Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Feature-based initial population generation for the optimization of job shop problems*

Jing CHEN†1, Shu-you ZHANG[1], Zhan GAO[1], Li-xin YANG[2]

([1]*State Key Lab of Fluid Power Transmission and Control, Zhejiang University, Hangzhou 310027, China*)

([2]*Top Vocational Institute of Information & Technology of Shaoxing, Shaoxing 312000, China*)

†E-mail: jingchen.zju@gmail.com

Received Nov. 15, 2009; Revision accepted June 1, 2010; Crosschecked Sept. 3, 2010

Abstract: A suitable initial value of a good (close to the optimal value) scheduling algorithm may greatly speed up the convergence rate. However, the initial population of current scheduling algorithms is randomly determined. Similar scheduling instances in the production process are not reused rationally. For this reason, we propose a method to generate the initial population of job shop problems. The scheduling model includes static and dynamic knowledge to generate the initial population of the genetic algorithm. The knowledge reflects scheduling constraints and priority rules. A scheduling strategy is implemented by matching and combining the two categories of scheduling knowledge, while the experience of dispatchers is externalized to semantic features. Feature similarity based knowledge matching is utilized to acquire the constraints that are in turn used to optimize the scheduling process. Results show that the proposed approach is feasible and effective for the job shop optimization problem.

Key words: Scheduling feature, Job shop problem (JSP), Scheduling optimization, Scheduling knowledge
doi:10.1631/jzus.C0910707 **Document code:** A **CLC number:** TP278

1 Introduction

The job shop problem (JSP) is about allocating limited resources to different tasks to obtain optimal scheduling solutions. The optimized scheduling solution plays an important role in the quest to achieve an efficient and orderly production process.

In recent years, the adoption of meta-heuristic algorithms has led to better results than classical dispatching or greedy heuristic algorithms. These meta-heuristic algorithms are widely used in view of their global optimization capabilities and flexibility. However, there are still several drawbacks. Simulated annealing may generate poor designs possessing a significant probability of bypassing local minima. Furthermore, convergence is slow. The search process of the tabu method is serial, thus leading to low efficiency. Ant colony optimization can achieve better solutions, but the search process tends to stagnation. Harmony search can generate different global optimal values, but is computationally expensive. Particle swarm may become trapped in local minima. The genetic algorithm (GA) architecture, in contrast, makes this method suitable for parallel computations. Furthermore, the global optimization capability is not subject to any restriction. However, GA considers only one new design, at a time. Furthermore, utilizing just one design the perturbation strategy may not be the best approach.

The poor local search ability of GA may result in premature convergence. The overall search capability could be improved by embedding local search methods in GA. For example, in this paper, we use the local search method to transform the temporary

* Project supported by the Important National Science and Technology Specific Projects (No. 2009ZX04014-031), the Science and Technology Pillar Program of Zhejiang Province (No. 2009C31120), and the Zhejiang Provincial Natural Science Foundation of China (No. Z1080339)

population. The local search method sets out from an initial solution, and uses a domain function to constantly seek for a better solution from the current field. Overall, the coding techniques and genetic operations of GA are relatively simple and well suited for the JSP. Since JSPs include soft constraints, the GA method is widely used to optimize scheduling processes. For example, GA performs well in scheduling coding design (Gao, 2003), crossover and mutation operator design (Watanabe *et al*., 2005; Zhang *et al.*, 2007), and algorithm framework design (Gao, 2003; Ida and Osawa, 2007). Hybrid genetic optimization methods have also been proposed (Tsai *et al.*, 2006; 2008). Watanabe *et al.* (2006) proposed a GA with an active solution space search, creating an active solution, while the solution is evaluated, in order to search the active solution space more effectively. Balas and Vazacopoulos (1998) developed a variable depth search procedure. Structural properties of the neighborhood were used to guide the search in promising directions. Zhang *et al*. (2007) presented a new full, active schedule relying on the operation-based representation to construct the schedule; an approach which can further reduce the search space. In addition, they proposed the precedence operation crossover operator for the operation-based representation. Satisfactory optimal solutions could be obtained by the above methods. However, the performance of GA depends on the algorithm parameters and requires a relatively long period of time. The effects of using a hybrid approach to improve the optimal solutions are relatively limited.

Performance of GA depends on various factors such as genetic operators, parameters, and operation strategies (Togan and Daloglu, 2008). Factors are directly related to the global convergence and search efficiency of GA. The selection of operating parameters should account for the diversity and speediness of practical optimization algorithms (Zhang *et al*., 2008).

Member grouping and initial population strategies are some examples of factors. While the member grouping strategy is adopted to reduce the size of the problem, the initial population strategy is applied to reduce the number of searches to reach the optimum design in the solution space (Togan and Daloglu, 2008). The initial population should scientifically denote the information of the solution space.

Data accumulation and the scheduling solution from application scheduling instances contain the implicated scheduling knowledge, which can be reused in real-time scheduling processes. In recent years, the adoption of knowledge-based scheduling methods such as semantics for the JSP (Gonzalez-Rodriguez *et al*., 2008), scheduling learning effect (Eren, 2009), and scheduling evaluation (Li *et al*., 2008) allowed better results to be obtained. These studies used knowledge-based methods to improve the optimization algorithms or evaluate the operating solutions for the local scheduling processes.

Note that a suitable initial value of a good (close to the optimal value) scheduling algorithm can greatly speed up the convergence rate. However, the initial population of the current scheduling algorithms is randomly determined. A scheduling task whose solutions and rules are already stored in the scheduling base is called a scheduling instance. Similar scheduling instances in the production process are hence not reused reasonably. Therefore, in the study we use the scheduling instances to generate the initial population of the scheduling task. The proposed approach is to externalize the operational experience to scheduling knowledge. The externalized scheduling knowledge supports the production process by data mining with the existing scheduling data. In this way, the experience of scheduling instances can be used throughout the activation process.

2 Job shop problem

The job shop problem is a special case of the general shop problem, which generalizes the flow shop problem (Brucker, 1995). It can be stated as follows (Tsai *et al*., 2006; Zhang *et al*., 2008): For a given set of jobs $\boldsymbol{J}=[J_1, J_2, \ldots, J_n]$ to be carried out on a set of physical resources (machines) $\boldsymbol{M}=[M_1, M_2, \ldots, M_m]$, the scheduling sequence of all jobs is described by the operational matrix $\boldsymbol{D}=[\theta_1, \theta_2, \ldots, \theta_n]^{\mathrm{T}}$. θ_i can be expressed as $[(i, 1), (i, 2), \ldots, (i, |\theta_i|)]$. Each element (i, j) in $\boldsymbol{D}$ represents the procedure j of job J_i. Sequence $(i, j-1)$ is called the pre-sequence of sequence (i, j); sequence $(i, j+1)$ is called the post-sequence of sequence (i, j). The devices with the same type constitute a set of resources. Each procedure is carried out by a corresponding processing equipment. The aim is

to obtain a scheduling solution that minimizes makespan. The cost function of the optimization problem is hence represented by the total time required for completing all operations on the available resources.

The operation-based encoding (Pezzella *et al.*, 2008) was generated according to the generated initial population. Each chromosome is composed of $N\times m$ genes that represent scheduling operations. Suppose the chromosome is [2 1 1 1 2 2 3 3 3]. With the order constraint, the operations sequence $S=(O_{21}, M_1)$, (O_{31}, M_2), (O_{22}, M_3), (O_{11}, M_1), (O_{12}, M_2), (O_{32}, M_1), (O_{23}, M_2), (O_{13}, M_3), (O_{33}, M_3) is shown as the scheduling Gantt in Fig. 1.

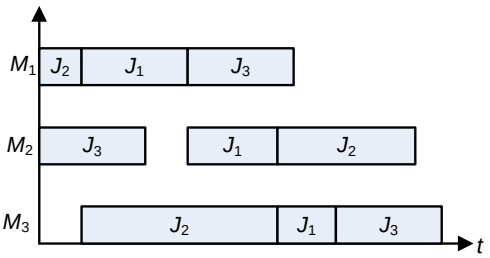


Fig. 1 Gantt based on operation coding

There are several constraints (Watanabe *et al.*, 2005): (1) a job does not visit the same machine twice; (2) there is no precedence constraint among the operations of different jobs; (3) operations cannot be interrupted; (4) each machine can process only one job at a time; and (5) all of the release times and the due dates are specified.

3 Generation of the initial population

Starting from an initial population, GA applies genetic operators to produce offsprings, which are presumably more fit than their ancestors (Pezzella *et al.*, 2008). For the same genetic operators, the characteristics and the size of the initial population determine the convergence speed and the scheduling results, as well as the crossover and mutation rate. Inappropriate selection increases the number of iterations and may even lead to a local optimal scheduling solution.

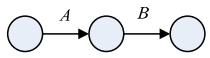
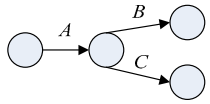
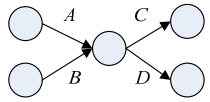
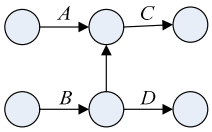
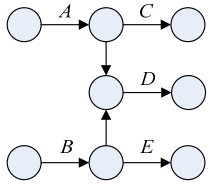
3.1 Network based scheduling features

The aim of the scheduling process is to obtain an operating plan consistent with technological constraints and production tasks. Performance indicators must be optimized under a certain production batch and corresponding processing sequences. The scheduling knowledge reflects the information constraints and implies the experience of the process parts, orders, and machines. According to the job shop scheduling theory and the features of actual production processes, scheduling knowledge is classified as a logic scheduling relationship and an organizational scheduling relationship. The logic relationship reflects the scheduling constraint information, while the organizational relationship matches the implied scheduling experience.

The scheduling node represents the processing procedure. The directed arc describes the scheduling sequence and the processing time between the procedures. Scheduling constraints present the static scheduling information inherently among the scheduling nodes. The scheduling node could not be arbitrarily changed. It accords with the processing path and equipment relationships. It is denoted as $R_i=\{(R_i \mid$ id, pre-order, post-order, cons), $i=1, 2, \ldots, n\}$, where id is the node number, pre-order is the description node before the current scheduling node, post-order is the description node after the current node, and the constraint expression cons is defined in Table 1, according to the processing relationship.

The organizational relationship of the JSP is constrained with the processing components, devices, and sequences. The choice of scheduling rules depends mainly on the experience of the scheduling personnel. The rules in the scheduling process are dynamically adopted according to the scheduling task. There is no fixed relationship between scheduling rules and the attributes of the scheduling process. However, different rules would affect the logical relationship, leading to different solutions for the problem. The dynamic scheduling knowledge is established according to the scheduling rules. The knowledge is then classified as the scheduling topology, scheduling breadth, and scheduling density. The scheduling knowledge is described as triple constituted by the data set and rule function of the scheduling topology, scheduling breadth, and scheduling density. The main event and the stage index of the scheduling process are stored in the data structure with a semantic definition.

Table 1 Constraint nodes of the scheduling logic relationship

Node icon	Logic relationship
	Operation sequence on machine *B* depends on the sequence on machine *A*
	Operation sequences on machines *B* and *C* depends on sequence on machine *A*
	Operation sequences on machines *C* and *D* depend on the sequences on machines *A* and *B*
	Operation sequence on machine *C* depends on the sequences on machines *A* and *B*; operation sequence on machine *D* depends on the sequence on machine *B*
	Operation sequence on machine *C* depends on the sequence on machine *A*; operation sequence on machine *D* depends on the sequences on machines *A* and *B*; operation sequence on machine *E* depends on the sequence on machine *B*

1. It is important to establish the scheduling topology nodes in a set of processing parts and scheduling sequences. The scheduling features of the topology nodes include the index of the current scheduling node, the existence of the pre-order for the current scheduling node, the existence of the post-order for the current scheduling node, the properties of the current scheduling node (including the information parts type, device index, and machine and processing time), and the annotation of the current scheduling node.

The scheduling node establishes the relationship between the pre-order and the post-order. In general, two or more procedures are joined by a scheduling node. The scheduling node could be an entry point of some pre-orders or an extraction point of some post-orders at the same time. The scheduling topology structure is finally established by further simplifying the scheduling network.

2. The features of scheduling breadth describe the relating information among the scheduling nodes in the network. The processing time is regarded as the size attribute of rule matching. The time parameter is defined in the topology graph with the edge information. The processing time is obtained from the length of each network edge. The features of scheduling breadth are defined by a coordination parameter and a controlling parameter.

The purpose of measuring the time parameters is to find the time relationship among the scheduling nodes. The features of scheduling breadth can be categorized into coordination parameters and control parameters. Coordination parameters are the total variable time and the local variable time for the scheduling planning; control parameters are the earliest possible finish time, the earliest possible planning start time, the earliest possible planning finish time, the latest necessary finish time of the plan, the latest necessary planning start time, and the latest necessary planning finish time.

3. The features of scheduling density include the device attributes such as bottleneck facilities, the equipment number, and processing capacity. Bottleneck facilities and processing capacity are simplified according to the scheduling network to obtain the reusable scheduling knowledge, which is the most important attribute of the dynamic rule matching method. The features of scheduling density are defined by the bottleneck device and the bottleneck device count. The attributes of the bottleneck device are the processing type, the bottleneck device index, and processing capacity of the current device. The processing capacity is calculated according to the equipment capacity, device load, parts count, and device count.

The bottleneck device is the equipment that controls the scheduling rhythm in the device set. During processing, the priority plan is arranged on the constraint resources. This feature reflects the efficiency of the scheduling process under the constraint resources. Device processing capacity is the maximum value of the processing capacity. The completion time is ignored with the rule in the production tasks. However, the efficiency of the device processing capacity is essential in the solution to scheduling problems. The value of the equipment processing capacity is an important indicator for simplifying the scheduling topology.

3.2 Reconfiguration of the scheduling network

A scheduling network is established with the processing orders and device attributes of the scheduling task. However, the network is affected by the

processing time, device quantity, and processing batch, which can lead to a significant change of the network structure. Therefore, the initial establishment of the scheduling must be enhanced in terms of structural simplification and combination.

Reconfiguration of a scheduling network is a process that converts the scheduling nodes to equivalent nodes without a processing burden. There are initially several paths between two scheduling nodes. The scheduling set between two nodes, which have no scheduling conflict with other scheduling paths, can be replaced by a simplified scheduling node. The processing time between the new nodes is the maximum length between the pre-nodes. The processing device for the topology feature Ar_i is denoted as $M_{Ar,i}$. The reconfiguration rules of the adjacent scheduling nodes are defined as follows.

1. If $M_{Ar,i}=M_{Ar,i+1}$ and there is no scheduling conflict on the processing devices, then the processing time between the new nodes is the maximum length between the pre-nodes (Fig. 2a).

2. If $M_{Ar,i}=M_{Ar,i+1}$ and there is a scheduling conflict on the processing devices, and the workpieces cannot be processed in batch, then the processing time between the new nodes is the summation length between $M_{Ar,i}$ and $M_{Ar,i+1}$ (Fig. 2b).

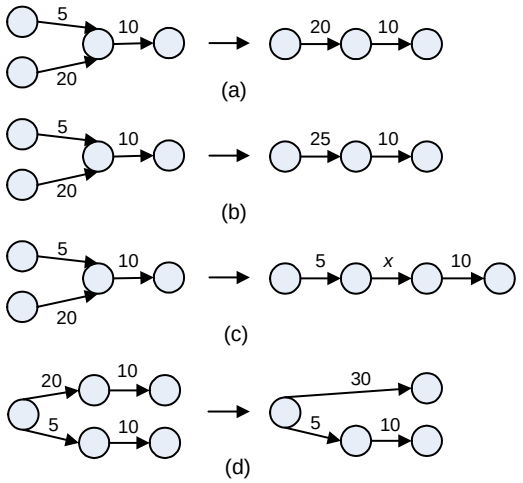


Fig. 2 Equivalent nodes in topology reconfiguration
(a) $M_{Ar,i}=M_{Ar,i+1}$, no scheduling conflict on the processing devices; (b) $M_{Ar,i}=M_{Ar,i+1}$, a scheduling conflict on the processing devices, and the workpieces cannot be processed in batch; (c) $M_{Ar,i}=M_{Ar,i+1}$, a scheduling conflict on the processing devices, and the workpieces can be processed in batch; (d) $M_{Ar,i}\neq M_{Ar,i+1}$, no scheduling conflict on the processing devices

3. If $M_{Ar,i}=M_{Ar,i+1}$ and there is a scheduling conflict on the processing devices, and the workpieces can be processed in batch, then the equivalent nodes are selected as the summation of the minimum length of the processing path and the spared processing batch. The scheduling time x between additional scheduling nodes is calculated by the interactive selection operation (Fig. 2c).

4. If $M_{Ar,i}\neq M_{Ar,i+1}$ and there is no scheduling conflict on the processing devices, then the processing time between the new nodes is the maximum length between $M_{Ar,i}$ and $M_{Ar,i+1}$ (Fig. 2d).

5. If $M_{Ar,i}\neq M_{Ar,i+1}$ and there is a scheduling conflict on the processing devices, then the scheduling topology cannot be simplified with the rule.

The equivalent points are selected according to the processing capability of the devices in the scheduling process. Assuming the device quantity is known, the device load state can be deduced by comparing the processing capability and the load balance capability. The objective function is defined as

$$\min \sum_{j=1}^{m} (T_j - d_j)^2$$

$$\text{s.t. } T_j = \sum_{i=1}^{n} t_{ij} h_i x_i, \quad 0 \le T_j \le \mathrm{Dc_b},\ j = 1, 2, \cdots, m,$$

where T_j denotes the capability provided by M_j, d_j denotes the load balance capability of M_j, m is the device quantity in the product balance operation, n denotes the type of processing parts, t_{ij} denotes the processing time of part i on M_j, h_i is the quantity of the processing part i, and $\mathrm{Dc_b}$ is the maximum processing capability of M_j.

If processing part j has been chosen, then $x_i=1$; otherwise, $x_i=0$. This objective function ensures the maximum load in the allowed range of the processing capability. Bottlenecks, as a starting point, guarantee the maximum utilization of the processing devices.

The static knowledge (logic relationship) provides information about the scheduling process, such as machines, part types, device quantities, processing sequences, and due dates. The dynamic knowledge (organizational relationship) describes the scheduling experience about the features of the scheduling network. Methods to derive the feasible scheduling solutions are mentioned in Section 3.3.

3.3 Rule matching

The scheduling rule can be obtained using the similarity matching method. The feature similarity describes the degree of similarity between the scheduling task and the scheduling instance. The scheduling rule is matched with the scheduling instances. The detailed algorithm is described as follows.

Algorithm 1 Rule matching algorithm

Step 1: Judge the bottleneck devices and processing capability according to the scheduling density feature. Find the topology nodes where a conflict occurs. Obtain the scheduling tasks that match the scheduling instance in the scheduling knowledge base.

Step 2: Define the set of the scheduling breadth features and the breadth feature guidance matrix. The set of the scheduling breadth features is described as $V=\{v_1, v_2, \ldots, v_q\}$, where v_i is the bottleneck device's breadth value of the scheduling task, and q is the device feature quantity after the process of nodes combination and simplification. The scheduling instance matrix is defined as $\boldsymbol{R}=[\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_p]$. The scheduling instance parameter $\boldsymbol{r}_i$ is the scheduling breadth value of the bottleneck device attribute. The parameter p is the length value of the edge after the process of nodes combination and simplification. The guidance matrix of the scheduling breadth features is described as

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1q} \\ r_{21} & r_{22} & \cdots & r_{2q} \\ \vdots & \vdots & & \vdots \\ r_{p1} & r_{p2} & \cdots & r_{pq} \end{bmatrix} = (r_{ij})_{p\times q}. \tag{1}$$

Step 3: Match the guidance matrix. The guidance matrix is finally obtained with the normalized disposal after the subtraction operation between the scheduling instance attributes and the scheduling task features.

$$\boldsymbol{\Delta} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1q} \\ d_{21} & d_{22} & \cdots & d_{2q} \\ \vdots & \vdots & & \vdots \\ d_{p1} & d_{p2} & \cdots & d_{pq} \end{bmatrix} = (d_{ij})_{p\times q},\ d_{ij} = \frac{\left|r_{0j} - r_{ij}\right|}{\sum_{k=1}^{p}\left|r_{0j} - r_{kj}\right|}, \tag{2}$$

where $d_{ii} = 0$. In the process of the normalized disposal, if $\sum_{k=1}^{p}\left|r_{0j} - r_{kj}\right| = 0$, then d_{ij}=0.

Step 4: Choose the weighted vector of each factor according to the processing capability $\mathrm{Db}_{\mathrm{c}}^{*}$:

$$\boldsymbol{W} = [\omega_1, \omega_2, \cdots, \omega_n]^{\mathrm{T}},\ w_i = \frac{\mathrm{Db}_{\mathrm{c\_}wi}\lg(N_w / n_w)}{\sqrt{\sum_{k=1}^{W_i}(\mathrm{Db}_{\mathrm{c\_}ki})^2\left(\lg(N_w / n_w)\right)^2}}, \tag{3}$$

where Db_{c} is the device processing capability, N_w is the scheduling device ratio of the scheduling instance, n_w is the key device ratio of the scheduling instance, and W_i is the key nodes count of the scheduling task. The weight of the feature is influenced by the value $\lg(N_w/n_w)$.

Step 5: The Euclidean distance between the scheduling task and the scheduling instance is defined as: $d_{0i} = \sqrt{\sum_{j=1}^{n}\omega_j d_{ij}^2}$. The similarity between the scheduling task and the scheduling instance can be denoted as: $\gamma_{0i} = 1 - d_{0i},\ 0 < \gamma_{0i} < 1$.

Step 6: The scheduling instances obtained from similarity matching are more similar with the increment of the value γ_{0i}. Record the scheduling rules satisfied by the similarity instances. The vector of the matching rules can be described as: $\boldsymbol{n}_w = [n_{w1}, n_{w2}, \cdots, n_{wl}]$.

The conflict judgment of the scheduling rules is a process to filter the conflict rules in the matching vectors obtained. The judgment of the scheduling rules is classified mainly as the judgment of contradiction rules and hypotaxis rules. The contradiction rule conflict is the circumstance of there being the same scheduling condition that leads to several conflict rules; the hypotaxis rule conflict is the circumstance of there being several different scheduling rules that lead to the same scheduling solution. If the rank of static constraint n_{w1} is larger than the rank of static constraint n_{w2}, then n_{w2} is defined as the hypotaxis rule in the scheduling. It is better to adopt the hypotaxis rule in the scheduling process when few rule conflicts occur. The scheduling instance, which appears frequently, is chosen as the constraint rule of the current scheduling task.

The scheduling features are put forward to implement the scheduling relating based on the feature similarity method. The initial feasible solution is

constructed with the scheduling rule obtained. Then the initial population could be finally generated following the G-T method (Giffler and Thomson, 1960).

4 Description of the genetic algorithm for the job shop problem

Appropriate scheduling rules are obtained by way of the relating method between the given scheduling tasks and scheduling instances in the knowledge base. The scheduling initial population is put forward to implement the scheduling relating based on the feature similarity method.

The strength of GA with respect to other local search algorithms is that several strategies can be combined to find individuals to add to the mating pool. This is true, either in the process of generating the initial population or in the dynamic generation phase (Watanabe *et al*., 2006). Based on the reconfiguration of scheduling features, the initial chromosomes are obtained using the matching method. The flow chart of the proposed GA for finding optimal scheduling is shown in Fig. 3.

The proposed algorithm is described in detail as follows.

Step 1: Scheduling topology can be constructed according to the density property and width property. A new scheduling topology is based on the node simplification of the bottleneck device. Scheduling instances must be matched to obtain the corresponding rule vectors $\{\boldsymbol{n}_w\}$. The initial population is built according to the G-T method. The size of initial population may affect GA performance, which decreases as the population becomes smaller in size. Increasing the population size allows local optima to be avoided, but entails higher computation costs. In this study, m ranges between 10 and 120.

Step 2: The makespan is computed for each chromosome in the current generation (Li *et al*., 2008). It is defined as a monotone decreasing function:

$$\text{Fitness} = T_{\max} - \max\{\text{ET}_{M_1}, \text{ET}_{M_2}, \cdots, \text{ET}_{M_m}\},$$

where $T_{\max}$ is the maximum completion time of the order, and ET_{M_i} (i=1, 2, …, m) is the maximum completion time of machine M_i. If Fitness<0, the current scheduling path is not a feasible solution. Different optimized makespan values correspond to different scheduling solutions. Scheduling solutions are different from the diverse orders of workpiece processes. Different processing sequences lead to different processing waiting time. The more is the time spent on waiting for a device, the later is the scheduling task completed (that is, the larger is the makespan). The scheduling solution is the result of the scheduling task, which differs little with test cases. The quantities of jobs and operations in the actual producing process are more than in the test cases.

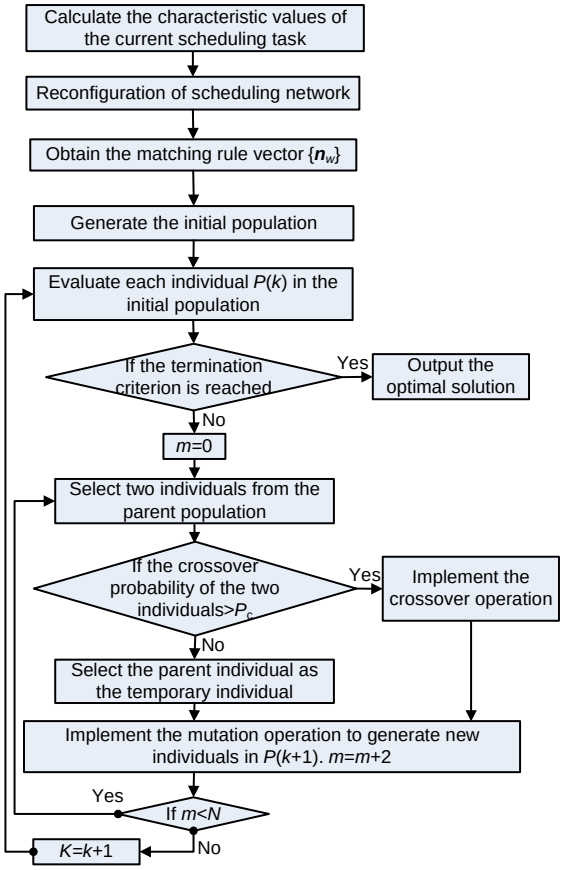


Fig. 3 The process of scheduling the genetic algorithm

Step 3: Set the termination criterion. If the number of generation becomes equal to the maximum value, the optimal solution is given in output. Otherwise, go to Step 4. The maximum number of evolutions allowed in this study ranges between 100 and 500. GA has the property of approaching the probability limit. But in practice, it is very difficult to achieve convergence. Therefore, the goal of optimization should be quick, with a certain optimization level, related to the operation design and the choice of

variables. In practical implementations, if the GA search does not terminate, optimization results may remain unknown. For this reason, in this study, the maximum number of evolutions was chosen as the termination criterion. If this criterion is satisfied, the algorithm stops and the best chromosome, together with the corresponding schedule, is provided as the output.

Step 4: The copy operation is implemented according to the selection strategy. The best chromosomes are chosen for reproduction by binary tournament in the selection phase: two individuals are randomly chosen from the population and the best of them is selected for reproduction.

Step 5: Crossover operators are implemented when the fitness values between P_1 and P_2 are not equal, and the crossover probability P_c is reached. The value of P_c ranges between 0.25 and 1. The precedence preserving order-based crossover proposed by Zhang *et al*. (2008) was used in this research. The crossover method selects children operations C_1 and C_2 from parents P_1 and P_2. The operation collection is divided into two non-empty collections J_1 and J_2. Copy the operations that include J_1 in P_1 to C_1, and copy the operations that include J_1 in P_2 to C_2. The remaining operations in P_1 and P_2 alternate as the original sequence. The mutation method selects an operation from a single parent chromosome and moves it to another position. Take the scheduling problem in Fig. 4 as an example. Suppose the parent chromosomes are P_1 [2 1 1 1 2 2 3 3 3] and P_2 [1 2 3 2 1 1 3 2 3]. Divide the workpiece collection into J_1={2} and J_2={1, 3}. Reserving collection J_1, we obtain the offspring chromosomes C_1 [2 1 3 1 2 2 1 3 3] and C_2 [1 2 1 2 3 3 3 2 1]. The offspring chromosome C_1 retains the sequence of workpiece {2} in parent chromosome P_1 and workpiece {1, 3} in parent chromosome P_2. The offspring chromosome C_2 retains the sequence of workpiece {1, 3} in parent chromosome P_1 and workpiece {2} in parent chromosome P_2. Therefore, the offspring chromosomes can inherit the superior characteristics of parent chromosomes.

Step 6: The mutation operators are implemented when the mutation probability P_m is reached. The value of P_m is between 0.001 and 0.1. Precedence preserving shift mutation operators (Zhang *et al*., 2008) were adopted. Select one gene into a random location in the chromosomes. Take the scheduling problem in Fig. 5 as an example. Suppose the parent chromosome is [1 2 3 2 1 1 3 2 3]. Select the workpiece 2 order 1 from chromosome into the location of workpiece 1 order 1. The offspring chromosome could be obtained as [2 1 3 2 1 1 3 2 3].

Step 7: Build a new generation of the population. Return to Step 3.

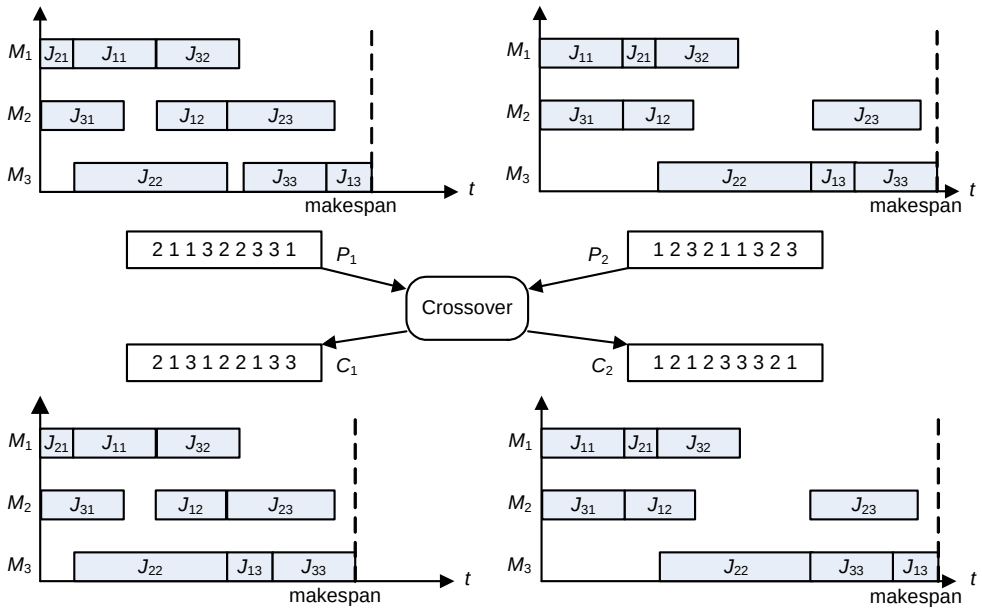


Fig. 4 Crossover operators of scheduling the genetic algorithm

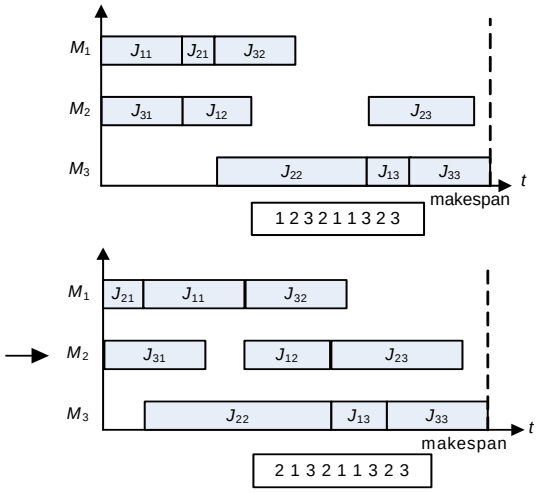


Fig. 5 Mutation operators of scheduling the genetic algorithm

5 Numerical results

The optimization algorithm was coded in C# programming language. The scheduling procedure described was implemented on a 2.4 GHz Pentium IV processor, and tested on a large number of problem instances from the literature (http://people.brunel.ac.uk). Table 2 compares the present cGA method with the algorithms proposed by Balas and Vcozacopados (1998), Watanabe *et al*. (2006), Zhang *et al*. (2007), and Tsai *et al*. (2008). The relative deviation of Ft10, Ft20, Abz8, La21, La25, and La36 were 0. Relative deviations observed in problems Abz7 and Abz8 were significantly smaller than those found for the asol algorithm.

Table 2 Comparison of the makespan values on JSP instances found by the different algorithms

Instance	$n\times m$	LB	Makespan (s)				
			cGA	asol	TS_{ED}	BV	TBGA
Ft10	10×10	930	930	930	930	930	930
Ft20	20×5	1165	1173	1173	–	–	1165
Abz7	20×15	656	657	700	657	666	–
Abz8	20×15	665	665	720	669	678	–
Abz9	20×15	679	681	–	680	693	–
La21	15×10	1046	1046	1074	1046	1046	1053
La25	20×10	977	977	–	977	979	984
La36	15×15	1268	1268	–	1268	1268	–

n: number of jobs; *m*: number of machines; LB: lower bound. cGA: algorithm proposed in this paper; asol: Watanabe *et al*., 2006; TS_{ED}: Zhang *et al*., 2007; BV: Balas and Vcozacopados, 1998; TBGA: Tsai *et al*., 2008

Table 3 compares the average makespan and CPU time required by the present algorithm and the algorithms proposed by Watanabe *et al*. (2006) and Zhang *et al*. (2007). The present algorithm improved the average makespan of instances compared to those reported in the literature. The present approach aims to improve the efficiency of the scheduling algorithm, attempting to match the most appropriate scheduling rules for generating the initial population. Deviation of the rules from the scheduling task is inevitable in this process. However, it can be seen from Table 2 that such a deviation affected marginally the scheduling results, while the scheduling efficiency was significantly increased through this initial population option (Table 3).

Table 3 Comparison of average makespan and CPU time of cGA

Instance	Makespan (s)			Makespan dev. (s)	Time (s)	Time dev. (s)
	cGA	asol	TS_{ED}			
Ft10	931.1	949.3	930.4	7.5	35	4
Ft20	1173.2	1182.4	–	1.7	39	5
Abz7	672.8	715.6	661.2	517.6	197	16
Abz8	666.3	733.6	670.2	4.4	199	23
Abz9	688.2	–	684.7	170.7	185	18
La21	1046.5	1092.1	1046.6	6.3	86	4
La25	979.2	–	977.1	16.5	44	5
La36	1268.3	–	1268.1	7.2	46	3

cGA: algorithm proposed in this paper; asol: Watanabe *et al*., 2006; TS_{ED}: Zhang *et al*., 2007

Different solutions were generated from the random sequences. The initial population was constructed by the rule vectors. This was a random matching process. The operators of selection, crossover, and mutation in the scheduling algorithm were also random. Therefore, different solutions were generated in each trial. Table 4 compares the best and the worst values of makespan found by the different algorithms. The randomly generated scheduling sequences caused solution instability. Less scheduling time was used to obtain the optimal solution, but the worst values were not always better than the results from TS_{ED}. Maintaining the trade off between scheduling time and stability performance of the algorithm is one of the problems to be dealt with in further research.

Table 4 Comparison of the best and worst values of makespan determined by the different algorithms

Instance	Best makespan (s)			Worst makespan (s)		
	cGA	asol	TS_{ED}	cGA	asol	TS_{ED}
Ft10	930	930	930	944	967	934
Ft20	1173	1173	–	1184	1199	–
Abz7	657	700	657	727	726	667
Abz8	665	720	669	676	746	672
Abz9	681	–	680	738	–	688
La21	1046	1074	1046	1053	1109	1047
La25	977	–	977	1000	–	978
La36	1268	–	1268	1283	–	1269

cGA: algorithm proposed in this paper; asol: Watanabe *et al*., 2006; TS_{ED}: Zhang *et al*., 2007

Two-hundred trials were run for cases Ft10 (Fig. 6a) and La25 (Fig. 6b). One hundred trials were run for cases Abz7 (Fig. 6c) and Abz9 (Fig. 6d).

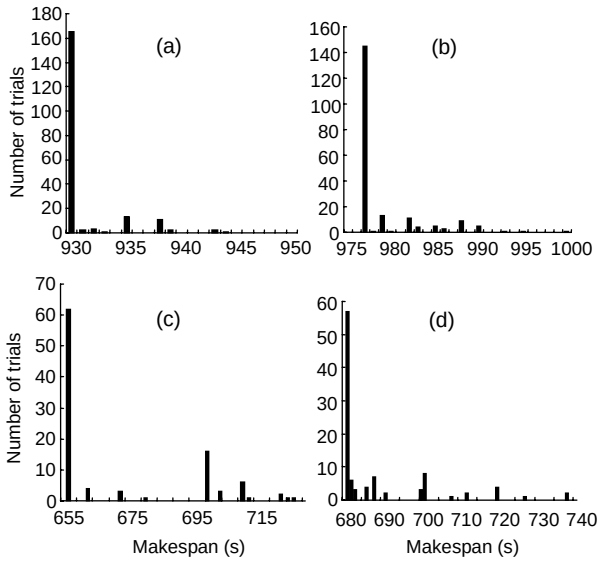


Fig. 6 Dispersion of optimized values for scheduling problems Ft10 (a), La25 (b), Abz7 (c), and Abz9 (d)
Two-hundred trials for (a) and (b), and one hundred trials for (c) and (d)

6 Conclusions

In this paper, we developed an initial population generating method for the JSP. The aim was to provide a method to improve the operating speed with the scheduling experience. Results showed that the proposed method takes less time to obtain the optimal scheduling solution. As a consequence, the scheduling experience, which could be hardly utilized otherwise, is converted to semantic scheduling knowledge, with procedures for finding an optimal scheduling solution.

There are still aspects to be investigated. Reconfiguration of scheduling topology should be closely associated with the personnel experience in the dynamic processing circumstance. Adapting the scheduling features to actual production environment more accurately will be studied in the future.

References

Balas, E., Vazacopoulos, A., 1998. Guided local search with shifting bottleneck for job shop scheduling. *Manag. Sci.*, **44**(2):262-275. [doi:10.1287/mnsc.44.2.262]

Brucker, P., 1995. Scheduling Algorithms. Springer Verlag, Berlin.

Eren, T., 2009. A bicriteria parallel machine scheduling with a learning effect of setup and removal times. *Appl. Math. Model.*, **33**(2):1141-1150. [doi:10.1016/j.apm.2008.01.010]

Gao, L., 2003. Shop Scheduling with Genetic Algorithms. Tsinghua University Press, Beijing, China (in Chinese).

Giffler, B., Thomson, G.L., 1960. Algorithms for solving production scheduling problems. *Oper. Res.*, **8**(4):487-503. [doi:10.1287/opre.8.4.487]

Gonzalez-Rodriguez, I., Puente, J., Vela, C.R., Varela, R., 2008. Semantics of schedules for the fuzzy job-shop problem. *IEEE Trans. Syst. Cybern. Part A: Syst. Hum.*, **38**(3):655-666. [doi:10.1109/TSMCA.2008.918603]

Ida, K., Osawa, A., 2007. Solution of job-shop scheduling problems by an idle time shortening. *Electr. Eng. Jpn.*, **159**(2):55-63. [doi:10.1002/eej.20389]

Li, N.X., Chen, Y.W., Yang, K.W., 2008. Multi-objective flexible job shop schedule: design and evaluation by simulation modeling. *Appl. Soft Comput.*, **9**(1):362-376.

Pezzella, F., Morganti, G., Ciaschetti, G., 2008. A genetic algorithm for the flexible job-shop scheduling problem. *Comput. Oper. Res.*, **35**(10):3202-3212. [doi:10.1016/j.cor.2007.02.014]

Togan, V., Daloglu, A.T., 2008. An improved genetic algorithm with initial population strategy and self-adaptive member grouping. *Comput. & Struct.*, **86**(11-12):1204-1218. [doi:10.1016/j.compstruc.2007.11.006]

Tsai, J.T., Chou, J.H., Liu, T.K., 2006. Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm. *IEEE Trans. Neur. Networks*, **17**(1):69-80. [doi:10.1109/TNN.2005.860885]

Tsai, J.T., Liu, T.K., Ho, W.H., Chou, J.H., 2008. An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover. *Int. J. Adv. Manuf. Technol.*, **38**(9-10):987-994. [doi:10.1007/s00170-007-1142-5]

Watanabe, M., Ida, K., Gen, M., 2005. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. *Comput. Ind. Eng.*, **48**(4):743-752. [doi:10.1016/j.cie.2004.12.008]

Watanabe, M., Ida, K., Gen, M., 2006. Active solution space and search on job-shop scheduling problem. *Electr. Eng. Jpn.*, **154**(4):61-67. [doi:10.1002/eej.20185]

Zhang, C.Y., Li, P.G., Guan, Z.L., Rao, Y.Q., 2007. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Comput. Oper. Res.*, **34**(11):3229-3242. [doi:10.1016/j.cor.2005.12.002]

Zhang, C.Y., Guan, Z.L., Liu, Q., Shao, X.Y., Li, P., 2008. New scheduling type applied to solving job-shop scheduling problem. *Chin. J. Mech. Eng.*, **44**(10):24-31. [doi:10.3901/JME.2008.10.024]