J Intell Manuf
DOI 10.1007/s10845-013-0841-z

A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem

Amir Jalilvand-Nejad · Parviz Fattahi

Received: 27 January 2011 / Accepted: 12 October 2013

Abstract Product orders are usually released with a cyclic manner, in the part manufacturing industries. Hence, scheduling cyclic jobs instead of single jobs is more practical in the real manufacturing environment. In this paper, a flexible job shop scheduling problem with cyclic jobs is studied. In this problem, jobs must be delivered in determined batch sizes with definite time intervals. This problem is practical for many manufacturing systems such as automotive parts making industries. Whereas jobs are repeating infinitely during time, a framework is suggested to break down the infinite programming horizon to smaller periods. Then a mixed integer linear programming (MILP) model is presented to schedule jobs in this short term horizon. The goal of the proposed model is minimizing the total cost including delay costs, setup costs and holding costs. Since the problem is well-known as NP-hard class, the proposed MILP model is effective just for small size problems. Therefore, two algorithms based on genetic and simulated annealing (SA) algorithms are developed to solve real size problems. The numerical experiments are used to evaluate the performance of the developed algorithms. Results show that the proposed genetic algorithm (GA) has a better performance than the SA algorithm. Also some intelligent approaches are presented and examined in the proposed algorithms. Results show the efficiency of the presented intelligent approaches as mutation operators for the proposed GA.

Keywords Cyclic scheduling problem · Flexible job shop · Mixed integer linear programming · Genetic algorithm · Simulated annealing

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

Introduction

The scheduling problem for a finite number of jobs has been widely considered in many researches. But in the real world, sometimes jobs are repeated infinitely during time. In this situation, jobs arrive successively to the shop with some time intervals. This condition generally is considered in two major categories: the *real-time* and the *cyclic* scheduling problems. The real-time models usually consider the scheduling of a set of simple tasks (single-operation jobs) with a regular or irregular repeating pattern. In this category, a due date is determined for each task which must be satisfied in the scheduling model (Leung 2004). Whereas the considered tasks in the real-time models are single-operation, it is not often suitable for job shop environments which include jobs with multiple operations. On the other hand, the cyclic job shop scheduling models are able to schedule high variety repeatable jobs in a job shop environment. But they have a restriction: The time intervals between successive tasks must be fixed and identical.

The cyclic job shop models are more suitable for the problem under consideration. Therefore in this paper, a specific type, which is called *cyclic flexible job shop scheduling problem*, is considered. Because of poor literatures of cyclic flexible job shop scheduling, the literatures of cyclic scheduling and cyclic job shop scheduling are reviewed as follows.

The cyclic operations were introduced as operations which must be processed infinitely with a determined time interval (Hanen and Munier 1997). They defined a linear constraint between each pair of cyclic operations. These linear constraints explain the precedence constraints between operations. Their objective was to optimize the cycle time.

Brucker and Kampmeyer (2008a) presented a general framework for modeling and solving the cyclic scheduling problems. Their proposed framework covers different versions of cyclic scheduling such as cyclic job shop, robotic cell, single hoist scheduling and tool transportation between the machines. They showed that these problems can be formulated as a mixed integer linear programming model. Before them, Hanen (1994) had presented a branch and bound method for such problems. Brucker and Kampmeyer (2008a) presented a tabu search algorithm to solve its large sizes. Also in another survey Brucker and Kampmeyer (2008b) they study blocking in cyclic job shop. Blocking is happen when operations must stay on a machine after finishing whereas the next machine is occupied by another job.

Cavory et al. (2005) presented a general taxonomy of cyclic scheduling problems. In their taxonomy one of the most common models is cyclic job shop problem (CJSP). Definition of jobs and machines in CJSP is similar to JSP but in CJSP each job must be processed frequently. Nonetheless the cycle time for various jobs may be different.

An exact solution for a CJSP with a single job has been found by Roundy (1992). His goal was to minimize the cycle time and the work-in-progress. He used a multi objective approach that aims at obtaining a compromise between these two criteria.

Song and Lee (1998) studied the cyclic job shop with blocking where each machine has an input buffer of finite capacity. They used Petri net models. Some other like Romanova and Servakh (2009) used dynamic programming based algorithms to solve special cases of CJSP. Bahroun et al. (1999) considered the scheduling problem with cyclic demand. The studied problem by them was mostly appropriate for automotive parts suppliers who their contracts are affected through the applying just-in-time policy by the automotive manufacturers. In their study, the average demand for each cycle was determined in long term but the exact values for demand and due dates was available just in short term. Though they considered a practical situation, but their model had some restrictions. The most important restriction of their work was that they considered the production of n items on a single facility. As a matter of fact, they studied the single machine problem with cyclic demand.

Ouenniche and Bertrand (2001) studied the multi-product, finite horizon and static demand scheduling problem in a job shop environment. Their approach included three main steps: sequencing, lot sizing and scheduling. They suggested a cyclic scheduling solution framework, called the multiple cycle (MC) method. They supposed a basic cycle time and their model relied on this assumption that the cycle time for each product is an integer multiple of this value. Their goal was to minimize the sum of setup costs, and work-in-progress and finished products inventory holding costs.

Ouenniche and Bertrand (2001) considered almost similar problem to ours. Nonetheless there are some differences between them:

- In this paper, jobs must be delivered in determined batch sizes defined by the customer.
- Each shipment has a due date and must be delivered on time. Delivering later than due dates causes penalties and delivering earlier, is not possible. Therefore preparing the product earlier than its due date, causes holding cost.
- It may be possible to process each operation by more than one machine. Moreover the processing time on each machine may be different. This characteristic leads to the flexibility in processing.

It has been tried to propose a model which is practical and especially appropriate for suppliers of automotive industries. Whereas most of automotive manufacturers use just-in-time approach as their inventory policy and order their needed parts and subsets by kanban policy, their suppliers are forced to produce and forward their products in distinct batches at determined due dates. Although the intervals between due dates for kanban orders are not predetermined, but usually the number of released kanbans in a long term period is determined by their concluded contracts. So, manufacturers often suppose a unique approximate interval between kanbans' due dates as cycle time, to regulate their production planning. Therefore, the results of this study can be used by automotive parts suppliers and other manufacturers who must deliver their products through such terms.

A framework is suggested to breakdown the infinite horizon to some short periods. All jobs must be processed for one or more shipments in this short term horizon and the main schedule can be obtained by repeating this short term schedule during time. For this purpose a mixed integer linear programming (MILP) model is developed to schedule jobs in short term horizon.

Since the model belongs to the NP-hard class, it is necessary to use a meta-heuristic approach for real-size problems. Therefore a genetic algorithm (GA) has been developed to solve the scheduling problem. Pezzella et al. (2008) has showed that integrating several strategies instead of using a single strategy is more efficient in the case of flexible job shop scheduling. Therefore, in the proposed GA, more than one strategy is adopted for generating the initial population, crossover, and mutation operators. Also three intelligent procedures over two random are defined, which are used as mutation operators in the proposed GA. The designed intelligent operators help to decrease some costs like setup costs or delay costs in the new solutions. Also, to evaluate the results of the proposed GA, it can be useful to compare it with other algorithms which prior studies have proved their performance for flexible job shop problems. Using a local search algorithm as a comparative algorithm has the advantage of comparing different approaches to generate new solutions. Among the local search algorithms, the simulated annealing (SA) has presented suitable efficiency in similar previous studies

(Fattahi et al. 2009). Hence, a SA is developed to compare with the proposed GA. The mentioned intelligent procedures are used as neighborhood search methods in the comparative SA.

The rest of this paper is organized as follows. In section "Problem definition" gives a definition of the problem and describes its characteristics. In section "The proposed framework for solving" has been assigned to describe the proposed framework for solving the problem. The mathematical model which can solve the problem in small sizes is specified in section "Mathematical model". For large sizes two meta-heuristic algorithms, based on genetic algorithm and simulated annealing, are given in sections "The proposed genetic algorithm" and "The proposed comparative algorithm". The proposed method has been tested and its results are demonstrated in section "Computational experiments". In "Conclusion and further research" section conclusions and future research opportunities are presented.

Problem definition

In this paper, the *cyclic flexible job shop scheduling problem* is considered. This study extends flexible job shop scheduling problem (FJSP) by assuming several demands for each job through infinite horizon with definite intervals. Jobs that have this character are called *cyclic jobs*. In part manufacturing industries, especially in automotive industries, orders are usually released with such cyclic manner. Hence, scheduling of cyclic jobs instead of some single jobs is more helpful and makes the results more appropriate to apply.

In the auto part manufacturing industries usually a job delivery means delivering a batch of parts which its size is determined by a bilateral contract. Each batch must be delivered on its assigned due date. The due date for delivering the kth batch of the job j is presented by $d_{j,k}$. Delivering batches of job j after their due dates causes delay cost dc_j per unit time. Figure 1 illustrates the structure of cyclic jobs.

Also each batch of a job must be performed through some operations. Therefore O_{jkh} denotes the hth operation of kth batch of job j. h_j is the number of operations of job j. There are some machines as set $M = \{M_1, M_1, \ldots, M_m\}$ in the workshop and it may be possible that more than one machine be able to process the operation O_{jkh}. So, the execution of each operation O_{jkh} requires one machine among a set of given machines called $M_{j,h}$ with process time $p_{i,j,h}$ for each alternative machine.

Performing an operation on a machine has a setup time and also a setup cost. The setup time and setup cost of processing the hth operation of job j on machine i is represented by $st_{i,j,h}$ and $sc_{i,j,h}$, respectively. Note that performing same operations of two consecutive batches of a job on a machine, leads to eliminate the setup cost and setup time between them. Also, stocking the finished product or work-in-progress (WIP), causes holding cost. Holding a batch of job j after performing its hth operation, result in holding cost

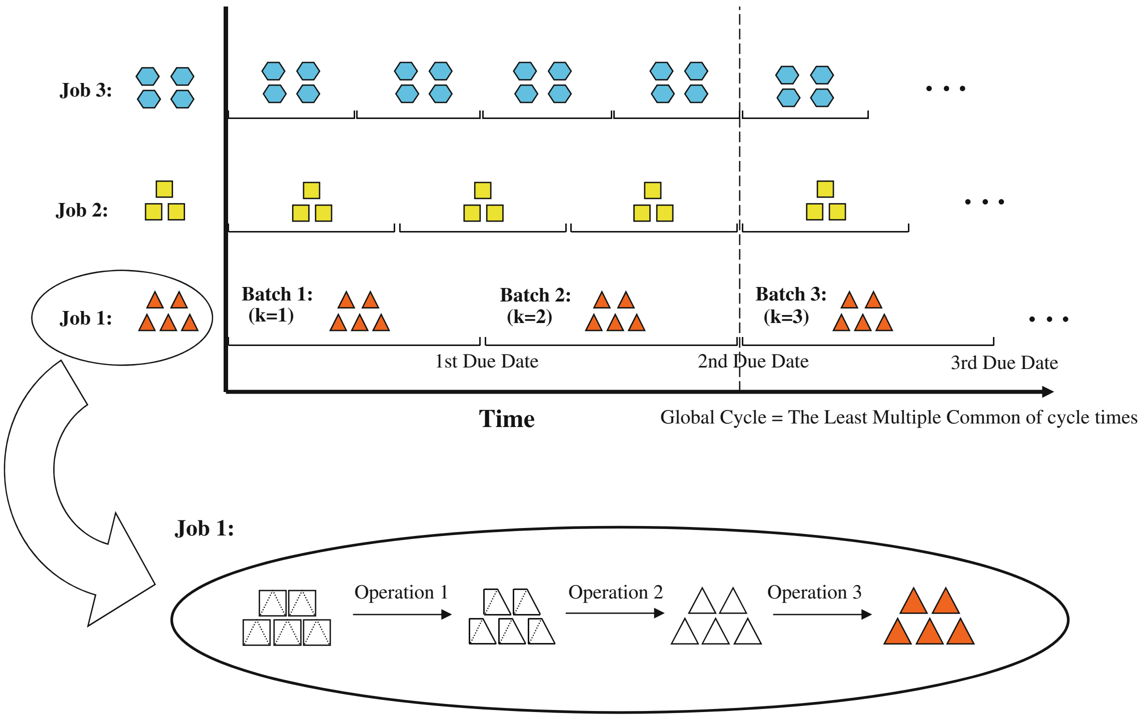

Fig. 1 A sample of cyclic jobs

$hc_{j,h}$ per unit time. The goal of the considered scheduling problem is to minimize the total cost including delay cost, setup cost, and WIP and finished product holding costs.

The proposed framework for solving

Since the scheduling horizon is infinite, a framework is suggested to breakdown it to short term horizons and then schedule operations in it. The scheduling plan for the infinite horizon can be produced by repeating this scheduling pattern through time. So, the selected length of the short term horizon must certify that the final schedule minimizes the total cost in long term.

For each job, the time intervals between its due dates are called cycle time. Since the cycle times for various jobs are different, if the least common multiple (LCM) of cycle times is nominated as global cycle (g) (global cycle is illustrated in Fig. 1), the short term horizon (H) is equal to an integer multiple of the global cycle:

$$H = \gamma \cdot g, \tag{1}$$

where γ is an integer number. Repeating the scheduling pattern which obtained for this short term horizon makes the general production schedule. Now, the problem is to find the proper γ which minimizes the total cost for the long term horizon.

The setup costs persuade the manufacturer to produce some batches together to eliminate the setup cost between successive production runs. On the other hand, processing too many batches together may interrupt the process of other jobs and also causes more holding costs. Hereupon, the optimum batches which should be produced together are dependent to the setup cost and holding cost. Note that it is possible that the optimum batches which should be produced together be even larger than the number of batches in the global cycle. This is the reason of defining γ in Eq. 1. In industries this condition may be seen where the setup costs are too large compared with the holding costs (where the cost is critical factor) or the setup times are too large compared with the production time of a batch (where the time is critical).

Through this framework, the scheduling process for short term horizon can be performed as a flexible job shop scheduling problem with the difference that each job may be executed k times during the short term horizon.

To determine the optimal γ it is necessary to solve the problem for various values of γ with ascending order. Initially, it is supposed that $\gamma = 1$ and the problem will be solved and the total cost will be calculated upon this assumption. Then for the next steps, the problem must be solved for larger values of γ which for each step γ is one unit larger than the previous step and the total cost per global cycle will be calculated. By increasing γ, if the processing batches together, still causes to decrease the total cost per global cycle, it is essential to examine a larger γ. Otherwise if it remains constant, the previous γ is optimum and displayed by γ^*. It means each step which its total cost is equal to the previous one without any decreases, determines the optimal γ. For γs larger than its optimal value, γ^*, it is impossible to reduce the total cost per global cycle any more. This descending manner of the cost values toward the integer multiples of global cycle are represented in Fig. 2.

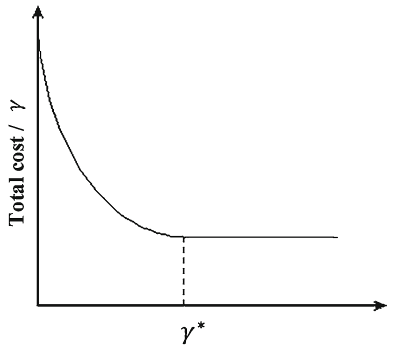

Fig. 2 The decreasing manner of total cost toward γ

Mathematical model

Parameters:

m	Total number of machines
n	Total number of jobs
h_j	Number of operations for job j
k_j	Number of batches in the short term horizon for job j
l_i	Maximum number of operations which can assign to machine i
$p_{i,j,h}$	Processing time of the operation h of job j on machine i
$st_{i,j,h}$	Setup time of hth operation of job j on machine i
$sc_{i,j,h}$	Setup cost of hth operation of job j on machine i
$hc_{j,h}$	Holding cost of job j after processing the hth operation/time unit
$d_{j,k}$	Due date for kth batch of job j
$dc_{j,k}$	Delay cost of delivering kth batch of job j after its due date / time unit
$a_{i,j,h}$	$\begin{cases} 1, & \text{if the } h\text{th operation of job } j \text{ is processable on machine } i \\ 0, & \text{Otherwise} \end{cases}$
M	An enough large number

Variables:

$t_{j,k,h}$	Start time of $O_{j,k,h}$(hth operation of kth batch of job j)
$Tm_{i,l}$	Start time of lth operation which is processed on machine i
$Ps_{j,k,h}$	Processing time of $O_{j,k,h}$ after machine assignment
$dt_{j,k}$	Delay time of delivering the kth batch of job j
$rt_{j,k}$	Rush time in preparing the kth batch of job j earlier than the due date
$y_{i,j,k,h}$	$\begin{cases} 1, & \text{if } O_{j,k,h} \text{ is assigned to machine } i \text{ for processing} \\ 0, & \text{Otherwise} \end{cases}$
$x_{i,j,k,h,l}$	$\begin{cases} 1, & \text{if } O_{j,k,h} \text{ is assigned to process at } l\text{th processing position on machine } i \\ 0, & \text{Otherwise} \end{cases}$
$e_{i,j,k,h}$	$\begin{cases} 1, & \text{if } O_{j,k,h} \text{ is immediately processed after } O_{j,k-1,h} \text{ on machine } i \\ 0, & \text{Otherwise} \end{cases}$
$w_{i,j,k,h,l}$	$\begin{cases} 1, & \text{if } O_{j,k,h} \text{ is immediately processed after } O_{j,k-1,h} \text{ at } l\text{th processing position on machine } i \\ 0, & \text{Otherwise} \end{cases}$

The considered problem can be formulated as a mixed integer linear programming (MILP) model:

$$MinTC = \sum_i \sum_j \sum_k \sum_h sc_{i,j,h} \cdot z_{i,j,k,h} + \sum_j \sum_k \mathrm{dc_j} \cdot \mathrm{dt_{j,k}} \tag{2}$$

$$+ \sum_j \sum_k \mathrm{hc_{j,h_j}} \cdot \mathrm{rt_{j,k}} + \sum_j \sum_k \sum_{h=1}^{h_j-1} \mathrm{hc_{j,h}} \times \left(\mathrm{t_{j,k,h+1}} - \left(\mathrm{t_{j,k,h}} + \mathrm{ps_{j,k,h}} + \sum_i (\mathrm{z_{i,j,k,h}} \cdot \mathrm{st_{i,j,h}}) \right) \right)$$

Subject to:

$$\sum_i y_{i,j,k,h} \cdot p_{ijh} = ps_{j,k,h} \quad \forall j,h,k; \tag{3}$$
$$t_{j,k,h} + Ps_{j,k,h} + st_{i,j,h} \cdot e_{i,j,k,h} \le t_{j,k,h+1} + \left(1 - y_{i,j,k,h}\right) \cdot M \quad \forall i,j,k,h = 1,\ldots,h_j - 1; \tag{4}$$
$$Tm_{i,l} + Ps_{j,k,h} + st_{i,j,h} \cdot e_{i,j,k,h} \le Tm_{i,l+1} + \left(1 - x_{i,j,k,h,l}\right) \cdot M \quad \forall i,j,k,h,l = 1,\ldots,l_i - 1; \tag{5}$$
$$Tm_{i,l} \le t_{j,k,h} + \left(1 - x_{i,j,k,h,l}\right) \cdot M \quad \forall i,j,k,h,l; \tag{6.1}$$
$$Tm_{i,l} + \left(1 - x_{i,j,k,h,l}\right) \cdot M \ge t_{j,k,h} \quad \forall i,j,k,h,l; \tag{6.2}$$
$$t_{j,k,h_j} + Ps_{j,k,h_j} + \mathrm{st}_{i,j,h} \cdot \mathrm{e}_{i,j,k,h} \le d_{j,k_j} + \left(1 - \mathrm{y}_{i,j,k,h}\right) \cdot \mathrm{M} \quad \forall i,j,k; \tag{7}$$
$$dt_{j,k} + \left(1 - \mathrm{y}_{i,j,k,h}\right) \cdot \mathrm{M} \ge \mathrm{t}_{j,k,h_j} + Ps_{j,k,h_j} + \mathrm{st}_{i,j,h} \cdot \mathrm{e}_{i,j,k,h} - d_{j,k} \quad \forall i,j,k; \tag{8}$$
$$rt_{j,k} + \left(1 - \mathrm{y}_{i,j,k,h}\right) \cdot \mathrm{M} \ge d_{j,k} - \mathrm{t}_{j,k,h_j} - Ps_{j,k,h_j} - \mathrm{st}_{i,j,h} \cdot \mathrm{e}_{i,j,k,h} \quad \forall i,j,k; \tag{9}$$
$$\left(x_{i,j,k,h,l} + x_{i,j,k+1,h,l+1}\right) - 1 \le \mathrm{w}_{i,j,k,h,l+1} \quad \forall i,j,h,k = 1,\ldots,k_j - 1, l = 1,\ldots,l_i - 1; \tag{10.1}$$
$$\tfrac{1}{2}\left(x_{i,j,k,h,l} + x_{i,j,k+1,h,l+1}\right) \ge \mathrm{w}_{i,j,k+1,h,l+1} \quad \forall i,j,h,k = 1,\ldots,k_j - 1, l = 1,\ldots,l_i - 1; \tag{10.2}$$
$$w_{i,j,k,h,1} = 0 \quad \forall i,j,k,h; \tag{11.1}$$
$$w_{i,j,1,h,l} = 0 \quad \forall i,j,h,l; \tag{11.2}$$
$$e_{i,j,k,h} = 1 - \sum_l \mathrm{w}_{i,j,k,h,l} \quad \forall i,j,k,h; \tag{12}$$
$$\left(y_{i,j,k,h} + e_{i,j,k,h}\right) - 1 \le \mathrm{z}_{i,j,k,h} \quad \forall i,j,k,h; \tag{13.1}$$
$$\tfrac{1}{2}\left(y_{i,j,k,h} + e_{i,j,k,h}\right) \ge \mathrm{z}_{i,j,k,h} \quad \forall i,j,k,h; \tag{13.2}$$
$$y_{i,j,k,h} \le a_{i,j,h} \quad \forall i,j,k,h; \tag{14}$$
$$\sum_j \sum_k \sum_h x_{i,j,k,h,l} \le 1 \quad \forall i,l; \tag{15}$$
$$\sum_i y_{i,j,k,h} = 1 \quad \forall i,j,k,h; \tag{16}$$
$$\sum_l x_{i,j,k,h,l} = y_{i,j,k,h} \quad \forall i,j,k,h; \tag{17}$$
$$Tm_{i,l} \le \mathrm{M} \sum_j \sum_k \sum_h x_{i,j,k,h,l} \quad \forall i,l; \tag{18}$$
$$t_{j,k,h}, Tm_{i,l}, Ps_{j,k,h}, dt_{j,k}, rt_{j,k}, l_i \ge 0 \quad \forall i,j,k,h,l; \tag{19}$$
$$x_{i,j,k,h,l}, y_{i,j,k,h}, e_{i,j,k,h}, w_{i,j,k,h,l}, z_{i,j,k,h} \in \{0,1\} \quad \forall i,j,k,h,l. \tag{20}$$

The objective function (2) minimizes the total cost including setup cost, delay cost, finished product holding cost and work-in-progress holding cost. Constraint (3) defines processing time after machine assignment for all operations. Constraint (4) restricts the start time of each operation after completion time of the previous operation of the same job. Constraint (5) restricts the start time of each operation on a machine after completion time of the previous operation on it. Constraints (6.1) and (6.2) make concurrent the start of each operation with the start of the assigned machine. Constraint (7) enforces all jobs to be completed in the short time horizon. Constraints (8) and (9) define $rt_{j,k}$ and $dt_{j,k}$. They assign the positive values of differences between completion time and due date of each batch to $rt_{j,k}$ and the negative values to $dt_{j,k}$. Constraints (10.1) and (10.2) are ordained to define nonlinear equation below:

$$w_{i,j,k,h,l} = x_{i,j,k,h,l} \cdot x_{i,j,k-1,h,l-1} \tag{21}$$

By the same reason constrains (13.1) and (13.2) are used to define the Eq. 22:

$$z_{i,j,k,h} = y_{i,j,k,h} \cdot e_{i,j,k,h} \tag{22}$$

Constraint (12) defines $e_{i,j,k,h}$. Constraint (14) causes the machine that is responsible for process of an operation to be chosen among the assignable machines. Constraint (15) leads to assign at most one operation to each processing position of a machine. Constraint (16) occasions to select just one machine among assignable machines for each operation. Constraint (17) causes to occupy just one position for processing an operation on a machine. Constraint (18) causes, all positions which are not assigned to operations, take consecutive positions at the first work of the machines. This constraint is necessary to disregard the setup times and setup costs between the same operations which are processed consecutively on a machine.

The proposed genetic algorithm

The flexible job shop scheduling problem is considered as a NP-hard problem (Fattahi et al. 2007). Therefore the mathematical model which was introduced in the previous section can solve our sample problems in admissible time only for small sizes. For larger problems it is necessary to develop a meta-heuristic to find near optimal solutions in reasonable times.

One of the recent works which has been studied on the flexible job shop scheduling problems is the Pezzella et al. (2008) work. They introduced a genetic algorithm which integrates several strategies for generating the initial population, selecting for reproduction and reproducing new individuals. They showed that using different strategies for GA operators has better performance on the flexible job shop scheduling problems in compare of the using single strategy. Therefore in this paper, this approach to the GA operators has been adopted. The algorithm which suggested by Pezzella et al. (2008) has been designed for flexible job shop scheduling problem and cyclic jobs are not discussed in it. Furthermore, its goal is minimizing the makespan, and many operators adopted by them reduce this performance measure, whereas the goal of the problem considered in this paper is to minimize the total cost. Nonetheless some strategies, which are used in their study as rules for generating initial population assignment and crossover operators, are useful for developing an algorithm.

By the adopted approach in the proposed algorithm, to generate new individuals, some individuals are used in a mating pool. Every pairs of individuals are selected randomly to produce two new children. After crossover process, mutation operators may cause some changes in the children chromosomes. The procedure will be repeated until the population size in the mating pool reaches to a predetermined level. After a stage of generation, selection operator selects some individuals and deletes others. The selected individuals will be used again in the mating pool to generate new population. This procedure will be continued to reach the maximum number of generation.

Fitness function

The fitness function of the proposed algorithm is the total cost of the solution which is presented by the chromosome. As it is described previously, the total cost contains delay cost, setup cost and holding cost for both finished products and work-in-progress.

Chromosomes

For each solution a 5-row matrix like Fig. 3 is developed as a chromosome. The specifications of each operation are displayed in a column. Hence, number of columns of the matrix is equal to the sum of all operations which must be executed in a cycle. This number is shown by s in Fig. 3. Three first rows display job, batch, and operation numbers of each column. The 4th row displays the processing position that the operation must be performed and the 5th row displays the machine number which is assigned to the operation. Note that rows of the matrix are sorted according to the position numbers.

Generating initial population

As Pezzella et al. (2008) for generating initial population assignment, the approach by localization of Kacem et al.

$$\begin{array}{r|ccccc|}
\textbf{Column} & 1 & 2 & 3 & \dots & s \\
\text{Job No.}\ (j) & 1 & 1 & 2 & \dots & j \\
\text{Batch No.}\ (k) & 1 & 2 & 1 & \dots & kj \\
\text{Operation No.}\ (h) & 1 & 1 & 1 & \dots & hj \\
\text{Position}\ (l) & 1 & 2 & 3 & \dots & s \\
\text{Assigned machine}\ (i) & 2 & 2 & 1 & \dots & i
\end{array}$$

Fig. 3 Chromosome structure

Table 1 Initial population generated by assignment rule 1

Stage	1			2			3			…	S		
Machine	m_1	m_2	m_3	m_1	m_2	m_3	m_1	m_2	m_3	…	m_1	m_2	m_3
O_{111}	2.1	2.7	–	2.1	**4.4**	–	**4.2**	4.4	–	…	*2.1*	2.7	–
O_{112}	–	2.2	2.8	–	**3.9**	2.8	–	3.9	2.8		–	2.2	*2.8*
O_{121}	2.1	2.7	–	2.1	**4.4**	–	**4.2**	4.4	–		*2.1*	2.7	–
O_{122}	–	2.2	2.8	–	**3.9**	2.8	–	3.9	2.8		–	*2.2*	2.8
O_{131}	2.1	2.7	–	*2.1*	**4.4**	–	–	–	–		*2.1*	2.7	–
O_{132}	–	2.2	2.8	–	**3.9**	2.8	–	3.9	*2.8*		–	2.2	*2.8*
O_{211}	2.6	–	3.3	2.6	–	3.3	**4.7**	–	3.3		*2.6*	–	3.3
O_{221}	2.6	–	3.3	2.6	–	3.3	**4.7**	–	3.3		2.6	–	*3.3*
O_{311}	3.1	2.7	–	3.1	**4.4**	–	**5.2**	4.4	–		3.1	*2.7*	–
O_{312}	–	*1.7*	2.3	–	–	–	–	–	–	…	–	*1.7*	2.3

Table 2 Initial population generated by assignment rule 2

Stage	1			2			3			…	S		
Machine	m_1	m_2	m_3	m_1	m_2	m_3	m_1	m_2	m_3	…	m_1	m_2	m_3
O_{121}	*2.1*	2.7	–	–	–	–	–	–	–	…	*2.1*	2.7	–
O_{132}	–	2.2	2.8	–	*2.2*	2.8	–	–	–		–	*2.2*	2.8
O_{221}	2.6	–	3.3	**4.7**	–	3.3	4.7	–	*3.3*		2.6	–	*3.3*
O_{112}	–	2.2	2.8	–	2.2	2.8	–	**4.4**	2.8		–	*2.2*	2.8
O_{211}	2.6	–	3.3	**4.7**	–	3.3	4.7	–	3.3		*2.6*	–	3.3
O_{122}	–	2.2	2.8	–	2.2	2.8	–	**4.4**	2.8		–	2.2	*2.8*
O_{111}	2.1	2.7	–	**4.2**	2.7	–	4.2	**4.9**	–		*2.1*	2.7	–
O_{312}	–	1.7	2.3	–	1.7	2.3	–	**3.9**	2.3		–	*1.7*	2.3
O_{131}	2.1	2.7	–	**4.2**	2.7	–	4.2	**4.9**	–		2.1	*2.7*	–
O_{311}	3.1	2.7	–	**5.2**	2.7	–	5.2	**4.9**	–	…	*3.1*	2.7	–

(2002) is adopted. Kacem et al. (2002) suggested two assignment rules which both of them lead to minimum possible values for both the operations' processing times and the machine workload.

Assignment rule 1: Searching the global minimum among the process times. Table 1 describes this rule. Process times for all operations on each machine are presented in this table in stage 1. For each stage, this rule searches and selects the global minimum value. Then the machine related to the selected number is assigned to the related operation and its process time will be added to the process time of all operations on the selected machine (see Table 1). In Table 1 italicized numbers show the selected number (global minimum time) and bold numbers show the numbers that are renewed in each stage by selecting a machine in previous stage. This procedure will be followed until machines assigned to all operations.

Assignment rule 2: Randomly permute operations in the table. This rule places operations randomly in the table and the assignment process begins from the first row. Other steps are similar to the rule 1.

To generate initial individuals one of mentioned rules should be selected with a predetermined probability.

In this survey the sequences of initial population are generated randomly (Table 2).

Crossover operators

Crossover operators are the main responsible for generating new individuals in genetic algorithms. In applying the crossover operators, the precedence constraints must be satisfied among operations and batches of the same job. Hence, as Kacem et al. (2002) the precedence preserving order-based crossover (POX) is adopted as one of two crossover rules.

POX operator leads to exchange the sequences of some operations. By the POX in each crossover operation two children will be generated from two parents. First, the algorithm selects a job randomly. Then POX copies all related operations to this job (including operations from all batches) from the first parent to the same positions of the first child. Then the remaining positions are completed by remaining operations as they appear in the second parent. A similar process is done for the second child.

Another crossover operator which is called random crossover just influences on machine assignments. For this purpose the operator first selects a subset of operations randomly (for example, all operations which have same batch number). For the first child, it uses the assignment pattern of the selected operations from the first parent and for the other operations, from the second pattern. A similar order will be used for the second child.

Mutation operators

According to the fitness function, five mutation operators are developed. Two operators are designed based on exchanging machine assignments and three operators based on exchanging positions in the processing positions on the same machines. After crossover operation, an assignment mutation and a sequencing mutation may be created with predetermined probabilities. These five mutation operators are described below:

- *Intelligent assignment mutation*: This operator first selects a batch of a job which causes most delay cost. Then it selects an operation for the selected job and changes its assigned machine if it is possible. This mutation may lead to reduce delay cost.
- *Randomly assignment mutation*: This operator randomly selects an operation among the operations which can be processed on more than one machine and then assigns another machine to it.
- *Intelligent sequencing mutation 1:* Same as the assignment case, this operator selects a batch of a job that causes most delay cost. Then it selects an operation for the selected job and exchanges its processing position with the operation which has the previous position if possible. Reducing one unit of position of an operation may cause reducing in the delay time and thereupon reducing in the delay cost.
- *Intelligent sequencing mutation 2:* By this operator an operation of a job which must be processed in more than one batch through a cycle is selected. Then the operator selects a batch of that job randomly. If possible, it reduces the position of the selected O_{jkh} so far as it can be processed instantly after its previous batch. Therefore this mutation leads to eliminate a setup cost between processing two similar operations of two consecutive batches.
- *Randomly sequencing mutation:* This operator selects two neighbor operations and exchanges their priorities if the precedence constraints allow.

Selection

Selection operation is responsible for selecting some individuals of the population and entering them to the mating pool to generate the new population. In this paper, among two well-known selection method (i.e. tournament and linear ranking), the linear ranking method is selected. This method first sorts all individuals according to their fitness and assigns a number between 1 and s to each individual (s is the number of all operations). The best individual gets rank s and the worst case gets rank 1. Then the probability of choosing each individual is calculated by the formula below:

$$P_i = \frac{2 \times rank_i}{s(s+1)}, \quad i = 1, 2, \ldots, \tag{23}$$

Now, if the stacked probabilities for all individuals are calculated, generating a random number $p \sim U(0, 1)$ can determine the individual for entering to the mating pool. Note that by entering each individual to the mating pool, ranks, probabilities and stacked probabilities must be calculated again for the remaining individuals so far as the mating pool become full.

Parameter setting

After each crossover operation, one of two assignment mutations and one of three sequencing mutations may be selected and cause some changes in the children chromosomes. A meta-heuristic algorithm is used to adjust the probability of using these operators. For this purpose a simulated annealing algorithm is developed which in this paper is called *regulator algorithm*. The solution seed of this algorithm is a five-member array which contains the probabilities for every five mutation operators. A small sample problem which its result (f^*) is available by MILP model, is used to evaluate the performance of the proposed genetic algorithm. The goal of the regulator algorithm is to minimize the average number of GA generations by adjusting the probability of five mutation operators. Therefore for each set of probabilities, the main algorithm runs until it reaches to at least three chromosomes with the fitness function equal to the f^*. For every probability set, the main algorithm runs ten times and the average number of generation is recorded as a factor of the main algorithm performance per the set of the probabilities. The regulator algorithm tries to achieve maximum performance by checking neighbor sets through a SA procedure. The set of probabilities which results the best performance are selected through five times regulator algorithm runs. This set is presented in Table 3. Whereas this method of parameter setting checks much more points in compare of the Design of Experience methods, it is expected that the result of this method has more accuracy.

Table 3 The results of parameter setting

Parameter	Value
Population size	100
Ratio of initial size of mating pool/population size	0.24
Number of generation	1,000
Initial population assignment probabilities	
Global minimum	10 %
Random minimum	90 %
Crossover probabilities	
POX	55 %
Randomly	45 %
Assignment mutation probabilities	
Intelligent	8 %
Randomly	5 %
Sequencing mutation probabilities	
Intelligent 1	12 %
Intelligent 2	28 %
Randomly	23 %

Results of the examinations on three different size problems showed that using both POX and random crossover simultaneously in every crossover operation, has worse effect on the fitness function. Therefore for each crossover operation just one operator applies with the adjusted probability. This probability is set by changing the values of them and fixing other parameters.

The values for all parameter in the algorithm are presented in Table 3.

The proposed comparative algorithm

To evaluate the performance of the proposed algorithm, it can be helpful to compare it with other efficient algorithms which previous experiences show their performance for similar problems. So the efficient algorithms developed for flexible job shop scheduling problems may be effective for this purpose.

On the other hand, it can be useful to apply an algorithm based on local search approach to compare the results of both approaches. Among the local search algorithms, simulated annealing is frequently applied in many studies and has been demonstrated that can be efficient especially in the case of flexible job shop problems. Therefore, another algorithm was developed based on simulated annealing approach (SA) and was compared with the proposed GA algorithm. The solution seed of this algorithm, same as the GA chromosome, includes a two dimensional matrix. Also all five mutation operators which had been used in GA procedure were used as neighborhood search methods in the SA algorithm. The initial solution for this algorithm has been generated randomly. Adjusted parameters for this algorithm are presented in Table 4.

Table 4 Adjusted parameters for the SA algorithm

Parameter	Value
l_max(number of better neighborhoods that are searched in each temperature)	20
Cooling rate	0.002
tf/t0 (final temperature/initial temperature)	1/2,000
Assignment neighborhood search probabilities	
Intelligent	15 %
Randomly	15 %
Sequencing neighborhood search probabilities	
Intelligent 1	15 %
Intelligent 2	30 %
Randomly	25 %

Computational experiments

To illustrate the efficiency of the proposed algorithm it has been implemented by C^{++} programming language. All experiments are implemented on a PC which utilizes a 3.00 GHz processor and 1 GB RAM. Twenty sample problems have been defined in various sizes. Problem specifications and number of constraints and variables are given in Table 5. m, n, kl and hl, stands for number of machines, number of jobs, maximum number of batches in a cycle for each job and maximum number of operations for each job, respectively.

To determine the short term horizon, the optimal integer multiple of global cycle, γ^*, must be selected. For this purpose, integer numbers, beginning from 1, has been tested as γ_is. The smallest γ_i which the total cost per cycle for γ_i and $\gamma_i + 1$ do not have significant difference, has been chosen as the γ^*. Since meta-heuristic algorithms follow a stochastic process, hypothesis testing is used to test the equality of total cost per cycle for two consecutive γ_i and γ_{i+1}. This testing hypothesis, tests:

$$\begin{cases} \overline{TC}_1 = \overline{TC}_2, & \textit{hypothesis}\, 0 \\ \overline{TC}_1 \neq \overline{TC}_2, & \textit{hypothesis}\, 1 \end{cases} \tag{24}$$

The appropriate statistic for this hypothesis is Z with normal standard distribution and is determined as below:

The critical area is $Z \leq -Z_{\alpha/2}$ and $Z \geq Z_{\alpha/2}$ where Z is determined as below:

$$Z = \frac{\overline{TC}_1 - \overline{TC}_2}{\sqrt{\frac{\sigma_1^2}{n} + \frac{\sigma_2^2}{n}}} \tag{25}$$

$\overline{TC}_1$ and $\overline{TC}_2$ denote the mean of total cost with different γs. If α states the confidence level of the hypothesis then the hypothesis 0 will be accepted if $Z \geq -Z_{\alpha/2}$ and $\leq Z_{\alpha/2}$. In this study $\alpha = 0.05$ is assumed.

Table 5 Sample problems definition

Problem no.	m	n	kl	hl	Number of constraints	Number of variables	Number of integer variables
P1	2	2	3	3	2,343	2,486	2,354
P2	2	3	2	2	3,983	2,743	2,595
P3	2	3	3	2	6,311	4,798	4,590
P4	3	3	4	2	4,016	3,680	3,520
P5	3	3	4	2	5,768	6,088	5,880
P6	3	4	6	3	6,932	5,770	5,578
P7	3	5	6	3	11,825	15,214	14,886
P8	4	7	6	5	55,505	79,232	78,532
P9	5	6	6	5	224,882	292,672	291,394
P10	5	7	6	6	78,408	118,989	118,157
P11	6	7	6	6	83,147	137,234	136,366
P12	7	7	6	7	483,532	783,948	781,928
P13	8	8	12	7	515,728	841,962	836,074
P14	8	9	12	7	479,197	1,106,658	1,104,190
P15	9	9	12	9	2,188,734	6,436,948	6,391,981
P16	10	12	12	9	3,509,843	11,495,292	11,414,911
P17	10	13	12	10	4,896,724	10,273,592	10,201,702
P18	10	14	12	10	7,235,749	12,047,634	11,963,394
P19	10	15	12	11	10,375,832	12,509,346	12,421,875
P20	10	15	12	12	12,498,357	14,293,180	14,114,923

Table 6 Determining best γ for problem no. 2

Problem no. 2	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 4$
Mean ($\overline{TC}$)	2	3.688	5.252	6.964
SD (s)	0	0.273	0.431	0.601
Mean ($\overline{TC}$) per global cycle	2	1.844	1.751	1.741
SD (s) per global cycle	0	0.137	0.144	0.15
Z	–	−6.257	−2.576	−0.252

Let define s as the sample standard deviation. Each problem has been run 30 times, i.e. $n = 30$. Therefore, it is supposed that the mean of fitness function has a normal distribution. So s can be substituted for σ in the above formula.

For instance, finding best γ procedure, for problem number 2 is presented in Table 6. The mean and the standard deviation of total cost for short term horizon and also for global cycle are represented in this table. Initially, the problem was run for $\gamma = 1, 2$. Since $Z_{0.025} = 1.96$ and for this step $= -6.257 \leq -Z_{0.025}$, the hypothesis 0 is rejected. Hence, it is necessary to consider larger γs. Finally for $\gamma = 4$, Z is located in the acceptable area and hypothesis 0 will be accepted. This indicates the total cost per global cycle for both $\gamma = 3, 4$ do not have significant difference. Thus the optimal integer multiple of the global cycle is $\gamma = 3$.

For problems No. 11–20 running 30 times became too time-consuming. So each member of this group has been run 10 times. Despite $n < 30$, because $n_1 = n_2 = 10$ it is possible to use the below formula to test the equality of mean total costs hypothesis:

$$T = \frac{\overline{TC}_1 - \overline{TC}_2}{S_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \tag{26}$$

where:

$$S_p^2 = \frac{(n_1 - 1)\, s_1^2 + (n_2 - 1)\, s_2^2}{n_1 + n_2 - 2} \tag{27}$$

Note that the statistic T follows t-distribution and the critical areas for these problems are $T < -t_{0.025,9}$ and $T > t_{0.025,9}$.

All 20 sample problems has been solved by both the proposed GA, and the comparative SA algorithm. The results of these experiments including the best γ, the mean and the best results in their short term horizon are represented in Table 7. In this table, the best γ for each problem is stand in Column 2. The Mean and the best results for two algorithms and their solution times are presented in the next columns. As a result, the SA shows better performance for small size problems but for larger ones, as is shown in Table 7 and Fig. 4 there is a

Table 7 Results of the proposed GA and the comparative SA

Problem specification		Proposed GA				Comparative SA			
Problem no.	γ	Mean	Best solution	CPU time (s)	SD	Mean	Best solution	CPU time (s)	SD
P1	2	50.14	42.40	5.08	12.85	41.55	37.2	9.08	2.26
P2	3	5.25	4.73	6.03	0.43	5.64	5.53	15.59	0.06
P3	3	76.18	73.80	7.36	2.21	83.51	79.65	26.31	2.77
P4	2	61.24	44.00	5.13	11.31	62.46	48.4	27.44	6.27
P5	2	127.26	95.61	6.71	35.35	198	166.4	43.20	14.34
P6	1	66.05	62.00	8.93	6.50	107.41	93.5	37.9	7.12
P7	1	65.95	60.14	9.89	4.28	86.90	83.3	34.12	2.79
P8	1	250.04	217.12	30.29	19.80	296.60	280.3	37.04	11.6
P9	2	299.21	2,285.12	66.34	37.66	366.71	328.7	23.89	22.85
P10	1	895.62	784.61	29.17	79.73	1,062.01	1,026.8	22.36	15.71
P11	1	211.75	186.62	32.91	19.12	278.42	261.1	15.74	10.30
P12	2	231.17	205.85	33.93	14.41	338.47	318.4	57.97	12.24
P13	2	232.73	213.19	35.13	18.84	357.72	337.8	59.22	16.63
P14	1	547.52	498.42	87.94	44.37	849.31	779.8	55.54	31.60
P15	2	530.14	482.04	88.45	33.71	848.10	817.4	53.24	23.81
P16	2	790.56	728.99	107.16	52.20	1,276.81	1,138.9	63.78	61.29
P17	1	1,034.03	985.30	157.98	47.35	1,629.50	1,551.4	122.8	65.62
P18	1	1,180.81	1,098.71	168.23	64.94	1,989.24	1,815.8	140.6	188.57
P19	1	1,242.32	1,185.11	159.59	43.82	2,007.08	1,816.8	144.8	96.79
P20	1	1,236.34	1,178.42	161.08	42.90	2,171.81	2,011.2	121.1	119.52

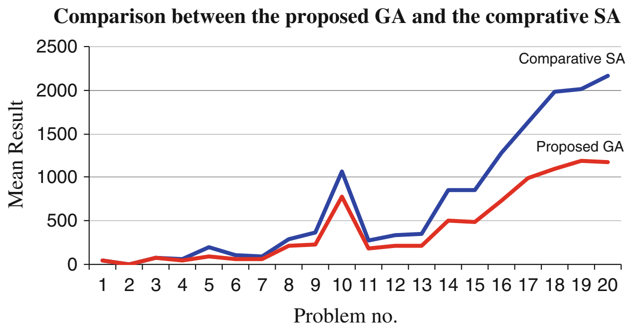

Fig. 4 Comparison between the GA and the SA results

meaningful difference between them, and the proposed GA displays better results.

Table 8 compares the results of these algorithms. The relative deviation criterion is used to compare the results of other algorithms with the proposed GA. This criterion is obtained as below:

$$Dev = \left(\frac{\overline{TC}_{comp} - \overline{TC}_{prop}}{\overline{TC}_{prop}} \right) \times 100\,\%, \tag{28}$$

where $\overline{TC}_{prop}$ denotes the mean total cost of the proposed GA and $\overline{TC}_{comp}$ is the mean total cost of other algorithm which the proposed GA has been compared to.

Table 8 illustrates that the proposed GA results for some problems are even more than 30 % better than the SA results. Also to evaluate the effect of the intelligent mutation operators that has been used in the proposed GA, all sample problems are solved by the similar GA which utilizes just randomly mutation operators. Numerical results for this algorithm are represented in Table 8. Relative deviations for results of this algorithm are given in Table 8 and also Fig. 5 clearly demonstrates the efficiency of the designed intelligent mutations.

Finally to display the performance of two proposed algorithms, it is useful to compare them with the MILP model which searches the solution space for the global optimal point. Therefore, because of its long computation time, this method for NP-hard problems is too much deficient and cannot be used for practical purposes. The considered MILP model given in the mathematical model section has been coded by software package LINGO 9.0. Table 9 displays the results of problems number 1–8. All light gray rows are related to samples which have been defined in Table 9 and their optimal multiples of global cycle, γ^*, are demonstrated. Since the mathematical model cannot be used for medium or

Table 8 Considering the performance of the proposed algorithm

Problem specification		Proposed GA	Proposed GA with random mutation		Comparative SA	
No.	γ	Mean	Mean	Dev (%)	Mean	Dev (%)
P1	2	50.14	51.16	2.00	41.55	−20.67
P2	3	5.25	5.01	−4.80	5.64	6.91
P3	3	76.18	78.54	3.00	83.51	8.77
P4	2	61.24	58.88	−4.01	62.46	1.95
P5	2	127.26	132.73	4.12	198.00	35.73
P6	1	66.05	93.74	29.53	107.35	38.47
P7	1	65.95	70.07	5.88	86.86	24.07
P8	1	250.04	303.09	17.50	296.55	15.68
P9	2	299.10	401.70	25.54	366.70	18.43
P10	1	895.62	1,274.28	29.72	1,062.00	15.67
P11	1	211.75	244.47	13.38	278.36	23.93
P12	2	231.17	253.65	8.86	338.47	31.70
P13	2	232.73	247.42	5.94	357.73	34.94
P14	1	547.52	614.48	10.90	849.29	35.53
P15	2	530.14	538.87	1.62	848.10	37.49
P16	2	790.56	787.16	−0.43	1,276.79	38.08
P17	1	1,033.96	1,044.31	0.99	1,629.53	36.55
P18	1	1,180.77	1,216.25	2.92	1,989.17	40.64
P19	1	1,242.27	1,297.78	4.28	2,006.96	38.10
P20	1	1,236.26	1,288.38	4.05	2,171.80	43.08

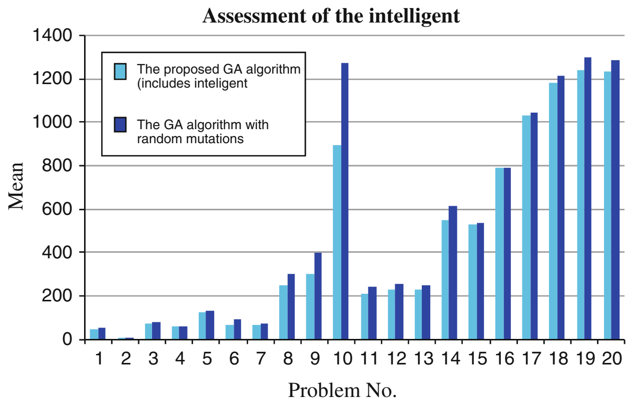

Fig. 5 Considering the efficiency of the intelligent mutation

large sizes, the small size problems with $\gamma = 1$ are used to increase the number of suitable problems for solving and consequently for better compare the results of the proposed algorithm and LINGO. The solutions of these samples are given as the dark gray rows. Finding every global optimal solution is important to evaluate the performance of the proposed algorithm and Table 9 shows that this purpose has been achieved by expanding the small size problems.

In Table 9, for each problem that the global optimal solution could not be achieved in admissible time (4 h for our survey), the best local optimal solution has been reported. In addition, the results of solving relaxed model as a linear programming model by eliminating all integer variable constraints are given in column 14. Also for those problems that the global optimal solutions are not available, the best achieved lower bounds from the software are exhibited. Note that for the cases larger than problem number 5, neither solution nor lower bound could be found by MILP model.

As shown in Table 9, the MILP model obtained global optimal solution just for problems number 1 to number 4 with assumption $\gamma = 1$. For these four cases the proposed GA has been achieved near optimal solution with an admissible gap and significantly less solution time. Also, the SA has been reached to the global optimal solution successfully. For other problems there are substantial differences between the proposed algorithms and the MILP model and the proposed GA and SA have a great performance in competition. Furthermore, the large gap between the local solution and the best upper bound for these problems is a weak point of the MILP model.

As another measure to evaluate the performance of the proposed algorithm in larger size cases, relaxing the model to a linear programming model was adopted, but as it can be seen in Table 9, for those problems which the global optimal solutions are available, the solution of the LP relaxed model has a great distance from the optimal solution and conse-

Table 9 Comparison between the proposed GA, comparative SA and the MILP model

Problems		Proposed GA				Comparative SA				MILP-mixed integer programming model				
Problem no.	γ	Ave. of solutions	Best solution	CPU Time (s)	SD	Ave. of solutions	Best solution	CPU time (s)	SD	Solution	Type of optimal solution	Best lower bound	LP	CPU time
P1	1	22.70	22.70	2.02	0.00	22.70	22.70	12.39	0.00	22.70	Global	–	11	36 s
P1	2	50.14	42.40	5.08	12.85	41.55	37.2	9.08	2.26	–	–	15.04	11	4 h
P2	1	2.00	2.00	0.69	0.00	1.915	1.915	1.633	0.00	1.915	Global	–	1.3	6 s
P2	3	5.25	4.73	6.03	0.43	5.64	5.53	15.59	0.06	19.95	Local	1.5	1.3	4 h
P3	1	32.07	32.00	2.42	0.38	32.0	32.0	0.97	0.00	32.0	Global	–	25	78 s
P3	3	76.18	73.80	7.36	2.21	83.51	79.65	26.31	2.77	196.3	Local	25	25	4 h
P4	1	34.32	30.60	2.59	3.83	30.6	30.6	23.46	0.00	30.6	Global	–	22	314 s
P4	2	61.24	44.00	5.13	11.31	62.46	48.4	27.44	6.27	320.8	Local	23.51	22	4 h
P5	1	80.59	64.40	3.51	14.96	64.4	64.4	56.156	0.00	94.6	Local	52	44	4 h
P5	2	127.26	95.61	6.71	35.35	83.51	79.65	26.31	2.77	475.2	Local	44.26	44	4 h
P6	1	66.05	62.00	8.93	6.50	198	166.4	43.20	14.34	–	–	–	14	4 h
P7	1	65.95	60.14	9.89	4.28	107.41	93.5	37.9	7.12	–	–	–	19	4 h
P8	1	250.04	217.12	30.29	19.80	296.60	280.3	37.04	11.6	–	–	–	53	4 h

quently they are not a confident measure to evaluating other solvers or methods.

As a result from Table 9, for such small problems the SA algorithm gives better solutions than the proposed GA but as illustrated in Table 7, for larger problems the SA algorithm loses its efficiency rapidly.

Conclusion and further research

In this paper a specific problem of a cyclic scheduling problem that is faced by automotive part suppliers and other manufacturers who must deliver their products to the customer through kanban policy is considered. The goal of this model is to minimize the total cost. A method is introduced to determine a short term horizon schedule and then generalize the result to long term horizon. To find the schedule in the short term horizon a mathematical model for small size problems and a genetic algorithm for medium and large size problems are introduced. The performance of the proposed genetic algorithm and its intelligent mutations are illustrated by some comparative experiments. The proposed algorithm could find better solution as compared to the other algorithms in short computation times.

There are many research opportunities to modify the traditional scheduling problems for using in different industries. Through this paper we studied cyclic jobs as a common feature in part manufacturing industries. Another characteristic in such workshops is human constraints. Usually in scheduling studies, machine is considered as the single limit resource to process jobs, whereas human resource is another constraint for such corporations. To process a job, besides specific machine, there is need to assign suitable operator with specific skills to handle the machine. Therefore a practical scheduling program in such industrial environments should schedule both machine and human simultaneously to perform jobs.

Another difficulty for implying the results of considered paper in a part manufacturing firm is machine failures. As an opportunity to develop the considered paper is to study the cyclic flexible job shop scheduling problem including machine failures.

References

Bahroun, Z., Baptiste, P., Campagne, J. P., & Moalla, M. (1999). Production planning and scheduling in the context of cyclic delivery schedules. *Computers and Industrial Engineering*, *37*(1), 3–7.

Brucker, P., & Kampmeyer, T. (2008a). A general model for cyclic machine scheduling problems. *Discrete Applied Mathematics*, *156*(13), 2561–2572.

Brucker, P., & Kampmeyer, T. (2008b). Cyclic job shop scheduling problem with blocking. *Journal of Intelligent Manufacturing*, *159*(1), 161–181.

Cavory, G., Dupas, R., & Goncalves, G. (2005). A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints. *European Journal of Operational Research*, *161*(1), 73–85.

Fattahi, P., SaidiMehrabad, M., & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, *18*(3), 331–342.

Fattahi, P., Jolai, F., & Arkat, J. (2009). Flexible job shop scheduling with overlapping in operations. *Applied Mathematical Modeling*, *33*(7), 3076–3087.

Hanen, C. (1994). Study of a NP-hard cyclic scheduling problem: The recurrent job shop. *European Journal of Operational Research*, *72*(1), 82–101.

Hanen, C., & Munier, A. (1997). Cyclic scheduling on parallel processors: An overview. In P. Chretienne, E. G. Coffman, J. K. Lenstra, & Z. Liu (Eds.), *Scheduling theory and its applications*. New York: Wiley.

Kacem, I., Hammadi, S., & Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible job shop scheduling problems. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, *32*(1), 1–13.

Leung, J. (2004). *Handbook of scheduling algorithms, models and performance analysis*. Boca Raton, FL, USA: CRC Press.

Ouenniche, J., & Bertrand, J. W. M. (2001). The finite horizon economic lot sizing problem in job shops: The multiple cycle approach. *International Journal of Production Economics*, *74*(1), 49–61.

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the flexible job shop scheduling problem. *Computers and Operations Research*, *35*(10), 3202–3212.

Romanova, A. A., & Servakh, V. V. (2009). Optimization of processing identical jobs by means of cyclic schedules. *Journal of Intelligent Manufacturing*, *3*(4), 496–504.

Roundy, R. (1992). Cyclic schedules for job shops with identical jobs. *Mathematics of Operations Research*, *17*(4), 842–865.

Song, J.-S., & Lee, T.-E. (1998). Petri net modeling and scheduling for cyclic job shop with blocking. *Computers and Industrial Engineering*, *34*(2), 281–295.