International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: http://www.tandfonline.com/loi/tprs20

A GRASP algorithm for flexible job-shop scheduling with maintenance constraints

M. Rajkumar , P. Asokan & V. Vamsikrishna

To cite this article: M. Rajkumar , P. Asokan & V. Vamsikrishna (2010) A GRASP algorithm for flexible job-shop scheduling with maintenance constraints, International Journal of Production Research, 48:22, 6821-6836, DOI: 10.1080/00207540903308969

To link to this article: https://doi.org/10.1080/00207540903308969

Published online: 18 Jan 2010.

International Journal of Production Research
Vol. 48, No. 22, 15 November 2010, 6821–6836

Taylor & Francis
Taylor & Francis Group

A GRASP algorithm for flexible job-shop scheduling with maintenance constraints

M. Rajkumar*, P. Asokan and V. Vamsikrishna

Department of Production Engineering, National Institute of Technology, Tiruchirappalli-620 015, Tamilnadu, India

(*Received 15 July 2009; final version received 18 August 2009*)

In most realistic situations, machines may be unavailable due to maintenance, pre-schedules and so on. The availability constraints are non-fixed in that the completion time of the maintenance task is not fixed and has to be determined during the scheduling procedure. In this paper a greedy randomised adaptive search procedure (GRASP) algorithm is presented to solve the flexible job-shop scheduling problem with non-fixed availability constraints (FJSSP-nfa). The GRASP algorithm is a metaheuristic algorithm which is characterised by multiple initialisations. Basically, it operates in the following manner: first a feasible solution is obtained, which is then further improved by a local search technique. The main objective is to repeat these two phases in an iterative manner and to preserve the best found solution. Representative FJSSP-nfa benchmark problems are solved in order to test the effectiveness and efficiency of the proposed algorithm.

Keywords: flexible job-shop scheduling; non-fixed availability; GRASP; preventive maintenance

1. Introduction

Most of the papers in the scheduling literature assume that machines are available during the whole planning horizon. This assumption may no longer be true in real time situations, since breakdowns can happen at any moment, making one or several machines unavailable for processing jobs. Also, machines can be voluntarily stopped to allow some maintenance tasks, such as washing or control operations, to be realised. Limited availability of machines may also result from pre-schedules. In the presence of pre-schedules, the availability of resources for any subsequent plan is limited. Therefore, a more realistic scheduling model should take into account the availability constraints of machines. The flexible job-shop scheduling problem (FJSSP) under availability constraints problems is strongly nondeterministic polynomial time hard (NP-hard) since the problem without unavailability periods is already strongly NP-hard (Nozha and Borne 2005).

In this paper, the flexible job-shop scheduling problem has been studied with non-fixed availability (FJSSP-nfa) constraints. We consider the deterministic case where the unavailability periods, corresponding to some preventive maintenance tasks, are known in advance and fixed. Moreover, we suppose that the operations are strictly non pre-emptive, which means that once started, the execution of an operation can be

*Corresponding author. Email: vathilairaj@gmail.com

ISSN 0020–7543 print/ISSN 1366–588X online

DOI: 10.1080/00207540903308969
http://www.informaworld.com

interrupted either by a maintenance task, or by another operation. The objective functions are minimisation of makespan, maximum workload and total workload.

There are various kinds of availability constraints in production systems. Roughly, they can be categorised into two types: fixed and non-fixed. The unavailable period of a fixed availability constraint starts at a fixed time point. For non-fixed availability constraints, the starting time of the unavailable period is supposed to be flexible and must be determined during the production scheduling procedure. For example, we generally allow certain flexibility in time to implement preventive maintenance (PM), and give each PM task a time window in which the actual starting time of the PM task can be moved within (Gao *et al*. 2006).

Lee (1997) considered a two-machine flow shop scheduling problem where if a job cannot be finished before the next down period of a machine then the job will have to partially restart when the machine has become available again. Lee (1999) extensively investigated flow shop scheduling with two machines. In particular, he defined the resumable, non-resumable and semi-resumable models. Lee and Chen (2000) investigated scheduling jobs and maintenance on parallel machines. Schmidt (2000) reviewed most results related to deterministic scheduling problems with availability constraints and a survey on existing methods for solving scheduling problems under availability constraints as well as complexity results can be found in this work.

Breit *et al*. (2001) presented an approximation algorithm for a two-machine open shop scheduling problem, in which one machine is not available for processing during a given time interval. Aggoune and Portman (2006) addressed a general flow shop scheduling problem with limited machine availability and minimal makespan. Aggoune (2006) applied a genetic algorithm (GA) and a tabu search (TS) for flow shop scheduling and considered more than one unavailability period on each machine and showed the stability of the two algorithms.

Allaoui and Artiba (2006) optimised a hybrid flow shop scheduling problem under maintenance constraints with the objectives of flow time and due date. They showed the proposed simulated annealing algorithm results and performance. Allaoui *et al*. (2008) dealt with the problem of jointly scheduling n immediately available jobs and the preventive maintenance in a two-machine flow shop with the objective of minimising the makespan.

Allaoui and Artiba (2009) recently applied Johnson's algorithm to the two-machine flow shop to minimise the makespan in polynomial time and also dealt with the extension of Johnson's algorithm. They analysed the importance of Johnson's algorithm and showed the results of flow shop scheduling problems. Naderi *et al*. (2009) investigated job shop scheduling with sequence-dependent setup times and preventive maintenance polices. They used four algorithms to solve the benchmark problems and proved that the hybridised GA gives superior results. Nozha and Borne (2005) proposed a GA to solve job shop with multi purpose machines (MPM) with limited availability problems and showed the quality and the adequacy of the proposed algorithm.

Many researchers have effectively solved traditional flexible job shop scheduling problems by various evolutionary algorithms such as GA (Zhang and Gen 2005), fuzzy logic (Kacem *et al*. 2002a), TS (Saidi and Fattahi 2007), particle swarm optimisation (PSO) (Xia and Wu 2005, Zhang *et al*. 2009), approach by localisation (Kacem *et al*. 2002b) and ant colony optimisation (ACO) (Liouane *et al*. 2007, Xing *et al*. 2008), etc. The traditional FJSSP problem is associated with two difficulties. The first one is the routing problem (i.e., the assignment of operations to machines) and the second one is the scheduling problem

(i.e., determining the starting and end time of each operation). But the FJSSP-nfa problem adds a third decision, scheduling of preventive maintenance tasks, to the traditional FJSSP. The combination of such decisions adds additional complexity and new problems. For these reasons, only a few researchers are interested in solving the FJSSP with resource constraints.

Chen *et al*. (2006) developed a GA based approach to solve the FJSSP under resource constraints and generated their results. Gao *et al*. (2006) solved FJSSP-nfa problems by using a hybrid genetic algorithm (hGA) and showed that the proposed hGA gives better results than those of Xia and Wu (2005). For more than a decade, the GRASP algorithm has been effectively applied to solve combinatorial problems in routing and scheduling, such as single machine problems (Bard *et al*. 1995, Gupta and Smith 2006), Chinese postman routing problem (Corberan *et al*. 2002), aircraft routing problem (Arguello and Bard 1997), flow shop scheduling (Prabhaharan *et al*. 2006) and job shop scheduling problems (Aiex *et al*. 2003).

We propose a greedy randomised adaptive search procedure (GRASP) algorithm to solve the FJSSP-nfa problems. A GRASP is an iterative process and each GRASP iteration consists of two phases, a construction phase and a local search phase. The best overall solution is kept as the result. This paper is organised with the following sections. Section 2 presents the problem formulation, notation and hypotheses considered. The GRASP algorithm is explained in Section 3. Section 4 depicts the computational results and presents a study on well-known FJSSP-nfa problems. Some final concluding remarks are provided in Section 5.

2. Problem formulation, notation and hypotheses

The flexible job shop scheduling problem with non-fixed availability constraints is stated as follows: n jobs are to be scheduled on m machines. Each job i represents a number of n_i non pre-emptive ordered operations. The execution of each operation k of job i (noted as O_{ik}) requires one machine selected from a set of available machines called A_{ik} and will occupy that machine t_{ikj} time units until the operation is completed. There are L_j maintenance tasks that have to be processed on each machine j during the planning horizon. Each maintenance task corresponds to a predefined time window, in which the completion time of the maintenance task can be moved within. We assume that the operation processing is non-resumable, i.e., an operation must be reprocessed fully after the maintenance if its processing is interrupted by the maintenance activity on a machine. The FJSSP-nfa problem is formulated to assign operations to a machine, schedule the job operations, and the maintenance task is subject to the constraints:

(1) The operation sequence for each job is prescribed;
(2) Each machine can process only one operation or PM task at a time;
(3) Each PM task has to be completed within its pre-specified time window.

In this study, we manage to minimise the following three criteria:

(1) Makespan (C_{m}) of the jobs;
(2) Maximal machine workload ($W_{\max}$), i.e., the maximum working time spent on any machine;
(3) Total workload of the machines (W_{t}), which represents the total working time over all machines.

Hypotheses considered in this paper are summarised as follows:

(1) In a flexible job shop, n jobs are to be scheduled on m machines;
(2) The ith job has n_i ordered operations that have to be processed;
(3) The execution of each operation O_{ik} requires one machine selected from a set of available machines called A_{ik};
(4) Every machine processes only one activity (including both operations and PM tasks) at a time;
(5) The setup time for the operations is machine independent and is included in the processing time;
(6) There are L_j maintenance tasks on each machine j during the planning horizon;
(7) Each maintenance task has a predefined time window, in which the completion time of the task can be moved within;
(8) When a PM task is performed on a machine, no operation can be processed on that machine.

The notation used in this paper is summarised as follows:

Indices

i, h: index of jobs, $i, h = 1, 2, \ldots, n$;
j: index of machines, $j = 1, 2, \ldots, m$;
k, g: index of operation sequence, $k, g = 1, 2, \ldots, n_i$;
l: index of maintenance tasks, $l = 1, 2, \ldots, L_j$.

Parameters

n: total number of jobs;
m: total number of machines;
n_i: total number of operations of job i;
L_j: total number of PM tasks on machine j;
O_{ik}: the kth operation of job i;
PM_{jl}: the lth preventive maintenance task on machine j;
A_{ik}: the set of available machines for the operation O_{ik};
t_{ikj}: processing time of O_{ik} on machine j;
p_{jl}: duration of the maintenance task PM_{jl};
$[t_{jl}^{\mathrm{E}}, t_{jl}^{\mathrm{L}}]$: time window associated with PM_{jl}, where t_{jl}^{E} is the early completion time and t_{jl}^{L} is the late completion time.
c_{ik}: completion time of the operation O_{ik};
y_{jl}: completion time of PM_{jl}.

Decision variable

$$x_{ikj} = \begin{cases} 1, & \text{if machine } j \text{ is selected for the operation } O_{ik} \\ 0, & \text{otherwise} \end{cases}.$$

Our model is presented as follows:

$$F_1 = \min C_{\mathrm{m}}(x, c, y) = \max_{1 \le i \le n} \left\{ c_{in_i} \right\} \tag{1}$$

$$F_2 = \min W_{\max}(x, c, y) = \max_{1 \le j \le m} \left\{ \sum_{i=1}^{n} \sum_{k=1}^{n_i} t_{ikj} x_{ikj} + \sum_{l=1}^{L_j} p_{jl} \right\} \tag{2}$$

$$F_3 = \min W_t(x, c, y) = \sum_{j}^{m} \sum_{i=1}^{n} \sum_{k=1}^{n_i} t_{ikj} x_{ikj} + \sum_{j}^{m} \sum_{l=1}^{L_j} p_{jl}. \tag{3}$$

Where:

F_1 = minimisation of makespan;
F_2 = minimisation of maximum workload;
F_3 = minimisation of total workload.

The weighted sum of the above three objective values are taken as the combined objective function:

$$\text{Minimise } F(c) = W_1 \times F_1(c) + W_2 \times F_2(c) + W_3 \times F_3(c). \tag{4}$$

Subject to:

$$\begin{aligned} W_1 &\in (0, 1) \\ W_2 &\in (0, 1) \\ W_3 &\in (0, 1) \\ W_1 + W_2 + W_3 &= 1 \end{aligned} \tag{5}$$

3. GRASP algorithm

The GRASP algorithm is a metaheuristic algorithm which is characterised by multiple initialisations. Basically it consists of two phases: construction phase and local search phase. The GRASP algorithm for the minimisation problem is given as follows:

Algorithm 1: The GRASP algorithm (the minimisation problem)

cut = dimension of the problem
For $k = 1, \ldots,$ iterations_max **make**:
  Construction:
  Solution = 0
  Evaluate the incremental cost of each individual candidate
  If **find** Dimension(*Solution*) $\neq$ *cut*:
    Build LLC: the limited list of the candidates
    Select randomly an element *e* from RCL
    Solution = Add(*Solution*, *e*)
  end

```
Procedure Greedy_Randomised_Construction (Seed)
1   Solution ← φ;
2   Evaluate the incremental costs of the candidate elements;
3   While Solution is not a complete solution do
4      Build the restricted candidate list (RCL);
5      Select an element s from the RCL at random;
6      Solution ← Solution ∪ {s};
7      Re-evaluate the incremental costs;
8   end;
9   return Solution;
end Greedy_Randomised_Construction.
```

Figure 1. Pseudo-code of the construction phase.

Search ***Local***:
If **find** the solution locally non-optimal:
Find $s' \in$ *the*Neighbourhood of (*Solution*) such that $f(s') < f(Solution)$
$Solution = s'$
end
$better\_solution = \min(Solution, better\_solution)$
end
return *better_solution*

3.1 *GRASP construction phase*

Figure 1 illustrates the construction phase with its pseudo code. Resende and Ribeiro (1998) explained the GRASP construction phase. Each iteration of this construction phase lets the set of candidate elements to be formed by all elements that can be incorporated to the partial solution under construction without destroying feasibility. The selection of the next element for incorporation is determined by the evaluation of all candidate elements according to a greedy evaluation function. This greedy function usually represents the incremental increase in the cost function due to the incorporation of this element into the solution under construction. The evaluation of the elements by this function leads to the creation of a restricted candidate list (RCL) formed by the best elements, i.e., whose incorporation to the current partial solution results in the smallest incremental costs (this is the greedy aspect of the algorithm). The element to be incorporated into the partial solution is randomly selected from those in the RCL (this is the probabilistic aspect of the heuristic). Once the selected element is incorporated to the partial solution, the candidate list is updated and the incremental costs are re-evaluated (this is the adaptive aspect of the heuristic).

3.2 *GRASP local search phase*

A local search algorithm functions in an iterative manner by successively replacing the current solution by a better solution in the neighbourhood of the current solution. It terminates when no better solution is found in the neighbourhood. The neighbourhood structure N for a problem P relates a solution s of the problem to a subset of solutions $N(s)$. A solution s is said to be locally optimal if there is no better solution in $N(s)$. Given a neighbourhood structure N, a local search algorithm has the general form as stated in Figure 2 (Feo and Resende 1995).

```
Procedure Local (P, N (P), s)
1  for s not locally optimal →
2     Find a better solution t ∈ N(s) ;
3     Let s = t;
4  rof;
5  return (s as local optimal for P);
end Local.
```

Figure 2. Pseudo-code of the local search phase.

3.3 *Bias function*

In this paper we have considered linear bias function for execution of GRASP. Various other bias functions like random bias, logarithmic bias, exponential bias and polynomial bias can be used in GRASP. From general observations the linear bias function is the best among the bias functions (Resende 1998).

3.4 *Structure of GRASP for routing*

The FJSSP-nfa problem is an extension of the classical job shop scheduling problem. It consists of two sub-problems, such as routing and scheduling. For the routing sub-problem in FJSSP, the objective is to assign an operation to a machine and PM tasks such that total work load is minimised and work load on a machine is maximised. The procedure is explained as follows:

Let:

σ refer to the set of assigned operations of jobs and PM tasks;
σ' refer to the set of unassigned operations of jobs and PM tasks;

consider an empty set for initialisation, i.e., $\sigma = \{0\}$.

Step 1: Set the number iterations (t).

Step 2: Compute the objective function value (OFV), i.e., total work load.

Step 3: Determine the minimum and maximum value of the processing times of work load of the operation.

Step 4: Determine the range using the formula shown below:

$$Range = (\text{maximum OFV} - \text{minimum OFV}).$$

Step 5: Randomly choose the value for greedy parameter α from discrete set of α values, and $\alpha \in (0, 1)$.

Step 6: Determine the width as: $Width = (Range \times \alpha)$

Step 7: Now select the processing times of operation for entry in RCL:

$$\text{RCL} = \big\{(\text{minimum OFV}), \quad ((\text{minimum OFV}) + Width)\big\}.$$

Step 8: Rank the candidates in RCL.

Step 9: Bias the candidate from RCL by using linear bias function. Linear bias: $\text{bias}(r) = 1/r$, where r is the rank of the candidate.

Step 10: Find the probability of selecting the candidate $\prod(O_{ik})$ using the formula:

$$\prod(\sigma) = \frac{bias(r(O_{ik}))}{\sum_{\sigma' \in RCL} bias\left(r(O'_{ik})\right)}. \tag{6}$$

Step 11: Update σ.

Step 12: Repeat Steps 2 to 9 until $\sigma' = \{0\}$.

Step 13: Compute the OFV for the complete sequence.

Step 14: Repeat all the steps t times for the different values of α.

3.5 *Structure of GRASP for scheduling*

For the scheduling sub-problem in FJSSP-nfa, the objective is to minimise the makespan by sequencing the operations and PM tasks, which are allocated on machines. The input is taken from the allocation sub-problem. The procedure is as follows:

Let:

σ refer to the set of scheduled operations of jobs and PM tasks;
σ' refer to the set of unscheduled operations of jobs and PM tasks;

consider an empty set for initialisation, i.e., $\sigma = \{0\}$.

Step 1: Set the number of iterations (t).

Step 2: Compute the objective function value, i.e., makespan.

Step 3: Determine the minimum and maximum value of the processing times of work load of the operation.

Step 4: Determine the range using the formula shown below:

$$Range = (\text{maximum OFV} - \text{minimum OFV}).$$

Step 5: Randomly choose the value for greedy parameter α from discrete set of α values, and $\alpha \in (0, 1)$.

Step 6: Determine the width as: $Width = (1 + Range \times \alpha)$

Step 7: Now choose the processing times of operation for entry in RCL:

$$\text{RCL} = \left\{(\text{minimum OFV}), ((\text{minimum OFV}) + Width)\right\}.$$

Step 8: Rank the candidates in RCL.

Step 9: Bias the candidate from RCL by using linear bias function. Linear bias: $\text{bias}(r) = 1/r$, where r is the rank of the candidate.

Step 10: Find the probability of selecting the candidate$\prod(O_{ij})$ using the formula:

$$\prod(\sigma) = \frac{bias(r(O_{ik}))}{\sum_{\sigma' \in RCL} bias\left(r(O'_{ik})\right)}. \tag{7}$$

Step 11: Update σ.

Step 12: Repeat Steps 2 to 9 until $\sigma' = \{0\}$.

Step 13: Compute the OFV for the complete sequence.

Step 14: Repeat all the steps t times for the different values of α.

4. Computational results

In this paper, the GRASP algorithm procedure was implemented in a C++ environment on a personal computer with Centrino Duo 1.0 G CPU. Four representative instances based on practical data have been selected. Each instance can be characterised by the following parameters: number of jobs (n), number of machines (m), and each operation O_{ik} of job i. Four problem instances (8×8, 10×10, 15×10 and 4×5) are taken from Gao *et al*. (2006).

4.1 *Problem 8 × 8*

We applied the GRASP algorithm to a middle scale instance of 8×8 with 27 operations, which is shown in Table 1. This is an instance of partial flexibility, i.e., some operations are

Table 1. Problem 8×8 with 27 operations (partial flexibility).

Job	O_{ik}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
J_1	$O_{1,1}$	5	3	5	3	3	–	10	9
	$O_{1,2}$	10	–	5	8	3	9	9	6
	$O_{1,3}$	–	10	–	5	6	2	4	5
J_2	$O_{2,1}$	5	7	3	9	8	–	9	–
	$O_{2,2}$	–	8	5	2	6	7	10	9
	$O_{2,3}$	–	10	–	5	6	4	1	7
	$O_{2,4}$	10	8	9	6	4	7	–	–
J_3	$O_{3,1}$	10	–	–	7	6	5	2	4
	$O_{3,2}$	–	10	6	4	8	9	10	–
	$O_{3,3}$	1	4	5	6	–	10	–	7
J_4	$O_{4,1}$	3	1	6	5	9	7	8	4
	$O_{4,2}$	12	11	7	8	10	5	6	9
	$O_{4,3}$	4	6	2	10	3	9	5	7
J_5	$O_{5,1}$	3	6	7	8	9	–	10	–
	$O_{5,2}$	10	–	7	4	9	8	6	–
	$O_{5,3}$	–	9	8	7	4	2	7	–
	$O_{5.4}$	11	9	–	6	7	5	3	6
J_6	$O_{6,1}$	6	7	1	4	6	9	–	10
	$O_{6,2}$	11	–	9	9	9	7	6	4
	$O_{6,3}$	10	5	9	10	11	–	10	–
J_7	$O_{7,1}$	5	4	2	6	7	–	10	–
	$O_{7,2}$	–	9	–	9	11	9	10	5
	$O_{7,3}$	–	8	9	3	8	6	–	10
J_8	$O_{8,1}$	2	8	5	9	–	4	–	10
	$O_{8,2}$	7	4	7	8	9	–	10	–
	$O_{8,3}$	9	9	–	8	5	6	7	1
	$O_{8,4}$	9	–	3	7	1	5	8	–

Table 2. PM tasks of JSSP-nfa problem 8×8 with 27 operations.

		$PM_{1,1}$	$PM_{2,1}$	$PM_{3,1}$	$PM_{4,1}$	$PM_{5,1}$	$PM_{6,1}$	$PM_{7,1}$	$PM_{8,1}$
Time window	t_{jl}^{E}	5	6	10	9	3	8	4	7
	t_{jl}^{L}	10	9	15	17	10	16	14	13
Duration		4	3	5	3	3	5	3	4

Table 3. Comparison of the results of FJSSP-nfa with hGA for 8×8 problem.

	hGA	GRASP
W_t	105	103
W_{max}	15	16
C_m	17	18
F(0.5-0.3-0.2)	60.4	59.9
F(0.5-0.2-0.3)	60.6	60.1

only available on a part of the available machine set. In the data of Table 1, symbol '-' indicates that the assignment is impossible. Table 2 shows the PM tasks on each machine of the 8×8 flexible job shop. The optimal values of the proposed algorithm are shown as follows: makespan (C_m) = 13, maximum work load (W_{max}) = 13 and total work load (W_t) = 73.

The comparison of the results of the proposed algorithm with the hybrid genetic algorithm (hGA) is presented in Table 3. It also shows that the combined objective function value ($F(c)$) for different weighting assigned to makespan, maximum work load, and total work load. The comparison results show that the proposed algorithm is more appropriate for solving problems of partial flexibility.

4.2 *Problem 10×10*

To test the performance of the proposed algorithm 10 jobs with 30 operations has been performed on 10 machines. It is a total flexibility problem; the detailed problem is given in Table 4. The PM tasks on each machine of the 10×10 flexible job shop are shown in Table 5. The result shows that the proposed algorithm gives best value in total work load (W_t). The optimal solution and comparison with the hGA algorithm is shown in Table 6. The combined objective function ($F(c)$) value is lower (33.9) if we consider total workload as most important, maximum workload as more important, and makespan is important.

4.3 *Problem 15×10*

A large scale problem is chosen to test the performance of the proposed GRASP algorithm. This problem contains 15 jobs with 56 operations that have to be processed on 10 machines with total flexibility. The detailed problem is shown in Table 7. The PM tasks on the 10 machines of the flexible job shop are shown in Table 8. The comparison of the proposed algorithm with the hGA is shown in Table 8. The result shows that the proposed

Table 4. Problem 10×10 with 30 operations (total flexibility).

Job	O_{ik}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	5
	$O_{1,2}$	4	1	1	3	4	8	10	4	11	4
	$O_{1,3}$	3	2	5	1	5	6	9	5	10	3
J_2	$O_{2,1}$	2	10	4	5	9	8	4	15	8	4
	$O_{2,2}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,3}$	6	11	2	7	5	3	5	14	9	2
J_3	$O_{3,1}$	8	5	8	9	4	3	5	3	8	1
	$O_{3,2}$	9	3	6	1	2	6	4	1	7	2
	$O_{3,3}$	7	1	8	5	4	9	1	2	3	4
J_4	$O_{4,1}$	5	10	6	4	9	5	1	7	1	6
	$O_{4,2}$	4	2	3	8	7	4	6	9	8	4
	$O_{4,3}$	7	3	12	1	6	5	8	3	5	2
J_5	$O_{5,1}$	7	10	4	5	6	3	5	15	2	6
	$O_{5,2}$	5	6	3	9	8	2	8	6	1	7
	$O_{5,3}$	6	1	4	1	10	4	3	11	13	9
J_6	$O_{6,1}$	8	3	10	8	4	2	7	8	3	10
	$O_{6,2}$	7	3	12	5	4	3	6	9	2	15
	$O_{6,3}$	4	7	3	6	3	4	1	5	1	11
J_7	$O_{7,1}$	1	7	8	3	4	9	4	13	10	7
	$O_{7,2}$	3	8	1	2	3	6	11	2	13	3
	$O_{7,3}$	5	4	2	1	2	1	8	14	5	7
J_8	$O_{8,1}$	5	7	11	3	2	9	8	5	12	8
	$O_{8,2}$	8	3	10	7	5	13	4	6	8	4
	$O_{8,3}$	6	2	13	5	4	3	5	7	9	5
J_9	$O_{9,1}$	3	9	1	3	8	1	6	7	5	4
	$O_{9,2}$	4	6	2	5	7	3	1	9	6	7
	$O_{9,3}$	8	5	4	8	6	1	2	3	10	12
J_{10}	$O_{10,1}$	4	3	1	6	7	1	2	6	20	6
	$O_{10,2}$	3	1	8	1	9	4	1	4	17	15
	$O_{10,3}$	9	2	4	2	3	5	2	4	10	23

Table 5. PM tasks of JSSP-nfa problem 10×10 with 30 operations.

		$PM_{1,1}$	$PM_{2,1}$	$PM_{3,1}$	$PM_{4,1}$	$PM_{5,1}$	$PM_{6,1}$	$PM_{7,1}$	$PM_{8,1}$	$PM_{9,1}$	$PM_{10,1}$
Time window	t_{jl}^{E}	2	2	1	2	3	2	2	3	2	3
	t_{jl}^{L}	4	7	6	5	7	6	5	7	5	6
Duration		2	1	1	2	1	2	2	3	2	3

Table 6. Comparison of the results of FJSSP-nfa with hGA for 10×10 problem.

	hGA	GRASP
W_t	61	60
W_{max}	7	7
C_m	8	9
F(0.5-0.3-0.2)	34.2	33.9
F(0.5-0.2-0.3)	34.3	34.1

Table 7. Problem 15×10 with 56 operations (total flexibility).

Job	O_{ik}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	4
	$O_{1,2}$	1	1	3	4	8	10	4	11	4	3
	$O_{1,3}$	2	5	1	5	6	9	5	10	3	2
	$O_{1,4}$	10	4	5	9	8	4	15	8	4	4
J_2	$O_{2,1}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,2}$	6	11	2	7	5	3	5	14	9	2
	$O_{2,3}$	8	5	8	9	4	3	5	3	8	1
	$O_{2,4}$	9	3	6	1	2	6	4	1	7	2
J_3	$O_{3,1}$	7	1	8	5	4	9	1	2	3	4
	$O_{3,2}$	5	10	6	4	9	5	1	7	1	6
	$O_{3,3}$	4	2	3	8	7	4	6	9	8	4
	$O_{3,4}$	7	3	12	1	6	5	8	3	5	2
J_4	$O_{4,1}$	6	2	5	4	1	2	3	6	5	4
	$O_{4,2}$	8	5	7	4	1	2	36	5	8	5
	$O_{4,3}$	9	6	2	4	5	1	3	6	5	2
	$O_{4,4}$	11	4	5	6	2	7	5	4	2	1
J_5	$O_{5,1}$	6	9	2	3	5	8	7	4	1	2
	$O_{5,2}$	5	4	6	3	5	2	28	7	4	5
	$O_{5,3}$	6	2	4	3	6	5	2	4	7	9
	$O_{5,4}$	6	5	4	2	3	2	5	4	7	5
J_6	$O_{6,1}$	4	1	3	2	6	9	8	5	4	2
	$O_{6,2}$	1	3	6	5	4	7	5	4	6	5
J_7	$O_{7,1}$	1	4	2	5	3	6	9	8	5	4
	$O_{7,2}$	2	1	4	5	2	3	5	4	2	5
J_8	$O_{8,1}$	2	3	6	2	5	4	1	5	8	7
	$O_{8,2}$	4	5	6	2	3	5	4	1	2	5
	$O_{8,3}$	3	5	4	2	5	49	8	5	4	5
	$O_{8,4}$	1	2	36	5	2	3	6	4	11	2
J_9	$O_{9,1}$	6	3	2	22	44	11	10	23	5	1
	$O_{9,2}$	2	3	2	12	15	10	12	14	18	16
	$O_{9,3}$	20	17	12	5	9	6	4	7	5	6
J_{10}	$O_{10,1}$	5	8	7	4	56	3	2	5	4	1
	$O_{10,2}$	2	5	6	9	8	5	4	2	5	4
	$O_{10,3}$	6	3	2	5	4	7	4	5	2	1
	$O_{10,4}$	3	2	5	6	5	8	7	4	5	2
J_{11}	$O_{11,1}$	1	2	3	6	5	2	1	4	2	1
	$O_{11,2}$	2	3	6	3	2	1	4	10	12	1
	$O_{11,3}$	3	6	2	5	8	4	6	3	2	5
	$O_{11,4}$	4	1	45	6	2	4	1	25	2	4
J_{12}	$O_{12,1}$	9	8	5	6	3	6	5	2	4	2
	$O_{12,2}$	5	8	9	5	4	75	63	6	5	21
	$O_{12,3}$	12	5	4	6	3	2	5	4	2	5
	$O_{12,4}$	8	7	9	5	6	3	2	5	8	4
J_{13}	$O_{13,1}$	4	2	5	6	8	5	6	4	6	2
	$O_{13,2}$	3	5	4	7	5	8	6	6	3	2
	$O_{13,3}$	5	4	5	8	5	4	6	5	4	2
	$O_{13,4}$	3	2	5	6	5	4	8	5	6	4
J_{14}	$O_{14,1}$	2	3	5	4	6	5	4	85	4	5
	$O_{14,2}$	6	2	4	5	8	6	5	4	2	6
	$O_{14,3}$	3	25	4	8	5	6	3	2	5	4
	$O_{14,4}$	8	5	6	4	2	3	6	8	5	4
J_{15}	$O_{15,1}$	2	5	6	8	5	6	3	2	5	4
	$O_{15,2}$	5	6	2	5	4	2	5	3	2	5

Table 8. PM tasks of JSSP-nfa problem 15×10 with 56 operations.

		$PM_{1,1}$	$PM_{2,1}$	$PM_{3,1}$	$PM_{4,1}$	$PM_{5,1}$	$PM_{6,1}$	$PM_{7,1}$	$PM_{8,1}$	$PM_{9,1}$	$PM_{10,1}$
Time window	t_{jl}^{E}	2	3	1	3	2	1	3	1	2	3
	t_{jl}^{L}	5	7	3	10	8	6	7	5	5	8
Duration		1	1	1	3	2	1	1	1	1	1

Table 9. Comparison of the results of FJSSP-nfa with hGA for 15×10 problem.

FJSSP-nfa	hGA	GRASP
W_t	109	107
W_{max}	12	13
C_m	12	16
F(0.5-0.3-0.2)	60.5	60.3
F(0.5-0.2-0.3)	60.5	60.9

Table 10. Problem 4×5 with 12 operations (total flexibility).

Job	O_{ik}	M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	2	5	4	1	2
	$O_{1,2}$	5	4	5	7	5
	$O_{1,3}$	4	5	5	4	5
J_2	$O_{2,1}$	2	5	4	7	8
	$O_{2,2}$	5	6	9	8	5
	$O_{2,3}$	4	5	4	54	5
J_3	$O_{3,1}$	9	8	6	7	9
	$O_{3,2}$	6	1	2	5	4
	$O_{3,3}$	2	5	4	2	4
	$O_{3,4}$	4	5	2	1	5
J_4	$O_{4,1}$	1	5	2	4	12
	$O_{4,2}$	5	1	2	1	2

algorithm generates best value in the case of maximum work load and makespan. The optimal values are shown as follows: $C_m = 12$, $W_{max} = 12$ and $W_t = 109$.

4.4 *Problem 4×5*

Table 10 shows a small scale problem, with four jobs, five machines and 12 operations. To evaluate the effectiveness of our algorithm, finally we have applied it to the small scale instance as shown in Table 10. The FJSSP problem is adopted from Kacem *et al.* (2002a,b) and then it is extended to the FJSSP-nfa problem by considering one or two PM tasks on each machine. The PM tasks on the five machines of the flexible job shop are shown in Table 11. The optimal solution of the FJSSP-nfa problem obtained by the GRASP is shown in Table 12. Due to lack of availability of benchmark problems, it cannot be compared with other algorithms.

Table 11. PM tasks of JSSP-nfa problem 4×5 with 12 operations.

		$PM_{1,1}$	$PM_{2,1}$	$PM_{3,1}$	$PM_{4,1}$	$PM_{5,1}$
Time window	t_{jl}^{E}	1	2	1	2	6
	t_{jl}^{L}	4	6	6	5	10
Duration		1	2	2	1	2

Table 12. Optimal results of problem 4×5.

	GRASP
W_t	40
W_{max}	9
C_m	16
F(0.5-0.3-0.2)	25.9
F(0.5-0.2-0.3)	26.6

5. Conclusions

In this paper, a GRASP algorithm has been proposed to solve multi-objective FJSSP-nfa problems. The performance of the proposed algorithm has been validated by comparing the results available in the literature for four representative instances. The literature problems have been solved using a hybrid genetic algorithm (hGA). The combined objective considering the makespan, maximum work load, and total work load has been used as the performance measures to solve FJSSP-nfa problems.

There are many research directions that can be considered as useful extensions of this research. The proposed algorithm has been tested with four representative instances for its objective criteria. An exclusive computational study could be carried out to test the efficiency of the proposed algorithm. Research on solving other multi-objectives such as mean flow time, tardiness and balanced level of machine utilisation could be an interesting area for future researchers. The robustness of GRASP can be proved while solving other combinatorial problems such as flow shop, classical job-shop, vehicle routing and the travelling salesman problem.

References

Aggoune, R., 2006. Minimizing the makespan for the flow shop scheduling problem with availability constraints. *European Journal of Operational Research*, 153 (3), 534–543.

Aggoune, R. and Portmann, M.C., 2006. Flow shop scheduling problem with limited machine availability: a heuristic approach. *International Journal of Production Economics*, 99 (1), 4–15.

Aiex, R.M., Binato, S., and Resende, G.C., 2003. Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing*, 29 (4), 393–430.

Allaoui, H. and Artiba, A., 2006. Scheduling two-stage hybrid flow shop with availability. *Computers & Operations Research*, 33 (5), 1399–1419.

Allaoui, H., *et al.*, 2008. Simultaneously scheduling *n* jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan. *International Journal of Production Economics*, 112 (1), 161–167.

Allaoui, H. and Artiba, A., 2009. Johnson's algorithm: a key to solve optimally or approximately flow shop scheduling problems with unavailability periods. *International Journal of Production Economics*, 121 (1), 81–87.

Arguello, M.F. and Bard, J.F., 1997. A GRASP for aircraft routing in response to groundings and delays. *Journal of Combinatorial Optimization*, 1 (3), 211–228.

Bard, J.F., Venkatraman, K., and Feo, T.A., 1995. Single machine scheduling with flow time and earliness penalties. *Journal of Global Optimization*, 3 (1), 289–309.

Breit, J., *et al*., 2001. Two-machine open shop scheduling with an availability constraint. *Operation Research Letters*, 29 (1), 65–77.

Chen, F.T.S., Wong, T.C., and Chen, L.Y., 2006. Flexible job-shop scheduling problem under resource constraints. *International Journal of Production Research*, 44 (11), 2071–2089.

Corberan, A., Marti, R., and Sanchis, J.M., 2002. A GRASP heuristic for the mixed Chinese postman problem. *European Journal of Operational Research*, 142 (1), 70–80.

Feo, T.A. and Resende, G.C., 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6 (1), 109–133.

Gao, J., Gen, M., and Sun, L., 2006. Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. *Journal of Intelligent Manufacturing*, 17 (4), 493–507.

Gupta, S.R. and Smith, J.S., 2006. Algorithms for single machine total tardiness scheduling with sequence dependent setups. *European Journal of Operational Research*, 175 (2), 722–739.

Kacem, I., Hammadi, S., and Borne, P., 2002a. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, 60 (4), 245–276.

Kacem, I., Hammadi, S., and Pierre, B., 2002b. Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions Systems Man and Cybernetics*, 32 (1), 1–13.

Lee, C.Y., 1997. Minimizing the makespan in two-machine flow shop scheduling with availability constraint. *Operations Research Letters*, 20 (1), 129–139.

Lee, C.Y., 1999. Two machine flow shop scheduling with availability constraints. *European Journal of Operational Research*, 114 (2), 420–429.

Lee, C.Y. and Chen, Z.L., 2000. Scheduling jobs and maintenances on parallel machines. *Navel Research Logistics*, 47 (1), 145–165.

Liouane, N., *et al*., 2007. Ant systems & local search optimization for flexible job shop scheduling production. *International Journal of Computers Communications and Control*, 2 (2), 174–184.

Naderi, B., Zandieh, M., and Fatemi Ghomi, S.M.T., 2009. Scheduling sequence dependent setup time job shops with preventive maintenance. *International Journal of Advanced Manufacturing and Technology*, 43 (1–2), 170–181.

Nozha, Z. and Borne, P., 2005. Hybrid genetic algorithm for the flexible job shop problem under maintenance constraints, *Proceedings of the international conference on advances in natural computation (ICNC 2005)*. 27–29 August Changsha, China. Berlin: Springer, 259–268.

Prabhaharan, G., Shahul Hamid Khan, B., and Rakesh, L., 2006. Implementation of GRASP in flow shop scheduling. *International Journal of Advanced Manufacturing and Technology*, 30 (11–12), 1126–1131.

Resende, G.C., 1998. Computing approximate solutions of the maximum covering problem using GRASP. *Journal of Heuristics*, 4 (1), 161–171.

Resende, G.C. and Ribeiro, C.C., 1998. Greedy Randomised Adaptive Search Procedures (GRASP). Technical Report, AT&T Labs Research, Florham Park, NJ 07932 USA, pp. 119–249.

Saidi, M. and Fattahi, P., 2007. Flexible job shop scheduling with tabu search algorithms. *International Journal of Advanced Manufacturing Technology*, 32 (6), 563–570.

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

Xia, W. and Wu, Z., 2005. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, 48 (2), 409–425.

Xing, L., *et al*., 2008. Knowledge based ant colony optimization for the flexible job shop scheduling problems. *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications & Algorithms*, 15, 431–446.

Zhang, G., *et al*., 2009. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers and Industrial Engineering*, 56 (4), 1309–1318.

Zhang, H. and Gen, M., 2005. Multistage-based genetic algorithm for flexible job shop scheduling problem. *Complexity International*, 11 (2), 223–232.