Journal of the Society of Korea Industrial and Systems Engineering
Vol, 31, No. 4, pp.21－28, December 2008.

단일 Burn-In Oven에서 Total Weighted Earliness와 Tardiness를 최소화하기 위한 유전자 알고리즘의 활용

박 유 진†

중앙대학교 상경학부 경영학과

Minimization of Total Weighted Earliness and Tardiness on a Single Burn-In Oven Using a Genetic Algorithm

You-Jin Park†

Department of Business Administration, College of Social Sciences, Chung-Ang University

본 연구는 반도체 제조공정에서 사용되는 단일 Burn-In oven에서의 Total weighted earliness와 Tardiness를 최소화하기 위한 생산 스케줄링을 결정하는 문제를 다룬다. 본 연구에서는 모든 작업은 상시에 시작가능하고 각각은 서로 다른 가중치를 가지고 있다고 가정하였다. 일반적으로 단일 Burn-In oven은 다양한 작업들이 동시에 가능한 Batch processing 기계이다. 따라서 다양한 작업들로 구성된 하나의 Batch의 Processing time은 그 Batch 내에 있는 가장 긴 Processing time을 가지는 작업에 의해 결정된다. 본 연구에서 Batch size는 미리 결정되지 않은 상황이라고 가정한 후, 최적의 Batch 개수와 작업의 순서를 결정하기 위해 유전자 알고리즘을 적용하였다. 수리적 예제를 통해서 다양한 접근방법의 성능들을 비교한 결과, 유전자 알고리즘이 Total weighted earliness와 Tardiness를 최소화하는데 가장 뛰어난 성능을 가지고 있음을 알 수 있다.

Keywords : Scheduling Problem, Burn-In Oven, Genetic Algorithms, Earliness, Tardiness, Dynamic Programming

1. Introduction

This research is motivated by the problem of scheduling compatible jobs proceeded in burn-in ovens in semiconductor manufacturing. In burn-in operation, semiconductor chips are exposed to high temperatures in a fixed capacity oven in order to weed out chips susceptible to premature failure. Generally, different types of chips have different processing times and require different level of temperature in the oven. The operation of burn-in oven, as the final stage of the semiconductor manufacturing processes, is to test whether all integrated circuits are defective products or not before passing on to the customers (Mehta and Uzsoy, 1998; Azizoglu and Webster, 2001).

In this paper we consider the problem of scheduling a given set of jobs and determining the batch sizes on a single burn-in processing machine in order to minimize the total weighted earliness and tardiness. It is assumed that there is a set of jobs denoted by $j = 1, \cdots, n$, which can be grouped into size of B at maximum without any restriction to grouping. Also, it is assumed that a certain job j is available at time zero, and its processing time, due date, and weight

논문접수일 : 2008년 03월 31일　논문수정일 : 2008년 08월 14일　게재확정일 : 2008년 09월 18일
† 교신저자 eugenepark@cau.ac.kr
※ The authors wish to thank Y. C. Choung of Samsung Electro-nics Co., Ltd. for sharing his expertise.

are represented by p_j, d_j, and w_j, respectively. Since the burn-in oven is a batch processing machine which can process up to B number of jobs simultaneously, the processing time of the batch is equal to the longest processing time of a job in that batch (Chandru et al., 1993). Since the throughput of the burn-in operations is very critical for both satisfying customer and reducing the inventory cost, the effective scheduling and batching of jobs are needed. However, it is known that this scheduling problem is a very complicate problem (*NP*-hard problem) by the fact that there is a tremendous number of the possible combination of the batches and job sequences. Thus, we aim to find an optimal job schedule and batches that minimize the total weighted earliness and tardiness simultaneously on a single burn-in oven environment. In next sections, we review previous studies concerning the scheduling problems with earliness and tardiness and then apply a genetic algorithm and a dynamic programming approach to obtain the optimal number of batches as well as the optimal job schedule simultaneously under the assumption that the number of batches has not been determined.

2. Previous studies

Many production scheduling problems with varying criteria and assumptions relating to due dates and weights have been recognized for their complexity and treated in many literatures. Among these, the single machine scheduling problems have been extensively studied, and the performance measure of most of the previous work in single-machine scheduling problems was especially *tardiness* (Baker, 1974). However, in many manufacturing industries, with the advent of the Just-In-Time (JIT) concept, attention has been drawn towards minimization of earliness as well as tardiness because these two measures are important in reducing inventory cost and keeping the due dates to satisfy customers.

Gupta and Kyparisis (1987), and Sen and Gupta (1984) reviewed literatures related to variations of this problem [5, 6]. Sidney (1977), and Lakshminarayan et al. (1978) presented algorithms for minimizing the maximum penalty of early or late jobs. Baker and Scudder (1990) provided a comprehensive survey relating to earliness and tardiness models. Ferris and Vlach (1992) considered the general earliness and tardiness problem.

However, scheduling problems related to batch processing have not been extensively studied. Chandru, Lee, and Uzsoy (1993) have considered a problem of minimizing the total completion time on a batch processing machine with incompatible job families where the machine can process at most B jobs simultaneously as a batch and the processing time of a batch is dependent on the processing times of jobs in the batch. In Chandru et al. (1993), the set of jobs to be scheduled can be partitioned into a number of families, where all jobs in the same family have the same processing time. They analyzed the properties of an optimal schedule and developed a dynamic programming algorithm of polynomial time complexity when the number of job families is fixed.

Mehta and Uzsoy (1998) have considered a problem of minimizing the total tardiness on a batch processing machine with incompatible job families, where all jobs of the same family have identical processing times and jobs of different families cannot be processed together. They showed that a greedy Earliest Due Date (EDD) rule can be used to form batches for each family and then developed a dominance rule which reduces the time for sequencing batches. Based on these results, they developed a dynamic programming algorithm which has polynomial time complexity when the number of job families and the batch machine capacity are fixed, and then examined various heuristic methods which can provide near optimal solution in a reasonable amount of computation time. In order to evaluate the performance of the heuristic methods they developed, they carried out a series of computational experiments using randomly generated test problems (Mehta and Uzsoy, 1998).

Azizoglu and Webster (2001) have considered a problem of scheduling the burn-in operation in the fabrication of integrated circuits, and attempted to minimize total weighted completion time. In this research, they proposed a branch and bound procedure applicable to a batch processor model with incompatible job families, that is, jobs in a given family have identical job processing time, arbitrary job weights, and arbitrary job sized. As a result, they found that the branch and bound algorithm returns optimal solutions for problem instances with up to about 25 jobs within 30 minutes of CPU time, and that it can be adapted for use as a heuristic to save CPU time and to use as a tool for larger problem instances (Azizoglu and Webster, 2001).

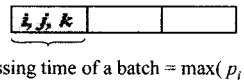
Wang and Uzsoy (2002) have considered a problem of minimizing maximum lateness on a batch processing machine in the presence of dynamic job arrivals, that is, jobs is arriving at the machine dynamically over time. They adapted a dynamic programming algorithm to determine whether

a due-date feasible batching exists for a given job sequence, and then combined this algorithm with a random keys encoding scheme to develop a genetic algorithm for solving this problem. As a consequence, they showed that computational experiments indicate this algorithm has excellent average performance with reasonable computational burden.

Mönch, Unbehaun, and Choung (2006) have considered a scheduling problem on a burn-in oven to reduce earliness-tardiness under the constraint that the maximum tardiness should be less than or equal to the maximum allowable time. They applied several heuristic algorithms including a genetic algorithm. More detailed discussions on the scheduling problem of minimizing the earliness and tardiness can be found in Garey (1988), Ow and Morton (1989), and Pinedo (1995).

3. Problem modeling and solution approaches

In this section, we describe the details and notations of the problem, and solution approaches. As mentioned above, the effectiveness and throughput of this operation in semiconductor manufacturing is very critical in both customer satisfaction and inventory cost point of view. We now consider a single batch processing machine and assume that there is a set of jobs $m = 1, \cdots, n$. A job j is assumed to be available at time zero, and it has a processing time p_j, a due date d_j, and weights w_j, respectively. Since the burn-in oven is a batch processing machine which can process up to B jobs simultaneously as a batch, the processing time of the batch is equal to the longest processing time of the jobs in the batch as shown below:



<Figure 1> Determination of processing time of a batch

Our objective is to find an appropriate job sequence and determine the size of batches that minimize the total weighted earliness and tardiness simultaneously. In a mathematical form, the objective function of this problem can represented as:

$$\text{Min } \Sigma \ (w_j E_j + w_j T_j),$$

where E_j, T_j, and w_j represent earliness, tardiness of job j, and assigned weight to job j, respectively. In order to obtain an optimal job sequence and batch form, we consider 5 different job sequencing rules and 2 different job batching rules as listed below:

- Job Sequencing Rules : SPT, LPT, EDD, Smallest PDW, and MST
- Batching Rules : LOE (Last Only Empty) and Random

For example, there are supposed to be 5 jobs with different processing times, due dates, and weights which are denoted by PT, DD, and W, respectively. If we rearrange the job sequence by Shorted Processing Time (SPT) first rule, Longest Processing Time (LPT) first rule, Earliest Due Date (EDD) first rule, Smallest ($p_j \times d_j$)/w_j (PDW) first rule, and Minimum Slack Time (MST) first rule, we can have 5 different job sequences as following table [15].

Now, in order to seeking for the optimal solution (as mentioned above, determining both optimal batch sizes and optimal job sequences simultaneously) of this scheduling, we can use a backward dynamic programming (DP), and the mathematical form of this scheduling problem as follows:

$$f(i) = \min_{1 \le j \le \min(i,B)} \left\{ \sum_{k=0}^{j-1} \left| d_{i-k} - \left(\max_{0 \le k \le j-1} p_{i-k} + C(i-j) \right) \right| \cdot (w_{i-k}) + f(i-j) \right\}$$

<Table 1> A job sequence and 5 new job sequences by SPT first, LPT first, EDD first, Smallest PDW first, and MST first rules

Job	PT	DD	W
1	8	14	4
2	1	4	2
3	3	9	2
4	12	5	3
5	5	8	1

Job_SPT	PT
2	1
3	3
5	5
1	8
4	12

Job_LPT	PT
4	12
1	8
5	5
3	3
2	1

Job_EDD	DD
2	4
4	5
5	8
3	9
1	14

Job_SPT	PDW
2	2
3	13.5
4	20
1	28
5	40

Job_MST	MST
4	-7
2	3
5	3
13	6
4	6

for $1 \le i \le n$, where i represents the number of jobs completed, j represents the number of jobs in the last batch, $f(i)$ is the minimum earliness and tardiness from first job to job i, B is the maximum batch size, and $C(i)$ is the minimum completion time of job i. We assume that $C(0) = f(0) = 0$. For example, when $i = 4$ and $B = 3$, since $1 \le j \le \min(i, B)$, then :

for $j = 1$:

$$f(4) = \sum_{k=0}^{0} \left| d_{4-k} - \left(\max_{0 \le k \le 0} (p_{4-k}) + C(4-1) \right) \right| \cdot (w_{4-k}) + f(4-1)$$
$$= \left| d_4 - p_4 - C(3) \right| \cdot (w_4) + f(3)$$

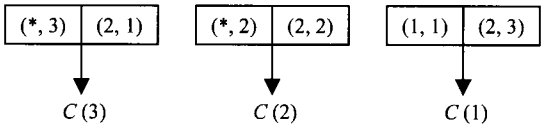
for $j = 2$:

$$f(4) = \sum_{k=0}^{1} \left| d_{4-k} - \left(\max_{0 \le k \le 1} (p_{4-k}) + C(4-2) \right) \right| \cdot (w_{4-k}) + f(4-2)$$
$$= \left| d_4 - \max(p_3, p_4) - C(2) \right| \cdot (w_4)$$
$$+ \left| d_3 - \max(p_3, p_4) - C(2) \right| \cdot (w_3) + f(2)$$

for $j = 3$:

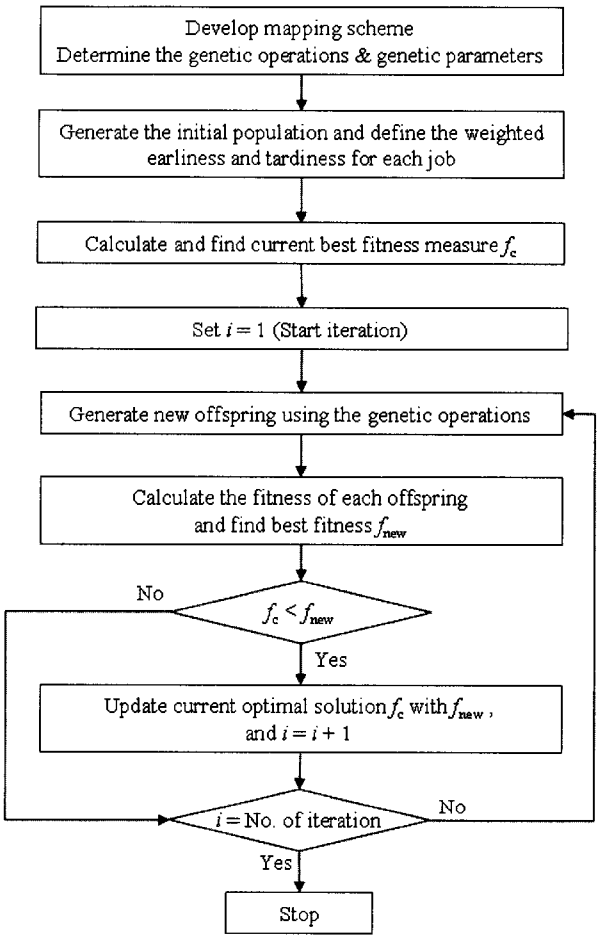
$$f(4) = \sum_{k=0}^{2} \left| d_{4-k} - \left(\max_{0 \le k \le 2} (p_{4-k}) + C(4-3) \right) \right| \cdot (w_{4-k}) + f(4-3)$$
$$= \left| d_4 - \max(p_2, p_3, p_4) - C(1) \right| \cdot (w_4)$$
$$+ \left| d_3 - \max(p_2, p_3, p_4) - C(1) \right| \cdot (w_3)$$
$$+ \left| d_2 - \max(p_2, p_3, p_4) - C(1) \right| \cdot (w_2) + f(1)$$

After calculating all $f(4)$ for j = 1, 2, and 3, we then choose the minimum $f(4)$ out of three solutions, and can illustrate this example as <Figure 2>. The values (a, b) in each box in <Figure 2> below represents the number of batches and the number of jobs, respectively. The notation * means that we do not know the specific number of batches, but we use the minimum value of $f(h)$ for $1 \le h \le i$-1.



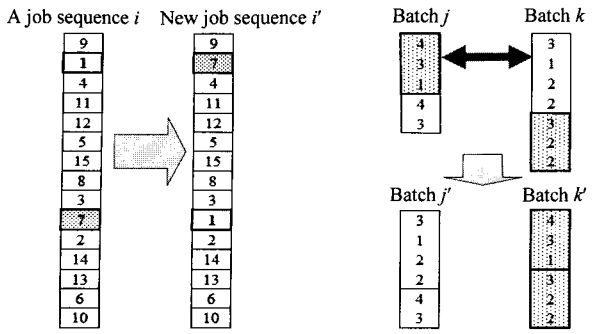
<Figure 2> All possible batch forms for i = 4, and j = 1, 2, and 3

In this research, a genetic algorithm (GA) as well as a backward dynamic programming approach is used for handling this problem. There are many applications of genetic algorithms to various scheduling problems. Since this problem is known as very difficult to solve due to the combinatorial aspects and the possibility of having numerous locally optimal solutions, we employ a genetic algorithm which is known to provide an optimal (or near optimal) solution within a reasonable computation time [17]. The general procedure for applying a genetic algorithm is shown in <Figure 3>.



<Figure 3> Genetic algorithm (GA) procedure

After generating the given number of initial population, then, in order to explore the optimal solution more efficiently, we generate new offspring for next generation by providing diversity through three different types of genetic operations to the problem; that is, cloning, mutation (swapping) for changing the job sequence, and crossover for constructing a new batch form. Each individual solution includes own batch form and job sequence. For example, consider a job sequence i in <Figure 4>. The job sequence i has initially 5 batches ([4, 3, 1, 4, 3]) denoted by j, and the sequence of jobs by [9, 1, 4, 11, 12, 5, 15, 8, 3, 7, 2, 14, 13, 6, 10]. Each value (defined as a chromosome in genetic algorithms) in the job sequence i represents the job number to be processed in machine (The index i in <Figure 3> repre-



<Figure 4> Mutation (Swapping) operation on a job sequence and crossover operation on batches

sents a job sequence, and the index j in 'Batch' represents a certain batch form). The mutation (swapping) operation is set to swap the job sequences which are randomly chosen where the number in a cell represents the job number, and the single-point crossover is conducted on only two batches where the number in one cell represents the number of jobs in one batch. The values in 'Batch' j and 'Batch' k represent the number of jobs assigned to each batch, for example, 4 in the Batch j means that the first 4 jobs are assigned to the first batch. For instance, if the job sequence i is set to be processed in Batch j, the first 4 jobs in job sequence i, that is, [9, 1, 4, 11], are assigned to the first batch in Batch j. The crossover operation is executed only when the number of jobs in the part exchanged in a batch is exactly same as that of the other batch. For example, since the total number of jobs in the gray part in Batch j, 8 (= 4 + 3 + 1), is same as that in the non-gray part in Batch k, 8 (= 3 + 1 + 2 + 2), the crossover operation can be executed. Through the mutation and crossover operations, the new job sequence i' obtained becomes [9, 7, 4, 11, 12, 5, 15, 8, 3, 1, 2, 14, 13, 6, 10], and the batch size of job sequence i is changed into 6 (Batch j of the job sequence i changes into Batch j', that is, [4, 3, 1, 4, 3] → [3, 1, 2, 2, 4, 3]).

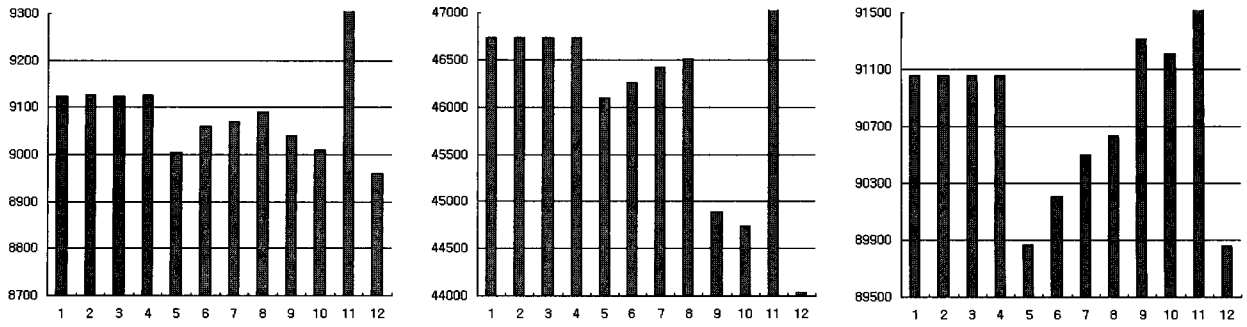
The cloning operation is to keep the given number of batches and sequences with outstanding performance. Through these operations, newly generated job sequences and batch forms are included in the new offspring. Note that if the number of jobs is n, the maximum number of batches is n because only one job can be assigned to one batch.

4. Computational results and analysis

In order to evaluate the performances of 10 different combined algorithms, dynamic programming approach, and a genetic algorithm, we generated 6 problem instances. We choose the number of iterations as 100 and initial population size as 100 and 200, and then we performed 10 times for each problem instance. We assume there are 100, 500, and 1000 jobs to be processed, and the maximum batch size (B) is assumed to be 3. The processing times of jobs for each problem instance are generated by a random number generator with $N(10, 2^2)$, and the due dates of jobs are generated by a random number generator with $N(45, 5^5)$, respectively. We also assume that there are 4 different weights (penalties) of jobs and assign the weights to jobs randomly. The summarized computational results of total weighted earliness and tardiness from 10 different algorithms for 6 different cases are shown as below. <Table 2> and <Table 3> contain values of total weighted earliness and tardiness when 10 different

<Table 2> Total weighted earliness and tardiness from simulation (Initial population size = 100) (B = Batch size, NJ = Number of Jobs, SPT = Shortest Processing Time First Rule, LPT = Longest Processing Time First Rule, EDD = Earliest Due Date First Rule, MST = Minimum Slack Time First Rule, LOE = Last Only Empty Rule)

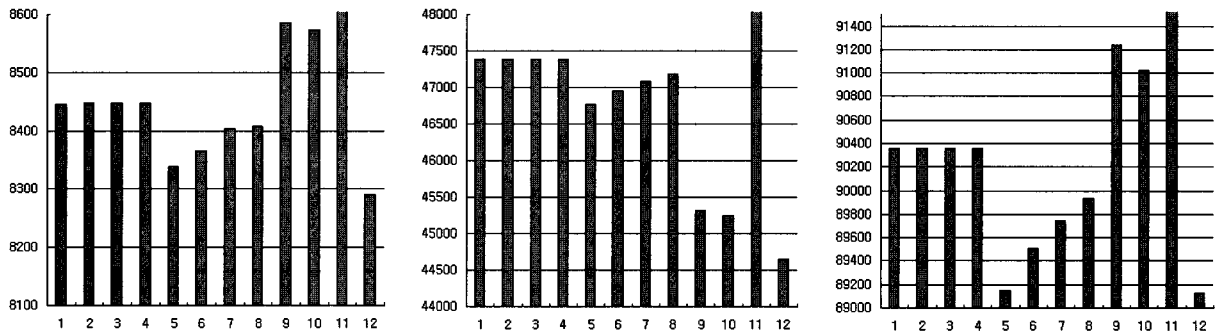
Sequencing Rule	Batch Rule	B = 3, NJ = 100	B = 3, NJ = 500	B = 3, NJ = 1000
SPT	LOE	9123.7	46738	91057
SPT	Random	9125.7	46740	91059
LPT	LOE	9124.3	46738	91057
LPT	Random	9125.9	46740	91059
EDD	**LOE**	**9005.7**	**46096**	**89867**
EDD	Random	9061.2	46263	90209
$(Pj \times Dj)/wj$	LOE	9069.8	46422	90496
$(Pj \times Dj)/wj$	Random	9089.6	46508	90634
MST	LOE	9039.3	44885	91313
MST	**Random**	9010.8	**44746**	91205
Dynamic Programming		1.27E+0.5	3.42E+06	1.36E+07
Genetic Algorithm		**8960.2**	**44042**	**89855**



<Figure 5> Comparison of total weighted earliness and tardiness for 3 different cases with initial population size of 100 (The numbers in row axis for each figure represent the combined scheduling-batch rules in order shown in <Table 2>)

<Table 3> Total weighted earliness and tardiness from simulation (Initial population size = 200) (B = Batch size, NJ = Number of Jobs, SPT = Shortest Processing Time First Rule, LPT = Longest Processing Time First Rule, EDD = Earliest Due Date First Rule, MST = Minimum Slack Time First Rule, LOE = Last Only Empty Rule)

Sequencing Rule	Batch Rule	B = 3, NJ = 100	B = 3, NJ = 500	B = 3, NJ = 1000
SPT	LOE	8445.9	47381	90358
SPT	Random	8447.4	47383	90360
LPT	LOE	8446.3	47381	90358
LPT	Random	8447.4	47383	90360
EDD	**LOE**	**8337.8**	46771	**89142**
EDD	Random	8366.2	46956	89506
$(Pj \times Dj)/wj$	LOE	8402.8	47087	89748
$(Pj \times Dj)/wj$	Random	8407.2	47177	89938
MST	LOE	8585.7	45321	91245
MST	**Random**	8573.4	**45245**	91020
Dynamic Programming		1.17E+0.5	3.49E+06	1.37E+07
Genetic Algorithm		**8290.1**	**44657**	**89125**



<Figure 6> Comparison of total weighted earliness and tardiness for 3 different cases with initial population size of 200 (The numbers in row axis for each figure represent the combined scheduling-batch rules in order shown in <Table 3>)

combined algorithms, dynamic programming approach, and a genetic algorithm applied. For example, the genetic algorithm produces total weighted earliness and tardiness of 8960.2 while dynamic programming approach produces total weighted earliness and tardiness of 1.27×10^5 for B = 3 and NJ = 100 case. Although the computation time of GA shows slightly longer than those of the other combined rules and backward dynamic programming approach, this seems to be

negligible.

As results, it is shown that total weighted earliness and tardiness of job schedules and batch forms obtained from 10 different combined algorithms and a dynamic programming are all greater than that obtained from GA for all instances considered. The backward dynamic programming approach provides the worst solution for all cases since the jobs are bound as a batch of the fixed size in EDD order without considering priorities of processing times and weights. We also see that the number of initial populations as well as a certain property of initial population affects the computational results, and, as the number of jobs increase, minimal difference exists between EDD-LOE rule or MST-Random rule and the genetic algorithm approach, which means that the EDD-LOE or MST-Random combined rules could be an attractive alternative to the other combined rules or a GA since the EDD-LOE or MST-Random combined rules could provide a better solution than the other combined rules including the backward dynamic programming approach do. However, further investigation will be needed in order to identify which algorithm provides better performance for the cases of the large number of jobs and large batch size. In conclusion, compared to the SPT, LPT, EDD, PDW, and MST rules combined with LOE and Random batching rules as well as a backward dynamic programming approach, the genetic algorithm performs very well and it gives better solution in total weighted earliness and tardiness point of view.

5. Conclusion and further study

These types of scheduling optimization problems frequently rise in many complicated manufacturing and service industries, and it is known that it is very difficult to obtain an optimal job sequence and batch size for the problem within reasonable computation time. Efficient algorithmic techniques have been developed and used to deal with these kinds of scheduling problems. In this research, we employ a genetic algorithm and provide near-optimal job sequence and batch size simultaneously to minimize total weighted earliness and tardiness on a single batch processing machine. We may improve the computational efficiency by modification or extensions to genetic algorithms GA-SA (Genetic Algorithm and Simulated Annealing) approach, or by using different types of genetic operations such as applying multi-point mutation (swapping) rule, or reverse order of the batches or two-point crossover on batches. As the task of next research, we can consider more realistic scheduling problems currently happening in the complex manufacturing processes, for example, a scheduling problem on minimization of completion time in multiple batch-parallel processing machines with different idle/release times and batch setup times.

References

[1] Azizoglu, M. and Webster, S.; "Scheduling a batch processing machine with incompatible job families," *Computers and Industrial Engineering*, 39, 325-335, 2001.

[2] Baker, K. R.; *Introduction to Sequencing and Scheduling*, Wiley, New York, NY, 1974.

[3] Baker, K. R. and Scudderb, G. R.; "Sequencing with Earliness and Tardiness Penalties : A Review," *Operations Researches*, 38(1), 22-36, 1990.

[4] Chandru, V., Lee, C. Y., and Uzsoy, R.; "Minimizing total completion time on a batch processing machine with job families," *Operations Research Letters*, 13, 61-65, 1993.

[5] Ferris, M. C. and Vlach, M.; "Scheduling with earli-ness and tardiness penalties," *Naval Research Logistics Quarterly*, 39, 229-245, 1992.

[6] Garey, M. R., Tarjan, R. E., and Wilfong, G. T.; "One-Processor Scheduling with Symmetric Earliness and Tardiness Penalties," *Mathematics of Operations Research*, 13, 330-348, 1988.

[7] Goldberg, D. E.; *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing, Boston, MA, 1989.

[8] Gupta, S. K. and Kyparisis, J.; "Single Machine Scheduling Research," *International Journal of Management Science*, 15(3), 207-227, 1987.

[9] Lakshminarayan, S., Lakshmanan, R., Papineau, R. L., and Rochete, R.; "Optimal single machine scheduling with earliness and tardiness penalties," *Operations Research*, 26(6), 1079-1082, 1978.

[10] Mehta, S. V. and Uzsoy, R.; "Minimizing total tardiness on a batch processing machine with incompatible job families," *IIE Transactions*, 30, 167-178, 1998.

[11] Mönch, L., Unbehaun, R. and Choung, Y. I.; "Minimizing earliness-tardiness on a single burn-in oven with a common due date and maximum allowable tardiness constraint," *OR Spectrum*, 28, 177-198, 2006

[12] Ow, P. S. and Morton, T. E.; "The Single Machine Early/Tardy Problem," *Management Science*, 35(2), 177-191, 1989.

[13] Pinedo, M.; *Scheduling : Theory, Algorithms, and Systems*. Prentice-Hall, NJ, 1995.

[14] Sen, T. and Gupta, S. K.; A State-of-the-Art Survey of Static Scheduling Research Involving Due Dates. *International Journal of Management Science*, 12(1), 63-69, 1984.

[15] Sidney, J.; Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties. *Operations Research*, 25(1), 62-69, 1977.

[16] Wang, C. S. and Uzsoy, R.; "A genetic algorithm to minimize maximum lateness on a batch processing machine," *Computers and Operations Research*, 29 : 1621-1640, 2002.