ICIC Express Letters
Part B: Applications
Volume **9**, Number **8**, August **2018**

ICIC International ©2018 ISSN 2185-2766
pp. **869–877**

PRODUCTION SCHEDULING IN STEEL MANUFACTURING WITH CUTTING AND PARALLEL OPERATIONS

Sujin Won[1], Hyerim Bae[1,*] and Riska Asriana Sutrisnowati[2]

[1]Department of Industrial Engineering
Pusan National University
Busandaehak-ro 63beon-gil 2 (Jangjeon-dong), Geum-jeong-gu, Busan 46241, Korea
woen3150@pusan.ac.kr; *Corresponding author: hrbae@pusan.ac.kr

[2]Pusan National University Dong-Nam Grand ICT R&D Center
QB e-Centum, 90 Centum Jungang-ro Haeundae Gu, Busan 48059, Korea
riska@pusan.ac.kr

Received January 2018; accepted April 2018

Abstract. *In a manufacturing process, there are many parts and some parts are divided into other parts. In this paper, we devise a steel company's schedule for cutting and parallel operations using our proposed algorithm based on the biased random-key genetic algorithm. All of the products of the steel industry are made of the same raw material in the forms of huge steel coils. The final product is obtained as the result of machine-sequence processing. Notably, the steel manufacturing process has unique features. A coil is split into several sub coils and in the annealing process, coils are processed at the same time. The annealing process in this sequence is subject to capacity constraints. We adapted our genetic algorithm to solve this scheduling problem and demonstrated its performance by testing it against real data from a steel company.*
Keywords: Job-shop scheduling, Biased random-key genetic algorithm, Simultaneous, Cutting operation, Parallel operation

1. **Introduction.** As customer needs have diversified, many manufacturers, seeking to meet these demands, have begun to customize their products to meet specific customer orders. Steel manufacturing is one such example. When a customer orders steel, he specifies (i.e., sets) its width, thickness, length, and hardness. Then, the steel-manufacturing company makes that customer's product to order via a given operations sequence. In this problem, there are N jobs, for each of which there is a set of operations that must be followed in a predefined sequence. This incurs the well-known NP-hard job-shop scheduling problem [2].

Steel offers the following two key properties: first, from its original, huge steel-coil form, it can be cut to size in the slitting and cold rolling mill processes; second, it can be rendered malleable in the annealing process. The annealing process is not conducted with only 1 coil but typically 2 or 3 coils. The number of simultaneous operating coils is determined by the number of bases of a machine. Domain experts determine how many coils are to be processed simultaneously in each base, which number can vary for each operation schedule. Consequently, this creates challenges in the scheduling process. Traditional job shop scheduling algorithms cannot handle our specific problems (e.g., slitting and simultaneous coils processing during annealing). Hence, in our model, as a contribution, we try to incorporate dynamic processing by finding the optimal number of coils processed simultaneously in each base and, eventually, to solve the steel-production scheduling problem using our proposed genetic algorithm (GA) that is based on the biased random-key GA.

DOI: 10.24507/icicelb.09.08.869

This paper is organized as follows. Sections 2 and 3 present the related work and problem definition, respectively. Section 4 discusses the proposed algorithms, while Section 5 shows the experimental results. Finally, Section 6 draws conclusions and anticipates future work.

2. Related Work.

2.1. **Job-shop scheduling problem.** Pinedo [4] defines the job-shop scheduling problem as a sequencing problem for a set of jobs in a set of machines that work in sequence (following a predetermined route). The machines are always available and interruption of jobs is not allowed. As stated in [2], the job-shop scheduling problem is an NP-hard problem. Researchers have introduced several ways to solve this problem either by heuristics [6,8] or meta-heuristics [5,7] methods. One algorithm that is widely used is the GA [9]. In [2], a survey of the related methods employed is provided. Researchers have experimented with the GA by way of searching the initial solution [12,13], chromosome representation [1], or operators [10,11].

2.2. **Random-key genetic algorithm.** The biased random-key GA was introduced by Bean in 1994 specifically for the sequencing problem [1]. It uses random-keys (between 0 and 1) to avoid the many infeasible solutions found by traditional GA and to present only good, feasible ones.

The general chromosome in the biased random-key GA can be seen in Figure 1. Every gene (7, 8, 4, 2, 6, 1, 5, 3) has a random-key (0.49, 0.35, 0.67, 0.32, 0.51, 0.38, 0.19 and 0.22, respectively) and all of the GA operators are processed by those random-keys. We can sort the random-keys in the following order: 0.19, 0.22, 0.32, 0.35, 0.38, 0.49, 0.51, 0.67. Afterwards, we can match all of the random-keys to genes from the smallest random-key. The smallest random-key of this chromosome is 0.19. Since 0.19 is in the 7^{th} position of this chromosome, the first gene is 7, and as 0.22 is the second smallest random-key, its position is 8^{th}; therefore, the second gene is 8. In this same way, random-keys represent the real chromosome. When this algorithm operates the GA operators (crossover, mutation), it uses only the random-keys.

Real chromosome	7	8	4	2	6	1	5	3
Random-key	0.49	0.35	0.67	0.32	0.51	0.38	0.19	0.22

FIGURE 1. Real chromosome of biased random-key genetic algorithm

The biased random-key GA is different for the selection operator. This can be seen in Figure 2. Whereas the general random-key GA selects parents randomly in all of the K-generation solutions, the biased random-key GA selects one parent in elite solutions and then selects the other parents randomly in non-elite solutions.

In the problem tackled in this paper, one chromosome represents an entire schedule; therefore, searching for optimal solutions takes a long time, and there are also many infeasible solutions. So, we designed, based on the biased random-key GA, a GA that can improve the performance of a scheduling algorithm.

3. Problem Definition.

In this paper, we devise the schedule of a steel manufacturing company. The company makes many kinds of products that differ by width, thickness, hardness, and other properties. In such a real steel manufacturing operation, accordingly, there are many kinds of processes, which are listed as follows: 1) Slitting (WS/SS): Cutting steel coil to customized size; 2) Cold Rolling Mill (CR): Pressing steel coil to fit

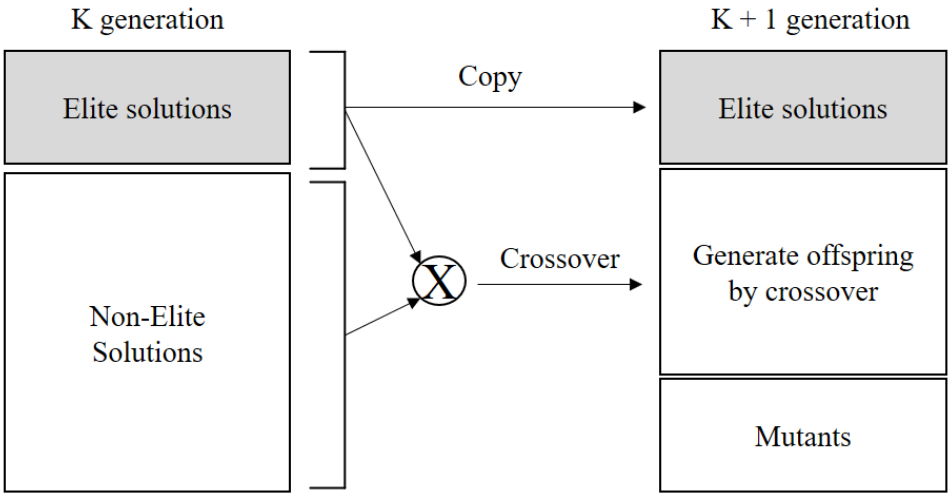

FIGURE 2. Structure of biased random-key genetic algorithm [3]

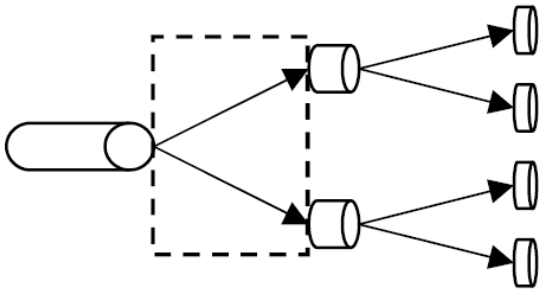

FIGURE 3. Cutting operation

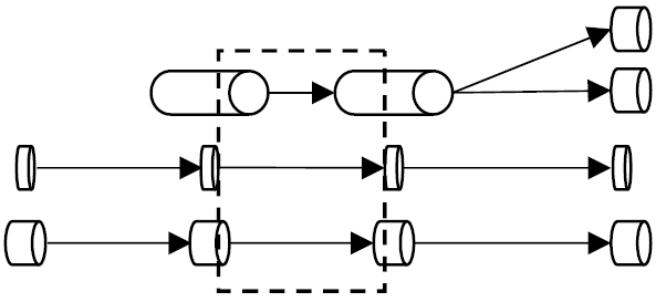

FIGURE 4. Parallel operation

customized thickness; 3) Pickling (PP): Oxidation treatment to clean surface of steel coil; 4) Annealing (AN): Heating and cooling steel coil to improve malleability; 5) Skin Pass Mill (SP): Pressing steel coil by weak force to improve quality of products; 6) Packing (PK): Packing of products; 7) Product (PR): Completion of products.

There are many kinds of products and each has a machine sequence. This is the same as in the job-shop scheduling problem. However, in this problem, a coil is divided into several parts and they are processed together (parallel operations) in an AN machine. In this paper, we consider this kind of process as cutting and parallel operations. In the cutting operation, one coil is divided into several coils (shown in Figure 3). These coils are handled at the same time.

In the parallel operation, some of coils are merged into the same process. Even though all of them are different, they are handled at the same time (as shown in Figure 4). In this paper, the AN process is a parallel operation. In the AN process, if the AN machine has 3

bases, 3 coils can be processed together. However, an AN machine also has its maximum capacity, which limits the number of simultaneous coils. In this case, the processing time is determined by which coil undergoes the longest processing time. Therefore, in this problem, the makespan changes according to how the coils are merged. After the AN process, each coil is processed according to its own machine sequence.

The problem of scheduling in a steel company can be stated as follows (Table 1 provides the index of this problem). Let i $(= 1, \ldots, I)$ be an index of coils, j $(= 1, \ldots, J)$ be an index of job sequences and $k = (1, \ldots, K)$ be an index of machines. *NoAN* is the number of machines for the AN process. Note that indexes smaller than or equal to *NoAN* are all used for AN machines. Index l is the simultaneous operation index. If l is 0, the coil is processed alone; otherwise, (if l is not 0), it is processed with other coils that have the same l index. *WC* is the maximum weight of coil.

Table 1. Notations

Index	Description
y_{ijkl}	Start time of coil i in the j-th sequence on machine k with l-th index
C_{ijkl}	Completion time of coil i in the j-th sequence on machine k with l-th index
$C_{\max}$	Maximum completion time (makespan)
p_{ijkl}	Processing time of coil i in the j-th sequence on machine k with l-th index
$x_{ii'k}$	1, if coil i' is processed before coil i on machine k; 0, otherwise.

The decision variables are y_{ijkl} and $x_{ii'k}$. The problem model is

$$\text{minimize } C_{\max} \tag{1}$$

s.t.

$$y_{ij'kl} - y_{ijkl} \geq p_{ijkl} \qquad \forall (i,j,k,l) \rightarrow (i,j',k,l) \tag{2}$$

$$C_{\max} - y_{ijkl} \geq p_{ijkl} \qquad \forall (i,j,k,l) \tag{3}$$

$$Mx_{ii'k} + (y_{ij'kl} - y_{ijkl}) \geq p_{ijkl} \qquad \forall (i,j,k,l),\ \forall (i',j',k,l) \tag{4}$$

$$M\,(1 - x_{ii'k}) + (y_{ijkl} - y_{i'jkl}) \geq p_{i'jkl} \qquad \forall (i,j,k,l),\ \forall (i'j',k,l) \tag{5}$$

$$y_{ijkl} = y_{i'j'kl} \qquad \forall (i,j,k),\ \forall (i',j',k),\ l = 1, \ldots, L \tag{6}$$

$$CN_{kl} \leq N \qquad \text{if } k \text{ is 'AN'},\ (k = 1, \ldots, NoAN) \tag{7}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{s} \sum_{k=1}^{m} W_{ijkl} \leq WC \qquad \text{if } k \text{ is 'AN'},\ (k = 1, \ldots, NoAN) \tag{8}$$

$$y_{ijkl} \geq 0 \qquad \forall (i,j,k,l) \tag{9}$$

The objective function (1) is to minimize the makespan. Constraint (2) ensures that all of the operations are processed in their proper sequence [4]. This is the same as the job-shop scheduling problem. We must perform all operations according to constraint (3). Constraints (4) and (5) determine the coil operation sequence for each machine. Constraint (6) makes simultaneous operations start at the same time. The AN process has two constraints: weight capacity and maximum number of simultaneous operations, which are presented in constraints (7) and (8), respectively.

4. **Proposed Algorithm.** In this paper, we devise the schedule of a steel company that has several jobs to perform in a predefined operations sequence. However, the AN machine and slitting machine can be run at the same time. In the job-shop scheduling problem, all operations cannot be processed at the same machines simultaneously. Therefore, the proposed algorithm, designed based on the biased random-key genetic algorithm, considers

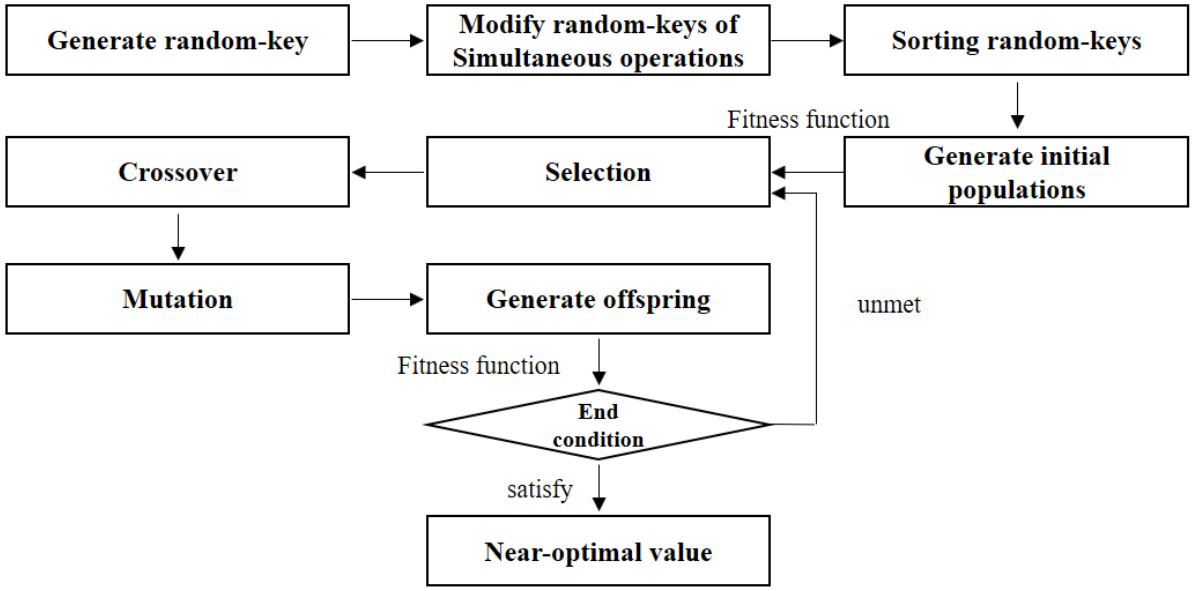

FIGURE 5. Overview of the proposed algorithm

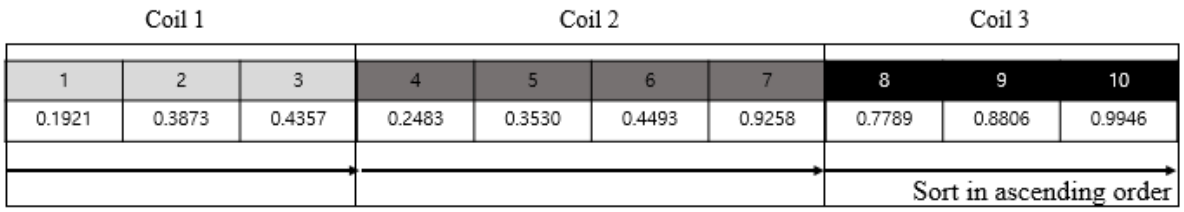

FIGURE 6. Generation of sorted random-keys in ascending order

this among the constraints and does not make infeasible solutions in the GA operators (Figure 5).

4.1. **Initial solution.** When the initial solution is generated, it must consider the order of jobs. Generation of the initial solution proceeds in 3 steps:

Step 1. Generate sorted random-keys in ascending order by each coil (shown in Figure 6)

Step 2. Sort all random-keys in the chromosome

Step 3. Complete initial solutions by merging the random-key of Step 1 and the real chromosome in Step 2.

Then, every initial solution is considered an operations sequence and thus is a feasible solution.

In this problem, there are simultaneous operations. To treat them, we modify their random-keys. If any of them are performed together, they share the same random-key. After generating the initial solution, the parents are selected. This is the same as in the biased random-key genetic algorithm. We calculate the fitness value of the initial solution according to the makespan. We copy the elite set to the next generation; then, we select the first parent in the elite set, and the other parent in the non-elite set.

4.2. **Crossover.** When we perform crossover, we also consider the sequence of jobs and simultaneous operations. As this algorithm is based on the biased random-key genetic algorithm, it uses only the random-key in the crossover operator. In Figure 7, there are 2 parents and 2 offsprings. The schedule consists of 3 coils and 15 operations. Each color is the same coil which must be processed in ascending order. When we perform crossover,

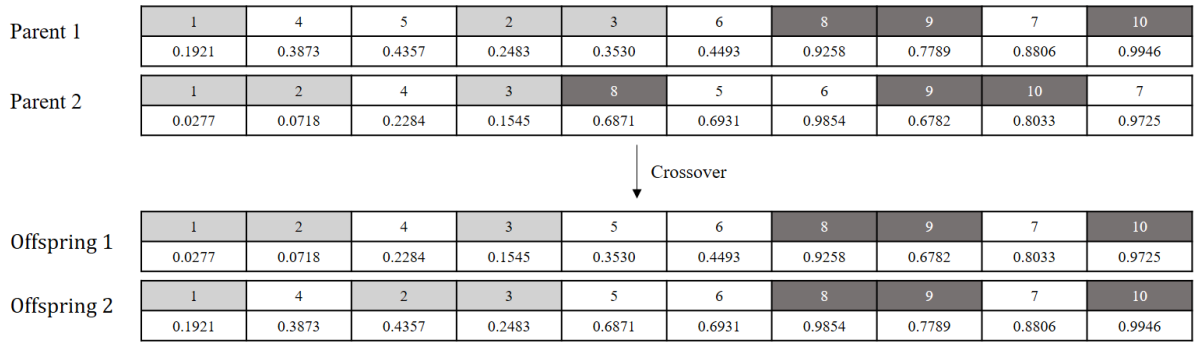

FIGURE 7. Crossover in the proposed algorithm

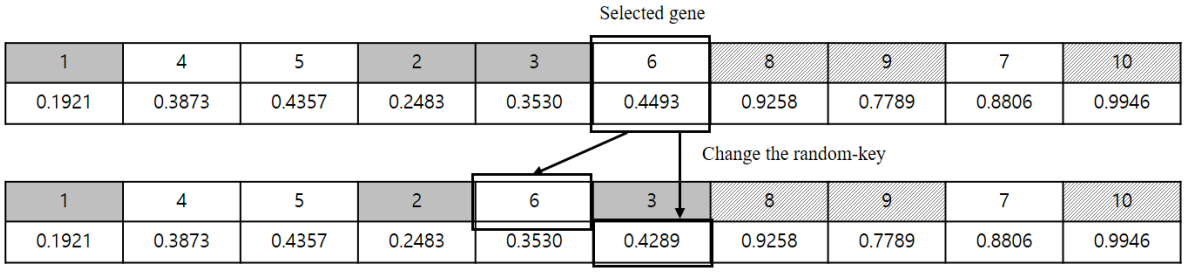

FIGURE 8. Mutation in the proposed algorithm

we compare each of the parents of the random-key at the same position. In Figure 7, the first random-key gene of parent 1 is 0.1921, and the first random-key gene of parent 2 is 0.0277. As 0.0277 is smaller than 0.1921, the first random-key gene of offspring 1 is 0.0277 and the first random-key gene of offspring 2 is 0.1921. The other genes in the offspring are generated in the same way.

4.3. **Mutation.** In the GA, mutation is necessary in order to avoid the local optima. In this algorithm, we design mutations according to the operations sequence and simultaneous operations. First, we select the chromosome to mutate with probability. Second, we select the gene to mutate. Third, we change the random-key of the selected gene within the operations sequence. In Figure 8, we select the 6^{th} gene to mutate. Then, we must consider the random-keys of the 5^{th} gene (0.3530) and the 7^{th} gene (0.9258) to keep the operations sequence. Note that the random-key of the 6^{th} gene must be larger than that of the 5^{th} gene and smaller than that of the 7^{th} gene. Therefore, it is changed to 0.4289. The original random-key of the 6^{th} gene (0.4493) is larger than the random-key of the 3^{rd} gene (0.4357); however, the changed random-key (0.4289) is smaller. Then, the real chromosome value is changed.

5. **Experiment.** The proposed algorithm was implemented on an Intel® Core™ i7-4790K 4.00Hz CPU (16GB RAM; Matlab R2016a). In our experiment, we tested different values of the parameters of this algorithm and obtained the following parameters: 1) population size: 30; 2) iteration limit: 500; 3) rate of crossover and mutation: 40% and 40%, respectively; 4) mutation probability: 60%.

Using a small size problem as shown in Gantt chart in Figure 9 and the respective encoded chromosome in Table 2, e.g., 6 coils and 15 operations, we compare the proposed GA, the precedence order crossover (POX) GA and the LP model (refer to Table 3). It shows that the makespan of all algorithms is 40, but that the processing of the proposed GA is less than that of the precedence order crossover (POX) GA and the mathematical model. Here, we are sure that our proposed method is advantageous since we can obtain an optimal solution for small-size data (or a nearly optimal solution for large size data)

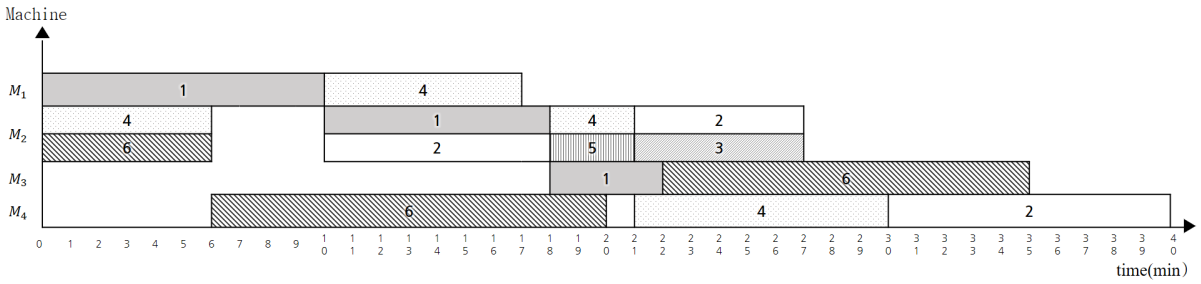

FIGURE 9. Result of simple example

TABLE 2. Result of simple example in chromosome

Coil	4	6	1	4	6	1	2	4	5	1	4	2	3	6	2
Operation	8	13	1	9	14	2	4	10	12	3	11	5	7	15	6
Machine	2	2	1	1	4	2	2	2	2	3	4	2	2	3	4
Random -key	0.1568	0.4983	0.7862	0.4983	0.8427	0.9574	0.8427	0.1086	0.1568	0.5222	0.7874	0.5222	0.1086	0.2248	0.8753

TABLE 3. Comparison result of simple example

	Proposed GA	POX GA	LP model
Processing time (sec)	0.85	1.19	41.35
Makespan	40	40	40

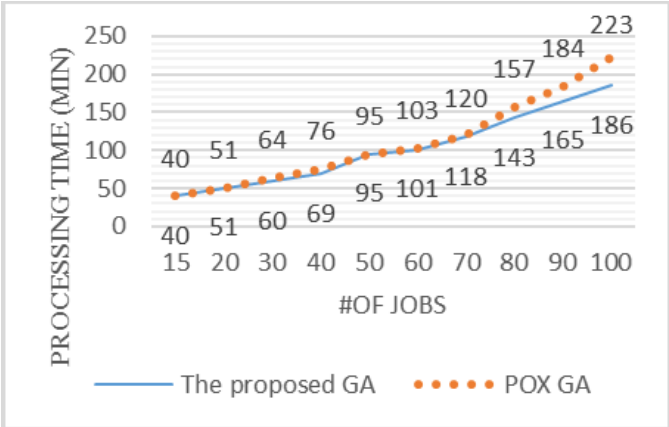

FIGURE 10. Job processing time comparison for the proposed GA and POX GA

in significantly less time. Figure 10 shows a further experiment comparing the POX GA and our proposed GA for an increasing number of jobs; the results consistently show that our proposed GA spent less time than the POX GA.

Next, we tested real data from the steel company. First, we tested data for 3 days (2016-01-02 00:00:00-2016-01-04 23:56:00): 245 products were produced over the course of 492 operations, as shown in Figure 11. Our algorithm took 300.4059 seconds to schedule the 3 days of real data. Because the makespan was changed according to the combination of coils in the AN, the average makespan oscillated up and down. The operation time for the real data was 4,316 minutes; however, the minimum makespan of our algorithm was 2,748 minutes, representing a significant reduction of about 36.3%. This indicated that our algorithm can improve the productivity of steel manufacturing with effective scheduling.

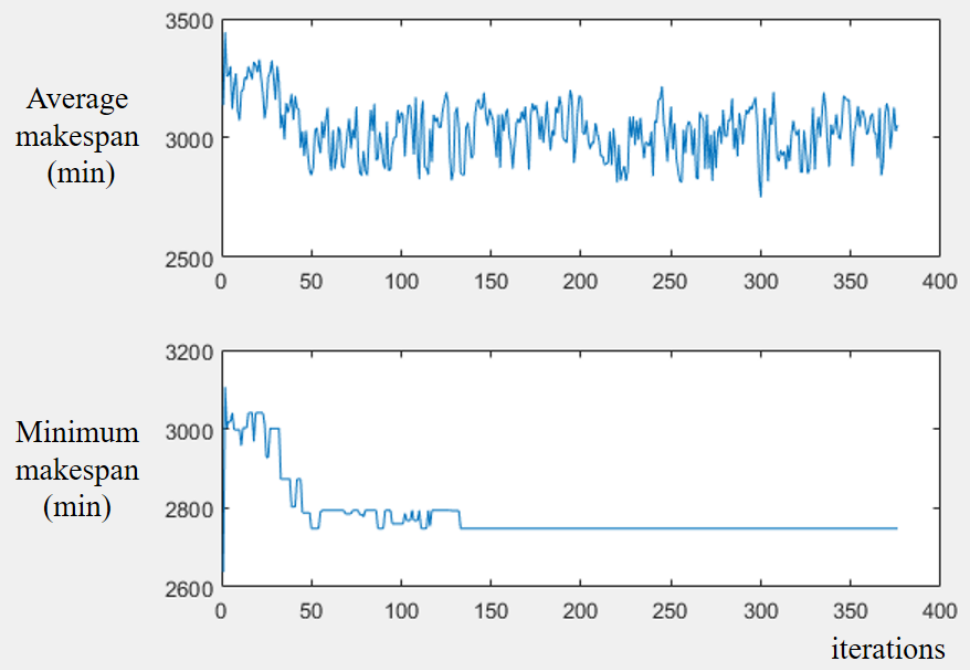

FIGURE 11. Average and minimum makespans of 492 operations in 3 days

6. **Conclusions.** As discussed in this paper, we designed a genetic algorithm based on the biased random-key genetic algorithm for the job-shop scheduling problem with cutting and parallel operations. In our algorithm, there were no infeasible solutions. The results of our computational study showed that our algorithm can find the optimal solution for a small-size problem more effectively than real data. The following are some of the goals of our upcoming research: forecasting of processing time to improve the accuracy of scheduling; comparison with other genetic algorithms for the same problem (cutting and parallel operations); consideration of other fitness functions for calculation of the fitness values of individuals and analysis of various problem sizes in testing the performance of our algorithm.

Acknowledgment. This research was partly supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Grand Information Technology Research Center support program (IITP-2017-2016-0-00318) supervised by the IITP (Institute for Information & Communications Technology Promotion); additional support was received in the form of a National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (No. NRF-2015R1D1A1A09061331).

REFERENCES

[1] J. C. Bean, Genetic algorithms and random keys for sequencing and optimization, *ORSA Journal on Computing*, vol.6, no.2, pp.154-160, 1994.
[2] A. Jain and S. Meeran, Deterministic job-shop scheduling: Past, present and future, *European Journal of Operational Research*, vol.113, pp.390-434, 1999.
[3] J. F. Gonçalves and M. G. C. Resende, Biased random-key genetic algorithms for combinatorial optimization, *Journal of Heuristics*, vol.17, pp.487-525, 2011.
[4] M. Pinedo, *Scheduling*, Springer, 2015.
[5] S. Knopp, S. Dauzère-Pérès and C. Yugma, A batch-oblivious approach for complex job-shop scheduling problems, *European Journal of Operational Research*, pp.50-62, 2017.
[6] A. Jamili, Robust job shop scheduling problem: Mathematical models, exact and heuristics algorithms, *Experts Systems with Applications*, pp.341-350, 2016.
[7] R. Mencía, M. R. Sierra, C. Mencía and R. Varela, Memetic algorithms for the job shop scheduling problem with operators, *Applied Soft Computing*, pp.94-105, 2015.

[8] I. Assaf, M. Chen and J. Katzberg, Steel production schedule generation, *International Journal of Production Research*, pp.467-477, 1997.

[9] M. Kurdi, An effective new island model genetic algorithm for job shop scheduling problem, *Computers & Operations Research*, pp.132-142, 2016.

[10] M. Watanabe, K. Ida and M. Gen, A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem, *Computer & Industrial Engineering*, pp.743-752, 2005.

[11] M. Amirghasemi and R. Zamani, An effective asexual genetic algorithm for solving the job shop scheduling problem, *Computers & Industrial Engineering*, pp.123-138, 2015.

[12] J. Li, Y. Huang and X. Niu, A branch population genetic algorithm for dual-resource constrained job shop scheduling problem, *Computers & Industrial Engineering*, pp.113-131, 2016.

[13] Y. Liu, Different initial solution generators in genetic algorithms for solving the probabilistic traveling salesman problem, *Applied Mathematics & Computation*, vol.216, no.1, pp.125-137, 2010.