J Intell Manuf (2007) 18:331–342
DOI 10.1007/s10845-007-0026-8

Mathematical modeling and heuristic approaches to flexible job shop scheduling problems

Parviz Fattahi · Mohammad Saidi Mehrabad · Fariborz Jolai

Received: March 2006/ Accepted: October 2006 / Published online: July 2007

Abstract Scheduling for the flexible job shop is very important in both fields of production management and combinatorial optimization. However, it is quite difficult to achieve an optimal solution to this problem in medium and actual size problem with traditional optimization approaches owing to the high computational complexity. For solving the realistic case with more than two jobs, two types of approaches have been used: hierarchical approaches and integrated approaches. In hierarchical approaches assignment of operations to machines and the sequencing of operations on the resources or machines are treated separately, i.e., assignment and sequencing are considered independently, where in integrated approaches, assignment and sequencing are not differentiated. In this paper, a mathematical model and heuristic approaches for flexible job shop scheduling problems (FJSP) are considered. Mathematical model is used to achieve optimal solution for small size problems. Since FJSP is NP-hard problem, two heuristics approaches involve of integrated and hierarchical approaches are developed to solve the real size problems. Six different hybrid searching structures depending on used searching approach and heuristics are presented in this paper. Numerical experiments are used to evaluate the performance of the developed algorithms. It is concluded that, the hierarchical algorithms have better performance than integrated algorithms and the algorithm which use tabu search and simulated annealing heuristics for assignment and sequencing problems consecutively is more suitable than the other algorithms. Also the numerical experiments validate the quality of the proposed algorithms.

Keywords Flexible job shop · Scheduling · Tabu search · Simulated annealing · Hierarchical approach

P. Fattahi (✉)
Department of Industrial Engineering, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran
e-mail: fattahi@basu.ac.ir

M. Saidi Mehrabad
Department of Industrial Engineering, Iran University of Science & Technology, Tehran, Iran

F. Jolai
Industrial Engineering Department, Faculty of Engineering, University of Tehran, Tehran, Iran

Introduction

Scheduling problems occur in all the economic domains, from computer engineering to manufacturing techniques. Most scheduling problems are complex combinatorial optimization problems and very difficult to solve. The job shop scheduling is a branch of production scheduling, which is among the hardest combinatorial optimization problems. The job shop scheduling problem is to determine a schedule of jobs that have pre-specified operation sequences in a multi-machine environment. In the classical job shop scheduling problem (JSP), n jobs are processed to completion on m unrelated machines. For each job, technology constraints specify a complete, distinct routing which is fixed and known in advance. Each machine is continuously available from time zero, and operations are processed without preemption. The general JSP is strongly NP-hard (Garey, Johnson, & Sethi, 1976). In order to match nowadays market requirements, manufacturing systems have to become more flexible and efficient. To achieve these objectives, the systems need not only the automated and flexible machines, but also the flexible scheduling systems. The flexible job shop scheduling problem (FJSP) extends JSP by assuming that, for each given operation, there is at least one instance of the machine type necessary to perform it. The scheduling problem of a FJSP

consists of a routing sub-problem, that is assigning each operation to a machine out of a set of capable machines and the scheduling sub-problem, which consists of sequencing the assigned operations on all machines in order to obtain a feasible schedule minimizing a predefined objective function. The FJSP mainly presents two difficulties. The first one is to assign each operation to a machine, and the second one is to schedule these operations in order to make a predefined objective minimal (Xia & Wu, 2005). The FJSP is a much more complex version of the JSP, so the FJSP is strongly NP-hard and combinatorial. It incorporates all of the difficulties and complexities of its predecessor JSP and is more complex than JSP because of the addition need to determine the assignment of operations to machines.

The literature of FJSP is considerably sparser than the literature of JSP. Bruker and Schlie (1990) were among the first to address this problem. They developed a polynomial algorithm for solving the flexible job shop problem with two jobs. For solving the realistic case with more than two jobs, two types of approaches have been used: hierarchical approaches and integrated approaches. In hierarchical approaches assignment of operations to machines and the sequencing of operations on the resources or machines are treated separately, i.e., assignment and sequencing are considered independently, where in integrated approaches, assignment and sequencing are not differentiated. Hierarchical approaches are based on the idea of decomposing the original problem in order to reduce its complexity. This type of approach is natural for FJSP since the routing and the scheduling sub-problem can be separated. Brandimarte (1993) was the first to use this decomposition for the FJSP. He solved the routing sub-problem using some existing dispatching rules and then focused on the scheduling sub-problem, which is solved using a tabu search heuristic. Saidi Mehrabad and Fattahi (2007) presented a mathematical model and tabu search algorithm to solve the flexible job shop scheduling problem with sequence-dependent setups. They used a hierarchical approach with two heuristics to solve this problem. The first one for assigning each operation to a machine out of a set of two capable machines and the second one for sequencing the assigned operations on all machines in order to obtain a feasible schedule minimizing the Makespan. Other literatures of this approach were represented by Kacem, Hammadi, and Borne (2002), Xia and Wu (2005) and Low, Yip, and Wu (2006). Integrated approaches were used by considering assignment and sequencing at the same time. Hurink, Jurisch, and Thole (1994) proposed a tabu search heuristic in which reassignment and rescheduling are considered as two different types of moves. The integrated approach which had been represented by Dauzere-Peres and Paulli (1997) was defined a neighborhood structure for the problem where there was no distinction between reassigning and resequencing an operation. Mastrololli and Gambardella (2002) improved Dauzere-Peres tabu search techniques and presented two neighborhood functions.

This paper considers jobs scheduling problems in a part making company. The company makes various parts and equipments for petrochemical industries. The company gets its orders in the start of each month. After receiving the orders, the company designs the operations for the jobs must be executed. After orders receiving, the process specifications (the operations must be executed for each job, the capable machines for each operation and the process time for the capable machine of each operation) are designed. The company have various flexible machines and facilities which able the company to can use various machines for an operation. Operation times are depended on the capable machine assigned to the operations. This designing is done based on the received orders and the facility specifications of the company. The information of jobs and their operation for each month is used in scheduling programs which proposed in this paper. The scheduling programs search the best machines assignment to operations and sequence of the assigned operations on each machine confirm to an objective function. The scheduling problem considered in this paper is based on a flexible job shop scheduling problem. For this reason, we present flexible job shop scheduling programs to present a scheduling system for this company. The scheduling programs presented in this paper are flexible and can be used for other flexible job shops. In this paper, a mathematical model and two heuristic approaches for flexible job shop scheduling are presented. Mathematical model is used to achieve the optimal solutions for small size problems. Since FJSP is NP-hard problem, two heuristic approaches involve of integrated and hierarchical approaches are developed. Six different hybrid searching structures depending on used searching approach and heuristics are presented in this paper.

The reminder of this paper is organized as follows: Section "Problem description and formulation" describes the problem under consideration and presents a mixed integer linear program to solve it. The heuristics approaches are presented in Section "Heuristic approaches." Section "Numerical experiments and discussion" presents numerical experiments and discussion. Section "Conclusion" includes concluding remarks.

Problem description and formulation

We formulate the FJSP problem as follows. This problem has m machine and n jobs. Each job consists of a sequence of operation $O_{j,h}$, $h = 1, \ldots, h_j$, where $O_{j,h}$ and h_j denote that hth operation of job j and the number of operations required for job j, respectively. The machine set is noted M, $M = \{M_1, M_2, \ldots, M_m\}$. Unless stated otherwise, index i denotes a machine, index j denote jobs and index h denote

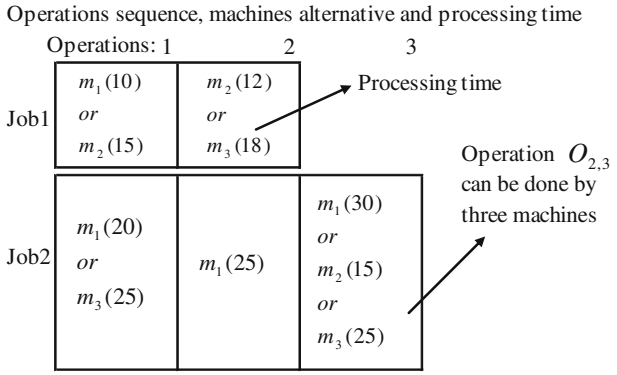


Fig. 1 Example of two jobs—three machines problem

operations throughout the paper. The execution of each operation h of a job j (noted $O_{j,h}$) requires one machine out of a set of given machines called $M_{j,h} \subset M$ and a process time, $P_{i,j,h}$, for each capable machine. The set $M_{j,h}$ is defined by $a_{i,j,h}$ as described below. We assign an index k for each machine which determines the sequence of the assigned operations on it. An example of a flexible job shop problem is presented in Fig. 1. This example describes a two jobs and three machines flexible job shop scheduling problem. The operations sequence, capable machines and processing times are shown in this figure.

We next introduce some notations and proceed by representing a mixed integer linear programming (MILP) formulation of the flexible job shop scheduling problems (FJSP).

Parameters: (*for* $i = 1, \ldots, m; j = 1, \ldots, n; h = 1, \ldots, h_j;$)

n Number of jobs

m Number of machines

$a_{i,j,h}$ Describe the capable machines set $M_{j,h}$ is assigned to operation $O_{j,h}$.

$$a_{i,j,h} = \begin{cases} 1 \text{ if } O_{j,h} \text{ can be performed on machine } i \\ 0 \text{ otherwise} \end{cases}$$

$p_{i,j,h}$ Processing time of operation $O_{j,h}$ if performed on machine $i (p_{i,j,h} > 0)$

L A large number.

Decision variables (*for* $i = 1, \ldots, m;\ j = 1, \ldots, n;\ h = 1, \ldots, h_j;\ k = 1, \ldots, k_i;$)

$C_{\max}$: Makespan or maximal completion time of jobs

$$y_{i,j,h} = \begin{cases} 1 & \text{if machine } i \text{ is selected for operation } O_{j,h} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{i,j,h,k} = \begin{cases} 1 & \text{if } O_{j,h} \text{ is performed on machine } i \text{ in priority } k \\ 0 & \text{otherwise} \end{cases}$$

$t_{j,h}$ Start time of the processing of operation $O_{j,h}$.

$Tm_{i,k}$ Start of working time for machine i in priority k.

k_i The number of assigned operations to machine i

$Ps_{j,h}$ Processing time of operation $O_{j,h}$ after select a machine.

Under these assumptions and notations, the problem is to find an assignment and a schedule that minimizes the makespan.

MILP models:

Minimize $C_{\max}$

s.t.

$$C_{\max} \geq t_{j,h_j} + Ps_{j,h_j} \quad \text{for } j = 1, \ldots, n; \tag{1}$$

$$\sum_i y_{i,j,h} . p_{i,j,h} = Ps_{j,h} \quad \text{for} j = 1, \ldots, n; \quad h = 1, \ldots, h_j; \tag{2}$$

$$t_{j,h} + Ps_{j,h} \leq t_{j,h+1} \quad \text{for } j = 1, \ldots, n;\ h = 1, \ldots, h_j - 1; \tag{3}$$

$$Tm_{i,k} + Ps_{j,h} \cdot x_{i,j,h,k} \leq Tm_{i,k+1} \quad \text{for } i = 1, \ldots, m; \quad j = 1, \ldots, n;\ h = 1, \ldots, h_j;\ k = 1, \ldots, k_i - 1; \tag{4}$$

$$Tm_{i,k} \leq t_{j,h} + (1 - x_{i,j,h,k}) \cdot L \quad \text{for } i = 1, \ldots, m; \quad j = 1, \ldots, n;\ h = 1, \ldots, h_j; k = 1, \ldots, k_i; \tag{5}$$

$$Tm_{i,k} + (1 - x_{i,j,h,k}) \cdot L \geq t_{j,h} \quad \text{for } i = 1, \ldots, m; \quad j = 1, \ldots, n;\ h = 1, \ldots, h_j;\ k = 1, \ldots, k_i; \tag{6}$$

$$y_{i,j,h} \leq a_{i,j,h} \quad \text{for } i = 1, \ldots, m;\ j = 1, \ldots, n; \quad h = 1, \ldots, h_j; \tag{7}$$

$$\sum_j \sum_h x_{i,j,h,k} = 1 \quad \text{for } i = 1, \ldots m;\ k = 1, \ldots, k_i; \tag{8}$$

$$\sum_i y_{i,j,h} = 1 \quad \text{for } j = 1, \ldots n;\ h = 1, \ldots, h_j; \tag{9}$$

$$\sum_k x_{i,j,h,k} = y_{i,j,h} \quad \text{for } i = 1, \ldots, m; \quad j = 1, \ldots, n;\ h = 1, \ldots, h_j; \tag{10}$$

$$t_{j,h} \geq 0 \quad \text{for } j = 1, \ldots, n;\ h = 1, \ldots, h_j; \tag{11}$$

$$Ps_{j,h} \geq 0 \quad \text{for } j = 1, \ldots, n;\ h = 1, \ldots, h_j; \tag{12}$$

$$Tm_{i,k} \geq 0 \quad \text{for } i = 1, \ldots, m;\ k = 1, \ldots, k_i; \tag{13}$$

$$x_{i,j,h,k} \in \{0, 1\} \quad \text{for } i = 1, \ldots, m; \quad j = 1, \ldots, n;\ h = 1, \ldots, h_j;\ k = 1, \ldots, k_i; \tag{14}$$

$$y_{i,j,h} \in \{0, 1\} \quad \text{for } i = 1, \ldots, m;\ j = 1, \ldots, n; \quad h = 1, \ldots, h_j; \tag{15}$$

Constraint (1) determines the makespan. Constraint (2) determines the processing time of operation $O_{j,h}$ by selected machine. Constraints (3) enforce each job to follow a specified operation sequence. Constraint (4) forces each machine to process one operation at a time. Constraints (5 and 6) force each operation $O_{j,h}$ can be start after its assigned machine is idle and previous operation $O_{j,h-1}$ is completed. Constraint (7) determines the capable machines for each operation. Constraint (8) assigns the operations to a machine and sequence assigned operations on all machines. Constraints (9) and (10) force each operation can be performed only on one machine and at one priority. Results of $x_{i,j,h,k}$ yield an assignment each operation on a machine and sequence assigned operations on all machines.

SA algorithm

Step 1: Initialization

Step 1.1: Obtain an initial solution F.

Step 1.2: Initiate the initial temperature T, final temperature T_f, and cooling rate r

Step 2: While not yet frozen, ($T > T_f$) or ($n_{bsc} < 20$), do the following.

Step 2.1: Perform the following loop L times.

Step 2.2.1: Neighborhood search. Select a neighbor F^c of F.

Step 2.2.2: Compute $\Delta = R_{F^c} - R_F$

Step 2.2.3: If $\Delta \le 0$ then $F = F^c$

Step 2.2.4: Compute $\Delta_b = R_{F^c} - R_{F^*}$

Step 2.2.5: If $\Delta_b < 0$, set $F^* = F^c$ and $n_{bsc} = 0$.

Step 2.2.6: If $\Delta > 0$, select a random variable $P \sim U(0,1)$.

If $e^{-\Delta/T} > P$, set $F = F^c$.

Step 2.2: Set $T = r \times T$.

Step 3: Return the best solution found for F^*.

Fig. 2 The pseudo-code for the SA algorithm

Heuristic approaches

The scheduling problem of a FJSP consists of a routing sub-problem, that is assigning each operation to a machine out of a set of capable machines and the scheduling sub-problem, which consists of sequencing the assigned operations on all machines in order to obtain a feasible schedule minimizing the predefined objective function. As discussed previously, FJSP is strongly NP-hard and combinatorial. For this reason many studies have focused on developing heuristic procedures for this problem. For solving the realistic case with more than two jobs, two types of approaches have been used: hierarchical approaches and integrated approaches. In hierarchical approaches assignment of operations to machines and the sequencing of operations on the resources or machines are treated separately, i.e., assignment and sequencing are considered independently, where in integrated approaches, assignment and sequencing are not differentiated. Hierarchical approaches are based on the idea of decomposing the original problem in order to reduce its complexity. Integrated approaches were used by considering assignment and scheduling at the same time. In this approach, we can use the reassignment and rescheduling as two different types of moves or a neighborhood structure for the problem where there is no distinction between assigning and sequencing an operation.

Simulated annealing

Simulated annealing (SA) extends basic local search by allowing moves to inferior solutions (cf. Cerny, 1985; Kirkpatrick, Gelatt, & Vecchi, 1983). The basic algorithm of simulated annealing may be described as follows: successively, a candidate move is randomly selected; this move is accepted if it leads to a solution with a better objective function value than the current solution. Otherwise the move is accepted with a probability that depends on the deterioration Δ of the objective function value. The probability of acceptance is usually computed as $e^{-\Delta/T}$, using a temperature T as control parameter. This temperature T is gradually reduced according to some cooling schedule, so that the probability of accepting deteriorating moves decreases in the course of the annealing process.

From a theoretical point of view, a simulated annealing process may provide convergence to an optimal solution if some conditions are met (e.g., with respect to an appropriate cooling schedule and a neighborhood which leads to a connected solution space); cf. Van Laarhoven and Aarts (1987) and Aarts, Korst, and van Laarhoven (1997) which give surveys on simulated annealing with theoretical results as one main topic. As convergence rates are usually too slow, in practice one typically applies some faster cooling schedule (giving up the theoretical convergence property).

We follow the robust parameterization of the general simulated annealing procedure as described by Johnson, Aragon, McGeoch, and Schevon (1989). T is initially high, which allows many inferior moves to be accepted, and is gradually reduced through multiplication by a parameter $r \prec 1$ according to a geometric cooling schedule. At each temperature size Factor L move candidates are tested (L denotes the current neighborhood size), before T is reduced to $r \cdot T$. The starting temperature is determined as follows: given a parameter T_0 and based on an abbreviated trial run, the starting temperature is set so that the fraction of accepted moves is approximately T_0. A further parameter T_f is used to decide whether the annealing process is frozen and should be terminated. Every time a temperature is completed with less than T_f of the candidate moves accepted, a counter is increased by one. This counter is reset every time a new best solution is found. The procedure is terminated when the counter reaches 20. Then, it is possible to reheat the temperature to continue the search by performing another annealing process. Briefly, the SA algorithm applied in this study in now described in Fig. 2.

Tabu search algorithm

Tabu search (TS) methods have been applied for finding the optimal solution of a number of combinatorial optimization problems. One can describe TS as a local search technique guided by the use of adaptive or flexible memory structures. The adaptive memory of TS is implemented with reference to short-term and long-term components. In the basic short-term scheme, the strategy for escaping from local minima is the following one. Even if there is no better solution than the current one, x_n in the neighborhood $V(x_n)$, one moves to the best possible solution x in $V(x_n)$ or a sub-neighborhood $V'(x_n) \subseteq V(x_n)$ in the case where $V(x_n)$ is too big to be explored efficiently. If the neighborhood structure is symmetric, i.e., if x_n belongs to the neighborhood $V(x_n)$ of x whenever $x \in V(x_n)$, there is a danger of cycling when we explore $V(x_n)$ during the next step; there is indeed a chance that x_n could be the best solution in $V(x_n)$ in which case we could come back to x_n and from then one, oscillate between x and x_n. To avoid this situation and more general cycling situation, we could store in a list $(x_{n-1}, \ldots, x_{n-L})$ called tabu list, a certain number L of the last solutions encountered. If x is in list, the move $x_n \rightarrow x$ is forbidden. An alternative is used to record a characteristic or an attribute of the moves. An aspiration criterion is introduced in tabu search to determine when tabu restriction can be overridden, thus removing a tabu classification otherwise applied to a move. A solution is above the current aspiration level if it is better than any solution met before. The main decisions to be made are: the specification of a neighborhood structure, the move attributes, the tabu list length, aspiration criterion and stopping rule (for more information about TS, refer to Pham and Karaboga, 2000). Briefly, the TS algorithm applied in this study in now described in Fig. 3.

Hierarchical algorithms

As discussed previously, FJSP consists of two sub-problem: assignment and sequencing. In hierarchical approaches assignment of operations to machines and the sequencing of operations on the resources or machines are treated separately, i.e., assignment and sequencing are considered independently, where in integrated approaches, assignment and sequencing are not differentiated. Assignment problem consists of assigning each operation to a machine out of a set of capable machines and scheduling problem consists of sequencing the assigned operations on all machines in order to obtain a feasible schedule minimizing the makespan objective function. The hierarchical approach used in this paper is described in Fig. 4. As shown in this figure, the assignment problem is the main problem and the sequencing problem is the sub-problem. It can be seen that during the hybrid search process the assignment search provides an initial solution for sequencing search. Each objective value for assignment search is computed by sequencing search. In fact sequencing search is only a sub-algorithm for the main search process. Assignment search uses the solution sequencing and evaluation by sequencing search to continue evaluation.

In this paper, we use both of SA and TS heuristics for assigning and sequencing problems with hierarchical approaches. Depended on which heuristic (SA and TS) is used for assignment and sequencing problems, four algorithms as described below are generated:

HSA/SA algorithm: this algorithm use hierarchical approach and SA heuristic for assignment problem and SA heuristic for sequencing problem.

HSA/TS algorithm: this algorithm use hierarchical approach and SA heuristic for assignment problem and TS heuristic for sequencing problem.

HTS/TS algorithm: this algorithm use hierarchical approach and TS heuristic for assignment problem and TS heuristic for sequencing problem.

HTS/SA algorithm: this algorithm use hierarchical approach and TS heuristic for assignment problem and SA heuristic for sequencing problem.

Integrated approaches

Integrated approaches were used by considering assignment and scheduling at the same time. This approach can use the assignment and sequencing as two different types of moves or a neighborhood structure for the problem where there is no distinction between assigning and sequencing an operation. In this research, we use TS and SA heuristics in this approach and present two algorithms: ISA (integrated approach with simulated annealing heuristic) algorithm and ITS (integrated approach with tabu search heuristic) algorithm. In ISA algorithm, assignment and sequencing problem are considered as two different types of moves. In this search process one of problem (assignment or sequencing) are selected randomly and a move are considered for it. ITS algorithm has a neighborhood structure where there is no distinction between assigning and sequencing an operation.

Heuristics elements

In SA and TS search algorithms, generating an initial solution, solution seed structure and neighborhood structure are three important factors in SA and TS performance. Therefore based on basely experiments we design the above elements to achieve good solutions. These elements are different for assignment and sequencing problems. In this research the parameter setting for the designed SA and TS heuristics is stated in detail. From the results of some preliminary experiments, the Parameters are obtained.

Fig. 3 The pseudo-code for the TS algorithm

TS algorithm

Step 1. Initialization

Select an initial solution x_1 in X ($x_1 \in X$)

Compute $F(x_1)$

Set $F^* \longleftarrow F(x_1)$ and $x^* \longleftarrow x_1$

F = objective function

Tabu list (TL) is empty.

Step 2. Iteration

Iteration $n = 1, 2, \ldots; xn$ denotes the current solution.

Assign $\overline{F}$ to ∞ at the beginning of the each step.

$\overline{F}$ = best accessible value of F met during the exploration of the sub neighborhood $V(x_n)$

For all x in $V(x_n)$:

If $F(x) < \overline{F}$ (if the move $x_n \longleftarrow x$ is not tabu or if the move is tabu but passes the aspiration criterion)

then $\overline{F} \longleftarrow F(x)$ and $\overline{x} \longleftarrow x$

reset $x_{n+1} \longleftarrow x$

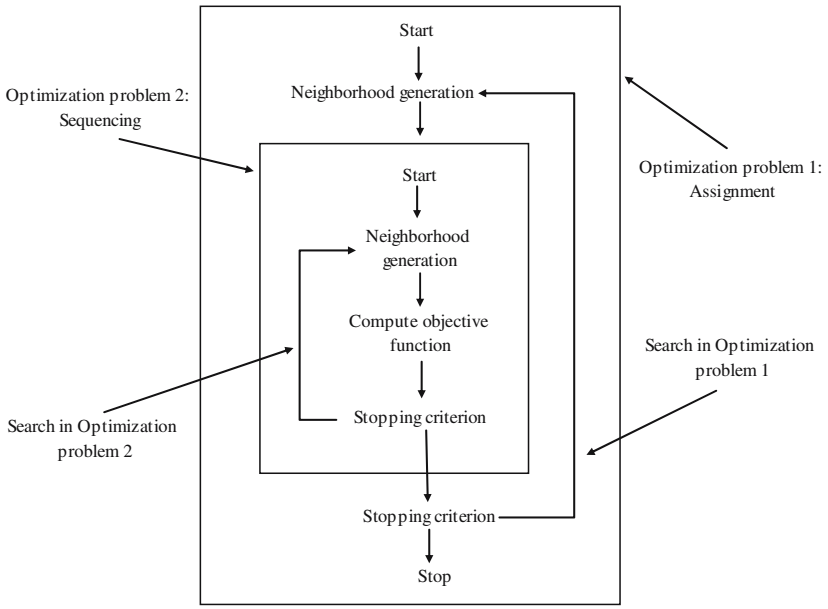
if $\overline{F} < F^*$ then $x^* \longleftarrow \overline{x}$ and $F^* \longleftarrow \overline{F}$

The appropriate characteristic of the move ($x_{n+1} \longleftarrow x_n$) enters the tabu list (TL) once the first entered characteristic has been removed if the list was full.

Step 3. Stop criterion

If the stopping criterion is fulfilled, then stop, else go to step 2

Fig. 4 The hierarchical approach for FJSP



The tabu tenure period in TS algorithms is the number of subsequent moves during which the last pair of solutions is to be forbidden. The rule considered for tabu tenure is a static rule in which tabu tenure $t = \sqrt{n}$ and n is number of neighborhood solutions. An aspiration criterion is introduced in tabu search to determine when tabu restriction can be overridden, thus removing a tabu classification otherwise applied to a move. A solution is above the current aspiration level if it is better than any solution met before. If all available moves are classified as tabu, and are not rendered admissible by some other aspiration criteria, then a "least tabu" move is selected. The used termination conditions in the algorithm steps are the number of iterations without improvement in solutions, and this considered number ranges from 50 to 200, depending on the size of the problem concerned.

From the results of some preliminary experiments, in SA heuristics, the initial temperature ranges from 50 to 300, depending on the size of the problem concerned; and the final temperature is determined to be 0.1 for all the cases of the problem. A temperature reduction factor, r, ranges from 0.9 to 0.99, depending on the size of the problem concerned.

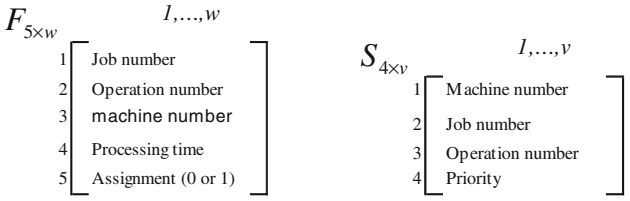


Fig. 5 The solution seed for assignment and sequencing problems

The maximum iteration length at a particular temperature L is set to be 20–150.

Solution seeds

In this paper, we use a matrix $F_{5\times w}$ to represent the solution seed for assignment problems. Index w denotes the total number of machine alternatives for all of operations. Also a matrix $S_{4\times v}$ is used to represent the solution seed for sequencing problems. These matrixes are described in Fig. 5.

Generation initial solutions

To generate an initial solution two algorithms are used in assignment and sequencing search processes. To generate an initial assignment matrix, we use the algorithm (namely ISAM algorithm) presented as follows. During the hybrid search process the assignment search provides an initial solution for sequencing search. The initial sequencing matrix is obtained from the related assignment matrix. Rows 1, 2 and 3 of this matrix are obtained from matrix F and row 4 determines the operations sequences of each machine. The initial sequencing matrix is obtained by the algorithm (namely ISSM algorithm) presented as follows. To explain these algorithms, consider the example is presented in Section "Problem description and formulation." The initial assignment for this example and an initial sequencing for the initial assignment are represented in Figs. 6 and 7.

```
ISAM algorithm

step 1. Compute the total number of
        capable machines for all of the
        operations (w) and generate
        matrix F5×w
step 2. Write row 1, 2, 3, 4 of this
        matrix for a given problem
        (these rows are fixed in search
        processes)
step 3. Choice a machine which has the
        shortest processing time for
        processing each operation from
        the set of capable machines
        (mark 1 in row 5 for selected
        machine and 0 for unselected
        machines)
step 4. return the matrix F5×w
```

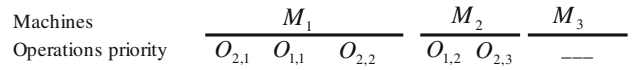


Fig. 7 The sequence illustration of Fig. 6

```
ISSM algorithm

Step 1. Compute the total number of
        perations for all jobs (v) and
        generate matrix S4×v
Step 2. For each operation write the
        number of selected machine from
        the related matrix F5×w and sort
        machine numbers. Set this sorted
        machine number in row 1 of this
        matrix
Step 3. Set the job number and operation
        number of selected machines
        according to the related matirx
        F5×w in rows 2, 3 of matrix S4×v
Step 4. For each machine set priority
        number to the associate
        operations randomly in row 4 of
        matrix S4×v
Step 5. Return the matrix S4×v
```

Neighborhood structures

In this research, the neighborhood structures for assignment and sequencing search process are obtained based on preliminary experiments. In assignment search process, given a matrix F, a new matrix F^c is obtained for F using the following method: randomly select one operation and two columns i and j for this operation (one of these columns must be have an assignment number equal 1) of the matrix F. A neighbor-

Fig. 6 The initial assignment for the presented example and an initial sequencing for this assignment

1	1	1	1	1	2	2	2	2	2	2
2	1	1	2	2	1	1	2	3	3	3
3	1	2	2	3	1	3	1	1	2	3
4	10	15	12	18	20	25	25	30	15	25
5	1	0	1	0	1	0	1	0	1	0

1	1	1	1	2	2
2	1	2	2	1	2
3	1	1	2	2	3
4	2	1	3	1	2

Table 1 Results of the branch and bound method for SFJSP problems

Problem no.	Size	Results of the branch and bound method				
		Integer variables	Non integer variables	Number of constraints	CPU time (S)	C_{max}
SFJS1	2.2.2	40	26	134	0	66
SFJS2	2.2.2	32	24	108	0	107
SFJS3	3.2.2	72	36	234	7	221
SFJS4	3.2.2	84	38	236	11	355
SFJS5	3.2.2	84	38	272	87	119
SFJS6	3.3.2	189	50	497	129	320
SFJS7	3.3.5	225	55	598	135	397
SFJS8	3.3.4	216	55	589	116	253
SFJS9	3.3.3	243	56	584	319	210
SFJS10	4.3.5	300	66	862	510	516
MFJS1	5.3.6	720	99	1,829	3,600	(396, 470)*
MFJS2	5.3.7	840	106	1,986	3,600	(396, 484)
MFJS3	6.3.7	1,260	131	2,819	3,600	(396, 564)
MFJS4	7.3.7	1,617	149	3,789	3,600	(496, 684)
MFJS5	7.3.7	1,617	149	3,726	3,600	(414, 696)
MFJS6	8.3.7	2,184	174	4,766	3,600	(469, 786)
MFJS7	8.4.7	3,584	219	7,883	3,600	(619, 1,433)
MFJS8	9.4.8	4,896	256	9,778	3,600	(619, 1,914)
MFJS9	11.4.8	7,040	308	14,190	3,600	(764, 2,908)
MFJS10	12.4.8	8,832	346	16,784	3,600	(944, 4,960)

* (IP Bound, Best IP), IP Bound: a bound on the best possible value of the objective can be attained, Best IP: the objective value of the best integer solution found by the branch and bound method

hood of F, F^c is obtained by interchanging the assignment number for the column i and j. In sequencing search process, given a matrix S, a neighborhood matrix S^c is obtained for S using the following method: randomly select one machine and one priority of this machine. A neighborhood of S, S^c is obtained by pairwise interchanging priority number of a pairwise operations of this machine.

Numerical experiments and discussion

This section describes the computational tests which are used to evaluate the performance of the presented algorithms. For this reason, various testing problems of flexible job shop scheduling are generated randomly. The testing problems are determined by size of the problem $(n/h/m)$ in which index n denotes number of jobs, h denotes the maximum number of operations for all jobs and m denotes the machine number. The testing problems are divided to two categories: small size flexible job shop scheduling problems (SFJS1:10) and medium and large size flexible job shop scheduling problems (MFJS1:10).

The mathematical model, the MILP model presented in Section "Problem description and formulation," is coded by lingo software which solves the model by the branch and bound method. The branch and bound method is a traditional optimization technique which obtain the optimum solutions. The small size problems are solved by the branch and bound method and their results are presented in Table 1. FJSP is strongly NP-hard and combinatorial. Solving of the flexible job shop scheduling problem with the branch and bound method is very time consuming and no method is able to generate optimal solutions for the medium and large size problems in polynomial time. For this reason, we calculate two bounds (upper lower bounds) of the optimal solution for the medium and large size problems. These bounds are obtained from lingo software and presented in Table 1.

Many studies have focused on developing heuristic procedures for medium and large or actual size problems. As mentioned in previous section, six different hybrid searching structures depending on used searching approach and heuristics are presented in this paper. Two approaches, integrated and hierarchical approaches, are used. In Integrated approach, two algorithms: ISA (integrated approach with simulated annealing heuristic) algorithm and ITS (integrated approach with tabu search heuristic) algorithm are presented. In hierarchical approaches four algorithms: HSA/SA, HSA/TS, HTS/TS and HTS/SA algorithms are presented.

Table 2 Results of the HSA/SA and HSA/TS algorithms for medium and large size problems

Problem no.	Size	Results of the HSA/SA algorithm				Results of the HSA/TS algorithm			
		CPU time (S)	Average of makespans	Best	D_{f^*} makespan	CPU time (S)	Average of makespans	Best makespan	D_{f^*}
SFJS1	2.2.2	12	66	66	0	1	66	66	0
SFJS2	2.2.2	13	107	107	0	1	107	107	0
SFJS3	3.2.2	14	221	221	0	1	221	221	0
SFJS4	3.2.2	14	355	355	0	1	355	355	0
SFJS5	3.2.2	14	119	119	0	2	119	119	0
SFJS6	3.3.2	18	320	320	0	3	320	320	0
SFJS7	3.3.5	16	397	397	0	4	397	397	0
SFJS8	3.3.4	17	256.2	253	0.013	5	253	253	0
SFJS9	3.3.3	19	210	210	0	6	210	210	0
SFJS10	4.3.5	21	516	516	0	7	516	516	0
MFJS1	5.3.6	22	503.2	479	0.07	55	504	491	0.07
MFJS2	5.3.7	62	511.8	495	0.12	55	496.6	482	0.08
MFJS3	6.3.7	82	577.3	553	0.08	70	595.4	538	0.11
MFJS4	7.3.7	102	667.2	656	0.08	85	680.2	650	0.1
MFJS5	7.3.7	105	667.2	650	0.06	110	690.4	662	0.10
MFJS6	8.3.7	125	776.2	762	0.08	130	823.2	785	0.14
MFJS7	8.4.7	197	1,032	1,020	0.09	290	1103.8	1,081	0.16
MFJS8	9.4.8	230	1,080	1,030	0.17	325	1,146	1,122	0.24
MFJS9	11.4.8	330	1238.4	1,180	0.12	660	1293.6	1,243	0.17
MFJS10	12.4.8	425	1621.8	1,538	0.17	600	1693.4	1,615	0.22

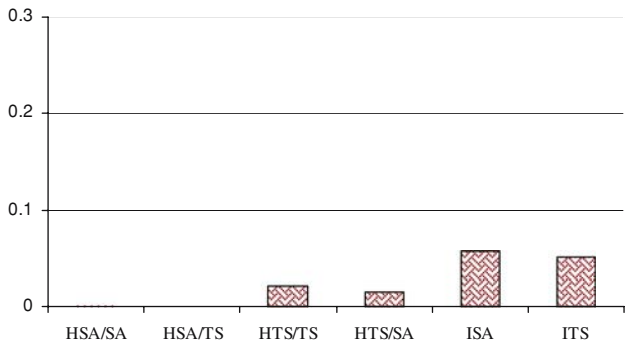


Fig. 8 Averages of index D_{f^*} for the small size problems

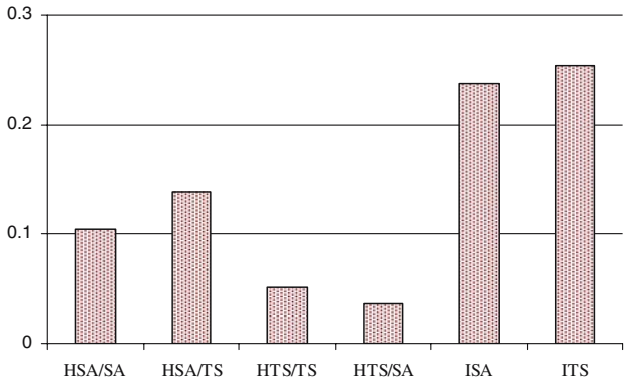


Fig. 9 Averages of index D_{f^*} for the large size problems

To verify the adaptability of the proposed algorithms, their performance is first evaluated based on preliminary experiments then a comparison with the optimal solutions is made. To solve the testing problems, the proposed algorithms are coded by the visual Fortran language. For each problem and algorithm, we run the algorithm 10 times. A heuristic is based on random search and may be obtain different solutions in different runs. To evaluate solutions, we use a factor namely D_{f^*} which determine the mean deviation of the best solution. This factor is described below, in which f^* denotes the best solution which obtained by the mathematical model or all of the algorithms. The results are presented in Tables 2–4.

$$D_{f^*} = \frac{\sum_{i=1}^{n} (f_i - f^*)}{n \cdot f^*}. \tag{16}$$

A review of the results in these tables shows that, the proposed algorithms are capable to attain the optimal solution for small size problems (SFJS1:3) and increasing in the size of problems leads to increasing in mean deviation of the algorithms. These tables show that, all of the algorithms can reach a final solution in a small time. The summarized mean devi-

Table 3 Results of the HTS/TS and HTS/SA algorithms for medium and large size problems

Problem no.	Size	Results of the HTS/TS algorithm				Results of the HTS/SA algorithm			
		CPU time(S)	Average of make-spans	Best makespan	D_{f*}	CPU time (S)	Average of make-spans	Best makespan	D_{f*}
SFJS1	2.2.2	1	phantom1,66	66	0	2	66	66	0
SFJS2	2.2.2	1	107	107	0	3	107	107	0
SFJS3	3.2.2	1	221	221	0	5	221	221	0
SFJS4	3.2.2	1	382	355	0.08	7	355	355	0
SFJS5	3.2.2	1	124.2	119	0.04	9	119	119	0
SFJS6	3.3.2	1	326	320	0.02	7	336	320	0.05
SFJS7	3.3.5	1	397	397	0	9	397	397	0
SFJS8	3.3.4	2	259.8	253	0.03	10	269.6	256	0.07
SFJS9	3.3.3	2	219	210	0.04	11	218	210	0.04
SFJS10	4.3.5	4	516	516	0	10	516	516	0
MFJS1	5.3.6	15	502.8	469	0.07	30	499	469	0.06
MFJS2	5.3.7	12	494.2	482	0.08	30	494.2	468	0.08
MFJS3	6.3.7	20	540.2	533	0.01	50	542	538	0.01
MFJS4	7.3.7	27	659.8	634	0.06	80	629.4	618	0.01
MFJS5	7.3.7	40	632.8	625	0.01	64	630.4	625	0.01
MFJS6	8.3.7	96	740.8	717	0.03	102	749.8	730	0.04
MFJS7	8.4.7	129	999.2	964	0.05	190	977	947	0.03
MFJS8	9.4.8	405	1,000	970	0.08	182	973.8	922	0.05
MFJS9	11.4.8	660	1157.8	1,105	0.04	330	1147.8	1,105	0.04
MFJS10	12.4.8	960	1511.8	1,404	0.09	430	1428.2	1,384	0.03

ations of the algorithms are shown in Figs. 8–11. The averages of index D_{f*} for the small size problems are shown in Fig. 8. This figure indicates that HSA/TS and HSA/TS algorithms can reach the optimal solutions for almost all the small size problem considered and HSA/TS algorithm have a better performance. For the medium and large size problem, as shown in Figs. 9 and 11, although none of these algorithms can guarantee the best results for all test problems, the performance of HTS/SA and HTS/TS are slightly better than other algorithms. Figure 11 shows the comparison of the best solutions obtained from each algorithm with the calculated lower and upper bounds for medium and large size problems. As shown in this figure, the HTS/SA and HTS/TS algorithms (the blue lines) have a better performance than other algorithms and these algorithms can reach a solutions within the calculated lower and upper bounds of the optimal solutions. Moreover, the computation of HTS/SA algorithm is faster than the other algorithms. Generally speaking, the hierarchical algorithms have better performance than integrated algorithms and HTS/SA algorithm is more suitable than the other algorithms.

The scheduling programs presented in this paper is able to integrate with the other factory-wide software systems such as enterprise resource planning (ERP), production activity control (PAC) and capacity requirement planning (CRP). The proposed scheduling programs get the production plan and orders from ERP and material requirement planning (MRP) systems and send the machines working plan and material requirement to PAC, CRP, and inventory control systems. Generally speaking, the proposed programs are able to integrate with the other elements of ERP software's.

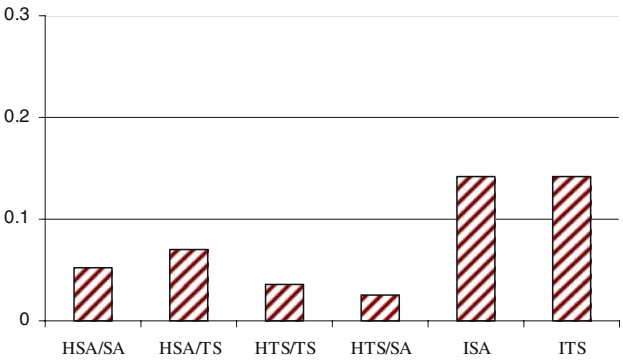


Fig. 10 Averages of index D_{f*} for all of the problems

Table 4 Results of the integrated approach algorithms for medium and large size problems

Problem no.	Size	Results of the ISA algorithm				Results of the ITS algorithm			
		CPU time (S)	Average of make-spans	Best makespan	D_{f*}	CPU time (S)	Average of make-spans	Best makespan	D_{f*}
SFJS1	2.2.2	25	66	66	0	1	66	66	0
SFJS2	2.2.2	35	107	107	0	1	107	107	0
SFJS3	3.2.2	40	231.6	221	0.10	1	221	221	0
SFJS4	3.2.2	45	375.2	355	0.05	1	390	390	0.11
SFJS5	3.2.2	50	137.6	119	0.16	1	137	137	0.15
SFJS6	3.3.2	50	336.6	320	0.05	2	320	320	0
SFJS7	3.3.5	55	397	397	0	2	397	397	0
SFJS8	3.3.4	35	254	253	0.004	2	253	253	0
SFJS9	3.3.3	55	228.3	215	0.09	3	218.3	215	0.04
SFJS10	4.3.5	55	570.3	516	0.11	3	623.6	617	0.21
MFJS1	5.3.6	60	517.8	488	0.10	9	584.2	548	0.25
MFJS2	5.3.7	60	494.4	478	0.08	8	544	457	0.20
MFJS3	6.3.7	107	610.8	599	0.14	8	606	606	0.13
MFJS4	7.3.7	195	797.8	703	0.29	9	870	870	0.40
MFJS5	7.3.7	240	706.2	674	0.13	10	729	729	0.13
MFJS6	8.3.7	330	889.5	856	0.24	50	876.5	816	0.22
MFJS7	8.4.7	480	1,307	1,066	0.38	240	1,127	1,048	0.19
MFJS8	9.4.8	610	1436.75	1,328	0.55	370	1,352	1,220	0.46
MFJS9	11.4.8	840	1218.25	1,148	0.10	680	1219.5	1,124	0.10
MFJS10	12.4.8	850	1733.25	1,546	0.25	763	1,737	1,737	0.25

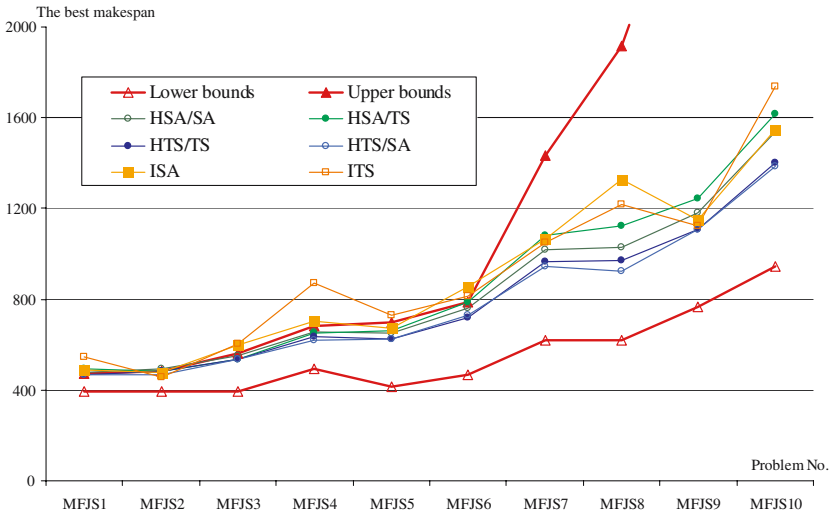


Fig. 11 Comparison of the best solutions of each algorithm with lower and upper bounds of the optimal solutions for medium and large size problems

Conclusion

In this paper, the flexible job shop scheduling is considered and a mathematical linear programming model is presented. Since FJSP is NP-hard problem, two heuristic approaches involve of integrated and hierarchical approaches are developed. In Integrated approach, we use TS and SA heuristics and presented two algorithms: ISA (integrated approach with simulated annealing heuristic) algorithm and ITS (integrated approach with tabu search heuristic) algorithm. In these algorithms, the assignment and sequencing are considered as two different types of moves and a neighborhood structure for the problem where there is no distinction between assigning and sequencing an operation, respectively. In hierarchical

approaches assignment of operations to machines and the sequencing of operations on the resources or machines are treated separately. In this approach, we use both of SA and TS heuristics in the algorithms and four algorithms (HSA/SA, HSA/TS, HTS/TS, HTS/SA) are presented. These algorithms use the SA and TS heuristics in assignment and sequencing problems.

To evaluate solutions, we use a factor namely D_{f^*} which determine the mean deviation of the best solutions. Various testing problems of flexible job shop scheduling are used to evaluate the performance of the presented algorithms. It is discussed that the proposed algorithms are capable to attain the optimal solution for small size problems (SFJS1:3) and increasing in the size of problems leads to increasing in mean deviation of the algorithms. Also all of the algorithms can reach a final solution in a small time. It is concluded that, HSA/TS and HSA/TS algorithms can reach optimal solutions for almost all the small size problem considered and HSA/TS algorithm have a better performance. For medium and large size problem, although none of these algorithms can guarantee the best results for all test problems, the performance of HTS/SA and HTS/TS are slightly better than the other algorithms. The HTS/SA and HTS/TS algorithms can reach a solutions within the calculated lower and upper bounds of the optimal solutions. Moreover, the computation of HTS/SA algorithm is faster than the other algorithms. Generally speaking, the hierarchical algorithms have better performance than the integrated algorithms and HTS/SA algorithm is more suitable than the other algorithms. Although the mathematical model and the hybrid algorithm are presented in the context of the makespan minimization, they can be easily adapted for multi-objective optimization problems. So use the multi-objective optimization technique to improve the approaches can be suggested for further research.

References

Aarts, E. H. L., Korst, J. H. M., & van Laarhoven, P. J. M. (1997). Simulated annealing. In E. Aarts, & J. Lenstra (Eds.), *Local search in combinatorial optimization* (pp. 91–120). Chichester: Wiley.

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by taboo search. *Annals of Operations Research, 41*, 157–183.

Bruker, P., & Schlie, R. (1990). Job shop scheduling with multi-purpose machines. *Computing, 45*, 369–375.

Cerny, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications, 45*, 41–51.

Dauzere-Peres, S., & Paulli, J. (1997). An integrated aooroach for modeling and solving the general multiprocessor job shop scheduling problem using tabu search. *Annals of Operations Research, 70*, 281–306.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research, 1*, 117–129.

Hurink, E., Jurisch, B., & Thole, M. (1994). Tabu search for the job shop scheduling problem with multi-purpose machines. *Operations Research Spektrum, 15*, 205–215.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1989). Optimization by simulated annealing: An experimental evaluation; part 1, Graph partitioning. *Operations Research, 37*, 865–892.

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

Kirkpatrick, S., Gelatt, Jr., C., & Vecchi, M. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Low, C., Yip, Y., & Wu, T. (2006). Modelling and heuristics of FMS scheduling with multiple objectives. *Computers & Operations Research, 33*, 674–694.

Mastrololli, M., & Gambardella, L. M. (2002). Effective neighborhood functions for the flexible job shop problem. *Journal of Scheduling, 3*(1), 3–20.

Pham, D. T., & Karaboga, D. (2000). *Intelligent optimization techniques: Genetic algorithms, tabu search, simulated annealing and neural networks*. London: Springer.

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

Van Laarhoven, P. J. M., & Aarts, E. H. L. (1987). *Simulated annealing: theory and applications*. Dordrecht: Reidel.

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