International Journal of Production Research

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

Flexible job-shop scheduling problem under resource constraints

F. T. S. Chan , T. C. Wong & L. Y. Chan

To cite this article: F. T. S. Chan , T. C. Wong & L. Y. Chan (2006) Flexible job-shop scheduling problem under resource constraints, International Journal of Production Research, 44:11, 2071-2089, DOI: 10.1080/00207540500386012

To link to this article: https://doi.org/10.1080/00207540500386012

Published online: 22 Feb 2007.

International Journal of Production Research,
Vol. 44, No. 11, 1 June 2006, 2071–2089

Flexible job-shop scheduling problem under resource constraints

F. T. S. CHAN*, T. C. WONG and L. Y. CHAN

Department of Industrial and Manufacturing Systems Engineering, University of Hong Kong, Pokfulam Road, Hong Kong

(*Revision received August 2005*)

A flexible job-shop-scheduling problem is an extension of classical job-shop problems that permit an operation of each job to be processed by more than one machine. The research methodology is to assign operations to machines (assignment) and determine the processing order of jobs on machines (sequencing) such that the system objectives can be optimized. This problem can explore very well the common nature of many real manufacturing environments under resource constraints. A genetic algorithm-based approach is developed to solve the problem. Using the proposed approach, a resource-constrained operations–machines assignment problem and flexible job-shop scheduling problem can be solved iteratively. In this connection, the flexibility embedded in the flexible shop floor, which is important to today's manufacturers, can be quantified under different levels of resource availability.

Keywords: Flexible job-shop scheduling; Operations–machines assignment; Sequencing; Resource constraints; Genetic algorithms

1. Introduction

In a classic job-shop scheduling problem (JSP), a set of n jobs $J = \{J_1, J_2, \ldots, J_n\}$ is processed on a set of m machines $M = \{M_1, M_2, \ldots, M_m\}$. Each job J_j is processed on machines in a given order with a given processing time Pt_{jk} and each machine M_i can only process one operation. Given MS_{jk} for $j = 1, \ldots, n$ and $k = 1, \ldots, m$, the problem becomes the determination of job processing orders on each machine for optimizing the objective value. However, due to vigorous technological developments over the past few years, the overlapping machine capabilities imply that a machine can process more than one operation (Baykasoğlu 2002, Kim *et al*. 2003). In this connection, a modified version of JSP called flexible JSP (FJSP) is studied. Specifically, the operations–machines assignment problem (OMAP) and FJSP will be solved using the proposed algorithm (Chan *et al*. 2005). Since all machines can process all operations, it is logical to restrict this freedom especially when resources are limited. Therefore, resource constraints are added to enhance the practicability of the previous algorithm. Nevertheless, FJSP is still NP-hard (Hartmann 2002, Kacem *et al*. 2002, Mastrolilli and Gambardella 2000). Since there are limited researches focusing on resource-constrained FJSP (RCFJSP), this study may provide

*Corresponding author. Email: ftschan@hkucc.hku.hk

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540500386012

practical values to today's manufacturers under rigid cost control and global competition.

The proposed approach makes use of an evolutionary optimization technique (Holland 1975), i.e. genetic algorithms (GA), due to its best-known optimizing ability for a class of combinatorial problems. The proposed algorithm is developed to solve mainly two sub-problems. Sub-problem 1 (SP1) is to assign operations to each machine, i.e. OMAP, such that a machine is dedicated to a set of operations. Sub-problem 2 (SP2) is to select machines to process certain operations out of the set and determine the processing order of jobs on each machine, i.e. FJSP. The model objectives are the minimization of makespan and machine idle cost (MIC) as shown in equation (1). Despite the fact that all machines are identical or flexible, the introduction of resource constraints will implicitly control the processing flexibility of each machine with fixed m. One reason is that if the degree of flexibility is too high, machines surely will process more operations. Since resources are limited, more flexible machines cannot guarantee that more operations will be executed on each machine. In other words, the total required processing time of each operation ($o$load) is shared by a number of machines ($m$load). If more flexible machines are available, $m$load of each machine tends to be small. In this case, MIC may be huge and makespan may not be satisfactorily minimized. Also, many manufacturers may have difficulties in dealing with the existing processing flexibility due to poor production planning. However, if the proposed algorithm is implemented, the assignment of operations–machines and shop floor performance can be optimized with the proper operations–machines mapping, operations–machines selection and job processing orders on each machine. This win–win situation may possibly provide practical values to manufacturers under severe cost control and global competition.

$$\text{Machine idle cost (MIC)} = \sum_{h=1}^{o}\sum_{i=1}^{m}\left(1 - \frac{m\text{load}_{hi}}{o\text{load}_{h}}\right) \times \text{PC}_{i} \tag{1}$$

2. Literature review

Although the complexity of FJSP is well-known, there are far more literatures focusing on JSP rather than on FJSP. Brucker and Schile (1990) are the pioneers in addressing FJSP. They proposed a polynomial algorithm to solve the assignment and scheduling problems in FJSP with two jobs using two types of approaches: hierarchical and integrated. In the hierarchical approach, the assignment and scheduling problems are solved independently. For examples of hierarchical approaches, see Brandimarte (1993) and Tung *et al.* (1999). In contrast, the integrated approach solves the assignment of operations and machines and the sequencing of jobs on machines dependently. For applications of the integrated approach, see Hurink *et al.* (1994) and Dauzère-Pérès and Paulli (1997). In the present paper, the proposed algorithm follows the integrated approach.

In general, there are different types of FJSP. Type I FJSP refers to a situation that jobs can have alternative operation sequences and alternative identical or non-identical machines for each operation. The problems then are to select operation sequences for jobs and determine job processing orders on machines. For work

on type I, see Baykasoğlu (2002), Choi and Choi (2002), Kacem *et al*. (2002) and Kim *et al*. (2003). Baykasoğlu proposed an approach based on linguistics, simulated annealing and a dispatching rule-based heuristic. He defined type I with alternative process plans (operation sequences). In addition, resource elements (RE), which mainly define the shared and unique capabilities of machine tools, have been proposed to denote operations. In the above studies, it has been claimed that RE could cluster machine capabilities and reduce the detail level by denoting unique and overlapping capabilities deliberately. However, experiments did not show clearly the extent of improvement over the traditional representation 'operations'. Also, resource constraints (fixtures and tools) are ignored. Kim *et al*. proposed an evolutionary algorithm to solve a type I in two sub-problems: (1) selection of processing plans to balance machine loads and (2) minimization of makespan and mean flow time in type I. But resource constraints are again neglected. Choi and Choi developed a mixed-integer program model and local search algorithm for solving a type I. Moreover, they considered explicitly sequence-dependent set-up. Computational results have shown that the local search scheme has enhanced the performance of several greedy based scheduling heuristics with reduced computational time. Kacem *et al*. presented a new pareto approach based on a fuzzy evolutionary optimization to solve a type I with a multi-objective function including makespan, total machine workload and the workload of the most loaded machine. They successfully demonstrated the robustness of the approach by showing that the obtained values of the multi-objective function are always near optimum. A shortcoming of their model is that they only defined a simple resource constraint on machines, i.e. the machine is idle or busy, which is not practical to modern a manufacturing environment.

A type II FJSP is when jobs can only have fixed operation sequences but alternative identical or non-identical machines for each operation (Lee *et al*. 1997, Norman and Bean 1999, Mastrolilli and Gambardella 2000, Rossi and Dini 2000, 2001). The problem is then to arrange jobs to machines with respect to their operation sequences such that the objective function is optimized. Norman and Bean developed a GA using random keys representation, elitist reproduction, Bernoulli crossover and immigration-type mutation to minimize the total tardiness with multiple non-identical machines. Resource constraints are defined as machine downtime and scarce tools only. However, they showed the superiority of the proposed GA over other benchmarked algorithms. Mastrolilli and Gambardella proposed a neighbourhood function for type II. They developed a tabu search method incorporating two neighbourhood functions to minimize the makespan solely. They proposed that a neighbourhood function is driven by removing and inserting an operation. For example, an operation is removed from its machine sequence and inserted to a chosen machine sequence such that the schedule is feasible until the insertion is the one with the lowest makespan. Computational experiments show that the proposed approach found 120 new upper bounds and 116 optimal solutions over 221 test problems. However, no attention is given to resource limitations. A GA-based approach called REGAL developed by Rossi and Dini (2000) is applied on type II for minimizing makespan (Rossi and Dini 2001). They explicitly considered sequence-dependent set-up and transaction time when scheduling jobs to several identical machines under a flexible manufacturing system (FMS). They successfully applied REGAL to several benchmarked problems and optimal

schedules were obtained almost for all problems. However, one assumption of their studies was that the assignment of operations–machines is somewhat restricted by merely adding more machines to process the same operation only, i.e. m is not fixed. This, however, may impede the processing flexibility of each machine to other operations implicitly. In fact, it is quite often to fix m due to an investment problem in real-life situations. Also, alternative operation sequences in type I are not always possible in a real-life manufacturing environment, e.g. a precision machining workshop requires ordered machining steps to ensure the tolerance level. Therefore, the current study will mainly focus on the OMAP and FJSP under resource constraints. According to the authors' knowledge, there is similar researchers trying to quantify the processing flexibility of machines in RCFJSP (Caprihan and Wadhwa 1997, Wadhwa and Rao 2000). A detailed description of FJSP will be given in the following section.

The paper is organized as follows. The problem nature and complexity are explained in Section 3. The proposed algorithm is depicted in Section 4. Experiments to examine the performance of the proposed algorithm are given in Section 5. An application of the proposed algorithm is illustrated in Section 6. Conclusions are given in Section 7 together with future research directions.

3. Problem nature and complexity

In this paper, FJSP defines a scenario that jobs can only have fixed operation sequences, but there are alternative identical (or flexible) machines for each operation. In this connection, Pt_{jk} is independent of M_i. Also, an operation-dependent set-up instead of a sequence-dependent set-up is considered. The reason is that the pallet design of modern CNC machines can be easily applied to different fixture types. It means that only if fixture changeover is required between jobs can set-up time be counted on the current operation of a job. Since all machines can process all operations, it is important to determine the operations–machines assignment for minimizing MIC and to solve FJSP for keeping a satisfactory shop floor performance (i.e. makespan). Hence, the problem can be divided into two sub-problems, SP1: OMAP; and SP2: FJSP. To describe clearly the problem, take the sample workshop (figure 1) as an example. Suppose there are total eight machines and four operations. SP1 is first to determine the assignment of operations and machines. It means that if Op_x and Op_y are assigned to M_i, M_i can only process Op_x and Op_y. Given OS_{jk} for $j = 1, \ldots, n$ and $k = 1, \ldots, ON_j$, SP2 is to determine the processing sequence of J_j on the machine chosen from the predefined machine set for the operation. For example, if the operation sequence

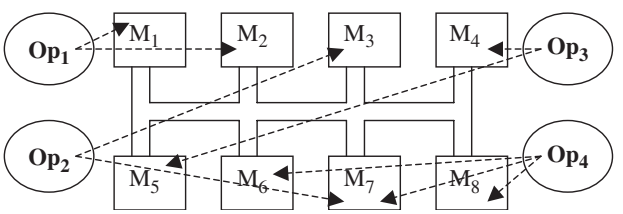

Figure 1. Job-shop with eight machines and four operations.

of J_1 is $Op_1 \rightarrow Op_2 \rightarrow Op_3 \rightarrow Op_4$ and the assignment of operations and machine is obtained as $Op_1\{M_1, M_2\}$, $Op_2\{M_3, M_7\}$, $Op_3\{M_4, M_5\}$, $Op_4\{M_6, M_7, M_8\}$ (figure 1), M_1 or M_2 can be selected to process Op_1 of J_1; M_3 or M_7 for Op_2; M_4 or M_5 for Op_3; M_6 or M_7 or M_8 for Op_4 and the processing orders of J_1 on the selected machine will be determined also. Under resource restriction, it is unwise to allow minimum or maximum machine flexibility. The former case will cause more delayed jobs due to long queuing time on non-flexible machines. The latter one also will lead to high investment cost for configuring machines to adapt to more resource types. In this connection, this research tends to seek a balance between resource limitation and machine flexibility.

The problem complexities can be stated as follows: multiple flexible machines for operations, fixture and tool constraints, operation precedent constraints, and operation-dependent set-up time. 'Fixture' is used to fix a job on the machine for completing operations. Since jobs may have different geometries, each job should have an associated fixture, otherwise, no operation can be started. 'Tool' is to complete an operation on jobs and each tool is dedicated to one specific operation. Therefore, an operation of a job can be processed on a machine only if the associated fixture and tool are available. The complexity of resource constraints is further extended by assuming that each operation of J_j requires only one tool T_{jk}, but no duplicated tool is permitted for the same job where $j = 1, \ldots, n$ and $k = 1, \ldots, ON_j$. For example, if $ON_j = 5$, it is required to have one fixture (F_j) and five distinct tools (T_{jk} where $k = 1, \ldots, ON_j$) in order to complete all operations of J_j. It is assumed that all fixtures and tools are renewable resources. At any time interval t, M_i starts processing J_j. F_j and T_{jk} can be released after time $t + SU_{jk} + Pt_{jk}$ if fixture changeover is required, otherwise $SU_{jk} = 0$. Therefore, the mathematical formulation of the problem can be stated as follows:

Objective function (Z_1) of SP1 (OMAP):

$$\begin{aligned} \text{Min.}Z_1 &= W1 \times Z_2 + W2 \times \text{Machine idle cost (MIC)} \\ &= W1 \times Z_2 + W2 \times \sum_{h=1}^{o} \sum_{i=1}^{m} \left(1 - \frac{m\text{load}_{hi}}{o\text{load}_h}\right) \times \text{PC}_i \end{aligned} \tag{2}$$

where $W1 + W2 = 1.0$, $W1 \rightarrow$ weighting on Z_2 and $W2 \rightarrow$ weighting on MIC.

Objective function (Z_2) of SP2 (FJSP):

$$\text{Min.}Z_2 = \max.\{C_j\} \quad \forall j = 1, \ldots, n \tag{3}$$

Subject to constraints:

$$St_{jk+1} \geq SU_{jk} + Pt_{jk} + St_{jk} \quad \forall j = 1 \ldots n, k = 1, \ldots, ON_j - 1 \tag{4}$$

$$St_{jk} \geq 0 \quad \forall j = 1 \ldots n, \; k = 1, \ldots, ON_j \tag{5}$$

The objective function (Z_1) of SP1 is the function of Z_2 and MIC as shown in equation (2). The objective function (Z_2) of SP2 is the minimization of makespan defined in equation (3) subject to constraints (4) and (5). Constraint (4) ensures the processing order of operations of each job corresponding to the predetermined order and (5) defines the earliest start time of all operations.

4. Proposed algorithm

Since RCFJSP is NP-hard (Hartmann 2002, Kacem *et al.* 2002, Mastrolilli and Gambardella 2002), GA is employed due to its best-known optimizing ability for classical JSP. Figure 2 gives an overview of the proposed algorithm. GA1 is applied to solve SP1: OMAP and GA2 is developed to solve SP2: FJSP. For each SP1 solution, a new SP2 is formed. After SP2 is solved, the results are employed to evaluate the quality of this SP1 solution. Based on the evaluation, GA1 will evolve to obtain better SP1 solutions. In this connection, the proposed algorithm is solving SP1 and SP2 iteratively (or in serial), namely iterative GA (IGA). For OMAP, the assignment of operations and machines is determined by the algorithm, unlike the common approach which assume the assignment is already known so that the results may be only sub-optimized. According to the authors' knowledge, there is currently no research applying this innovative approach to the same problem.

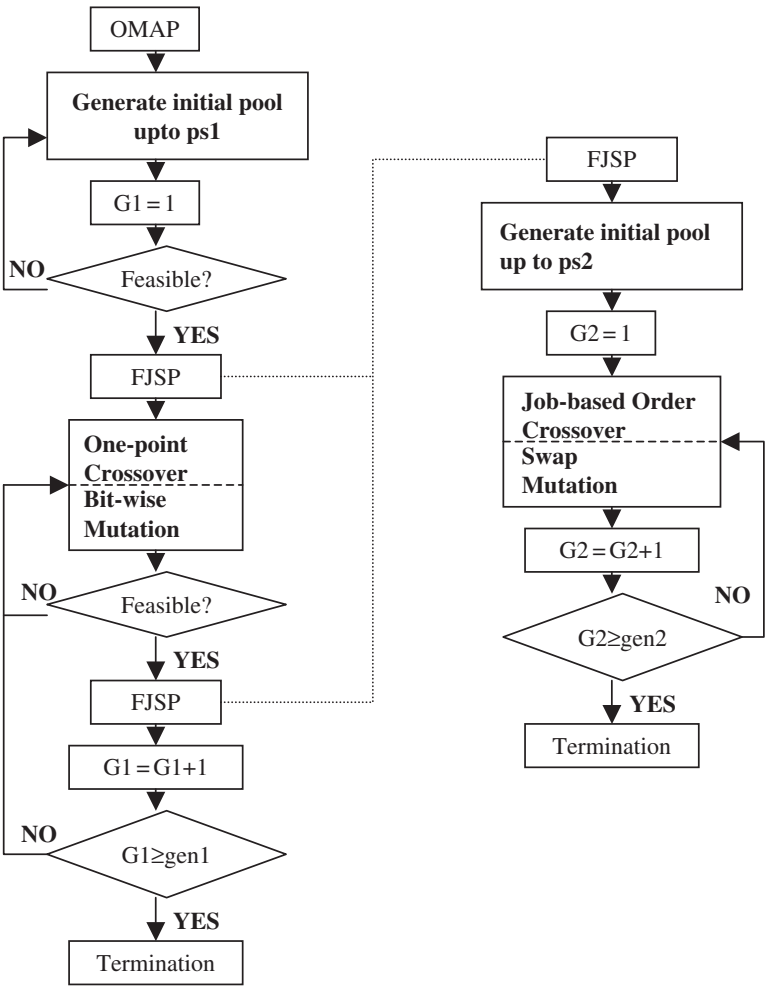

Figure 2. Overview of the proposed algorithm.

4.1 *Solution encoding*

To implement GA successfully, solutions should be encoded into so-called chromosome structure. Let $\Phi(1,\ldots,o,\ 1,\ldots,m)$ and $\Omega(1,\ldots,m,\ 1,\ldots,n)$ both be two-dimensional structure of chromosomes to GA1 and GA2, respectively, so each GA1 solution (Φ) can be defined as:

$$\Phi = \begin{pmatrix} (1,1) & (1,2), & \ldots, \\ (1,m) & & \\ (2,1) & (2,2), & \ldots, \\ (2,m) & & \\ ,\ldots, & & \\ ,\ldots, & & \\ o,1 & o,2, & \ldots, \\ o,m & & \end{pmatrix}$$

where $\Phi(x1,\ y1)$ is a binary gene (0 or 1).

If $\Phi(x1,\ y1)=1$, it means that M_{y1} can process Op_{x1}, otherwise M_{y1} is not allowed for Op_{x1}. Rule (6) demands each operation should be assigned to at least one machine. Rule (7) requires each machine should at least process one operation.

Rules:

$$\sum_{y1=1}^{m} \Phi(x1,y1) \geq 1 \quad \forall x1 = 1,\ldots,o \tag{6}$$

$$\sum_{x1=1}^{o} \Phi(x1,y1) \geq 1 \quad \forall y1 = 1,\ldots,m \tag{7}$$

Each GA2 solution (Ω) can be defined as:

$$\Omega = \begin{pmatrix} (1,1)\ (1,2),\ldots,(1,n) \\ (2,1)\ (2,2),\ldots,(2,n) \\ ,\ldots, \\ ,\ldots, \\ (m,1)(m,2),\ldots, \\ (m,n) \end{pmatrix}$$

where $\Omega(x2,\ y2)$ is the job located at position $y2$ in the processing sequence on machine M_{x2}.

Suppose $\Omega(u1,\ v1)=\mathrm{J}_a$ and $\Omega(u2,\ v2)=\mathrm{J}_b$. If $u1=u2$ and $v1 < v2$, it indicates that J_a is ranked above J_b on $\mathrm{M}_{u1}=u2$. Applying the earliest possible start time heuristic, J_a will be assigned to one machine M_{y1} (with the earliest St_{ay1}) for processing kth operation if $\Phi(\mathrm{OS}_{ak},\ y1=1,\ldots,\ m)=1$ and the associated resources (fixture and tool) are available, similarly for J_b. If both jobs compete on the same machine, J_a is more preferable than J_b since $v1 < v2$. If $u1 \neq u2$, no comparison between J_a and J_b should be made. This representation scheme (called a current scheme) allows available jobs with higher priorities in the list to be processed first, unlike the

absolute one (called absolute scheme) that strictly control the job processing orders. Therefore, non-delay schedules would be constructed. Hartmann (2002) studied the self-adapting GA with the above two solution representation schemes for optimizing some measures. This self-adapting GA will converge in such a way that solutions encoded in one of the schemes will grow while another will fade out. Experimentally, if resources are scarce, 76% of the final population contains the solutions encoded in the current scheme while the remaining belongs to the absolute scheme. Therefore, it is believed that the current scheme would work well to the current problem under tight resource level. Rule (8) defines the feasible genes to $\Omega(1, \ldots, m, 1, \ldots, n)$ and (9) is self-explanatory.

Rules:

$$1 \leq \Omega(x2, y2) \leq n \quad \forall x2 = 1 \ldots m, y2 = 1, \ldots, n \tag{8}$$

$$\text{Any } J_j \text{ should only appear once in } \Omega(1, \ldots, m, y2) \text{ for all } y2 = 1, \ldots, n \tag{9}$$

For example, if $o = 3$ and $m = n = 4$, solutions in GA1 and GA2 can be encoded as:

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix}$$

The top GA1 solution refers to Op_1 $\{M_1, M_2\}$, Op_2 $\{M_2, M_4\}$ and Op_3 $\{M_3, M_4\}$. The bottom GA2 solution means M_1: $J_1 > J_2 > J_3 > J_4$; M_2: $J_2 > J_1 > J_3 > J_4$; M_3: $J_1 > J_3 > J_4 > J_2$; and M_4: $J_3 > J_2 > J_4 > J_1$. Note: M_i: $J_a > J_b$ means J_a is more preferable than J_b on M_i.

4.2 *Fitness evaluation*

For each GA1 chromosome (SP1 solution), the best GA2 chromosome (SP2 solution) is obtained by solving SP2. Then ps1 × gen1 GA1 chromosomes are evaluated in this way until the best combination of GA1 and GA2 chromosomes is obtained with the optimum objective value. Both GA1 and GA2 employ equation (10) to transform the objective value into normalized fitness value (FV). Chromosomes with higher FV will have larger opportunities to be selected in pairs. Each pair of selected chromosomes will perform crossover and mutation operations. Then only their offspring will enter the new generation. Some of the advantages using weighted performance criteria in scheduling can be referred to Hui and Lau (2003) and Wong *et al.* (2003).

$$\text{Fitness value (FV)} = \frac{\text{Max.} - Z}{\text{Max.} - \text{Min.}} \tag{10}$$

$$\text{MAX} \rightarrow \text{maximum } Z \text{ over the same generation}$$
$$\text{MIN} \rightarrow \text{minimum } Z \text{ over the same generation}$$

4.3 *Crossover and mutation*

Besides solution encoding, it is also important to decide the crossover and mutation operators. It is noted that the selection process for crossover operation follows a roulette wheel selection scheme. This scheme selects a pair of chromosomes for mating with a probability proportional to its FVs. For GA1, one-cut-point crossover (1X) operator and bit-wise mutation which work well with bit string (Cheng *et al.* 1999) are applied. To this end, any GA1 solution that violates (6) or (7) will be discarded and regenerated again until a feasible solution is obtained. Then a pair of offspring will enter the new generation each time after crossover and mutation operations are executed on a pair of parental chromosomes. For GA2, job-based order crossover (JOX) operator that preserves the job order on all machines well (Ono *et al.* 1996) is used. Mutation is by randomly swapping the processing order of two jobs on the same machine. Since job processing preference list on each machine is employed, infeasible GA2 solutions will not be generated. Like GA1, only offspring of their parental chromosomes will enter the new generation each time. For more information on crossover and mutation operators, please refer to Cheng *et al.* (1999).

The mechanism of 1X and bit-wise mutation are as follows:

(i) Choose 2 Φ, namely $\Phi 1$ and $\Phi 2$
(ii) Generate one random cut point r, where $1 \le r < m$
(iii) For example,

$$\text{if } r=2,\ \Phi 1\begin{pmatrix}1&1&0\\1&0&0\\0&1&1\end{pmatrix}+\Phi 2\begin{pmatrix}1&1&1\\0&1&1\\0&1&0\end{pmatrix}=\Phi 1'\begin{pmatrix}1&1&1\\1&0&1\\0&1&0\end{pmatrix}$$

} Crossover

(iv) Generate a number $r1$, where $1 \le r1 \le o$
(v) Generate $r2$ and $r3$, where $1 \le r2 \le o$ and $1 \le r3 \le m$
(vi) If $\Phi 1'(r2, r3) = 1$ or 0, then set $\Phi 1'(r2, r3) = 0$ or 1
(vii) Repeat (v) to (vi) $r1$ times
(viii) The same to $\Phi 2'$

} Mutation

The mechanism of JOX and Swap mutation are as follows:

(i) Choose 2 Ω, namely $\Omega 1$ and $\Omega 2$
(ii) Generate a random number p, where $1 \le p \le n$
(iii) Generate a number of distinct jobs up to p and store p jobs in a set Δ
(iv) Imitate J_j from $\Omega 1(1, \ldots, m, 1, \ldots, n)$ if $J_j \in \Delta$ to $\Omega 1'$
(v) For example, if $p = 1$ and

$$\Delta\{J_1\},\ \Omega 1\begin{pmatrix}J_1&J_2&J_3\\J_1&J_3&J_2\\J_3&J_2&J_1\end{pmatrix}=>\Omega 1'\begin{pmatrix}J_1&x&x\\J_1&x&x\\x&x&J_1\end{pmatrix}$$

(vi) Imitate J_j from $\Omega 2(1, \ldots, m, 1, \ldots, n)$ if $J_j \notin \Delta$ to $\Omega 1'$
(vii) Then, $\Omega 2\begin{pmatrix}J_3&J_1&J_2\\J_1&J_2&J_3\\J_2&J_1&J_3\end{pmatrix}=>\Omega 1'\begin{pmatrix}J_1&J_3&J_2\\J_1&J_2&J_3\\J_2&J_3&J_1\end{pmatrix}$

} Crossover

(viii) Generate a number $p1$, where $1 \le p1 \le m$
(ix) Generate $p2$ and $p3$, where $1 \le p2, p3 \le n$
(x) Interchange $\Omega 1'(p1, p2)$ and $\Omega 1'(p1, p3)$
(xi) The same to $\Omega 2'$

} Mutation

5. Computational results

In this section a number of test problems will be employed to examine the performance of the proposed algorithm. It is noted that the proposed algorithm IGA has been coded in C language and tested on a Pentium computer. First, 15 classical JSP problems (one machine to one operation), namely LA01–15 from Lawrence (1984), will be tested on IGA and the results compared with REGAL, which has been tested on a computer, reported by Rossi and Dini (2001). The comparison is shown in table 1. Preliminary experiments suggest that the GA2 parameters are gen2 = 80, ps2 = 80 (200 for REGAL) and mr2 = 0.01. In table 1, IGA outperforms REGAL in five of 15 instances by 0.29–9.38%. Moreover, IGA computes the optimal schedule (marked with a '∗') in 12 of 15 problems while REGAL only gives nine optimal solutions. Column 'D2%' indicates the per cent difference between OPT and IGA varies from −4.36 to 0.0%. The last column shows the computational time for solving each problem using IGA. On average, REGAL takes 7.19 min for each problem and IGA uses only 2.11 min. To solve JSP, one-to-one mapping is applied to M_i and Op_h for all $i = h$, i.e. M_i can only process Op_h. Therefore, the gene structure of GA1 is $\Phi(x1, y1) = 1$ for all $x1 = y1$, otherwise $\Phi(x1, y1) = 0$. Then only GA2 is executed to compute the job processing order on each machine and the actual machine sequence of all jobs.

Next, ten type II FJSP will be tested by modifying LA01–10 as reported by Rossi and Dini (2001). For each of the five operations, two machines are available to process it. Also each job is duplicated once, i.e. two copies of the same job co-exists. Hence, for five operations and ten jobs, there will be ten machines and 20 jobs after the modification, i.e. $o = 5$, $m = 10$ and $n = 20$. Comparison again is made between REGAL and IGA as shown in table 2 for modified Lawrence instances (Lawrence 1984), namely LA01′–10′. Experimentally, IGA outperforms REGAL in nine type II FJSP by 2.63–13.22%. It is expected the performance

Table 1. REGAL versus IGA on JSP.

Problems	$o(m)$	n	OPT	REGAL	IGA	D1%	D2%	Minimums
LA01	5	10	666	668	666∗	0.29	0.0	1.46
LA02	5	10	655	670	670	0.0	−2.29	1.55
LA03	5	10	597	648	623	3.86	−4.36	1.58
LA04	5	10	590	614	611	0.49	−3.56	1.55
LA05	5	10	593	593∗	593∗	0.0	0.0	1.52
LA06	5	15	926	926∗	926∗	0.0	0.0	2.03
LA07	5	15	890	943	890∗	5.62	0.0	2.08
LA08	5	15	863	863∗	863∗	0.0	0.0	2.08
LA09	5	15	951	951∗	951∗	0.0	0.0	2.23
LA10	5	15	958	958∗	958∗	0.0	0.0	2.12
LA11	5	20	1222	1222∗	1222∗	0.0	0.0	2.82
LA12	5	20	1039	1039∗	1039∗	0.0	0.0	2.63
LA13	5	20	1150	1150∗	1150∗	0.0	0.0	2.62
LA14	5	20	1292	1292∗	1292∗	0.0	0.0	2.68
LA15	5	20	1207	1332	1207∗	9.38	0.0	2.75

D1% = (REGAL − IGA)/REGAL × 100%.
D2% = (OPT − IGA)/OPT × 100%.

Table 2. REGAL versus IGA on type II FJSP.

Problems	o	m	n	OPT	REGAL	IGA	D1%	D2%	Minimums
LA01′	5	10	20	666	717	668	6.83	−0.3	5.75
LA02′	5	10	20	655	770	692	10.13	−5.65	5.83
LA03′	5	10	20	597	734	637	13.22	−6.7	6.13
LA04′	5	10	20	590	681	629	7.64	−6.61	5.87
LA05′	5	10	20	593	638	595	6.74	−0.34	5.95
LA06′	5	10	30	926	951	926∗	2.63	0.0	8.13
LA07′	5	10	30	890	963	891	7.48	−0.11	8.62
LA08′	5	10	30	863	966	863∗	10.66	0.0	8
LA09′	5	10	30	951	984	954	3.05	−0.32	7.75
LA10′	5	10	30	958	958∗	958∗	0.0	0.0	8.27

D1% = (REGAL − IGA)/REGAL × 100%.
D2% = (OPT − IGA)/OPT × 100%.

difference will be enlarged when problem size grows with m and n. Also, column 'D2%' indicates the percent difference between OPT and IGA varies from −6.7 to 0.0%. Regarding computational time, REGAL takes 7.7–320.6 min while IGA takes 5.75–8.62 min to solve ten type II FJSP. In fact, IGA is tested on a faster computer; however, REGAL shows instability (7.7–320.6 min) when solving a type II FJSP due to the algorithm itself. To solve a type II FJSP, two-to-one mapping is applied to M_i and Op_h, i.e. two machines can process one specific operation. Therefore, the gene structure of GA1 is $\Phi(x1, y1) = 1$ for $x1 = 2(y1) - 1, 2(y1)$ where $x1 = 1, \ldots, o$ and $y1 = 1, \ldots, m$, otherwise $\Phi(x1, y1) = 0$. Afterwards, GA2 can be executed to compute the job processing order on each machine and actual machine sequence of all jobs.

For instance, a workshop with a number of CNC machines is a typical problem background of the proposed algorithm. Due to the robust design of CNC machines, it is possible that all machines are functionally identical and can process the same operations. To examine the performance of IGA, some harder JSP will be employed to model a CNC workshop. First, ten harder JSP (seven from Lawrence 1984; three from Fisher and Thompson 1963) will be chosen and modified to be harder FJSP, i.e. one machine to many operations instead of one machine to one operation is allowed. Then IGA is applied to minimize the makespan without set-up time and resource constraints. The results will be compared with the theoretical lower bound (LB) defined in equation (11). Experimentally, GA1 is compared with a random (R) approach to OMAP and GA2 is the only approach to FJSP. The results are shown in table 3, where GA1 can give schedules 47.41% worse than LB, 16.1% better than OPT (optimum makespan with no flexibility), and 6.81% better than R on average. If ratio m/n is small, i.e. $n > > m$, it is hard to obtain schedules near to the LB as illustrated in FT5 × 20 and LA31. However, the performance gains are significant (16.1%) if flexibility is considered in the scheduling process:

$$\mathrm{LB} = \max\left\{\sum_{k=1}^{\mathrm{ON}_j} \mathrm{Pt}_{jk}\right\} \quad \forall j = 1, \ldots, n \tag{11}$$

Next, if resource constraints are introduced, the previous ten FJSP will be modified to RCFJSP as follows: $o = m$, $\mathrm{TF} = 0.5n$, $\mathrm{TT} = 0.4(o \times n)$, SU_{jk} [1, 5],

Table 3. GA1 versus *R* on harder FJSP.

Problems	*o*(*m*)	*n*	LB	OPT	R	GA1	D1%	D2%	D3%
1. FT6 × 6[a]	6	6	47	55	47	47	0.0	14.55	0.0
2. FT5 × 20[a]	5	20	387	1165	1325	1030	−166.15	11.59	22.26
3. FT10 × 10[a]	10	10	655	930	657	655	0.0	29.57	0.3
4. LA02	5	10	394	655	573	565	−43.4	13.74	1.4
5. LA03	5	10	349	597	588	503	−44.13	15.75	14.46
6. LA04	5	10	369	590	615	508	−37.67	13.9	17.4
7. LA16	10	10	717	945	717	717	0.0	24.13	0.0
8. LA21	10	15	717	1046	870	852	−18.83	18.55	2.07
9. LA29	10	20	723	1130	1150	1050	−45.23	7.08	8.7
10. LA31	10	30	717	1784	1592	1568	−118.69	12.11	1.51
						Average	−47.41	16.1	6.81

D1% = (LB − GA1)/LB×100%.
D2% = (OPT − GA1)/OPT×100%.
D3% = (*R* − GA1)/*R*×100%.
[a] Source: Fisher and Thompson (1963).

Table 4. Experimental shop floor environment.

Problems	LB	*o*(*m*)	*n*	TF	TT	AF	AT
1. FT6 × 6	47	6	6	3	14	3	2.77
2. FT5 × 20	387	5	20	10	40	2.22	2.63
3. FT10 × 10	655	10	10	5	40	2.5	2.56
4. LA02	394	5	10	5	20	2	3.13
5. LA03	349	5	10	5	20	2	3.13
6. LA04	369	5	10	5	20	2	3.13
7. LA16	717	10	10	5	40	2.5	2.63
8. LA21	717	10	15	7	60	2.5	2.68
9. LA29	723	10	20	10	80	2.22	2.63
10. LA31	717	10	30	15	120	2	2.7

AF, average number of jobs per fixture type; AT, average number of operations per tool type.

PC_i [0.01, 1], F_j [1, TF] and T_{jk} [1, TT] (table 4). AF represents the average number of jobs sharing the same fixture type, while AT indicates the average number of operations sharing the same tool type, respectively. If AF and AT are large, it implies that the shop performance is highly sensitive to resource level (RL). In this paper, RL is controlled by CF_f and CT_t. To illustrate the impact of RL, two RL extremes are modelled: tight and loose. Tight RL means that CF_f and $CT_t = 1$ for all t and f, hence there is no extra copy for each fixture and tool, i.e. high share rate of resources. Loose RL means that CF_f = total number of jobs for which $F_j = f$ and CT_t = total number of jobs for which $T_{jk} = t$, where $j = 1, \ldots, n$ and $k = 1, \ldots, ON_j$, so there will be no resource shortage. If RL is tight, large (small) AF and AT imply greater (smaller) chances to queue up for fixtures and tools, hence they increase average job processing lead-time. If RL is loose, the shop floor performance is free of AF and AT. In this experiment, AF and AT varies from 2 to 3 and from 2.56 to 3.13, respectively. First, the objective is to minimize makespan, i.e. $W1 = 1$ and $W2 = 0$,

Table 5. Makespan minimization under two RLs.

		Loose		Tight	
Problems	LB	Makespan	MIC	Makespan	MIC
1	47	49	227.32	138	94.08
2	387	1044	596.77	1435	246.11
3	655	656	536.43	3659	249.88
4	394	566	337.23	963	108.71
5	349	507	339.17	887	109.15
6	369	511	333.65	823	105.35
7	717	722	635.48	2667	269.09
8	717	868	811.68	2159	308.69
9	723	1068	1128.8	2158	451.68
10	717	1595	2078.98	2099	962.58
Average		758.6	702.55	1698.8	290.53

Table 6. MIC minimization under two RLs.

	Loose		Tight	
Problems	Makespan	MIC	Makespan	MIC
1	49	216.53	162	79.45
2	1123	596.05	1458	240.58
3	662	509.91	3674	237.4
4	585	334.86	975	102.31
5	529	336.93	903	102.36
6	536	331.91	867	99.34
7	735	609.98	2700	254.98
8	931	811.48	2209	300.09
9	1090	1101.11	2244	422.47
10	1679	2056.37	2173	938.95
Average	791.9	690.51	1736.5	277.79

with set-up time and resource constraints. Preliminary experiments suggest that gen1 = 10, ps1 = 10 and mr1 = 0.01 for GA1. Table 5 shows the makespan and MIC in the loose and tight RLs. The makespan obtained under loose RL (758.6) is lower than that under tight RL (1698.8) on average. MIC obtained under loose RL (702.55) is higher than that under tight RL (290.53) on average. These results show that RL is highly correlated to the operations–machines assignment. On the other hand, in order to optimize the machine utilization, the objective here is to minimize MIC, i.e. $W1 = 0$ and $W2 = 1$. Similar results (as reported in table 5) are obtained in table 6.

The results also indicate that IGA always can minimize makespan or MIC with respect to the corresponding objective function under two different RLs. In this connection, it is shown that IGA can work well to the objective weightings. Also, it is noted that fewer operations are assigned to each machine in consideration of machine idleness, thus the lowest MIC (average 284.16) is obtained under tight

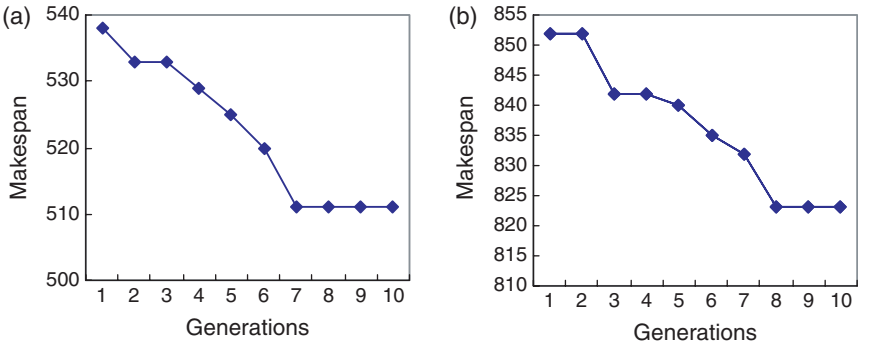

Figure 3. Convergence of makespan under (a) loose and (b) tight resource levels.

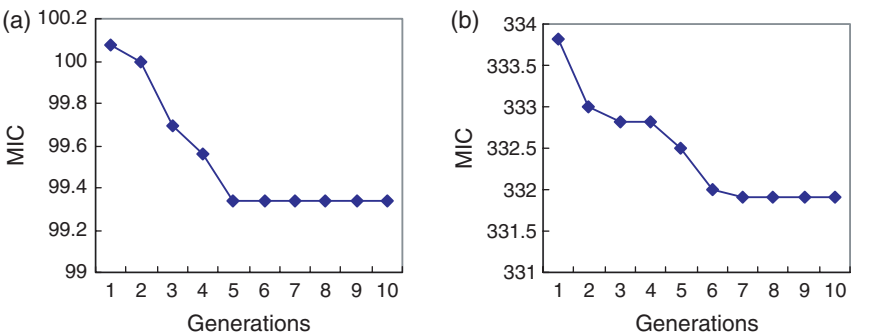

Figure 4. Convergence of MIC under (a) loose and (b) tight resource levels.

RL as compared with the highest MIC (average 696.53) under loose RL. However, the worst shop floor performance or makespan (average 1717.65) is obtained under tight RL as compared with the best performance (average 775.25) under loose RL. To this end, the experiment has explicitly indicated the potential conflicts between makespan and MIC in RCFJSP. There are also two extremes noted in RCFJSP: (1) the system under loose RL favours makespan minimization ($W1 = 1$ and $W2 = 0$) and (2) the system under tight RL favours MIC minimization ($W1 = 0$ and $W2 = 1$). In reality, the trade-off between makespan and MIC is also vital under different RLs but always difficult to obtain as well as to quantify. Thus, the proposed model may provide solutions to this type of problem. Figures 3 and 4 show the convergence rate of IGA to minimize makespan and MIC in LA04 under two RLs respectively. Generally speaking, the results indicate that sound convergence rate over generations is obtained.

6. Illustrative example

In this section, the proposed model will be applied to an example of $o = m = 5$ and $n = 10$, hence $TF = 5$ (five distinct fixture types) and $TT = 20$ (20 distinct tool types) under tight RL. The job information is given in table 7. For example, the

Table 7. Job information.

J_j	F_j	ON_j	SU_{jk}	T_{jk}	OS_{jk} (Pt_{jk})
1	5	5	5, 2, 4, 5, 5	6, 18, 10, 17, 13	2(6), 1(5), 4(14), 5(5), 3(1)
2	1	5	1, 2, 1, 5, 1	17, 15, 9, 18, 10	2(19), 3(8), 4(10), 5(16), 1(16)
3	5	5	4, 2, 4, 1, 2	12, 4, 16, 17, 19	3(6), 4(17), 2(16), 1(6), 5(17)
4	4	5	5, 2, 5, 2, 2	14, 16, 19, 13, 2	5(11), 1(4), 2(7), 3(8), 4(12)
5	3	5	5, 3, 3, 1, 2	6, 4, 9, 11, 3	1(17), 4(17), 2(10), 5(18), 3(17)
6	1	5	2, 5, 3, 5, 5	11, 17, 9, 6, 16	4(18), 1(18), 2(8), 3(5), 5(12)
7	3	5	2, 3, 2, 4, 4	6, 18, 4, 2, 19	3(8), 2(12), 1(14), 5(10), 4(7)
8	2	5	5, 5, 2, 4, 3	18, 1, 4, 10, 6	2(8), 4(2), 1(2), 3(7), 5(11)
9	1	5	3, 4, 2, 5, 2	12, 3, 17, 7, 5	2(10), 3(16), 1(20), 5(16), 4(15)
10	1	5	5, 5, 4, 2, 4	9, 3, 16, 18, 1	5(13), 3(7), 4(9), 2(20), 1(19)

information of J_1 is as follows: fixture type is 5, number of operations is five, operation-dependent set-up time for the five operations (from left to right), tool types for the five operations, and the operation sequence for the five operations with the associated processing time in blanket.

Applying the proposed model, decision-makers then can assign relative weightings among makespan and MIC. Since there are limited approaches to this problem, Random (R) machine assignment again is employed to compare with IGA. If $W1 = 1$ and $W2 = 0$, makespan is the only objective now. The model generates schedule with makespan = 276 ($R = 277$), MIC = 132.42, and average number of operations per machine (ANOM) = 3.0. If $W1 = 0$ and $W2 = 1$, the model will now minimize the MIC only. Schedule with makespan = 289, MIC = 130.78 ($R = 130.83$) and ANOM = 1.8 is obtained. Obviously, the model works well with the objective function. However, in real-life manufacturing world, the trade-off between makespan and MIC is even more important then any single objective. In this connection, if $W1 = W2 = 0.5$, the schedule with makespan = 280 ($R = 281$), MIC = 131.4 ($R = 131.64$), and ANOM = 2.2 is obtained. Experimentally, IGA outperforms R with the same number of iterations in all cases. The details of the schedule generated by IGA under $W1 = W2 = 0.5$ can be shown in table 8. Table 9 shows the comparison between IGA and R for a number of test problems. Results indicate that IGA outperforms R for all problems.

6.1. *Updating procedure*

From the left-hand side of table 8, the machine sequences and completion time of all jobs are obtained via the proposed model. Due to the processing flexibility of machines, jobs can revisit the same machine for different operations. This processing flexibility level is explicitly defined by the operations–machines assignment stated on the right-hand side. It is noted that M_5 does not process any operation and this violate constraint (7). This help to depict the additional function of the proposed model, i.e. the updating procedure. After the feasible schedule is obtained, i.e. all jobs are completed and all constraints are satisfied, the updating procedure will be executed to denote any excessive machine. Excessive machine

Table 8. Schedule under $W1 = W2 = 0.5$.

				M_i				
J_j	MS_{jk}	C_j	Op_h	1	2	3	4	5
1	4, 3, 2, 2, 1	116	1	1	0	1	0	0
2	1, 1, 2, 2, 1	198	2	1	0	1	1	0
3	1, 4, 3, 3, 4	186	3	1	0	0	0	0
4	2, 3, 3, 1, 4	155	4	0	1	0	1	0
5	3, 2, 3, 2, 1	151	5	1	1	0	1	0
6	2, 1, 1, 1, 1	280						
7	1, 3, 3, 2, 2	221						
8	3, 4, 3, 1, 1	181						
9	1, 1, 1, 1, 2	127						
10	1, 1, 2, 1, 1	237						

Table 9. IGA versus R for test problems.

		$W1 = 1;\ W2 = 0$			$W1 = 0;\ W2 = 1$		
$o(m)$	n	R	IGA	D%	R	IGA	D%
5	10	277	276	0.36	130.83	130.78	0.04
5	20	291	286	1.72	250.04	249.61	0.17
5	30	397	390	1.76	387.77	385.24	0.65

$D\% = (R - IGA)/R \times 100\%$.

means that a machine is assigned to process operation(s) but no jobs are actually processed on it for those operation(s) even feasible schedule is generated. In this connection, the updating procedure unassigns the particular operation(s) from the corresponding machine(s). Thus, the model can also denote any excessive machine such that decision-makers can have a better plan to utilize the machines under tight RL.

7. Conclusion

In this study, an innovative GA-based approach called IGA is proposed to RCFJSP. Through several experiments, it is shown that IGA works well with different objective weightings. Also, the proposed algorithm indicates the potential improvement in FJSP especially with multiple identical machines under a resource-constrained environment. Thus, the conflicts between makespan and MIC can be solved even under different RLs. Moreover, the problem can be easily modified to a type II FJSP with multiple non-identical machines under resource constraints. Applying IGA, it is expected that sound trade-off between several objectives can also be obtained in RCFJSP. However, the trend to optimize shop performance with a number of identical or flexible machines is promising because of continuous improvement in manufacturing technology over the past few decades. In this

connection, the proposed model may be the pioneer in quantifying the flexibility between machines under limited resources. It is generally agreed that if processing flexibility can be measured properly, the increase in resource utilization can implicitly lessen the operating cost significantly, which sometimes may be underestimated by some manufacturers. Since the assignment of objective weightings is quite subjective, further investigation will be carried out to this issue.

A.1 Appendix: Notation

JSP	job-shop scheduling problem
FJSP	flexible JSP
RCFJSP	resource-constrained FJSP
OMAP	operations to machines assignment problem
IGA	iterative GA
REGAL	real-time GA
RL	resource level
MIC	machine idle cost
ANOM	average number of operations per machine
FV	fitness
o	total number of operations
m	total number of machines
n	total number of jobs
Op_h	operation h
M_i	machine i
J_j	job j
ON_j	total number of operations for J_j
OS_{jk}	kth operation of J_j
MS_{jk}	machine for kth operation of J_j
ST_{jk}	start time of kth operation of J_j
PT_{jk}	processing time of kth operation of J_j
SU_{jk}	set-up time of kth operation of J_j
C_j	completion time of J_j
PC_i	maximum penalty cost for idleness of M_i
$o\mathrm{load}_h$	sum of PT_{jk} on Op_h if $\mathrm{OS}_{jk} = \mathrm{Op}_h$ for all j, k
$m\mathrm{load}_{hi}$	sum of PT_{jk} on M_i for processing Op_h for all j, k
TF	total types of fixture
TT	total types of tool
AF	average number of jobs share the same fixture type
AT	average number of operations share the same tool
F_j	type
T_{jk}	fixture for J_j
CF_f	tool for kth operation of J_j
CT_t	total copies of fixture f
Z_1	total copies of tool t
Z_2	objective function of GA1
Φ	objective function of GA2
Ω	solution representation (set) to GA1

ps1	solution representation (set) to GA2
ps2	population size of GA1
gen1	population size of GA2
gen2	maximum generation of GA1
mr1	maximum generation of GA2
mr2	mutation rate of GA1 mutation rate of GA2

References

Baykasoğlu, A., Linguistic-based meta-heuristic optimization model for flexible job shop scheduling. *Int. J. Prod. Res.*, 2002, **40**, 4523–4543.

Brandimarte, P., Routing and scheduling in a flexible job shop by taboo search. *Ann. Oper. Res.*, 1993, **41**, 157–183.

Brucker, P. and Schile, R., Job-shop scheduling with multi-purpose machines. *Computing*, 1990, **45**, 369–375.

Caprihan, R. and Wadhwa, S., Impact of routing flexibility on the performance of an FMS — a simulation study. *Int. J. Flex. Manuf. Sys.*, 1997, **9**, 273–298.

Chan, F.T.S., Wong, T.C. and Chan, P.L.Y., A genetic algorithm-based approach to machine assignment problem. *Int. J. Prod. Res.*, 2005, **43**, 2451–2472.

Cheng, R., Gen, M. and Tsujimura, Y., A tutorial survey of job-shop scheduling problems using genetic algorithms, Part II: Hybrid genetic search strategies. *Comput. Ind. Eng.*, 1999, **36**, 343–364.

Choi, I.C. and Choi, D.S., A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent set-up. *Comput. Ind. Eng.*, 2002, **42**, 43–58.

Dauzère-Pérès, S. and Paulli, J., An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Ann. Oper. Res.*, 1997, **70**, 281–306.

Fisher, H. and Thompson, G.L., Probabilistic learning combinations of local job-shop scheduling rules. In *Industrial Scheduling*, edited by J.F. Muth and G.L. Thompson, pp. 225–251, 1963 (Prentice-Hall, Englewood Cliffs, NJ).

Hartmann, S., A self-adapting genetic algorithm for project scheduling under resource constraints. *Nav. Res. Logist.*, 2002, **49**, 433–448.

Holland, J.H., *Adaptation in the Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, MI, 1975).

Hui, I.K. and Lau, H.C.W., A dynamic inventory-production scheduling model for small scale organizations. *Int. J. Adv. Manuf. Tech.*, 2003, **22**, 89–100.

Hurink, J., Jurisch, B. and Thole, M., Tabu search for the job shop scheduling problem with multi-purpose machines. *Oper. Res. Spektr.*, 1994, **15**, 205–215.

Kacem, I., Hammadi, S. and Borne, P., Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *J. Math. Comput. Simul.*, 2002, **60**, 245–276.

Kim, Y.K., Park, K. and Ko, J., A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Comput. Oper. Res.*, 2003, **30**, 1151–1171.

Lawrence, S., *Resource-constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques*, 1984 (Carnegie Mellon University, Pittsburgh, PA) GSIA.

Lee, M.K., Luong, L.H.S. and Abhary, K., A genetic algorithm based cell design considering alternative routing. *Int. J. Comput.-Integr. Manuf.*, 1997, **10**, 93–108.

Mastrolilli, M. and Gambardella, L.M., Effective neighbourhood functions for the flexible job shop problem. *J. Schedul.*, 2000, **3**, 3–20.

Norman, B.A. and Bean, J.C., A genetic algorithm methodology for complex scheduling problems. *Nav. Res. Logist.*, 1999, **46**, 199–211.

Ono, I., Yamamura, M. and Kobayashi, S., A genetic algorithm for job-shop scheduling problems using job-based order crossover, in *Proceedings of an International Conference on Evolutionary Computation*, Nayoya University, Japan, 1996, p. 547–552.

Rossi, A. and Dini, G., An evolutionary approach to complex job-shop and flexible manufacturing system scheduling. *Proc. Inst. Mech. Eng. B*, 2001, **215**, 233–245.

Rossi, A. and Dini, G., Dynamic scheduling of FMS using a real-time genetic algorithm. *Int. J. Prod. Res.*, 2000, **38**, 1–20.

Tung, L.F., Li, L. and Nagi, R., Multi-objective scheduling for the hierarchical control of flexible manufacturing systems. *Int. J. Flex. Manuf. Sys.*, 1999, **11**, 379–409.

Wadhwa, S. and Rao, K.S., Flexibility: an emerging meta-competence for managing high technology. *Int. J. Tech. Manag.*, 2000, **7–8**, 820–845.

Wong, T.N., Chan, L.C.F. and Lau, H.C.W., Machining process sequencing with fuzzy expert system and genetic algorithms. *Eng. Comput.*, 2003, **19**, 191–202.