Expert Systems with Applications 38 (2011) 7609–7615

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Solving flexible flow-shop problem with a hybrid genetic algorithm and data mining: A fuzzy approach

H. Khademi Zare, M.B. Fakhrzad *

Department of Industrial Engineering, Yazd University, Yazd, Iran

ARTICLE INFO

Keywords:
Flexible flow-shop scheduling
Genetic algorithm
Data mining
Attribute-driven deduction
Fuzzy sets

ABSTRACT

In this paper, an efficient algorithm is presented to solve flexible flow-shop problems using fuzzy approach. The goal is to minimize the total job tardiness. We assume parallel machines with different operation times. In this algorithm, parameters like "due date" and "operation time" follow a triangular fuzzy number. We used data mining technique as a facilitator to help in finding a better solution in such combined optimization problems. Therefore, using a combination of genetic algorithm and an attribute-deductive tool such as data mining, a near optimal solution can be achieved. According to the structure of the presented algorithm, all of the feasible solutions for the flexible flow-shop problem are considered as a database. Via data mining and attribute-driven deduction algorithm, hidden relationships among reserved solutions in the database are extracted. Then, genetic algorithm can use them to seek an optimum solution. Since there are inherited properties in the solutions provided by genetic algorithm, future generation should have the same behavioral models more than preliminary ones. Data mining can significantly improve the performance of the genetic algorithm through analysis of near-optimal scheduling programs and exploration of hidden relationships among pre-reached solutions.

1. Introduction

Flow-shop with parallel machines (FSPM) problem is well-known to flexible flow-shop problem (Hoogeveen, Lenestra, & Veltman, 1996; Janiak & Lichtenstein, 2001; Tian, Jian, & Zhang, 1999). This problem includes a number of products with the same production steps. There are parallel machines in each production stage (work-station). All products have should pass step 1 to step "*n*" (Artiba & Elmaghrabhy, 1997; Shiau, Cheng, & Huang, 2008).

Each job is processed just by one machine in each step. Also, each machine is working on one job each time and equally each job is under processing just using one machine each time. Machine center can be filled by uniform or unrelated machines.

In this paper, we suppose that three types of machines in each center. These machines are equal in operation type and different in operation speed.

In recent two decades, many researchers focus on flexible flow-shop problem which is a NP-Hard problem. Most researchers considered maximum finish time as a goal function. As well as mathematical modeling, deterministic, heuristic and even meta-heuristic methods are used (Chen, 1995; Gupta, 1988; Haouari & Hallah, 1997). Others preferably applied deterministic and heuristic methods to solve small-size problems (Botta-Genoulaz, 2000; Braha & Loo, 1999; Kyparisis & Koulamas, 2001). The main parameter used in these papers is to minimize tardiness time for all jobs. Customer's orders represent input jobs to the system. For each job, the setup and process times are determined by process unit and due date is established by customers. Botta-Genoulaz (2000) presented six heuristic algorithms to minimize the maximum tardiness in a flexible flow-shop problem with different due dates. Lin and Liao (2003) developed a near-optimal heuristic algorithm to minimize maximum tardiness in a 2-stage flexible flow-shop problem. Bertel and Billaut (2004) proposed a heuristic algorithm for a scheduling problem in a flow-shop environment with regards to minimizing the numbers of operations with balanced tardiness. They have used a combination of integer-linear programming and genetic algorithm for the mentioned problem. Vob and Witt (2007) studied on a flexible flow-shop which is an actual scheduling problem including 16 processing steps to minimize balanced tardiness time. They have developed a mathematic model on the basis of scheduling problem with limitation of resources. Then they have solved the problem heuristically. Besides pointed out algorithm used to solve different types of problems, a number of other algorithms can be found. For example, there are lots of algorithms most of which use neighborhood search, tabu search and simulated annealing. Some of them also use branch and bound algorithm (Chen, Pan, & Lin, 2008; Garey & Johnson, 1979; Karimi & Zandieh, 2009).

The above techniques have significantly success to solve combinational optimization problem. The flexibility of these techniques

* Corresponding author.
E-mail address: mfakhrzad@yazduni.ac.ir (M.B. Fakhrzad).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.12.101

especially genetic algorithm has extended their application scope (Deborah & Freitas, 2004; Roshanaei, Seyyed Esfehani, & Zandieh, in press). In this paper, we have used a different analytical glass to the flexible series scheduling problem to improve the efficiency of the solution. We have also used data mining to study the properties of the solutions to find their hidden relationships. Data mining is an effective tool to manage huge databases. In the presented methodology, each feasible scheduling solution is considered as a database member. After clustering the data, their properties are recorded. Then, logical relationships among the operation's properties and their sequence have been developed through data mining deduction algorithm. The extracted rules help genetic algorithm to boost its speed to the optimal point. In each stage of scheduling, there are three types of machines with low, medium and high operation speed. A triangular fuzzy number is assigned for the time of operations. The structure of the remaining part of the paper is as follows: In Section 2, we have studied the flexible series scheduling problem. In Section 3, the application of data mining in the flexible series scheduling is studied. The genetic algorithm is applied to the problem in Section 4. The proposed methodology is discussed in Section 5. In Section 6, we analyzed the efficiency of the methodology and finally, we illustrate the findings in Section 7.

2. Flexible series scheduling problem

A fuzzy flexible flow-shop problem is considered as a difficult problem in production management and combinational optimization. In this problem, a set of jobs having same sequence are existed. There are n jobs which totally should pass m production steps. Each job in the whole process uses a unique machine. Each step includes a number of machines with different operation speed: low, medium or high (Behnamian, Fatemi Ghomi, & Zandieh, 2009). The time of operations for each job would be different on the basis of the assigned machine. The operation time follows a triangular fuzzy number. The objective of such problem is to sequence jobs on the existing machines so that the sum of tardiness time would be minimized. In general, the start time and operation time for each job can be different. In this paper, we suppose the starting times for all the jobs are equal.

Since the flexible flow-shop problem is an NP-Hard problem, the number of flexible solutions for sequencing jobs increase on the basis of exponential function.

For a flexible flow-shop machine including 6 jobs and 1 machine, the total number of feasible and infeasible solutions to sequence jobs are equal to 720 (i.e. 6!). Suppose the number of machines is 6, then, the answer is equal to 518400 sequencing routes. To encounter to this problem, a simple search mechanism can hardly find the optimal solution. Different search algorithms are developed to solve such problems but, they still cannot guarantee to reach the optimal solution.

It should be noted that since we can consider different idle times between two adjacent machines hence there are unlimited quantity of feasible scheduling programs regarding fuzzy approach. It is so clear that after sequencing the machines, the mentioned idle time between two adjacent machines isn't so effective. As a result, there is no need to search within all feasible solutions. Thus, a set of feasible times is defined and the solution space can be decreased. Now, the optimal solution will be in this space reasonably.

3. Flexible flow-shop problem and data mining

Today, human has to use new tools to capture the knowledge via analyze gathering information intelligently. Knowledge perception and mining data bases are some kinds of Artificial Intelligence tools which help us to analyze huge volume of information. At the other hand, knowledge perception includes process of evaluation and interpretation of models to capture what we call is knowledge. As well, data mining is a typical application of deduction algorithms which is used to identify behavioral models in data or trends in a specified context. Most of data mining algorithms are derived from machine-learning, trend recognition and statistical techniques. These algorithms include clustering, grouping and graphical models. Primary objective are described in these algorithms. Before mining data, the appropriate algorithm should be selected and data partially organized. In this paper, property-driven deduction algorithm is selected as a data mining tool. The first step in studying flexible flow-shop problem using property-driven deduction algorithm with fuzzy approach is to prepare data.

Firstly, the effective parameters for sequencing the jobs should be recognized. Then, concepts and levels of the hierarchical structure are presented. Also, defined concepts are applied on higher levels. The whole indicated procedure above must be done by an expert whose knowledge and intuition is sufficient to the problem space (Chen, Hong, & Vincent, 2009; Lin, Shi-Jinn Horng, Yuan-Hsin Chen, Rong-Jian Chen, & I-Hong Kuo, 2009). In this paper, we identified five effective parameters which can affect job sequencing. They are:

i. **Machine grade:** ***j***th job would be done by machine with grade m in ***k***th step. For each operation, there are three types of machines including low, medium and high speed. Note that lower grade machine equals to higher speed one.
ii. **Operation time:** The operation time needed for the ***j***th job in step ***k*** is determined by machine of grade ***m***.
iii. **Remaining time:** It is equal to the sum of processing time for the remaining ***j*** jobs after passing ***k*** steps
iv. **Load on machine:** Sum of the total processing time should be considered on machine without any specific sequencing.
v. **Delivery time:** Time needed for delivering ***j***th job to the customer.

4. Genetic algorithm and FFS problem

Genetic algorithm is a statistical method for the optimization and search problems. The algorithm is a member of evolutionary calculations in the field of Artificial Intelligence. Its unique property makes the algorithm so effective that we cannot assume it as a simple stochastic search method. In fact, the preliminary idea of genetic algorithm comes from Darvin's evolution theory (Casillas, Francisco, & Martínez-López, 2009). The algorithm is based upon natural behavior of human genes. To solve optimization problems, parameters like search space, coding type and method, shapes of chromosomes and their lengths, population size in each generation, fitness function and the most important ones including mutation and cross-over types must be considered. The search space refers to the space we are looking for optimal solution in it.

The shape and type of chromosomes in the genetic algorithm should be in such a way that the algorithm can solve the problem with predefined accuracy. It also must search the space well. The population size in each generation relates to the problem type. The fitness function shows the ideality level of the genetic algorithm. The chance of each parent chromosome to be selected to create child chromosomes is due to its fitness function. Most of the times, the best chromosome is transferred to the next generation. The mutation and cross-over operators have significant role on performance of genetic algorithm. The "pmx" and "6x" are two famous cross-over operators (Garey & Johnson, 1979). Besides,

the combination of data mining into genetic algorithm on the basis of a fuzzy approach significantly has improved the performance of the algorithm. As well, it has increased the convergence rate to the optimal solution (Casillas et al., 2009; Yang & Wang, 2008). During the recent decade, regular endeavors are made to solve flexible flow-shop problems using fuzzy approach and genetic algorithm. Many researchers are now working on combination of genetic algorithm and flexible scheduling problem. In their studies, it can be found two dominant questions (Chang, Chen, & Liu, 2007; Cheng & Chang, 2007):

(1) How it is possible to format a feasible solution into a chromosome scheme. Because of complicated constraints existed in FFS problems, the use of a simple binary coding method is not pleasant. Certainly, such coding results in impossible answers.
(2) How it is possible to increase the performance of the algorithm via traditional heuristic methods. Since the genetic algorithm is not an appropriate method for local search, there have been developed a lot of flexible methods from simple precedence rules to complex algorithms like branch and bound.

Different genetic algorithms are presented to solve FFS problems on the basis of fuzzy approach. These algorithms are classified into three groups including *adaptive*, *heuristic* and *combinational* (Cheng, Mitsuo & Yasuhiro, 1999). In this paper, we used combinational genetic algorithm. The most important characteristic of genetic algorithm are its mutation and cross-over operators. We have used "pmx" cross-over and substitution operators. The rate of replacement is 95%. The mutation rate is 5%. These percentages come from via trial and error. Based on the problem's condition, they can be different.

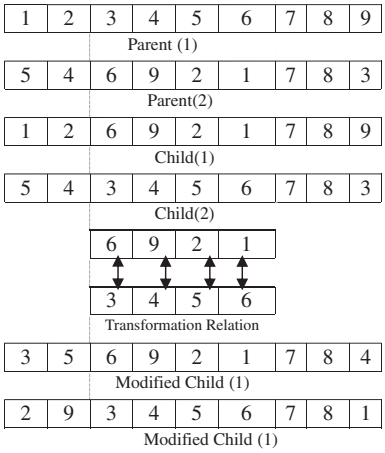

Fig. 1. Operator of "pmx".

4.1. Mutation and cross-over operators

In "pmx" and cross-over operator, two points having the same positions in the route are randomly selected. First, we substitute both sub sections of chromosomes with each other. According to the substitution, the substitution relationships can be found. Finally, the new children are modified. The mutation operator generates a random number for the gene on/off chromosome. IF the generated random number is fewer than mutation rate, the gene randomly is changed to the other forms; else, the gene is not change (Fig. 1).

4.2. Attribute-driven deduction

Attribute-driven deduction is a search method to search within a set. It makes a collection of inter-related jobs due to their properties. The method has been developed to extract the clustering rules from relational databases by means of a hierarchical concept through a deduction process. In fact, the deduction algorithm tries to replace the lower level concepts or raw data with higher level ones. Universalization process of the concepts of each attributes has a specific threshold. The process makes some parts of data to have identical attributes. Also, data with the same repetitive attributes are purged. A hierarchical concept is a sequence of relationships among a set of corresponding concepts which exist in higher levels. The hierarchical concept which can be created with sufficient knowledge to the problem is the key part of decision-making process in the attribute-driven deduction algorithm. When we move bottom-up in a sample hierarchical concept, concepts will be developed from specific to general. Absolutely, we can see general words on the top of the concept tree.

Example: For a better conceptualization, an example is presented here:

Suppose a series scheduling problem with 6 jobs, 6 steps and 3 types of machines. The problem data are shown in Table 1.

From the presented data, we can estimate the average load on machines, required processing time for each job and the remaining time for each job to be completely processed (Table 2).

The average load on machine i = sum of the operation time of the first grade machine + 4 $*$ sum of the operation time of the second grade machine + sum of the operation time of the third grade machine/6.

Expected time of doing operation and remained time can be seen in Table 3.

By first looking at the Table 1, it can be conceived that the least processing time among all operators are equal to 1. Also the maximum processing time is 12. Furthermore, Table 1 shows repetitive numbers. Therefore, we can classify the processing times into three groups: *"low", "medium"* and *"high"*. If the "processing times" are labeled as an attribute, then, "low", "medium" and "high" are the three concepts which define the attribute. Here, the "processing time" attribute is just decomposed into one level below. The procedure is repeated for all of the attributes. The decomposition of

Table 1
Time of doing the operation j in stage k by machine with grade m and delivery time of operation

Stage operation	First grade machine						Second grade machine						Third grade machine						Delivery time
	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	
1	3	5	8	9	5	8	2	4	7	8	4	7	1	3	6	7	3	6	40
2	10	7	12	12	12	6	9	6	11	11	11	5	8	5	10	10	10	4	35
3	7	6	10	11	3	9	6	5	9	10	2	8	5	4	8	9	1	7	45
4	7	7	7	5	10	11	6	6	6	4	9	10	5	5	5	3	8	9	45
5	11	5	7	6	5	3	10	4	6	5	4	2	9	3	5	4	3	1	65
6	5	5	11	12	6	3	4	4	10	11	5	2	3	3	9	10	4	1	40

Table 2
Average load on machines.

Stage	1	2	3	4	5	6
Average load on machines	24	29	49	49	35	34

"processing time" attribute into three concepts of "low", "medium" and "high" is shown in Fig. 2.

As a result, for the "operation grade", "machine load", "remaining time" and "delivery time" we have Figs. 3–6.

Although, the definition of concepts and determination of their quantities is selective, since the search process is made in a very complex space, the more decomposition makes more confusion. As we can see, the numbers shown in the last column (Table 3) are all zero, since the final process on jobs isn't a requisite for other jobs; hence their remaining time is zero.

4.3. Job assignment rules to machines

We follow the below rules for the job assignment:

(1) If the "remaining time" and "delivery time" are "low", then, "machine load" and "operation grade" are "medium".
(2) If the "remaining time" and "delivery time" are "medium", then, "machine load" and "operation grade" are "low".
(3) If the "remaining time" and "delivery time" are "high", then, "machine load" and "operation grade" are "low".
(4) If the "remaining time" and "delivery time" are "low", then, "machine load" and "operation grade" are "high".
(5) If the "remaining time" is "medium" and "delivery time" is "on time", then, "machine load" and "operation grade" are "medium".
(6) If the "remaining time" and "delivery time" are "low", then, "machine load" and "operation grade" are "medium".
(7) If the "remaining time" and "delivery time" are "low", then, "machine load" and "operation grade" are "medium".
(8) If the "remaining time" and "delivery time" are "low", then, "machine load" and "operation grade" are "medium".
(9) If the "remaining time" and "delivery time" are "low", then, "machine load" and "operation grade" are "medium".

In each cells of Table 4, there are four letters. The upper letter shows the "machine grade" or "machine speed" and the lower letter shows "machine load". We have used "L", "M", and "H" for "Low", "Medium" and "High" respectively.

4.4. Similarity concept

According to the "similarity concept", we can make divers preliminary population classes. Consider the following example for a better understanding:

Suppose a flexible series scheduling problem which we have two different solutions from a set of existing feasible solutions. They have shown in Tables 5 and 6.

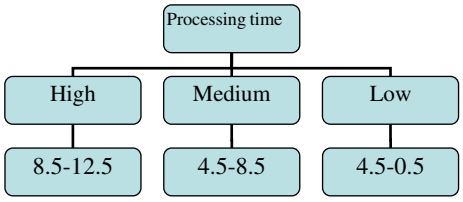

Fig. 2. Decomposition of "processing time".

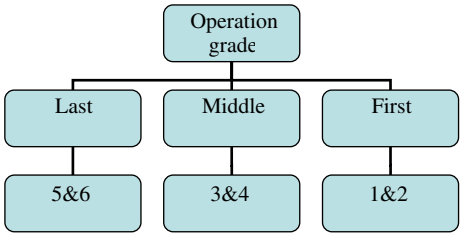

Fig. 3. Decomposition of operation grade.

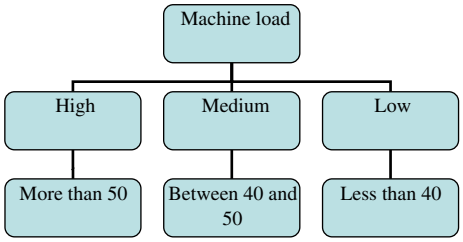

Fig. 4. Decomposition of machine load.

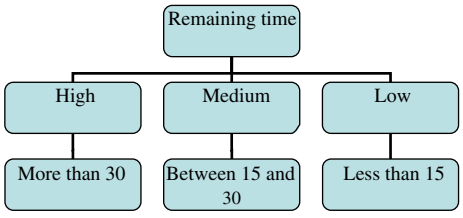

Fig. 5. Decomposition of remaining time.

The data presented in Tables 5 and 6 shows the machine types used for *j*th job in *k*th step. To calculate the similarity grade between these two programs, consider the following procedure:

As you can see the 1st job is processed by machine 1 in step 1 in both programs. Then, the similarity grade is equal to 1. The 2nd job uses the same machine in steps 1, 3 and 4 for both programs. Then, the similarity grade is 3. Finally, the similarity grades for the both 3rd and 4th jobs are 1. Having similarity grades for both programs,

Table 3
Expected time of doing operation and remaining time.

Operation	Expected time of doing operation *j* in stage *k*						Remaining time of operation *j*						Delivery time
	1	2	3	4	5	6	1	2	3	4	5	6	
1	2	4	7	8	4	7	30	26	19	11	7	0	40
2	9	6	11	11	11	5	44	38	27	16	5	0	35
3	6	5	9	10	2	8	34	29	20	10	8	0	45
4	6	6	6	4	9	10	35	29	23	19	10	0	45
5	10	4	6	5	4	2	21	17	11	6	2	0	65
6	4	4	10	11	5	2	32	28	18	7	2	0	40

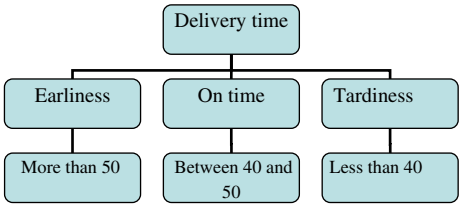

Fig. 6. Decomposition of delivery time.

we can calculate the normalized similarity grades. The normalized grades for the programs can be calculated by the feasible maximum similarity grade.

In general, the similarity grade for the same programs due to their machine types is 1. Usually, the similarity grade is less than 0.25 that hardly occurs especially in small-sized programs. In this paper, we consider 0.75 as the needed grade for generating different preliminary generations.

Normalized similarity grade = (0.25, 0.75, 0.25, 0.25).

5. The structure of the presented algorithm

The structure of the algorithm is described in this section includes:

STEP 1: Generate the preliminary population

(1-1) Consider the maximum flexible similarity grade of 0.75 according to the time interval of doing jobs.

(1-2) Generate timing programs due to the quantity of the preliminary population. In the generated programs, we consider the clause of similarity grade whereas the similarity grade of any programs shouldn't be more than the permitted maximum similarity grade. The developed timing programs are called preliminary population.

STEP 2: Developing classified jobs

(2-1) Calculates the "remaining time" and "machine load" for each machine using the input data.

(2-2) Identify effective parameters affect sequencing such as "delivery time", "machine load" and so on.

STEP 3: Determine a matrix for preliminary population classes including two dimensions: number of jobs and number of steps

(3-1) The machine type can be determined due to the acquired first generation. Each number in row "*i*" and column "*j*" shows the type of machine used in the process.

(3-2) To calculate the "similarity grade" of job "*i*" in both programs (i.e. 1 and 2), we should sum the number of processes with the same machines. Then, through dividing the number by the maximum similarity grade, similarity percentage of each job is attainable.

STEP 4: Perform the chromosome matrix

The developed matrix here has the same dimensions with the developed matrix in STEP 3. Generate a random number for each row of the matrix from 1 to *n*.

STEP 5: Generate new generation

(5-1) Develop a feasible timing program

(5-1-1) Define set *C* which includes all operations that they have no unfinished precedent jobs.

(5-1-2) Choose those operations that have the minimum completion time from operations in set *C*,

(5-1-3) Add selected operations from set *C* into the feasible timing program set.

(5-2) Repeat STEP 5–1 till all operations have been completed.

(5-3) Repeat STEP 5 for all the programs of each generation.

STEP 6: Perform new class matrix

Each value of the new class matrix is equal to the average of all values of programs in the same generation determined in STEP 5.

STEP 7: Calculate the objective function for all programs of existing generation. Apply the program with the best performance to create new chromosome. Each chromosome is displayed as a one-row matrix.

STEP 8: According to the answers from the objective function, arrange all programs in a descending manner (Except the best chromosome which directly moves to the next coming generation). Consider the quantity of objective function in the 1st column and its probability of selection in the 2nd column. Programs with better performance have the more chance to be selected as a parent.

STEP 9: Generate new chromosomes

(9-1) By generating two random numbers, select two chromosomes as parents.

(9-2) Apply the "pmx" operator and the mutation operator to generate two children.

(9-3) Repeat STEP 9 to complete the new matrix of chromosomes.

STEP 10: Restart from STEP 5 until the stopping criteria is fulfilled. (Here, the stopping criteria is to create 100 generations that each generation encompasses two chromosomes).

Table 4
The relationship among the "remaining time", "machine load" and "machine grade".

Delivery time	Remaining time		
	Low	Medium	High
Low	M/M	L/H	L/H
Medium	H/L	M/M	L/H
High	H/L	H/L	M/M

Table 5
Program 1 (preliminary population class matrix).

Operation *j*	Stage *k*			
	1	2	3	4
1	1	3	2	1
2	1	2	2	1
3	3	2	3	1
4	3	3	3	1

Table 6
Program 2 (preliminary population class matrix).

Operation *j*	Stage *k*			
	1	2	3	4
1	1	2	1	2
2	1	1	2	1
3	2	1	1	1
4	3	2	1	3

6. The efficiency of presented algorithm

To appraise the efficiency of the proposed algorithm, 300 problems are selected. To have a better estimation, the real conditions affect the solution are considered. The changing scope for the parameters due to their real conditions includes a 99% confidence interval. Required parameters to create problems are selected from confidence interval on the basis of uniform function. Confidence interval for each of parameters is defined as follows:

In each point of time, there are at least 10 jobs and maximum 200 jobs. The working steps are between 5 and 50. The number of machines in each work station is between 3 and 20. The machines do the same operations, but due to their operation's speed, they can be classified into three groups: *low*, *medium* and *high*. Setup times differ between 5 and 15. Operation times are between 1–3 months. Consider 25 working days in a month and 8 hr a day. Also, we have considered 1 hr as a break time in a day. As a result, there are 175 hr per month. To solve the problem, two programs were developed in MATLAB environment. The programs were coded on the basis of the proposed algorithm and genetic algorithm. Results derived from the proposed algorithm in comparison with genetic and linear programming algorithms are presented in Tables 7 and 8.

The presented results in Tables 7 and 8 shows that the proposed algorithm has created better answers. Besides, problems have solved in a better time. The solving time in the proposed method is reduced 48% of the ILP method whereas the GA method has decreased the solving time 14% of ILP method. On the other hand, the deviation percentage from the optimal solution in the proposed algorithm rather than ILP is 5% which is locally optimal whereas the deviation percentage from optimal solution in GA rather than ILP is about 20%.

7. Conclusion

In this paper, a hybrid algorithm is developed which is a combination of multiple algorithm including genetic algorithm, data mining, fuzzy sets, similarity algorithm and attribute-driven deduction algorithm. We solved the flexible series scheduling problem to minimize the length of timing period. The obtained answers are very pleasant. In this algorithm, the chromosomes are defined on the machines. The genes are representatives for assignment rules. Applications of data mining and attribute-driven deduction algorithm together with the genetic algorithm and fuzzy sets have significantly improved the performance of the algorithm. Also the convergence rate to the optimal solution has improved. To show the efficacy of the algorithm, a numerical example is presented. The results are compared against to GA and ILP. We have solved 300 typical real problems and compared the results against GA and ILP. The findings show a 48% time reduction in solving the problems. The local optimal solution is equal to 5%. As the application of data mining and attribute-driven deduction algorithm in fuzzy sets needs sufficient knowledge and intuition to the problem, the use of obtained information from the presented algorithm in genetic algorithm has improved the solutions.

Table 7
Comparison of objective function with GA and ILP algorithm.

Problem size (N.M.K)	Value objective function			Percent of deviation from the optimal solution	
	ILP	GA	GA/DM	$\frac{GA-ILP}{ILP}$	$\frac{(GA/DM-ILP)}{ILP}$
15 * 5 * 3	45.18	59.25	52.18	0.31	0.15
15 * 8 * 3	59.22	68.44	62.11	0.16	0.05
15 * 10 * 3	75.19	82.11	77.15	0.09	0.03
30 * 8 * 5	64.25	79.15	65.27	0.23	0.02
30 * 10 * 5	79.27	89.14	83.97	0.12	0.06
30 * 15 * 5	84.98	95.27	88.78	0.12	0.04
50 * 10 * 10	78.18	94.18	82.11	0.2	0.05
50 * 15 * 10	92.15	112.28	98.15	0.22	0.07
50 * 20 * 10	115.29	145.25	121.14	0.26	0.05
100 * 15 * 15	99.27	115.19	105.48	0.16	0.06
100 * 20 * 15	121.18	144.18	129.14	0.19	0.07
100 * 25 * 15	135.42	158.15	141.68	0.17	0.05
200 * 20 * 20	128.44	155.97	135.15	0.21	0.05
200 * 30 * 20	145.11	177.15	151.14	0.22	0.04
200 * 50 * 20	158.75	198.11	164.19	0.25	0.03
Total				3	0.82
Average				0.2	0.05

Table 8
Comparison of needed solving time with GA and ILP algorithm.

Problem size (N.M.K)	Time (min)			Percent of deviation from the optimal solution	
	ILP	GA	GA/DM	$\frac{GA-ILP}{ILP}$	$\frac{(GA/DM-ILP)}{ILP}$
15 * 5 * 3	15.17	14.15	8.25	−0.07	−0.46
15 * 8 * 3	25.25	22.18	15.11	−0.12	−0.4
15 * 10 * 3	56.24	48.11	34.27	−0.14	−0.39
30 * 8 * 5	75.14	62.15	48.92	−0.17	−0.35
30 * 10 * 5	90.17	81.12	65.14	−0.1	−0.28
30 * 15 * 5	145.28	122.17	95.15	−0.16	−0.35
50 * 10 * 10	150.25	125.17	96.92	−0.17	−0.35
50 * 15 * 10	214.48	182.14	112.27	−0.15	−0.48
50 * 20 * 10	260.17	215.24	125.14	−0.17	−0.52
100 * 15 * 15	270.15	220.12	141.15	−0.19	−0.48
100 * 20 * 15	374.11	310.17	172.75	−0.17	−0.54
100 * 25 * 15	480.12	390.25	180.15	−0.19	−0.62
200 * 20 * 20	515.11	475.18	192.17	−0.08	−0.63
200 * 30 * 20	690.17	580.29	212.48	−0.16	−0.69
200 * 50 * 20	824.29	715.11	243.27	−0.13	−0.7
Total				−2.17	−7.01
Average				−0.14	−0.48

References

Artiba, A., & Elmaghrabhy, S. E. (1997). *The planning and scheduling of production systems: Methodologies and applications*. Springer. pp. 163–198.

Behnamian, J., Fatemi Ghomi, S. M. T., & Zandieh, M. (2009). A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic. *Expert Systems with Applications, 36*(8), 11057–11069.

Bertel, S., & Billaut, J. C. (2004). A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. *European Journal of Operational Research, 159*(3), 651–662.

Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constrains and time legs to minimize maximum lateness. *International Journal of Production Economics, 64*, 101–111.

Braha, S. A., & Loo, L. L. (1999). Heuristics for scheduling in a flow shop with multiple processors. *European Journal of Operational Research, 113*, 113–122.

Casillas, J., Francisco, J., & Martínez-López (2009). Mining uncertain data with multiobjective genetic fuzzy systems to be applied in consumer behavior modelling. *Expert Systems with Applications, 36*(2), 1645–1659.

Chang, P. C., Chen, S.-H., & Liu, C.-H. (2007). Sub-population genetic algorithm with mining gene structures for multiobjective flow shop scheduling problems. *Expert Systems with Applications, 33*(3), 762–771.

Chen, B. (1995). Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage. *Journal of Operation Research Society, 46*, 234–244.

Chen, C. H., Hong, T. P., & Vincent, S. T. (2009). An improved approach to find membership functions and multiple minimum supports in fuzzy data mining. *Expert Systems with Applications, 36*(6), 10016–10024.

Chen, J.-S., Pan, J. C.-H., & Lin, C.-M. (2008). A hybrid genetic algorithm for the re-entrant flow-shop scheduling problem. *Expert Systems with Applications, 34*(1), 570–577.

Cheng, B. W., & Chang, C.-L. (2007). A study on flow shop scheduling problem combining Taguchi experimental design and genetic algorithm. *Expert Systems with Applications, 32*(2), 415–421.

Cheng, R., Mitsuo, G., & Yasuhiro, T. (1999). A tutorial survey of job shop scheduling problems using genetic algorithms: Part II. Hybrid genetic search strategies. *Computers and Industrial Engineering, 37*(1-2), 51–55.

Deborah, R. C., & Freitas, A. A. (2004). A hybrid decision tree/genetic algorithm method for data mining. *Information Sciences, 163*(1–3), 13–35.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability. A guide to the theory of NP-completeness*. Oxford, England: W.H. Freeman and Co.

Gupta, J. ND. (1988). Two stage, hybrid flow shop scheduling problem. *Journal of Operation Research Society, 39*(4), 359–364.

Haouari, M. M., & Hallah, R. (1997). Heuristics algorithms for the two-stage hybrid flow shop problem. *Operation Research Letters, 21*, 43–53.

Hoogeveen, J. A., Lenestra, J. K., & Veltman, B. (1996). Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard. *European Journal of Operation Research, 89*, 172–175.

Janiak, A., & Lichtenstein, M. (2001). Comparison of some heuristic algorithms for the flow shop problem with parallel machines to minimize the total earliness, tardiness and waiting time. *Operations Research Proceedings 2000, Selected Papers of the Symposium on Operations Research (OR 2000), Dresden, September 9–12, 2000* (pp. 59–63). Berlin: Springer.

Karimi, N. M., Zandieh, R., & Kar-Amooz, (in press). Bi-objective group scheduling in hybrid flexible flowshop: A multi phase approach. *Expert Systems with Applications*.

Kyparisis, G. J., & Koulamas, C. (2001). A note on weighted completion time minimization in a flexible flow shop. *Operation Research Letters, 29*, 5–11.

Lin, T. L., Horng, S.-J., Kao, T.-W., Chen, Y.-H., Run, R.-S., & Chen, R.-J. (in press). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*.

Lin, H. T., & Liao, C. J. (2003). A case study in a two-stage Hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics, 86*(2), 133–143.

Roshanaei, V., Seyyed Esfehani, M. M., & Zandieh, M. (in press). Integrating non-preemptive open shops scheduling with sequence-dependent setup times using advanced metaheuristics. *Expert Systems with Applications*.

Shiau, D. F., Cheng, S.-C., & Huang, Y.-M. (2008). Proportionate flexible flow shop scheduling via a hybrid constructive genetic algorithm. *Expert Systems with Applications, 34*(2), 1133–1143.

Tian, P., Jian, Ma., & Zhang, D. M. (1999). Application of the simulated annealing algorithm to the combinatorial optimization problem with permutation property: An investigation of generation mechanism. *European Journal of Operational Research, 118*, 81–94.

Vob, S., & Witt, A. (2007). Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-word application. *International Journal of Production Economics, 105*(2), 445–458.

Yang, H. L., & Wang, C.-S. (2008). Two stages of case-based reasoning – Integrating genetic algorithm with data mining mechanism. *Expert Systems with Applications, 35*(1–2), 262–272.