J. Ind. Eng. Int., 7(13), 35-43, Spring 2011
ISSN: 1735-5702

A genetic algorithm approach for $P / ST_{si,b} / \sum w_j f_j$ problem

E. Mehdizadeh [1]*; R. Tavakkoli-Moghaddam [2]

[1] Assistant Prof., Dep. of Industrial Engineering, Islamic Azad University, Qazvin Branch, Qazvin, Iran

[2] Professor, Dep. of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran

Received: 13 February 2009; Revised: 15 December 2009; Accepted: 13 April 2010

Abstract: In this paper, a genetic algorithm is presented for an identical parallel-machine scheduling problem with family setup time that minimizes the total weighted flow time ($P / ST_{si,b} / \sum w_j f_j$). No set-up is necessary between jobs belonging to the same family. A set-up must be scheduled when switching from the processing of family *i* jobs to those of another family *j*, $i \neq j$, the duration of this set-up being the sequence-independent set-up time *sj* for family *j*. This problem is shown to be NP-hard in the strong sense and obtaining an optimal solution for the large-sized problems in reasonable computational time is extremely difficult. Further, it is computationally evaluated the performance of the proposed genetic algorithm solutions obtained using a mixed integer programming (MIP) with the Lingo 8.0 software.

Keywords: *Genetic algorithm; Parallel machine scheduling; Setup time; weighted flow time*

1. Introduction

A machine scheduling problem is an extended field of research in various applications. The main elements of machine scheduling problems are machine configuration, job characteristics, and objective function. The machine configuration can be classified into single and multiple machine problems in a broad sense. Parallel-machine scheduling problems can be referred as a class of problems that relaxed from the multiple machine scheduling problems (Ahn and Hyun, 1990). A batch setup time (cost) occurs when jobs, e.g., machine parts, are processed in batches (pallets, containers, boxes) and a setup of a certain time or cost precedes the processing of each batch. The batch setup time (cost) can be sequence dependent or sequence-independent. It is sequence-dependent if its duration (cost) depends on the families of both the current and the immediately preceding batches, and is sequence-independent if its duration (cost) depends solely on the family of the current batch to be processed.

Three-field notation α/β/γ is adapted of Allahverdi *et al.* (2008) to describe a scheduling problem. The *α* field describes the shop (machine) environment. The *β* field describes the setup information, other shop conditions, and details of the processing characteristics, which may contain multiple entries. Finally, the *γ* field contains the objective to be minimized.

A parallel machine-scheduling problem studied with identical parallel machines, jobs arranged into families, and sequence-independent set-up time between jobs of different families on these machines i.e. $P / ST_{si,b} / \sum w_j f_j$.

Many researchers studied parallel-machine scheduling problems in the past. However, research on family scheduling models is relatively new to the literature. Allahverdi *et al.* (2008) conducted a comprehensive review of setup-time (or time) research for scheduling problems classifying into batch, non-batch, sequence-independent, and sequence-dependent setup. Brono *et al.* (2003) proved that even a two-machine system for finding the weighted sum of flow time with an unequally weighted set of jobs is NP-hardness. Blazewicz and Kovalyov (2002) proved the strong NP-hardness of the problem $P / ST_{si,b} / \sum C_j$ under the group technology assumption, and presented a polynomial-time dynamic programming algorithm for the special case with a given number of the machines.

Webster and Azizoglu (2001) and Azizoglu and Webster (2003) addressed the same problem with a weighted objective function, i.e., $P / ST_{si,b} / \sum w_j f_j$, or equivalently $P / ST_{si,b} / \sum w_j C_j$.

Two dynamic programming algorithms (a backward and a forward) were proposed by Webster and Azizoglu (2001), where they also identified the characteristics of the problems for which each algorithm is suitable. When the number of machines and families is fixed, the backward dynamic algorithm is polynomial in

*Corresponding Author Email: emehdi@qiau.ac.ir
Tel.: +98 2813670051

sum of the weights while the forward dynamic algorithm is polynomial in the sum of processing and setup time. Azizoglu and Webster (2003) presented several branch-and-bound algorithms for the problem and computationally evaluated the performance of each algorithm. They concluded that the algorithms can quickly generate optimal solutions for problems with up to 15 to 25 jobs, depending on the number of machines. Chen and Powell (2003) proposed column generation based branch-and-bound algorithms for the same problem, where they obtained optimal solutions for problems up to 40 jobs, 4 machines and 6 families. Dunstall and Wirth (2005a) presented another branch-and-bound algorithm for the same problem and they showed that their algorithm outperforms that of Azizoglu and Webster (2003). They solved problems with up to 25 jobs and 8 families using their branch-and-bound algorithm. Dunstall and Wirth (2005b) proposed several simple heuristics for the same problem. Clearly, the branch-and-bound algorithms of Chen and Powell (2003) and of Dunstall and Wirth (2005b) remain to be compared.

Zhu and Heady (2000) introduced a mixed integer programming formulation to minimize the earliness and tardiness of all jobs for a scheduling problem with a non-uniform parallel machine and setup consideration. Omar and Teo (2006) studied on minimizing the sum of earliness/tardiness in the presence of setups. They developed a mixed-integer programming formulation model to deal with such a scheduling problem. Biskup and Cheng (1999) focused on two objectives, namely small deviations from a common due date and short flow time. Therefore, they proposed a model to minimize the earliness, tardiness, and completion time penalties.

Balakrishnan *et al.* (1999) presented a compact mathematical model and described computational experience by using their model to solve small-sized problems. Dunstall *et al.* (2000) proposed a B/B algorithm for lower bounds for the problem of minimizing the weighted flow time on a single machine with family set-up time and static job availability. Thus, as the case of many NP-hard problems, research on efficient heuristics capable of high quality solutions is warranted. Meta-heuristic methods can be developed to solve such hard problems.

Caoa *et al.* (2005) developed a heuristic algorithm to obtain the near-optimal solutions based on a tabu search (TS) mechanism on parallel machines scheduling problems by minimizing the sum of machine holding costs and job tardiness costs. Ramachandra and Elmaghraby (2006) proposed a GA procedure to solve a two machines scheduling problem to minimize the weighted completion time.

Monch *et al.* (2005) attempted to minimize the total weighted tardiness on parallel batch machines with incompatible job families and unequal ready times of jobs. They proposed two approaches and applied genetic algorithm (GA) in both approaches. Li and Cheng (1993) considered the job scheduling problem in identical parallel-machine systems with an objective of minimizing the maximum weighted absolute lateness further, Cheng and Gen (1997) have applied genetic algorithms to the above problem. Kim *et al.* (2002) proposed a simulated annealing (SA) method for unrelated parallel-machine scheduling problems with setup times. Melve and Uzsoy (2007) also proposed a genetic algorithm to a problem of minimizing the maximum lateness on parallel, identical batch-processing machines with dynamic job arrivals, based on random keys encoding. Tavakkoli-Moghaddam and Mehdizadeh (2007) presented a novel, integer-linear programming (ILP) model for an identical parallel-machine scheduling problem with family setup times that minimizes the total weighted flow time (TWFT). A meta-heuristic based genetic algorithm is proposed and applied to the given problem, which obtains good and near-optimal solution, especially for large sizes.

The purpose of this paper is to extend a genetic algorithm to schedule job families on parallel machines by minimizing the total weighted flow time, in which the weight of a job is the cost rate for delaying its completion. Job families reflect the efficiencies associated with processing similar jobs together. The setup may reflect the need to change a tool or clean the machine. A machine must be set up when switching from one family to others. There is no setup time between two jobs from the same family.

The rest of this paper is given below. In Section 2, details of the given problem and the optimization model are presented. Section 3 presents a description and design of the proposed genetic algorithm. In Section 4, various test problems are presented and solved by the proposed genetic algorithm. Future research in this area and conclusions are presented in Section 5.

2. Problem formulation

The objective of the problem is to schedule identical parallel machines by minimizing the total weighted flow time. All jobs are available at time zero with known integer-processing times, setup time, and weights. Each job is related to a family, in which a setup time is required between two jobs from different families, and the family setup time is independent of the preceding family. A setup is also required prior to the processing of the first job on a machine. This is typical of environment when scheduling at the beginning of a new shift after machine down time. For a given schedule, the weighted flow time of a particular job is the product of its weight and job completion time, and the total weighted flow time of a schedule is the sum of weighted flow time over all jobs.

2.1. Definition of parameters

The following Parameters are used in the proposed model.

i, j	Job indices, where job 0 is a dummy job that is always at the first position on a machine (i, j = 0, 1, …, n).
k	Machine index (k=1, 2, …, m).
f, g	Family indices.
n	Number of jobs.
m	Number of identical parallel machines.
o	Number of families, ($o \leq n$).
M	Large positive number.
P_{if}	Processing time of job i from family of (f = 1, 2, …, o).
S_f	Setup time of family f.
W_{if}	Weight of job i from family f.
γ_{ifjg}	Equals to 1, if $f \neq g$; and equals to 0, otherwise.

2.2. Definition of decision variable

C_{if}	Completion time of job i from family f.
Y_{ifk}	Equals to 1, if job i from family f is assigned to machine k; and equals to 0, otherwise.
X_{ifjgk}	Equals to 1, if job j from family g immediately follows job i from family f on machine k; and equals to 0, otherwise.
X_{0ifk}	Equals to 1, if job i from family f on machine k is the first in the queue; and equals to 0, otherwise.

2.3. Proposed model

The proposed mathematical model is as follows:

$$Min\,TWFT = \sum_{i=1}^{n} C_{if} W_{if} \qquad (1)$$

Subject to:

$$\sum_{k=1}^{m} Y_{ifk} = 1 \qquad \forall i, f \qquad (2)$$

$$C_{if} \geq S_f Y_{ifk} + P_{if} Y_{ifk} \qquad \forall i, f, k \qquad (3)$$

$$C_{jg} \geq C_{if} + P_{jg} + S_{jg}\gamma_{ifjg} - M(1 - X_{ifjgk}), \quad \forall i, j, f, m, g, k \; ; \; i \neq j \; ; \; f \neq g \qquad (4)$$

$$\sum_{\substack{i=0 \\ j \neq i}}^{n} \sum_{k=1}^{m} Y_{ifjgk} = 1 \qquad \forall f, j, g \qquad (5)$$

$$\sum_{\substack{i=0 \\ j \neq i}}^{n} X_{ifjgk} = Y_{jgk} \qquad \forall f, j, g, k \qquad (6)$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} X_{ifjgk} \leq Y_{ifk} \qquad \forall i, f, g, k \qquad (7)$$

$$\sum_{i=1}^{n} X_{0ifk} \leq 1 \qquad \forall f, k \qquad (8)$$

$$Y_{ifk}, X_{ifjgk}, X_{0ifk} = 0,1;\; C_i \geq 0 \;\; \forall i, j, f, g, k \qquad (9)$$

Equation (1) represents the objective function minimizing the total weighted flow time. Equation (2) states each job from each family must be assigned to exactly one machine. Equation (3) ensures that completion time of a job from a family must be later or equal to its processing time and setup time.

Equation (4) guarantees that the completion time of a job must be later or equal to the completion time of its direct predecessor job in the sequence, and its processing and setup time (if setup is necessary). This constraint becomes redundant if jobs i and j are assigned to different machines. Equation (5) ensures that a job must be processed at one and only one position on a machine. Equation (6) states that job j should immediately follow other job on machine k if it is placed on this machine. Equation (7) states that if job i, $i \neq 0$, is processed on machine k, it will be immediately followed by at most one another job on this machine.

Equation (8) enforces that only at most one job immediately follows the dummy job 0 on each machine. Equation (9) states the properties of the decision variables.

3. The proposed genetic algorithm

3.1. Structure of genetic algorithm

Genetic Algorithm (GA) was first introduced by John Holland in the 1970s. It is a search technique based on the concept of the natural selection and evolution. GA is a stochastic search technique based on the mechanism of the natural selection and natural genetics. Genetic algorithm, differing from conventional search techniques, it starts with an initial set of random solutions called a population.

Each individual in the population is referred to a chromosome, representing a solution to the problem at hand. A chromosome is a string of symbols. Chromosomes evolve through successive iterations, namely generations. During each generation, chromosomes are evaluated by using some measures of fitness.

To create the next generation, new chromosomes, referred to offspring, are formed by either (1) merging two chromosomes from the current generation by using a crossover operator, or (2) modifying a chromosome by using a mutation operator. A new generation is formed by (1) selecting, according to the fitness value, some of parents and offspring, and (2) rejecting others so as to keep the population size constant. After several generations, the algorithm converges to the best chromosome, which hopefully represents the optimal or sub-optimal solution to the given problem.

A genetic algorithm consists of four search operators, namely selection, crossover, mutation, and reproduction, to transform a population of chromosomes while improving their ‘‘quality’’. Genetic search operators are then applied one after another to systematically obtain a new generation of chromosomes with a better overall quality. This process is repeated until the stopping criterion is met, and the best solution of the last generation is reported as the final solution. To efficiently search the GA process and find the proper solution structure, it is necessary that the initial population of schedules be a diverse representative of the search space.

3.2. Application of GA to the given problem

Chen and Gen have applied genetic algorithms to the job scheduling problem in identical parallel machine system with the objective of minimizing the maximum weighted absolute lateness (Cheng and Gen, 1995). This problem was first considered by Li and Cheng (1993) as follows.

3.2.1. Chromosome representation

There are two essential issues to be dealt with all types of multiple machine scheduling problems:

- Partition of jobs to machines.
- Sequence jobs for each machine.

Also, each job (e.g., k) belongs to a family (e.g., j) as shown with (j, k). Suppose n indicates the number of jobs, therefore, an extended representation is proposed to encode partition of jobs to machines and sequence jobs for each machine into a chromosome with n columns and two rows. Where first row represents all possible permutation of (j, k) (or sequence of (j, k)) and second row designates the partition of (j, k) to machines. Let us consider a simple example with three jobs, two families, and two machines subject to k_1 and k_2 belong to j_1 , and k_3 belong to j_2. Suppose there is a schedule as shown in Figure 1.

The chromosome can be represented as follows:

In general, for an n-job, f-family, and m-machine problem, a legal chromosome contains two rows with n columns. There are n symbols of (j, k) at the first row, and m machines at the second row.

3.2.2. Generation of the initial population

Initial population is generated at random.

Machine 2	(j_1 , k_1)	
Machine 1	(j_2 , k_3)	(j_1 , k_2)

Figure 1: Schedule for three-jobs, two-families and two-machines.

(2 3)	(1 2)	(1 1)
1	1	2

Figure 2: Representation for three-jobs, two-families and two-machines.

3.2.3. Evaluation

A simple way to determine the fitness value for each chromosome is to use the inverse of the total weighted flow time. Let $TWFT_k$ denote the total weighted flow time for the k^{th} chromosome. The fitness value ($eval\ (v_k)$) is then calculated as follows:

$$eval(v_k) = \frac{1}{TWFT_k} \tag{10}$$

where,

$$TWFT_k = \sum_{j=i}^{n} WFT_j \tag{11}$$

WFT_j is the weighted flow time for the j^{th} job that is computed as follows:

WFT_j = (completion time of j^{th} job) ×(weight of j^{th} job)

3.2.4. Selection

The purpose of the parent selection in GA is to offer additional reproductive chances to those population members that are the fittest. One common technique used in the proposed GA is the roulette wheel selection. Here, it is used the modified roulette wheel as follows that use the number of generation for improvement of fitness quality. The chromosome with higher fitness has higher chance for selection.

1. Calculate the fitness value $eval(v_k)$ for each chromosome v_k (k=1, 2, …, pop_size).

2. Calculate the total fitness for the population $F = \sum_{k=1}^{pop-size} eval(v_k)$.

3. Calculate the selection probability P_k for each chromosome v_k:

$$p_k = \frac{eval(v_k)}{F} \quad , (k=1, 2, \ldots, pop\_size)$$

4. Calculate $z_k = p_k^{(R)^{Generation}}$ and R is a real number.

5. Calculate the cumulative probability q_k for each chromosome v_k:

$$q_k = \sum_{j=i}^{k} z_j \quad , (k=1, 2, \ldots, pop\_size)$$

6. Generate a random number r from the interval [0, 1]

7. If $r \le q_1$, then select the first chromosome v_k ; otherwise, select the k^{th} chromosome v_k ($2 \le k \le pop\_size$) such that $q_{k-1} \prec r \le q_k$.

3.2.5. Genetic operators order crossover

There are several types of crossover operators. In this study, order crossover (OX) operator is used. The procedure is illustrated in Figure 3. By the OX procedure, two off springs in per iteration can be produced but the proposed crossover takes two parents and creates a single offspring.

During past years, several mutation operators have been proposed. The reciprocal exchange mutation (swapping mutation) is used here, in which two random positions are selected and then their genes are swapped.

Parent1

(1 3)	(1 5)	(3 4)	(2 1)	(3 6)	(2 2)	(1 7)
1	1	1	2	2	2	2

Offspring

(3 6)	(1 5)	(3 4)	(2 1)	(1 3)	(2 2)	(1 7)
1	1	1	2	2	2	2

Parent2

(3 6)	(1 5)	(3 4)	(1 3)	(2 1)	(2 2)	(1 7)
1	1	1	2	2	2	2

Figure3: Illustration of OX operator.

Parent

(1 3)	(1 5)	(3 4)	(2 1)	(3 6)	(2 2)	(1 7)
1	1	1	2	2	3	3

Offspring

(3 4)	(1 5)	(1 3)	(2 1)	(3 6)	(2 2)	(1 7)
1	1	1	2	2	3	3

(a)

Parent

(1 3)	(1 5)	(3 4)	(2 1)	(3 6)	(2 2)	(1 7)
1	1	1	2	2	3	3

Offspring

(1 3)	(3 6)	(3 4)	(2 1)	(1 5)	(2 2)	(1 7)
1	1	1	2	2	3	3

(b)

Figure4: (a) Swap two jobs within one machine; (b) swap two jobs within different machine.

3.2.6. Swapping mutation

The randomly swap genes may be either job or job and machines. Different combinations of job and job and machines result in two basic types of mutation.

1. If the two selected jobs are processed by the same machine. In this case, the mutation alters the job order for the machine as shown in Figure 4(a).

2. Another case is that two jobs are processed by different machines. In this case, the mutation alters both job order and job partition to machines for the chromosome as shown in Figure 4(b).

3.2.7. Stopping condition

The GA is terminated after a pre-selected number of generations. A reasonable number can be arrived at with a few preliminary test runs. 150 generations are founded to be sufficient.

4. Computational results and performance evaluation

The proposed genetic algorithm (GA) can be used for more complex and widely applicable models of scheduling problems in industries.

The proposed algorithms are coded in Visual Basic 6 and run on a PC with Intel(R) Core(TM) 2Duo CPU 1.8GHz, 2GB of RAM.

In Table 1 we compare the performance of the proposed GA with the Lingo 8 software in terms of computational time. The proposed GA has a better solution than the Lingo software. Further, when the number of jobs increases, we can see that the computational time increases exponentially because of the NP-hard nature of the given problem. Other test instances have generated according to the same rules followed by Duntsall and Wirth (2000).

For each (G; N; M) jobs randomly assigned to families by first drawing a random number for each family and then distributing jobs approximately in proportion to these random numbers, subject to the constraint that total the number of jobs sums to N. Job processing time and weights are randomly sampled from the ranges [1; 100] and [1; 10] respectively, and setup time is randomly sampled from the range [0; 50]. 27 sample size and 5 instances have generated for each (G; N; M).

In line with the study by Duntsall and Wirth (2000), the genetic algorithm have tested over each combination of (G; N; M) from $G \in \{3,5,8\}$, $N \in \{10,20,40\}$ and $M \in \{2,3,5\}$.

Because of the stochastic nature of the proposed GA, five trials are performed for 150-generation each and the best solution amongst the five is considered as the final solution. The associated computational results are summarized in Table 1.

It can be seen the input and output of the designed software in Figures 5 and 6. Tables 2 and 3 show the convergence of the average and the best fitness in each generation. The near-optimal solution is achieved after approximately 50 generations. The results show that the GA consistently converged to the optimal solution. Since, GA is a stochastic search algorithm, one aspect of investigating the efficiency of the proposed GA is the sensitivity analysis to the GA operators used in this paper. Thus, 27 test problems are solved, which, are a modified versions of the problems given in Duntsall and Wirth (2005b). The data sets of these problems are shown in Table 4 and the last column of the table shows that all (STD/Average) ×100 of the problems are less than 2%. Thus, the proposed GA is a robust optimization algorithm. Figure 7 shows the (STD/Average) ×100 of the best solution to each problem.

Table 1: Comparison of the proposed GA and Lingo8.

Problem size (N × G× M)	Lingo		Old GA (Cheng and Gen, 1995)		New GA	
	OFV	Time (sec)	OFV	Time (sec)	OFV	Time (sec)
3×2×2	10	~0	10	~0	10	~0
5×2×2	46	0.1	46	<0.1	46	~0
7×3×3	147	12	147	5	147	0.014
10×3×2	96	>36000	96	32	96	0.155

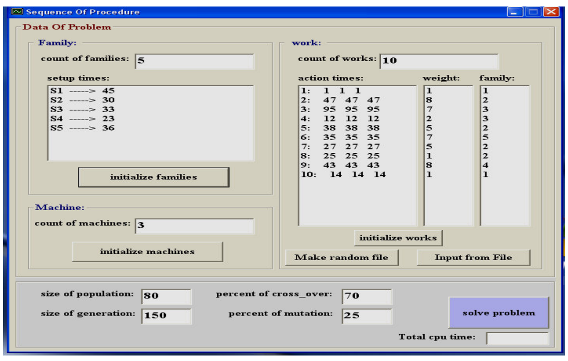

Figure 5: The input of designed software.

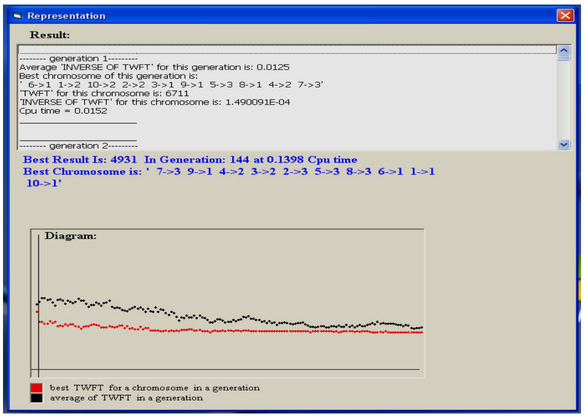

Figure 6: The output of the designed software.

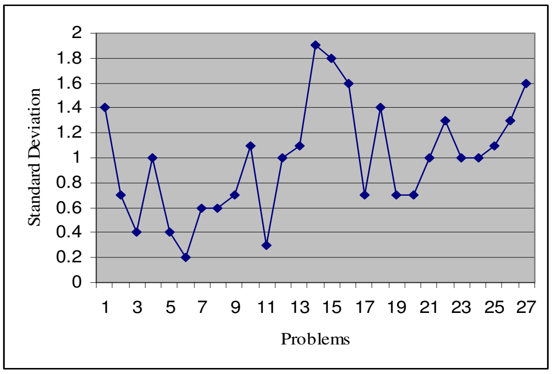

Figure7: Standard deviation of fitness function.

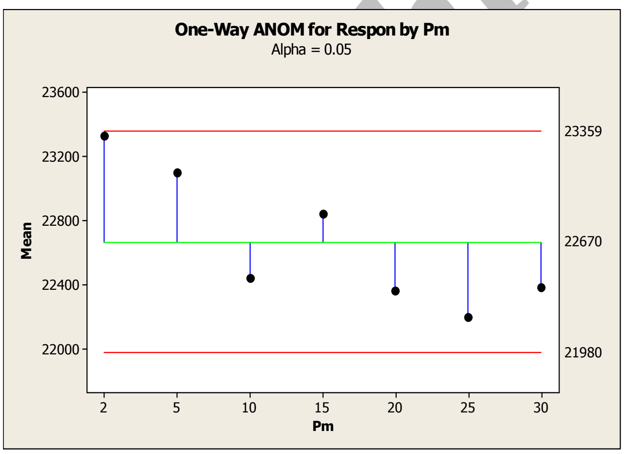

Figure 8: Tree Level ANOVA Results.

Table 2: General linear model: Respon versus Pc;Pm.

Factor	Type	Levels	Value
Pc	Fixed	3	25; 35; 45
Pm	Fixed	4	2; 5; 10; 15

Table 3: Analysis of variance for Respon, using adjusted SS for tests.

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Pc	2	1682772	1682772	841386	0.68	0.506
Pm	3	65543694	65543694	21847898	17.72	0.000
Pc*Pm	6	530733	530733	88456	0.07	0.999
Error	228	281094281	281094281	1232870		
Total	239	348851481				

4.1. Parameter tuning

As discussed above, the genetic search method is guided by the 'tuning' of two parameters, namely crossover rate (P_c), and mutation rate (P_m) which choosing a proper value for them affects the search behavior and improves the quality of convergence.

For choosing these parameters, a factorial design in the design of experiments (DOE) with three levels and the parameters ranges are showed next.

- Crossover rate: 25%, 35% and 45%.
- Mutation rate: 2%, 5%, 10% and 15%.

The results are given in figure 8 and they show that only mutation parameter have to be estimated. Therefore, another experiment designed with one levels and the mutation parameter ranges are considered as 2%, 5%, 10%, 15%, 20%, 25% and 30%. The results are given in figure 9 and 25% rate chosen for mutation rate.

5. Conclusion

The parallel-machine scheduling problem is an extended field of study in various applications. This type of the problem is one of classical machine scheduling problems. In this paper, a genetic algorithm presented for an identical parallel-machine scheduling problem with family setup time that minimizes the total weighted flow time i.e. $P/ST_{si,b}/\sum w_j f_j$). This problem is shown to be NP-hard in the strong sense and obtaining an optimal solution for the large-sized problems in reasonable computational time is extremely difficult. This is the motivation for using genetic algorithms (GAs). The proposed GA is more flexible in the sense that the practitioner is not limited to a single solution. Some properties and solution methods for a generalized model consisting of job due dates and penalties for completing both early and tardy jobs can be used in further research.

Table 4: GA performance for 27 problems.

Problem	N	M	G	Trial					Average	The Best	STD	(STD/Average) ×100
				1	2	3	4	5				
1	10	2	3	4959	5017	4959	4959	5124	5004	4959	72	1.4
2	10	2	5	7823	7707	7823	7838	7825	7803	7707	54	0.7
3	10	2	8	7800	7751	7740	7710	7740	7748	7710	33	0.4
4	10	3	3	5339	5331	5229	5262	5229	5278	5229	54	1.0
5	10	3	5	4705	4744	4705	4704	4724	4716	4704	18	0.4
6	10	3	8	6919	6925	6941	6958	6923	6933	6919	16	0.2
7	10	5	3	7466	7538	7466	7540	7453	7493	7453	43	0.6
8	10	5	5	3176	3176	3182	3208	3220	3192	3176	20	0.6
9	10	5	8	6145	6148	6225	6145	6224	6177	6145	43	0.7
10	20	2	3	26444	26446	26794	26015	26208	26381	26015	293	1.1
11	20	2	5	21807	21931	21778	21819	21779	21823	21778	63	0.3
12	20	2	8	18509	18108	18090	18102	18243	18210	18090	178	1.0
13	20	3	3	15583	15864	15503	15917	15756	15725	15503	178	1.1
14	20	3	5	19992	20264	19830	20789	19967	20168	19830	381	1.9
15	20	3	8	19556	20081	19417	20276	19658	19798	19417	365	1.8
16	20	5	3	13147	12839	13096	13236	13410	13146	12839	209	1.6
17	20	5	5	13324	13170	13250	13140	13099	13197	13099	90	0.7
18	20	5	8	10326	10472	10312	10684	10420	10443	10312	150	1.4
19	40	2	3	77985	77665	77373	76706	76901	77326	76706	528	0.7
20	40	2	5	80430	80363	80208	81545	81019	80713	80208	558	0.7
21	40	2	8	81985	81786	82122	81036	80195	81425	80195	805	1.0
22	40	3	3	68963	69256	69019	68161	70660	69212	68161	909	1.3
23	40	3	5	75363	75182	76900	76543	76115	76021	75182	740	1.0
24	40	3	8	71912	72058	71473	70637	72632	71742	70637	744	1.0
25	40	5	3	40454	41337	40185	41056	40558	40718	40185	468	1.1
26	40	5	5	34489	34988	35214	35062	34179	34786	34179	435	1.3
27	40	5	8	40807	39474	39409	39273	39687	39730	39273	620	1.6

References

Ahn, B. H.; Hyun, J. H., (1990), Single facility multi-class job scheduling. *Computers and Operations Research*, 17(3), 265-72.

Allahverdi, A.; Cheng, T. C. E.; Kovalyov, Y. M., (2008), A survey of scheduling problems with setup times or costs. *European Journal of Operational Research,* 187(3), 985-1032.

Azizoglu, M.; Webster, S., (2003), Scheduling parallel machines to minimize weighted flowtime with family set-up times. *International Journal of Production Research*, 41(6), 1199-1215.

Balakrishnan, N.; Kanet, J.; Sridharan, S. V., (1999), Early/tardy scheduling with sequence dependent setups on uniform parallel machines. *Computer and Operations Research*, 26(3),127-141.

Biskup, D.; Cheng, T. C. E., (1999), Multiple-machine scheduling with earliness, tardiness and completion time penalties. *Computer and Operations Research*, 26(3),45-57.

Blazewicz, J.; Kovalyov, M. Y., (2002), The complexity of two group scheduling problems. *Journal of Scheduling*, 5(6), 477-485.

Caoa, D.; Chen, M.; Wan, G., (2005), Parallel machine selection and job scheduling to minimize machine cost and job tardiness. *Computers and Operations Research*, 32(8), 1995-2012.

Chen, Z. L.; Powell, W. B., (2003), Exact algorithms for scheduling multiple families of jobs on parallel machines. *Naval Research Logistics*, 50(7), 823-840.

Cheng, R.; Gen, M., (1995), Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithm. *Computer and Industrial Engineering*, 29(1-4), 513-517.

Dunstall, S.; Wirth, A., (2005a), A comparison of branch-andbound algorithms for a family scheduling problem with identical parallel machines. *European Journal of Operational Research*, 167(2), 283-296.

Dunstall, S.; Wirth, A., (2005b), Heuristic methods for the identical parallel machine flowtime problem with set-up times. Computers and Operations Research, 32(9), 2479-2491.

Dunstall, S.; Wirth, A.; Baker, K., (2000), Lower bounds and algorithms for flow time minimization on a single machine with set-up times. *Journal of Scheduling*, 3(2), 51-69.

Gen, M.; Cheng, R., (1997), *Genetic algorithms and engineering design*. John Wiley & Sons, New York.

Kim, D. W.; Kim, K. H.; Wooseung- Jang, F.; Chen, F., (2002), Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer Integrated Manufacturing*, 18(3-4), 223-231.

Li, C.; Cheng, T., (1993), The parallel machine min-max weighted absolute lateness scheduling problem. *Naval Research Logistics*, 41(1), 33-46.

Melve, S.; Uzsoy, R., (2007), A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. *Computers and Operations Research*, 34(10), 3016-3028.

Monch, L.; Balasubramanian, H.; Fowler, W. J.; Pfund, E. M., (2005), Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. *Computers and Operations Research*, 32(11), 2731-2750.

Omar, M. K.; Teo, S. C., (2006), Minimizing the sum of earliness/tardiness in identical parallel machines schedule with incompatible job families: An improved MIP approach. *Applied Mathematics and Computation*, 181(2), 1008-1017.

Ramachandra, G.; Elmaghraby, S. E., (2006), Sequencing precedence-related jobs on two machines to minimize the weighted completion time. *International Journal of Production Economics*, 100(1), 44-58.

Tavakkoli-Moghaddam, R.; Mehdizadeh, E., (2007), A new ILP model for identical parallel machine scheduling with family setup times minimizing the total weighted flow time by a genetic algorithm. *International Journal of Engineering*, 20(2), 183-194.

Webster, S.; Azizoglu, M., (2001), Dynamic programming algorithms for scheduling parallel machines with family setup times. *Computers and Operations Research*, 28(4), 127-137.

Zhu, Z.; Heady, R. B., (2000), Minimizing the sum of earliness / tardiness in multi machine scheduling with sequence dependant setups on uniform parallel machines. *Computer and Industrial Engineering*, 38(1), 297-305.