $j$th operation of job $i$ is an unscheduled operation and machine $k$ is capable of processing this operation), calculate the value of *TC*.
- For all unscheduled PM operations, calculate the value of *TC*.
- Select the operation (either job operation or PM operation) with minimum *TC* and schedule it on the last position of current partial sequence on the corresponding machine.

}

---

Fig. 1. General outline of the proposed heuristic algorithm

---

***Initialization:***

- **Sort the jobs in increasing order of their *sj$_i$* and call the resulting set: *i_sort*. Let *i_sort$_z$* be *z*th job of the list *i_sort*.**
- **Sort the machines in increasing order of their *sk$_y$* and call the resulting set: *k_sort*. Let *k_sort$_y$* be *y*th machine of the list *k_sort*.**

***Constructive Algorithm:***

```
for x1:=L_x1 to U_x1 do
for x2:=L_x2 to U_x2 do
⋮
for x6:=L_x6 to U_x6 do
for x7:=0 to 1 do
{
     % Beginning of a schedule generation

     until  all job operations and all PM operations are
scheduled, repeat the following steps:
     {
          for j:= 1 to maxJ do
          {
               Set TC*:=M
```

```
for i′:=1 to n do
{
     Set i:=x7.(i_sorti′)+(1-x7).(i_sort(n-i′+1)),
     Set b:=0,

     if (   1. j ≤ Ji, and
          2. j=1 or (j-1)th operation of job i
          is already scheduled, and
          3. jth operation of job i is an
          unscheduled operation) then
     {
          for k′:=1 to m do
          {
               Set k:= k_sort(m-k′+1),

               if (machine k is capable of
               processing jth operation of job
               i) then
               {
                    if ( 1. PM of machine
                         k is already
                         scheduled; or
                         2.PM task of machine
                    k is not already
                    scheduled, and: Cij <= tLk -
                    durk,      (such      that,
                         Cij=max (Cmaxk,  Ci,j-
                    1)+tijk )
                        )      then
                    {
                         Set TC:=∑_{r=1}^{5} wr.xr.Cr
                         if TC<TC* then
                         {
                              Set TC*:= TC
                              Set z:=i
                              Set y:=k
                              Set b:=0
                         }
                    }
               }

if ( PM operation of machine k is an unscheduled operation) then
                              {
                                   Set TC:= x6.{∑_{r=1}^{5} wr.xr.C′r}
                                   if TC<TC* then
                                   {
                                        Set TC*:= TC
```

```
                                        Set b:=1
                                        Set y:=k
                                    }
                                }
                            }
                        }
                    }

                    if TC*<M then
                    {
                        if b:=1 then schedule the PM task of machine y on the last position of the current partial sequence on this machine, and set its completion time as max (Cmax_y+ dur_y, tE_y).

                        else schedule jth operation of job z on the last position of the current partial sequence on machine y to finish at time c_zj.
                    }
                }
            }

            % End of a schedule generation

            If the objective value of the obtained sequence (Obj) is less than the best objective value obtained so far (Obj*), then set Obj*:=Obj and x_r*=x_r (r=1,2,…, 7) corresponding to Obj*.
}
```

Fig. 2. Pseudocode of the proposed heuristic method

In the above algorithm, each unscheduled job operation ($i$, $j$) (operation $j$ of job $i$) to be scheduled on machine $y$ is evaluated by the following criterion (2):

$$TC = \sum_{r=1}^{5} w_r . x_r . C_r , \qquad (2)$$

Also, unscheduled PM operation of machine $y$ is evaluated by the following criterion (3):

$$TC = x_6 . \left\{ \sum_{r=1}^{5} w_r . x_r . C'_r \right\}, \qquad (3)$$

and the unscheduled operation (either job operation or maintenance operation) with minimum $TC$ is selected for scheduling.

$C_1$ to $C_5$ and $C'_1$ to $C'_5$ are calculated as (4) to (13):

$C_1 = max\ (Cmax_y, C_{i,j-1}) + t_{ijy}$ (4)

$C_2 = max\ (0,( C_{i,j-1} - Cmax_y))$ (5)

$C_3 = t_{ijy}$ (6)

$C_4 = w'_T + t_{ijy}$ (7)

$C_5 = w'_y + t_{ijy}$ (8)

$C'_1 = max\ (Cmax_y + dur_y, tE_y)$ (9)

$C'_2 = max\ (0,(tE_y - dur_y - Cmax_y))$ (10)

$C'_3 = dur_y$ (11)

$C'_4 = w'_T + dur_y$ (12)

$C'_5 = w'_y + dur_y$ (13)

*TC* is weighted sum of some criteria which are established based on the factors affecting the objective function value. Minimization of TC in the process of schedule generation leads to improvement in solution quality. $w_r$ ($r$=1,2,…,5) are constants and $x_r$ ($r$=1,2,…,6) are integer variables used to increase the flexibility and effectiveness of criterion *TC* and have a significant impact on the performance of the algorithm. The constant weights ($w_r$) are preliminary estimated weights assigned to criteria according to their importance, and the coefficients $x_r$ are variables bounded in a given range and used to refine the TC. $Cmax_y$ is the maximum completion time across all the operations scheduled on machine *y;* i.e., the completion time of last operation (either job operation or PM operation) scheduled on machine *y*. $w'_T$ is the total workload of machines for the partial schedule. $w'_y$ is the workload of machine *y* for the partial schedule. $C_1$ and $C'_1$ are applied to decrease $Cmax_y$; $C_2$ and $C'_2$ are used to decrease idle times; clearly, both these objectives ($Cmax_y$ and idle times) affect the main objective function, i.e. $C_{max}$. The values of other objective functions, i.e. $W_T$ and $W_{max}$, are directly affected by ($C_4$ and $C'_4$) and ($C_5$ and $C'_5$), respectively. For assigning operations to a machine, their processing time are also taken into account by $C_3$ and $C'_3$.

Other notations used in the pseudocode of the proposed heuristic are as follows:

***TC*:*** denotes the best value of *TC*. After each operation is scheduled, *TC** is reset to M.

***L_x$_r$ (r=1,2,…,6):*** lower limit of $x_r$.

***U_x$_r$ (r=1,2,…,6):*** upper limit of $x_r$.

***b:*** a binary variable taking value 1 if PM task of a machine is selected for scheduling, and 0 if an operation of a job is selected for scheduling.

As it can be seen in Fig. 2, the algorithm first sorts the jobs in increasing (decreasing) order of their $sj_i$ and then uses this order for evaluating their operations. Therefore, if two unscheduled operations belonging to two different jobs have the same value of *TC*, then according to this sorting of the jobs, the operation of job with smaller (greater) $sj_i$ is selected for scheduling sooner than the other operation. Binary variable $x_7$ is applied for setting the order of the sorting (i.e. either increasing order or decreasing order), it takes a value of 1 for increasing order and 0 for decreasing one. Similarly, the algorithm first sorts the machines in decreasing order of their $sk_y$ and then uses this order to evaluate assigning the operations to each of them. In our preliminary computational experiments, we used these sortings of the jobs and machines instead of randomly selecting them, and interestingly observed that these sortings can lead to better solutions. Specially, the results showed that in most cases, the sorting of the machines in decreasing order of their $sk_y$ leads to better solutions in comparison with increasing order. It is because the machines with larger $sk_y$ which are firstly selected for scheduling have more sensibility and effect on the objective value. In other words, the schedule of these machines determines the performance of overall schedule of the problem. Therefore, we have used only decreasing order of them in the computational experiments. $x_r^*(r=1,2,\ldots,7)$ are the best values of variables $x_r$ (i.e. the values

corresponding to the best solutions). Indeed, for various values of $x_r$ ($r$=1,2,…,7), the algorithm of Fig. 1 is run and a complete schedule is generated. Among all these schedules, the one with minimum *Obj* is reported as the final solution. The values of variables $x_r$ for this best solution are also reported and denoted by $x_r^*$ (see Table 2). This best schedule obtained from the heuristic is next improved by a shift neighborhood based local search procedure. The pseudocode of this local search is shown in Appendix.

As mentioned earlier, the evaluation of the operations for scheduling them is done using the criterion *TC*, i.e. the unscheduled operation with minimum *TC* is selected for scheduling.

## 4.Computational Results

This section describes the computational experiments conducted in order to evaluate the performance of the proposed heuristic method. First, some preliminary experiments have been conducted for the parameter settings. Regarding the test on various values for the parameters of the algorithm and considering the computational results, we used the settings of Table 1 for benchmarking the presented algorithm.

Table 1. Parameter settings for the heuristic

| Parameter | Value | Parameter | Value | Parameter | Value |
|---|---|---|---|---|---|
| $L\_x_1$ | 0 | $U\_x_1$ | 2 | $w_1$ | 1 |
| $L\_x_2$ | 0 | $U\_x_2$ | 2 | $w_2$ | 1 |
| $L\_x_3$ | 0 | $U\_x_3$ | 2 | $w_3$ | 1 |
| $L\_x_4$ | 0 | $U\_x_4$ | 2 | $w_4$ | 0.5 |
| $L\_x_5$ | 0 | $U\_x_5$ | 2 | $w_5$ | 0.2 |
| $L\_x_6$ | 0 | $U\_x_6$ | 2 | | |

The algorithm was coded in C language and run on a Pentium IV, 2.2 GHz and 2.0 GB RAM PC. The benchmark problems used were the set of 4 instances presented by Kacem et al. [20,21,22] and extended by Gao et al. [9] and Rajkumar et al. [15] to problems involving maintenance constraints. All these instances have exactly one PM activity on each machine in the planning horizon. Table 2 shows a comparison of the results of our algorithm with those of two recently published algorithms: the hybrid genetic algorithm (hGA) presented by Gao et al. [9] and the greedy randomized adaptive search procedure (GRASP) developed by Rajkumar et al. [15]. The results are obtained under two objective functions: $Obj1=0.3\times C_{max}+0.5\times W_T+0.2\times W_{max}$, and $Obj2=0.2\times C_{max}+0.5\times W_T+0.3\times W_{max}$. *Weights* denotes the weights of the objectives. *Name* and *Size* refer to the name of each instance and its size in terms of the number of jobs, machines and operations, respectively. *Cmax*, $W_T$ and $W_{max}$ stand for the makespan, total workload and maximum workload, respectively. *RPD* is the relative percentage deviation and calculated as (14):

$$RPD = \frac{Obj_{alg} - Obj^*}{Obj^*} \times 100, \qquad (14)$$

where $Obj_{alg}$ is the objective function (*Obj*) value generated by the algorithm and $Obj^*$ is the best value of *Obj* obtained from the three algorithms. *Sol.1* and *Sol.2* show the results obtained by the heuristic algorithm and local search procedure, respectively. *Time(s)* indicates the computational time to solve each instance by the heuristic (including the time spent on the local search) in seconds. The best values of variables $x_r$ (i.e. $x_r^*$), $r$=1,2,…,7; are also reported in Table 2. Symbol

'–' denotes that the result was not presented in the given reference. As it can be seen in the table, for the results corresponding to *Obj2*, i.e. cases with larger weights for $W_{max}$, the value of $x_7$ is equal to 0.25, i.e. near to zero. It means that the sorting of the jobs in decreasing order of their $sj_i$ leads to better solutions in comparison with increasing order. It is because the jobs with larger $sj_i$ which are firstly selected for scheduling have more impact on $W_{max}$. The results of Table 2 also show that the value of $x_6$ is equal to zero in almost all instances, meaning that TC for PM operations is set to zero, i.e. PM operations are started at earliest possible time. This intuitively gives more opportunities to job operations to be inserted in good positions of the overall schedule. The average value of each variable $x_r$, $r$=1,2,…,5 can be considered as the relative effect of the corresponding criterion on the quality of solutions. Of course, as it can be seen in the table, the values of each variable $x_r$, $r$=1,2,…,5 have relatively high variance, meaning that they are strongly dependent on the specifications of problem instance under consideration and on the values of other variables $x_r$. The proposed algorithm selects for each instance, best combination of $x_r$ values leading to best result.

In Table 2, an interesting observation is that the proposed heuristic is better than the other two metaheuristic algorithms in terms of the average RPD, considering that it is very fast and needs only 0.5 sec. on average. Figures 3 and 4 show a graphical comparison of the RPD of the three methods for *Obj1* and *Obj2*, respectively.

Table 2. Computational results for benchmark instances

| | | | Heuristic | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Sol.1 | | | | | | | | | | | Sol.2 | | | | | |
| Weights | Name | Size | Cmax | Wt | Wmax | Obj1 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | Cmax | Wt | Wmax | Obj1 | RPD | Time(s) |
| w1=0.3 w2=0.5 w3=0.2 | 4-5-nfa | 4,5,12 | 15 | 40 | 9 | 26.3 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 13 | 40 | 9 | 25.7 | 0 | 0.05 |
| | 8-8-nfa | 8,8,27 | 31 | 103 | 16 | 64 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 20 | 103 | 16 | 60.7 | 0.998 | 0.22 |
| | 10-10-nfa | 10,10,30 | 11 | 60 | 8 | 34.9 | 2 | 1 | 1 | 0 | 2 | 0 | 1 | 9 | 60 | 8 | 34.3 | 0.587 | 0.38 |
| | 15-10-nfa | 15,10,56 | 20 | 108 | 13 | 62.6 | 2 | 1 | 2 | 0 | 1 | 1 | 1 | 15 | 104 | 14 | 59.3 | 0 | 1.45 |
| | Average | | | | | | 1.25 | 0.75 | 1 | 0.75 | 1.25 | 0.25 | 0.5 | | | | | 0.396 | 0.52 |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| Weights | Name | Size | Cmax | Wt | Wmax | Obj2 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | Cmax | Wt | Wmax | Obj2 | RPD | Time(s) |
| w1=0.2 w2=0.5 w3=0.3 | 4-5-nfa | 4,5,12 | 15 | 40 | 9 | 25.7 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 13 | 40 | 9 | 25.3 | 0 | 0.05 |
| | 8-8-nfa | 8,8,27 | 31 | 103 | 16 | 62.5 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 20 | 103 | 16 | 60.3 | 0.668 | 0.22 |
| | 10-10-nfa | 10,10,30 | 11 | 60 | 8 | 34.6 | 2 | 1 | 1 | 0 | 2 | 0 | 1 | 9 | 60 | 8 | 34.2 | 0.885 | 0.36 |
| | 15-10-nfa | 15,10,56 | 27 | 104 | 14 | 61.6 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 14 | 105 | 12 | 58.9 | 0 | 1.09 |
| | Average | | | | | | 0.75 | 0.5 | 0.5 | 1.25 | 1.5 | 0 | 0.25 | | | | | 0.388 | 0.43 |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | hGA | | | | | | GRASP | | | | | | | | | | |
| Weights | Name | Size | Cmax | Wt | Wmax | Obj1 | RPD | | Cmax | Wt | Wmax | Obj1 | RPD | | | | | | |
| w1=0.3 w2=0.5 w3=0.2 | 4-5-nfa | 4,5,12 | – | – | – | – | – | | 16 | 40 | 9 | 26.6 | 3.502 | | | | | | |
| | 8-8-nfa | 8,8,27 | 17 | 105 | 15 | 60.6 | 0.832 | | 18 | 103 | 16 | 60.1 | 0 | | | | | | |
| | 10-10-nfa | 10,10,30 | 8 | 61 | 7 | 34.3 | 0.587 | | 9 | 60 | 7 | 34.1 | 0 | | | | | | |
| | 15-10-nfa | 15,10,56 | 12 | 109 | 12 | 60.5 | 2.024 | | 16 | 107 | 13 | 60.9 | 2.698 | | | | | | |
| | Average | | | | | | 1.147 | | | | | | 1.55 | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | |
| Weights | Name | Size | Cmax | Wt | Wmax | Obj2 | RPD | | Cmax | Wt | Wmax | Obj2 | RPD | | | | | | |
| w1=0.2 w2=0.5 w3=0.3 | 4-5-nfa | 4,5,12 | – | – | – | – | – | | 16 | 40 | 9 | 25.9 | 2.372 | | | | | | |
| | 8-8-nfa | 8,8,27 | 17 | 105 | 15 | 60.4 | 0.835 | | 18 | 103 | 16 | 59.9 | 0 | | | | | | |
| | 10-10-nfa | 10,10,30 | 8 | 61 | 7 | 34.2 | 0.885 | | 9 | 60 | 7 | 33.9 | 0 | | | | | | |
| | 15-10-nfa | 15,10,56 | 12 | 109 | 12 | 60.5 | 2.716 | | 16 | 107 | 13 | 60.3 | 2.377 | | | | | | |
| | Average | | | | | | 1.479 | | | | | | 1.187 | | | | | | |

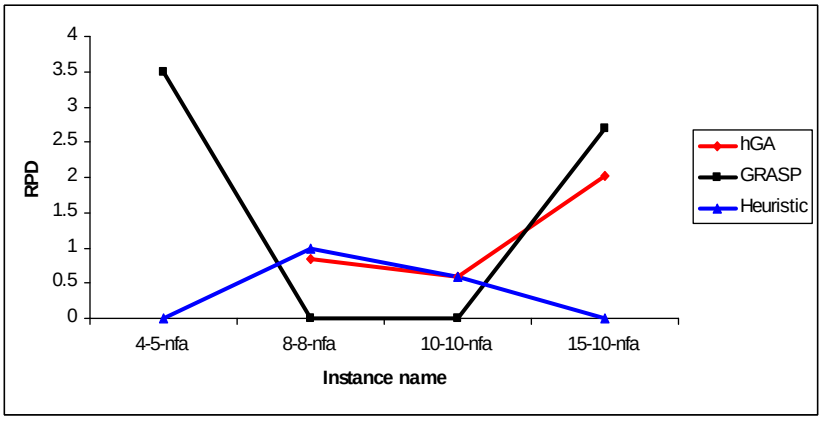


Fig. 3 Comparison of the three methods for *Obj1* (i.e. $W_1$=0.3, $W_2$=0.5, $W_3$=0.2)

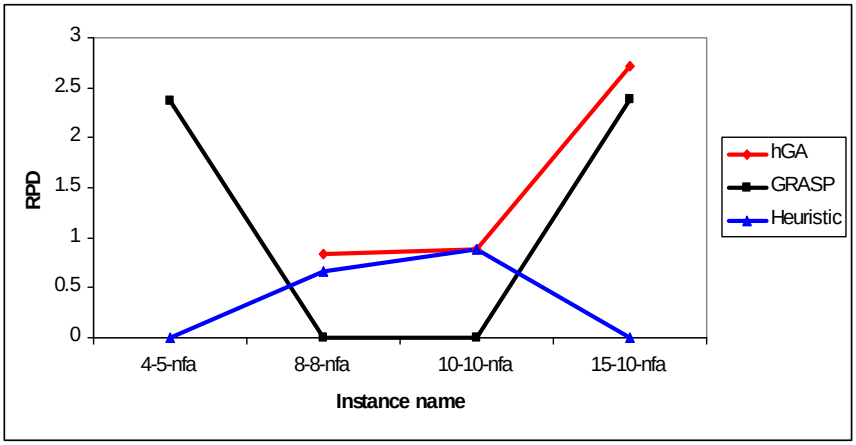


Fig. 4 Comparison of the three methods for *Obj2* (i.e. $W_1$=0.2, $W_2$=0.5, $W_3$=0.3)

## 5.Conclusion

This paper investigates the flexible job shop scheduling problem with preventive maintenance constraints. The objective is to minimize the makespan, the total workload of machines and the workload of most loaded machine. The main purpose is to produce reasonable schedules very quickly. A simple and easily extendable heuristic based on a constructive procedure is presented. The proposed approach uses an accurate, relatively comprehensive and flexible criterion for scheduling job operations and PM operations and constructing a feasible high-quality solution. In this criterion, several factors affecting the quality of solutions are used and to each of these factors, a variable weight is assigned. By setting different values to these variable weights, different solutions are generated and evaluated. The algorithm is tested on benchmark instances from the literature in order to evaluate its performance. The computational results show that the proposed approach can yield very good solutions with very little computational time. Since the presented method is a heuristic, its results cannot be compared in a meaningful way with those of the methods evaluated as they are metaheuristic based algorithms. However, the computational results show that the proposed heuristic outperforms the other metaheuristic methods evaluated,

in terms of both the average RPD and the computational time. Further research needs to be conducted in applying other criteria in the *TC* in order to improve the solution quality and to adapt the approach to other objectives and process constraints.

**Appendix.** Shift neighborhood based local search procedure($l$ and $l'$ are indices of position in the machine path. $L$, $L'$ denote the number of operations assigned to machines $y$ and $y'$ in the current solution, respectively.)

```
Repeat:
        for y:=1 to m do
                for l:=1 to L do
                        for y′:=1 to m do
                                for l′:=1 to L′ do
                                        if (y′≠y or (y′=y and l′≠l)) then
                                        {
                                                • Remove the operation placed in position l
                                                on machine y and insert it into position l′
                                                on machine y′, leaving all other relative
                                                operation orders unchanged.
                                                • If the objective value of the obtained
                                                sequence (Obj) is less than the best
                                                objective value obtained so far (Obj*), then
                                                set Obj*:=Obj; Otherwise, insert the
                                                operation into its previous position.
                                        }
until  (no improvement occurs over the best solution)
```

## References


[1] Jain, A.S., Meeran, S., Deterministic job-shop scheduling: Past, present and future, European Journal of Operational Research 113 (2) (1998) 390–434.

[2] Baker, K., Introduction to sequencing and scheduling, NewYork: Wiley (1974).

[3] Pinedo, M., Scheduling: theory, algorithms and systems, Englewood cliffs, NJ: Prentice-Hall (2002).

[4] Garey, M.R., Johnson, D.S., Sethi, R., The complexity of flow shop and job-shop scheduling, Mathematics of Operations Research 1 (2) (1976) 117–129.

[5] Dell'Amico, M., Trubian, M., Applying tabu-search to the job-shop scheduling problem, Annals of Operations Research 4 (1993) 231–252.

[6] Matsuo, H., Suh, C., Sullivan, R., A controlled search simulated annealing method for the general job-shop scheduling problem, Tech. Rep. 03-04-88, Dept. of Management, The University of Texas, Austin (1988).

[7] Van Laarhoven, P., Aarts, E., Lenstra, J., Job shop scheduling by simulated annealing, Operations Research 40 (1992) 113–125.

[8] Xiong, J., Xing, L.-N., Chen, Y.-W., Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns, International Journal of Production Economics 141 (2013) 112–126.

[9] Gao, J., Gen, M., Sun, L., Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm, Journal of Intelligent Manufacturing 17 (2006) 493–507.

[10] Schmidt, G., Scheduling with limited machine availability, European Journal of Operational Research 121 (1) (2000) 1–15.

[11] Ma, Y., Chu, C., Zuo, C., A survey of scheduling with deterministic machine availability constraints, Computers & Industrial Engineering 58 (2010) 199–211.

[12] Dalfard, V. M., Mohammadi, G., Two meta-heuristic algorithms for solving multi-objective flexible job-shop scheduling with parallel machine and maintenance constraints, Computers & Mathematics with Applications 64 (6) (2012) 2111–2117.

[13] Chan, F.T.S., Wong, T.C., Chan, L.Y., Flexible job-shop scheduling problem under resource constraints International Journal of Production Research 44 (11) (2006) 2071–2089.
[14] Zribi, N., Kamel, A.E., Borne, P., Minimizing the makespan for the MPM job-shop with availability constraints, International Journal of Production Economics 112 (1) (2008) 151–160.
[15] Rajkumar, M., Asokan, P., Vamsikrishna, V., A GRASP algorithm for flexible job-shop scheduling with maintenance constraints, International Journal of Production Research 48 (22) (2010) 6821–6836.
[16] Moradi, E., Ghomi, S.M.T.F., Zandieh, M., Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem, Expert Systems with Applications 38 (6) (2011) 7169–7178.
[17] Vilcot, G., Billaut, J.-C., A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem, European Journal of Operational Research 190 (2) (2008) 398–411.
[18] Chan, F.T.S., Chung, S.H., Chan, L.Y., Finke, G., Tiwari, M.K., Solving distributed FMS scheduling problems subject to maintenance: Genetic algorithms approach, Robotics and Computer-Integrated Manufacturing 22 (5-6) (2006) 493–504.
[19] Wang, S., Yu, J., An effective heuristic for flexible job-shop scheduling problem with maintenance activities, Computers & Industrial Engineering 59 (2010) 436–447.
[20] Kacem, I., Hammadi, S., Borne, P., Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems, IEEE Transactions on Systems, Man, and Cybernetics, Part C, 32(1) (2002) 1–13.
[21] Kacem, I., Hammadi, S., Borne, P., Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic, Mathematics and Computers in Simulation 60 (2002) 245–276.
[22] Xia, W., Wu, Z., An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems, Computers & Industrial Engineering 48 (2005) 409–425.