A NEW ILP MODEL FOR IDENTICAL PARALLEL-MACHINE SCHEDULING WITH FAMILY SETUP TIMES MINIMIZING THE TOTAL WEIGHTED FLOW TIME BY A GENETIC ALGORITHM

R. Tavakkoli-Moghaddam*

Department of Industrial Engineering, Faculty of Engineering
University of Tehran, P. O. Box 11365/4563, Tehran, Iran
tavakoli@ut.ac.ir

E. Mehdizadeh

Department of Industrial Engineering, Qazvin Branch
Islamic Azad University, Qazvin, Iran
mehdizadeh@qazviniau.ac.ir

*Corresponding Author

(Received: November 21, 2006 – Accepted in Revised Form: May 31, 2007)

Abstract This paper presents 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). Some researchers have addressed parallel-machine scheduling problems in the literature over the last three decades. However, the existing studies have been limited to the research of independent jobs, and most classical optimization methods are focused on parallel-machine scheduling problems without considering setup times and relationship between jobs. This problem is shown to be NP-hard one in the strong sense. Obtaining an optimal solution for this type of complex, large-sized problems in reasonable computational time is extremely difficult. A meta-heuristic method, based on genetic algorithms, is thus proposed and applied to the given problem in order to obtain a good and near-optimal solution, especially for large sizes. Further, the efficiency of the proposed algorithm, based on various test problems, is compared with the Lingo 8.0 software.

Keywords Parallel Machine Scheduling, ILP Model, Family Setup Times, Total Weighted Flow Time, Genetic Algorithm

چکیده در این مقاله یک مدل برنامه‌ریزی مختلط (عدد صحیح – خطی) جدید برای مساله زمانبندی ماشین‌های موازی یکسان با در نظر گرفتن زمان‌های آماده‌سازی خانواده کارها به منظور حداقل نمودن کل زمان جریان ساخت وزنی می‌گردد. در سال‌های اخیر مسائل زمانبندی ماشین‌های موازی توسط محققین زیادی مورد مطالعه قرار گرفته است، اما تحقیقات موجود محدود به کارهای مستقل بوده و اغلب روش‌های بهینه‌سازی متداول روی مسائل ماشین‌های موازی بدون در نظرگرفتن زمان‌های آماده‌سازی و ارتباط بین کارها متمرکز می‌باشند. بدست آوردن جواب بهینه برای حل این نوع مسائل پیچیده و با مقیاس بزرگ در زمان محاسباتی معقول بسیار مشکل می‌باشد. بنابراین برای حل مسئله مورد نظر، ارائه یک الگوریتم فرا ابتکاری بر اساس الگوریتم ژنتیک برای دستیابی به جواب مناسب و نزدیک بهینه، مخصوصاً برای مسائل با اندازه بزرگ، پیشنهاد می‌گردد. بعلاوه، کارایی الگوریتم پیشنهادی بر اساس چند مسئله نمونه با نرم افزار لینگو نسخه ۸ مورد مقایسه قرار می‌گیرد.

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 to 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 [1]. In an identical parallel machine system, all machines are identical and a job can be processed by any free machine. Three

issues are dealt with this scheduling as follows:

- What machine to be allocated to which jobs?
- How to order the jobs in an appropriate processing sequence?
- How to rationalize the reasonableness of the schedule?

The main criteria measures can also be classified to three distinct groups as follows [2]:

1. Completion-time-based measures.
2. Due-date-based measures.
3. Idleness-penalty-based measures.

1.1. Minimum Weighted Flow Time Problem The minimum weighted flow time problem belongs to the first above-mentioned group.

Let W_j denotes the weight associated with job j. This problem finds an optimal non-preemptive job schedule by minimizing the weighted flow time as follows:

$$\underset{s}{Min}\ f(s) = \sum_{j=i}^{n} w_j c_j$$

s refers to a given schedule and $f(s)$ is the corresponding objective function value.

General assumptions are listed below [3]:

- Each job can be processed only by one operation.
- No job can be processed on more than one machine simultaneously.
- Any machine can process any job.
- No machine may process more than one job at a time.
- Machines never break down and are available during the scheduling period.
- Job processing times are independent of the scheduling.
- Number of jobs is fixed.
- Number of machines is fixed.
- Processing time of jobs on machine is given and fixed.

In the other hands, scheduling problems in practice often require a trade-off between the system efficiency and timely completion of individual orders. A major reason for this is that there are typically efficiencies associated with processing similar parts together. This exerts pressure for scheduling long runs of similar jobs at the expense of some delays in other jobs.

1.2. Literature Review Many researchers studied parallel-machine scheduling problems in past. Cheng and Sin [4] surveyed a parallel-machine scheduling problem and Allahverdi et al. [5] investigated a comprehensive review of setup-time (or time) research for scheduling problems classifying into batch, non-batch, sequence-independent, and sequence-dependent setup. Potts and Kovalyov [6] reviewed the literature on family scheduling models with single-machine, shop problems, and parallel machine. Research on family scheduling models is relatively new in the literature, and most of this work has been focused on single machine models [7]. Brono et al. [8] proved that even a two-machine system for finding the weighted sum of flow times with an unequally weighted set of jobs is NP-hardness. This is true even when the problem is simplified by assuming no family setup times [9], or assigning weight 1 for each job [10].

Ahn and Hyun [1], Bruno and Sethi [11], Mason and Anderson [12], and Monma and Potts [13] proposed some algorithms to minimize the total weighted flow time on a single machine with family setup times. Mason and Anderson [12] described a branch-and-bound (B/B) method and a depth-first search algorithm. Ramachandra and Elmaghraby [14] proposed a binary integer program (BIP) and a dynamic program (DP) on two machines to minimize the weighted completion time. They also introduced a genetic algorithm (GA) procedure to solve the above-mentioned problems. Other studies have been applied dynamic programming (DP) algorithms to the problem. Monma and Potts [13], while not explicitly defining an algorithm, identified the approach for extending their single-machine DP algorithm to accommodate parallel machines.

Webster and Azizoglu [7] described the problem of scheduling jobs with family setup times on identical-parallel machines to minimize the total weighted flow time. They presented two dynamic programming algorithms to solve the given

problem and identified characteristics of problems, in which each algorithm is best suited. However, they did not develop an algorithm to be directly implemented in practice. They believed that most real-world problems are much more complicated than the model under consideration, and hoped to take a step towards effective and practical solutions methods for more complex and more widely applicable models of scheduling problems in industries.

Nessah et al. [15] presented an identical parallel-machine scheduling problem with sequence-dependent setup times and release dates to minimize the total completion time. They proved a condition for local optimality as a priority rule, and defined a dominant subset based on this condition. They also proposed heuristic algorithms based on this condition to build a schedule belonging to a subset, and then developed the lower bound computed in a polynomial time. They constructed a branch-and-bound (B/B) algorithm in such way that the heuristic, lower bound, and dominance properties were incorporated. Monch et al. [16] 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. Leung et al. [17] analyzed efficient heuristics for the scheduling orders for multiple product types to minimize the total weighted completion time without release dates.

Zhu and Heady [18] 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 [19] 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 [20] focused on two objectives, namely small deviations from a common due date and short flow times. Therefore, they proposed a model to minimize the earliness, tardiness, and completion time penalties.

Caoa et al. [21] developed a combinatorial optimization model and a heuristic algorithm to obtain the optimal or 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. Balakrishnan et al. [22] presented a compact mathematical model and described computational experience by using their model to solve small-sized problems. Dunstall et al. [23] 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 times 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. Park et al. [24] applied a neural network approach for scheduling jobs on parallel machines. Wardono and Fathi [2] developed a tabu search algorithm for the problem of scheduling N jobs on parallel machines in L successive stages with limited buffer capacities between stages. Ramachandra and Elmaghraby [14] proposed a genetic algorithm (GA) procedure to find a sequence of precedence-related jobs on two machines that minimizes the weighted completion time. Li and Cheng [25] considered the job scheduling problem in identical parallel-machine systems with an objective of minimizing the maximum weighted absolute lateness as follows: there are a set of jobs associated with known processing time s and weights, several parallel and identical machines, and a common due date that is not too early to constrain the scheduling decision. The objective is to find an optimal job schedule so as to minimize the maximum weighted absolute lateness. Further, Cheng and Gen [26] have applied genetic algorithms to the above problem. Kim et al. [27] proposed a simulated annealing (SA) method for unrelated parallel-machine scheduling problems with setup times.

Tavakkoli-Moghaddam et al. [28] presented a new mathematical model for a multi-criteria parallel-machine scheduling problem that minimizes the total earliness and tardiness penalties, and machine costs. A meta-heuristic method, based on genetic algorithms, is proposed and developed. Melve and Uzsoy [29] 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. Leonardi and Raz [30] proposed an approximate of

the total flow time on parallel machines. Kang and Ng [31] presented a fully polynomial-time approximation scheme for the parallel-machine scheduling with deteriorating jobs.

The purpose of this paper is to develop a combinatorial optimization model and describe a meta-heuristic algorithm in order 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 [7]. 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 bellow. 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, several small-sized instances are solved by a combinatorial model with the Lingo 8.0 software to demonstrate the nature of the problem and the features of the model. In Section 5, various test problems are presented and solved by the proposed genetic algorithm. Computational results obtained by the proposed algorithm show its effectiveness of finding optimal or near-optimal solutions, especially for larger-sized problems. Future research in this area and conclusions are presented in Section 6.

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 times, and weights. Each job is related to a family, in which a setup time is required between two jobs from deferent 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 Following Parameters are used in the proposed model:

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

2.2. Definition of Decision Variable

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

2.3. Proposed Model The proposed mathematical model is as follows:

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

s.t.

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

$$C_{if} \geq S_f Y_{ifk} + P_{if} Y_{ifk} \quad \forall i, f, k \tag{3}$$

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

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

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

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

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

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

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. The usual form of GA was described by Goldberg [32]. 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 [32].

Usually, initialization is assumed to be random. There are only two types of operators in genetic algorithms:

- Genetic operations: crossover, mutation
- Evolution operations: selection

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. The structure of GA is illustrated in Figure 1.

3.2. Application of GA to the Given Problem

3.2.1. Chromosome representation There are two essential issues to be dealt with all types of multiple machine scheduling problems [32]:

- 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). An extended permutation representation is proposed to encode these things into a chromosome. Where (j, k) represent all possible permutation of (j, k) (or sequence of (j, k)) and asterisks * designate the partition of (j, k) to machines. Each * in a chromosome is a *gene* of it. 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 shown in Figure 2.

The chromosome can be represented as follows:

$[(j_2, k_3)\ (j_1, k_2)\ *\ (j_1, k_1)]$

In general, for an n-job, f-family and m-machine problem, a legal chromosome contains n symbols of (j, k) and m-1 partitioning symbols resulting in the total size of ($n + m$-1).

3.2.2. Generation of the initial population Initial population is randomly generated.

3.2.3. Evaluation a simple way to determine the fitness value for each chromosome is to use the inverse of 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:

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

Where,

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

WFT_j is 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. The roulette wheel can be constructed as follows:

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

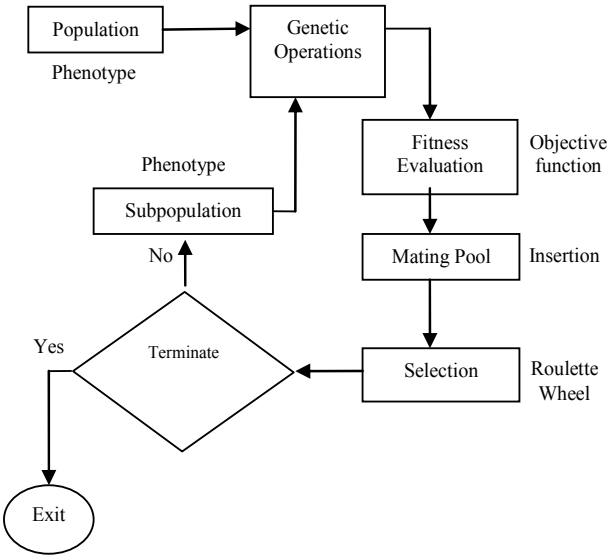

Figure 1. Structure of the proposed GA.

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

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

$$F = \sum_{k=1}^{pop-size} eval(v_k))$$

3. Calculate selection probability P_k for each chromosome v_k: $p_k = \frac{eval(v_k)}{F}$, ($k$ = 1, 2, …, pop_size)

4. Calculate cumulative probability q_k for each chromosome v_k: $q_k = \sum_{j=i}^{k} p_j$, (k = 1,2,…,pop_size)

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

6. 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. The genetic operators Crossover is the main genetic operator. It generates offspring by combining both chromosomes' features. To a great extend, the performance of genetic algorithms depends on the type of the crossover operator used. The crossover rate is defined as the ratio of the number of offspring produced in each generation to the population size [32]. There are several types of crossover operators. In this study, we use the order crossover (OX) operator.

Order Crossover (OX) The OX works as follows:

- Select a substring from one parent at random.
- Produce a proto-child by copying the substring into the corresponding positions.
- Delete the jobs which are already in the substring from the second parent. The resulted sequence of jobs contains the jobs that the proto-child needs.
- Place the jobs into the unfixed positions of the proto-child from left to right according to the order of the sequence to produce an offspring.

The procedure is illustrated in Figure 3. By the OX procedure, we can produce two offspring in per iteration but the proposed crossover takes two parents and creates a single offspring.

Mutation is a background operator which produces spontaneous random changes in various chromosomes. The mutation rate is defined as the percentage of the total number of gene in the population. During past years several mutation operators have been proposed such as inversion, insertion, displacement, reciprocal exchange mutation [32]. The reciprocal exchange mutation (swapping mutation) is used here, in which we select two random positions and then swap their genes.

Swapping Mutation The randomly swapped genes may be either job or asterisk. The different combinations of job and asterisk result in four basic types of mutation.

1. If both genes are job, two cases may occur: One case is that 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. If both genes are asterisk, the mutation performs a trivial operation as shown in Figure 4(c).
4. If one gene is asterisk and another is job, the mutation alters both job order and job partition to machines for the chromosome as shown in Figure 4(d).

The last one is the only genetic operation, which can alter the position of asterisks.

4. ILLUSTRATED EXAMPLES

To efficiently perform the proposed GA, two small-sized test problems are solved by using the Lingo 8.0 optimization software compared with the proposed GA.

Parent1

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

Offspring

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

Parent2

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

Figure 3. Illustration of OX operator.

Parent

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

Offspring

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

(a)

Parent

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

Offspring

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

(b)

Parent

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

Offspring

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

(c)

Parent

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

Offspring

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

(d)

Figure 4. (a) Swap two jobs within on machine; (b) Swap two jobs within different machine; (c) Trivial swap; (d) Swap the position of a job and an asterisk.

Example 4.1. First, we consider a simple example of a three-job, two-family, and two-machine problem given in Webster and Azizoglu [7]. The primary information is summarized in Table 1.

The globally optimal solution is found at iteration 514 with the best objective function 10, in which the associated sequence is given bellow:

M1: (1 2)
M2: (2 3) → (1 1)

Example 4.2. For the second example, we consider an identical parallel-machine scheduling problem with seven jobs, three families, and three machines. The other associated information is given in Table 2.

The globally optimal solution is found at iteration 165723with the best objective function 147, and the associated Gantt chart for the best one is depicted in Figure 5. A comparison of these two examples also reveals the complexity of the problem.

5. COMPUTATIONAL RESULTS AND PERFORMANCE EVALUATION

Most real-world problems are more complicated than the previous examples. The proposed genetic algorithm can be used for more complex and widely applicable models of scheduling problems in industries.

The genetic algorithm is first used to solve the same two small-sized test problems as presented in Section 4. The related genetic solutions are compared with the optimal solutions obtained by the Lingo 8.0 software package. It is then used to solve one medium-sized problem to schedule 20 jobs from 10 families on five machines with various P_c and P_m.

Example 5.1. For example 4.1, the genetic parameters are set as follows: pop-size = 5, max-gen = 1, *Pc* = [0.3-0.8], and *Pm* = [0.1-0.6]. We run the proposed GA about 50 times and obtain an optimal schedule in any effort in first time. There are two distinct optimal schedules in the proposed GA (e.g., jobs (1,2) and (1,1) on one machine, and job (2,3) on the other machine, or alternatively, jobs (2,3) and (1,1) on one machine, and job (1,2) on the other machine). The associated computational results are presented in Table 3.

TABLE 1. Input Data.

JOB (I)	1	2	3
Processing time (P_{if})	3	1	1
Weight (W_{if})	1	2	1
Family (f)	1	1	2
Setup time for each family (S_f)	1	1	0

TABLE 2. Input Data for the 2th Example.

i	1	2	3	4	5	6
P_{if}	3	5	7	6	4	2
W_{if}	4	2	3	1	2	3
f	2	2	1	3	1	3
S_f	3	3	2	4	2	4

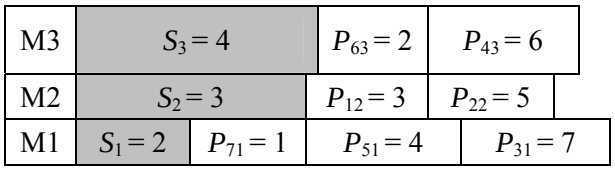

Figure 5. Gantt chart for best schedule for the 2nd example.

TABLE 3. Comparison of GA with DP and ILP Methods.

Method	Machine	Schedule	TWFT
DP	MI	(1 2)	10
	M2	(2 3) → (1 1)	
ILP	M1	(1 2)	10
	M2	(2 3) → (1 1)	
GA	M1	(1 2)	10
	M2	(2 3) → (1 1)	

Example 5.2. For example 4.2, the genetic parameters are set as follows: pop_size = 20, max_gen = 150, Pc = 0.3, Pm = 0.1. We run the genetic algorithm 10 times. The computational results obtained are summarized in Table 4.

Example 5.3. Table 5 shows an identical parallel-machine problem with 20 jobs, 10 families, 5 machines and other information generated at random.

In this section, to compare various P_c and P_m, we choose the appropriate crossover rate and mutation rate for parallel machines with family setup times. Other parameters are tuned up as follows: pop_size = 10, max_gen = 20. We run the genetic algorithm 15 times. The associated computational results are summarized in Table 6.

As discussed above, the genetic search method is guided by the 'tuning' of three parameters, namely population size, crossover rate (P_c), and mutation rate (P_m). We choose these parameters empirically within the ranges as shown in Table 7. In Table 8 we compare the performance of the proposed GA with the Lingo 8 software in terms of computational times. The proposed GA has 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.

6. CONCLUSION

The parallel-machine scheduling problem is an extended field of study in various applications. This type of problem is one of classical machine scheduling problems. This problem with family setup times is considered in the parallel machines problem, and shown to be NP-hardness in strong sense. We presented a new integer-linear programming (ILP) model of the foregoing problem. Further, we proposed a genetic algorithm (GA) that minimizes the total weighted flow time of jobs on identical parallel machines with family setup times. 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. Result of Example with Seven Jobs

Total run	Best one	Worst one	Average
10	147	178	169.2

TABLE 5. Information for Example 5.3.

i	P_{if}	W_{if}	f	S_f	i	P_{if}	W_{if}	f	S_f
1	10	4	1	5	11	5	2	4	10
2	12	2	1	5	12	5	2	3	10
3	5	1	2	5	13	10	4	2	5
4	12	3	3	10	14	15	2	9	5
5	15	1	4	10	15	20	3	10	5
6	10	2	5	5	16	10	4	8	10
7	15	5	6	5	17	5	5	5	5
8	18	4	7	10	18	5	1	3	10
9	15	2	6	5	19	15	2	2	5
10	10	5	5	5	20	10	5	1	5

TABLE 6. The Comparison Various Mutation and Crossover Rates.

No. of Job	No. of Machine	No. of Family	P_c	P_m	TWFT of Best One	Average
20	5	10	0.2	0.1	2758	3087
20	5	10	0.3	0.1	2755	3158
20	5	10	0.5	0.1	2570	2960
20	5	10	0.7	0.1	2760	3397
20	5	10	.3	0.2	2491	3085
20	5	10	0.5	0.2	2608	3151
20	5	10	0.7	0.2	2671	3253
20	5	10	0.3	0.4	2659	3036
20	5	10	0.3	0.6	2498	3008
20	5	10	0.5	0.7	2813	3171

TABLE 7. Basic Setting for GA Parameters.

Parameters	Range
Pop-size	5 - 50
Max-gen	20 - 100
P_c	0.2 - 0.5
P_m	0.05 - 0.2

TABLE 8. Comparison of the Proposed GA and Lingo 8.

No. of Job	No. of Machine	No. of Family	Lingo		GA	
			OFV	Time (sec)	OFV	Time (sec)
3	2	2	10	< 1	10	< 1
5	2	2	46	1	46	< 1
7	3	3	147	40	147	5
10	3	3	77	4906	77	20
10	3	2	68	9614	68	32

7. REFERENCES

1. Ahn, B. H. and Hyun, J. H., "Single facility multi-class job scheduling", ***Computers and Operations Research***, Vol. 17, (1990), 265-72.
2. Wardono, B. and Fathi, Y., "A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities", ***European Journal of Operational Research***, Vol. 155, (2004), 380-401.
3. Bjorndal, M, Caprara, A., Cowling, P., Croce, P.

Lourence, H., Malucelli, F., Orman, A., Pisinger, D., Rego, C. and Salazar, J., "Some thoughts on combinatorial optimization", ***European Journal of Operational Research***, Vol. 83, (1990), 253-270.
4. Cheng, T. and Sin, C., "A state-of-the-art review of parallel-machine scheduling research", ***European Journal of Operational Research***, Vol. 47, (1990), 271-292.
5. Allahverdi, A., Ng, C. T., Cheng, T. C. E. and Kovalyov, M. Y., "A survey of scheduling problems with setup times or costs", ***European Journal of Operational Research***, to appear in, (2007), DOI: 10.1016/j.ejor., (2006).06.060.
6. Potts, C. N. and Kovalyov, M. Y., "Scheduling with batching: A review", ***European Journal of Operational Research***, Vol. 120, (2000), 228-249.
7. Webster, S. and Azizoglu, M., "Dynamic programming algorithms for scheduling parallel machines with family setup times", ***Computers and Operations Research***, Vol. 28, (2001), 127-137.
8. Bruno, L, Coofman, J. and Sethi, R., "Scheduling independent tasks reduce mean finishing time", ***Communications on ACM***, Vol. 17, (1974), 382-387.
9. Garey, M. R. and Johnson, D. S., "Computers and intractability: a guide to the theory of NP-completeness", San Francisco, CA: Freeman, (1979).
10. Webster, S. T., "The complexity of scheduling job families about a common due date", ***Operations Research Letters***, Vol. 20, (1997), 65-74.
11. Bruno, J. and Sethi, R., "Task sequencing in a batch environment with setup times", ***Foundations of Control Engineering***, Vol. 3, (1978), 105-117.
12. Mason, A. J. and Anderson, E. J., "Minimizing flow time on a single machine with job classes and setup times", ***Naval Research Logistics***, Vol. 38, (1991), 333-350.
13. Monma, C. L. and Potts, C. N., "On the complexity of scheduling with batch setup times", ***Operations Research***, Vol. 37, (1989), 798-804.
14. Ramachandra, G. and Elmaghraby, S. E., "Sequencing precedence-related jobs on two machines to minimize the weighted completion time", ***Int. J. Production Economics***, Vol. 100, (2006), 44-58.
15. Nessah, R., Chu, C. and Yalaoui, F., "An exact method for $P_m / sds, r_i / \sum_{i=1}^{n} C_i$ or $(P_m / sds, r_i / \sum_{i=1}^{n} C_i)$ problem", ***Computers and Operations Research***, Vol. 34, (2007), 2840-2848.
16. Monch, L., Balasubramanian, H., Fowler, W. J. and Pfund, E. M., "Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times", ***Computers and Operations Research***, Vol. 32, (2005), 2731-2750.
17. Leung, J. Y. - T., Li, H. and Pinedo, M. L., "Scheduling orders for multiple product types to minimize total weighted completion time", ***Discrete Applied Mathematics***, Vol. 155, (2007), 945-970.
18. Zhu, Z. and Heady, R. B., "Minimizing the sum of earliness/tardiness in multi machine scheduling with sequence dependant setups on uniform parallel machines", ***Computer and Industrial Engineering***, Vol. 38, (2000), 297-305.
19. Omar, M. K. and Teo, S. C., "Minimizing the sum of earliness/tardiness in identical parallel machines schedule with incompatible job families: An improved MIP approach", ***Applied Mathematics and Computation***, Vol. 181, (2006), 1008-1017.
20. Biskup, D. and Cheng, T. C. E., "Multiple-machine scheduling with earliness, tardiness and completion time penalties", ***Computer and Operations Research***, Vol. 26, (1999), 45-57
21. Caoa, D., Chen, M. and Wan, G., "Parallel machine selection and job scheduling to minimize machine cost and job tardiness", ***Computers and Operations Research***, Vol. 32, (2005), 1995-2012.
22. Balakrishnan, N., Kanet, J. and Sridharan, S. V., "Early/tardy scheduling with sequence dependent setups on uniform parallel machines", ***Computer and Operations Research***, Vol. 26, (1999), 127-141.
23. Dunstall, S., Wirth, A. and Baker, K., "Lower bounds and algorithms for flow time minimization on a single machine with set-up times", ***Journal of Scheduling***, Vol. 3, (2000) 51-69.
24. Park, Y., Kim, S. and Lee, Y. - H., "Scheduling jobs on parallel machines applying neural network and heuristic rules", ***Computers and Industrial Engineering***, Vol. 38, (2000), 189-202.
25. Li, C. and Cheng, T., "The parallel machine min-max weighted absolute lateness scheduling problem", ***Naval Research Logistics***, Vol. 41, (1993), 33-46.
26. Cheng, R. and Gen, M., "MinMax earliness/tardiness scheduling in identical parallel machine system using genetic algorithm", ***Computer and Industrial Engineering***, Vol. 29, No. 1-4, (1995), 513-517.
27. Kim, D. - W., Kim, K. - H. Wooseung, Jang, F. and Chen, F., "Unrelated parallel machine scheduling with setup times using simulated annealing", ***Robotics and Computer Integrated Manufacturing***, Vol. 18, (2002), 223–231.
28. Tavakkoli-Moghaddam, R., Jolai, F., Khodadadeghan, Y. and Haghnevis, M., "A mathematical model of a multi-criteria parallel machine scheduling problem: a genetic algorithm", ***International Journal of Engineering, Transactions A: Basic***, Vol. 19, No. 1, (2006), 79-86.
29. Melve, S. and Uzsoy, R., "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***, Vol. 34, (2007), 3016-3028.
30. Gen, M. and Cheng, R., "Genetic algorithms and engineering design", John Wiley and Sons, New York, (1997).
31. Leonardi, S. and Raz, D., "Approximating total flow time on parallel machines", ***Journal of Computer and System Science***, Vol. 73, (2007), 875-891.
32. Kang, L. and Ng, C. T., "A note on a fully polynomial-time approximation scheme for parallel-machine scheduling with deteriorating jobs", ***International of Journal of Production Economics***, Vol. 109, (2007), 180-184.