Expert Systems with Applications xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Expert Systems with Applications

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

A hybrid genetic algorithm for the hybrid flow shop scheduling problem with nighttime work and simultaneous work constraints: A case study from the transformer industry

Sungbum Jun, Jinwoo Park *

Department of Industrial Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 151-742, Republic of Korea

A R T I C L E I N F O

Article history:
Available online xxxx

Keywords:
Hybrid flow shop scheduling problem
Nighttime work
Simultaneous work
Genetic algorithm
Local search
Transformer industry

A B S T R A C T

This paper addresses a hybrid flow shop scheduling problem with real-world constraints, and proposes a novel algorithm for its solution. We first discuss the distinguishing characteristics of nighttime and simultaneous work in the transformer manufacturing process. To solve the problem within a reasonable time, we propose a hybrid genetic algorithm. This algorithm combines the Nawaz–Enscore–Ham (NEH) heuristic, a local search algorithm, and a machine allocation rule with the aim of minimizing the total tardiness. Our experimental results show that the proposed algorithm outperforms the NEH algorithm, a simple genetic algorithm, and five existing dispatching rules in terms of average total tardiness performance and relative deviation index. The proposed algorithm is also shown to be competitive with respect to its efficiency and robustness.

1. Introduction

The flow shop scheduling problem is common in many production systems. In certain environments, parallel machines are made up of multiple copies and grouped into stages. For these production environments, the traditional flow shop scheduling model is inappropriate, because some stages utilize parallel machines. This type of problem can be defined as a hybrid flow shop scheduling problem (HFSP).

The hybrid flow shop is an extension of the production system in a traditional flow shop. It consists of two or more stages in series and one or more parallel machines at each stage to increase productivity and flexibility. Examples of hybrid flow shop problems are floor covering production, glass-bottle industry, and so on (Lopez & Roubellat, 2008).

In this type of shop, the major issues are the allocation of jobs to machines at each stage, and the sequence of jobs assigned to each machine. HFSPs have been extensively studied; however, most examples are NP-hard (Linn & Zhang, 1999).

This paper focuses on the scheduling problem in hybrid flow shops with two distinguishing constraints: the consideration of daytime and nighttime work teams and simultaneous work of specific order types. Our research is motivated by an industrial transformer manufacturing system with a number of availability conditions between various product types and machines. In this case, a feasible solution that minimizes the total tardiness (that is, the total time by which order processing is delayed) is vitally important, because the penalty cost of tardy jobs has a detrimental effect on a company.

In addition to the characteristics of the general HFSP, there are constraints on the waiting times between successive stages of a job, as well the consideration of nighttime work and simultaneous work at each stage.

This paper is organized as follows. In the next section, previous research into hybrid flow shop scheduling is reviewed. The problem and constraints of a transformer manufacturing system are defined in Section 3, and a hybrid genetic algorithm to solve this problem is then proposed in Section 4. Section 5 summarizes the results of experiments to verify our approach. Finally, our conclusions and areas for further research are discussed in Section 6.

2. Literature review

Arthanari and Ramamurthy (1971) considered the HFSP, and proposed the first Branch and Bound method. Kochhar and Morris (1987) developed heuristic algorithms to minimize the mean flow time for the flexible flow line problem with finite buffers. They divided the problem into two sub problems: entry point sequencing and dispatching. The two-stage HFSP was shown to be

* Corresponding author. Tel.: +82 2 880 7179.
E-mail addresses: junsb87@mailab.snu.ac.kr (S. Jun), autofact@snu.ac.kr (J. Park).

http://dx.doi.org/10.1016/j.eswa.2015.03.012
0957-4174/© 2015 Published by Elsevier Ltd.

NP-hard by Gupta (1988). Gupta, Hariri, and Potts (1997) then showed that a non-preemptive two-stage HFSP is NP-hard in the strong sense.

Exact approaches based on mathematical modeling can ensure higher performance than heuristic methods in finding optimal solutions of HFSP. Fattahi, Hosseini, Jolai, and Tavakkoli-Moghaddam (2014) developed a branch-and-bound algorithm that considered the setup time and assembly operations to minimize the makespan for HFSP. Sun and Yu (2015) deal with a two-stage HFSP with batch constraints and the variable processing times through a Lagrangian relaxation approach. However, because of their NP-hard nature, exact approaches are only applicable to small-scale problems. Thus, heuristic algorithms are widely used to obtain good approximations within a reasonable time (Ribas, Leisten, & Framiñan, 2010). Examples of such heuristic algorithms are the neighborhood search, simulated annealing, and genetic algorithms (GAs).

Heuristic approaches have been devised for solving the HFSP constraints that arise in actual applications. Holland (1975) first proposed the GA concept in his book "Adaptation in Natural and Artificial Systems". In traditional GAs, mutation is used to produce small changes to chromosomes, resulting in a varied population. Unlike traditional GAs, Tsujimura and Gen (1999) proposed a mutation operator with a neighborhood search technique to determine near-optimal solutions. Botta-Genoulaz (2000) proposed a heuristic algorithm based on the earliest due date (EDD) sequencing method with First Available Machine and Last Busy Machine allocation rules for the HFSP. Engin, Ceran, and Yilmaz (2011) proposed an efficient GA for hybrid flow shop scheduling with multiprocessor tasks. Liao, Tjandradjaja, and Chung (2012) proposed a particle swarm optimization (PSO) algorithm for the HFSP with a minimum makespan objective. They developed a hybridizing PSO with a bottleneck heuristic and simulated annealing to help escape from local optima. Bożejko, Pempera, and Smutnicki (2013) designed a parallel tabu search algorithm for an HFSP derived from automated manufacturing lines. Costa, Cappadonna, and Fichera (2014) considered a GA for the HFSP with parallel batching and eligibility constraints. Li, Pan, and Wang (2014) combined a neighborhood search algorithm with both chemical-reaction optimization and an estimation of distribution to minimize the HFSP makespan. Rossi, Pandolfi, and Lanzetta (2014) developed dynamic set-up rules for HFSP with parallel batching machines. They introduced heuristics based on the critical ratio between the setup and processing times to minimize makespan and the number of tardy jobs.

There are still two issues relevant to the majority of flow shop scheduling research. The first issue is the great complexity of real-world problem sizes. Unfortunately, although exact approaches such as MILP and dynamic programming can find an optimal solution, they are often impractical because of the extremely long calculation time for large problems. On the other hand, heuristic approaches such as GAs can be applied to more complex problems. However, the execution time and solution quality vary with the design of the algorithm. Thus, there is a significant need for efficient heuristic or meta-heuristic methods.

The second issue is the determination of various constraints in industry and their consideration in an algorithm. In real-world problems, a typical flow shop with a single machine at each stage rarely exists. Generally, there will be a variety of machines with different abilities placed in parallel at stages to increase capacity and balance the workload (Naderi, Gohari, & Yazdani, 2014). Although there have been a number of previous research articles on HFSPs in manufacturing systems, the assumptions made when developing their algorithms mean they have limited applicability (Ruiz & Vázquez-Rodríguez, 2010). Thus, consideration of other constraints, such as unrelated parallel machines and eligibility, is a significant step towards increasing the possibility of application in the field, and is thus worthy of further research.

The limitations of previous research with regard to these two issues make the study of a hybrid approach to HFSP more interesting. In this paper, Section 3 broaches the second issue by presenting the distinguishing constraints in a transformer production factory. Section 4 then deals with the first issue by describing a hybrid algorithm that efficiently incorporates a GA into heuristic methods.

3. Problem definition

In consideration of increasing market competition and the need to present a range of voltages and capacities, several types of transformer should be included in the scheduling process. In addition, there are a number of parallel machines (workbenches and drying furnaces) at each stage of the process, each with their distinguishing constraints. The entire process of transformer production is summarized in Fig. 1.

The problem is to schedule a hybrid flow shop (HFS) with m stages. Each stage has several machines operating in parallel, but the flow of jobs through stages is unidirectional. Some stages may have only one machine, but at least one stage must have multiple machines. The type of parallel machines can be identical, uniform, or unrelated. An operation refers to a specific period of processing by the selected machine.

Using the well-known three-field notation (Pinedo, 2008), the transformer production problem can be denoted by $FH2, \left(\left(RM^{(k)}\right)_{k=1}^{2}\right)|r_j|\sum T_j$ (Ruiz & Vázquez-Rodríguez, 2010). The type of parallel machines is the unrelated parallel machine that the processing time depends on the allocated machine. In certain practical applications with continuous job processing, such as in the plastics industry, there is limited intermediate storage space between stages (Moradinasab, Shafaei, Rabiee, & Ramezani, 2013). In this case, the number of jobs in intermediate storage should be minimized to reduce inventory costs. This implies that the waiting queue between two successive stages operates under the FIFO principle.

The following assumptions are also considered in this paper.

1. The number of stages and number of machines at each stage are known in advance. The number of jobs and their processing times are also known in advance.
2. Each machine can process only one job at a time. Pre-emption is not allowed.
3. All the machines are available for the entire period of scheduling, and there are no machine breakdowns.
4. The objective is to minimize the total tardiness. The total tardiness is defined as:

$$\text{Total Tardiness} = \sum_{i=1}^{n} max(0, C_i - d_i)$$

where C_i is the completion time of job i, d_i is the due date of job i, and n is the number of jobs.

3.1. Distinguishing constraints

3.1.1. Nighttime work

Work teams can be divided into three subteams: two daytime teams and one nighttime team, as in Fig. 2. In a transformer production plant, a dividable work team generally has two workbenches to process their Stage 1 operations, i.e., each daytime

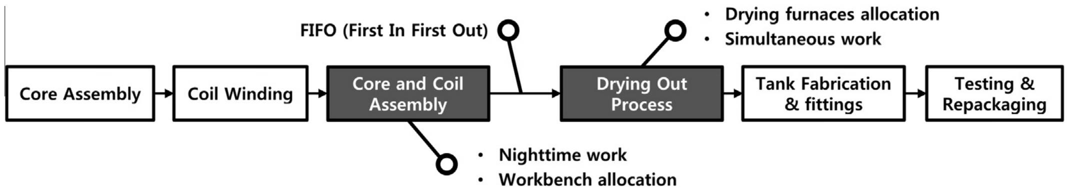

Fig. 1. Transformer manufacturing process.

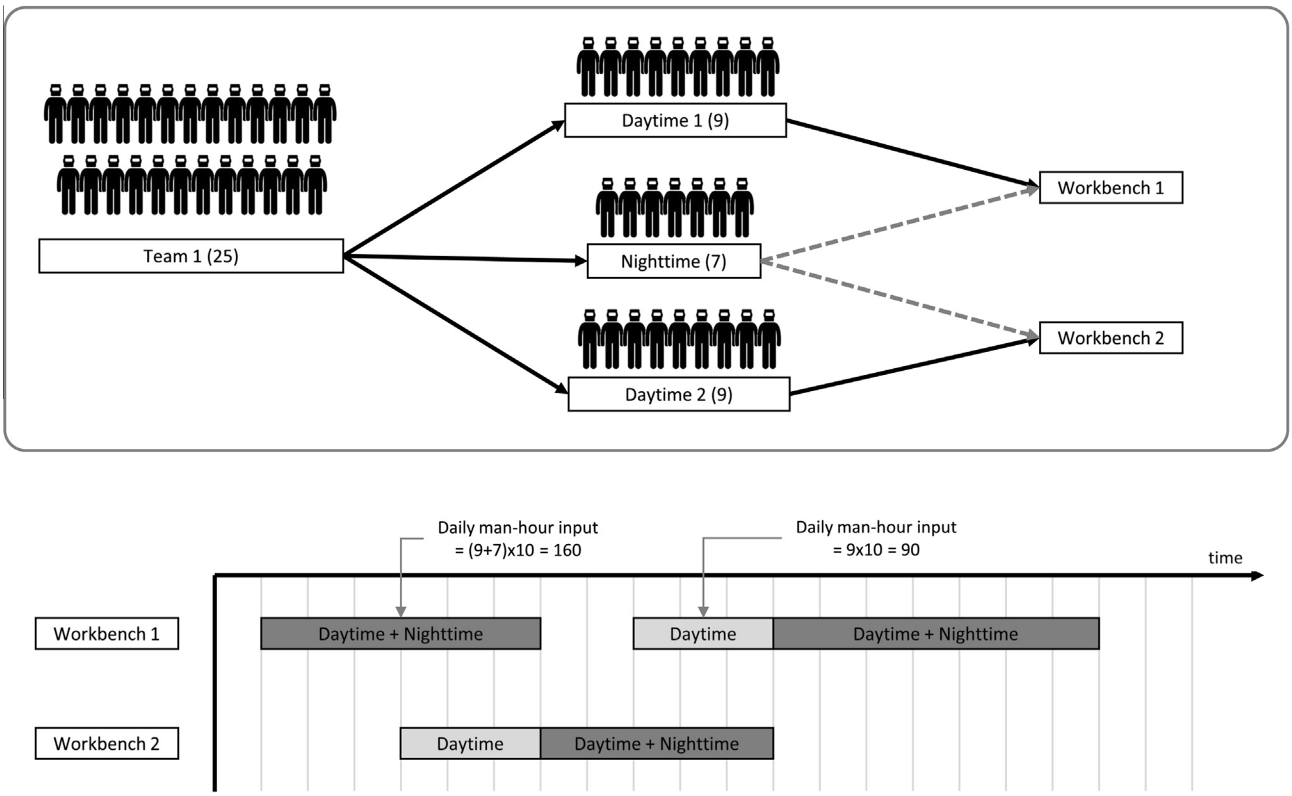

Fig. 2. Example of nighttime work.

team has their own workbench. However, the nighttime team can be assigned to either workbench. In the transformer production process, each nighttime team is assigned to the first available workbench. The allocation of teams to workbenches is important to ensure orders are completed on time. Process times can vary according to the assigned daytime/nighttime teams, because the processing time of a transformer is calculated based on production man-hours. This nighttime work system allows a flexible response to meet the due date by allocating workers to imminent jobs. This flexible allocation system is very important in the field because of the huge penalty cost of tardy orders and the problems of an unbalanced workload.

3.1.2. Simultaneous work

In the transformer production process, workers should determine an available drying furnace by measuring the length, width, and height of the transformer. Some transformers in a waiting queue are small enough to fit into a particular machine, so some machines can process two orders simultaneously, as shown in Fig. 3. In this case, the orders should be processed simultaneously to increase the utilization of drying furnaces and shorten the total process time. Furthermore, this will decrease the overall cost of electricity and labor, and would allow for processing more orders.

3.2. An illustrative example

We now present a small example problem to illustrate the concepts described in Fig. 4. The first stage includes a nighttime work constraint, and the second stage considers simultaneous working.

Stage 1 consists of three parallel workbenches and two work teams. Every work team must be assigned a workbench to process an order at Stage 1. Team 1 can be divided into Daytime 1, Daytime 2, and Nighttime teams. Team 2 cannot be divided. Table 1 lists the machines' order availability conditions and Table 2 shows the due date, man-hours required at Stage 1, and the release time of each order. Table 3 represents the worker allocation of each team.

Stage 2 is composed of two parallel machines. These can process two orders simultaneously, so long as they have the same voltage and capacity, and are small enough to fit into the machine. From Table 1, Orders 1 and 2 can be processed as a simultaneous work by Furnaces 1 and 2. The simultaneous work availability is expressed as ◎ in Table 1. Table 2 shows the process time of each order in Stage 2.

As regards the total tardiness, the second schedule, which allows simultaneous work in Stage 2, is better than the first schedule (see Figs. 5 and 6). Furthermore, the second schedule can decrease the operating cost of furnace 2 because two different orders are processed simultaneously. Thus, we should consider

Please cite this article in press as: Jun, S., & Park, J. A hybrid genetic algorithm for the hybrid flow shop scheduling problem with nighttime work and simultaneous work constraints: A case study from the transformer industry. *Expert Systems with Applications* (2015), http://dx.doi.org/10.1016/j.eswa.2015.03.012

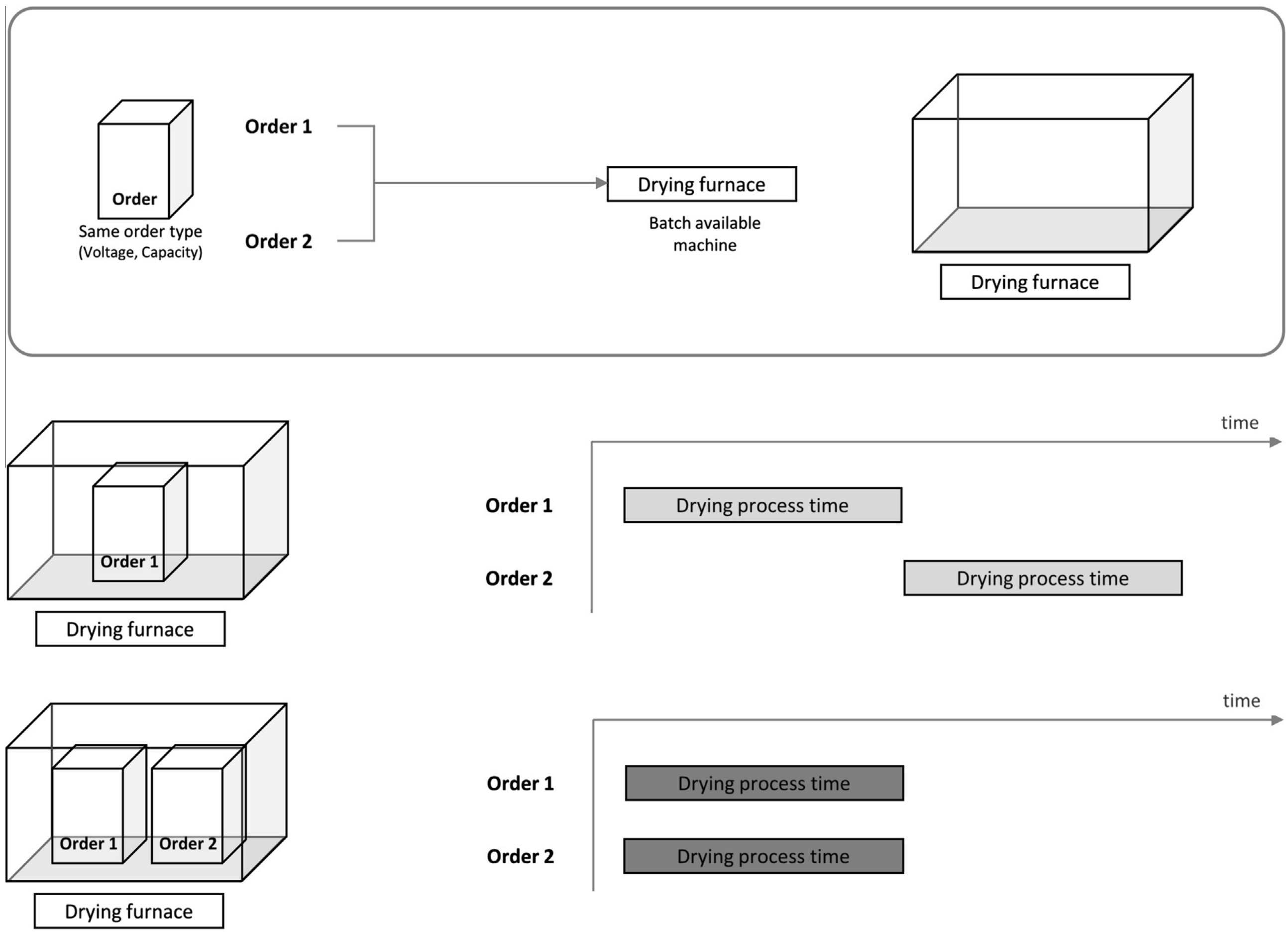

Fig. 3. Example of simultaneous work.

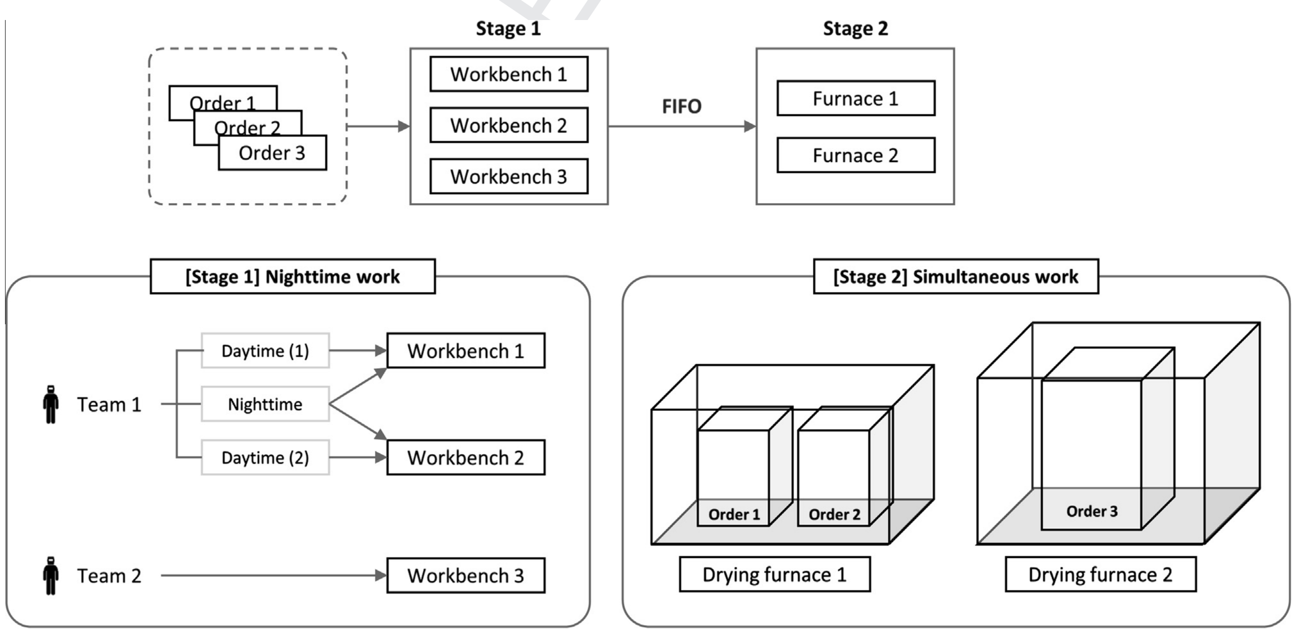

Fig. 4. Illustration of the hybrid flow-shop in the two-stage example problem.

Table 1
Order-machine availability condition matrix.

Order	Stage 1			Stage 2	
	Workbench 1	Workbench 2	Workbench 3	Furnace 1	Furnace 2
1	○	○	X	◎	◎
2	○	○	X	◎	◎
3	○	○	X	X	○
4	X	X	○	X	○

the precise distinguishing constraints of the problem to develop an efficient scheduling algorithm that can be employed in a real environment. The proposed heuristic algorithm will be discussed in the next section.

4. Proposed algorithm

We now present the proposed hybrid genetic algorithm (HGA) methodology. The entire HGA framework is first described, and then the detailed procedure for an HFS is explained. The proposed algorithm can be summarized as follows (see Fig. 7).

When considering the HFSP in the real world, the most important issue is to determine a list of jobs at the entry point, and allocate these jobs to the available machines. The list of jobs is determined in the GA phase and local search phase. In the HGA algorithm, we incorporate a neighborhood search (a type of local search technique) into the mutation, crossover, and selection loops of the GA. If the best solution in the population shows no improvement within N_Threshold steps, the GA phase is stopped, and we return to the local search phase. Our HGA applies the GA as a global exploration of the selected population, whereas the neighborhood search performs a local exploitation of each chromosome.

The actual allocation of these jobs to available machines is conducted in the chromosome decoding phase. The detailed schedule and fitness value is calculated by the machine allocation rule. The previous three phases aim to compute the objective function, whereas the function in the decoding chromosome phase describes the real schedule.

4.1. NEH algorithm phase

The Nawaz–Enscore–Ham (NEH) heuristic gives the optimal solution to the permutation flow shop scheduling problem with the makespan minimization (Ruiz & Maroto, 2005). If due dates are taken into consideration, there are several ways of sorting the jobs. If jobs are sorted according to the earliest due date (EDD), this method is known as NEH_{edd} (Vallada, Ruiz, & Minella, 2008). In this phase, the initial solution for the next GA is generated by NEH_{edd}.

NEH algorithm

Step 1: Order the job list by non-increasing due date.
Step 2: Take the first two jobs, and schedule them so as to minimize the total tardiness.
Step 3: For $k = 3$ to n, do Step 4.
Step 4: Insert the kth job into the schedule so as to minimize the partial total tardiness among the k possible values.

Table 2
Order information matrix.

Order	Release time	Due date	Required man-hour at stage 1	Process time at stage 2
1	4	15	580	5
2	3	15	580	5
3	5	18	720	6
4	7	24	1400	7

4.2. Genetic algorithm phase

4.2.1. Chromosome representation

A chromosome denotes the sequence of jobs to be considered for scheduling in the first stage. This job-permutation-based representation has been widely applied in the literature for scheduling problems.

4.2.2. Initial population

Each initial chromosome is randomly generated from the mutation and crossover operators based on five dispatching rules (EDD, Slack, Critical Ratio, COVERT and MDD). These rules are commonly used in practice and as the initial sequence of heuristic algorithms (Tari & Olfat, 2013). The job sequence is determined according to the non-decreasing order of each rule's index.

Mutation and crossover operations can maintain diversity in a population, and allow the hybrid algorithm to avoid local minima by preventing solutions from becoming too similar.

4.2.3. Crossover

A two-point crossover method is applied in the proposed algorithm, because this ensures that at least three genes are swapped between each pair of chromosomes (Korytkowski, Wiśniewski, & Rymaszewski, 2013). A brief example is illustrated in Fig. 8.

4.2.4. Mutation

We use the swap mutation method in the proposed algorithm, because this produces more variations than other mutation operators (Feng, Lu, & Li, 2009). In swap mutation, two genes are selected at random and their positions exchanged. An example of how to implement swap mutation is depicted in Fig. 9.

4.3. Local search phase

The main benefit of hybridizing GAs with a local search algorithm is the improvement in convergence to local optima. The local search is also applied to elite solutions inherited from previous populations. In this phase, the applied local search procedure can be written as follows:

Local search procedure

Step 1: Specify a seed solution s.
Step 2: Generate a neighborhood set N. This is obtained from s by interchanging all adjacent pairs of jobs.
Step 3: Select a schedule n in the neighborhood set N generated by the seed solution s, and compute its fitness value.
Step 4: If all neighborhood solutions of s have been already examined, check the neighborhood solution with the minimum fitness value and improvement ratio. If there is no neighborhood solution that improves the overall solution, terminate this procedure. Otherwise, replace the seed solution with the neighborhood solution with the minimum fitness value and return to Step 3.

Table 3
Worker allocation of Teams 1 and 2 at Stage 1.

	Team 1			Team 2
	Daytime (1)	Daytime (2)	Nighttime	Daytime Only
	Workbench 1	Workbench 2	Workbench 1, 2	Workbench 3
Number of workers	10	10	8	14
Total		28		14

Please cite this article in press as: Jun, S., & Park, J. A hybrid genetic algorithm for the hybrid flow shop scheduling problem with nighttime work and simultaneous work constraints: A case study from the transformer industry. *Expert Systems with Applications* (2015), http://dx.doi.org/10.1016/j.eswa.2015.03.012

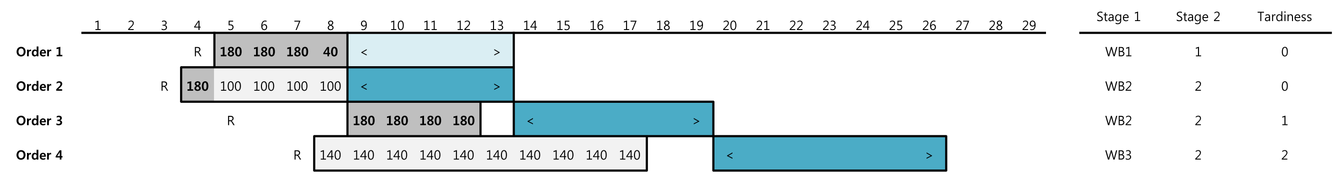

Fig. 5. First possible schedule for the example problem.

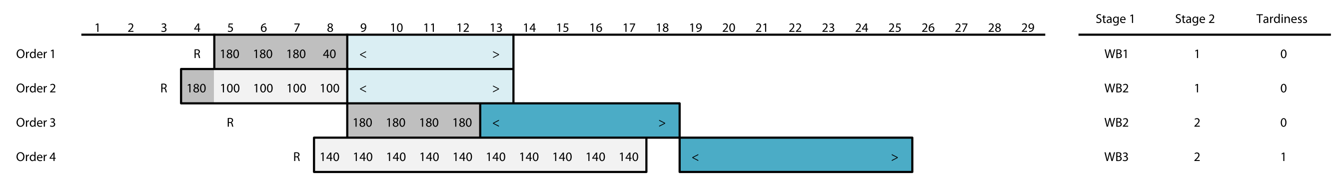

Fig. 6. Second possible schedule for the example problem.

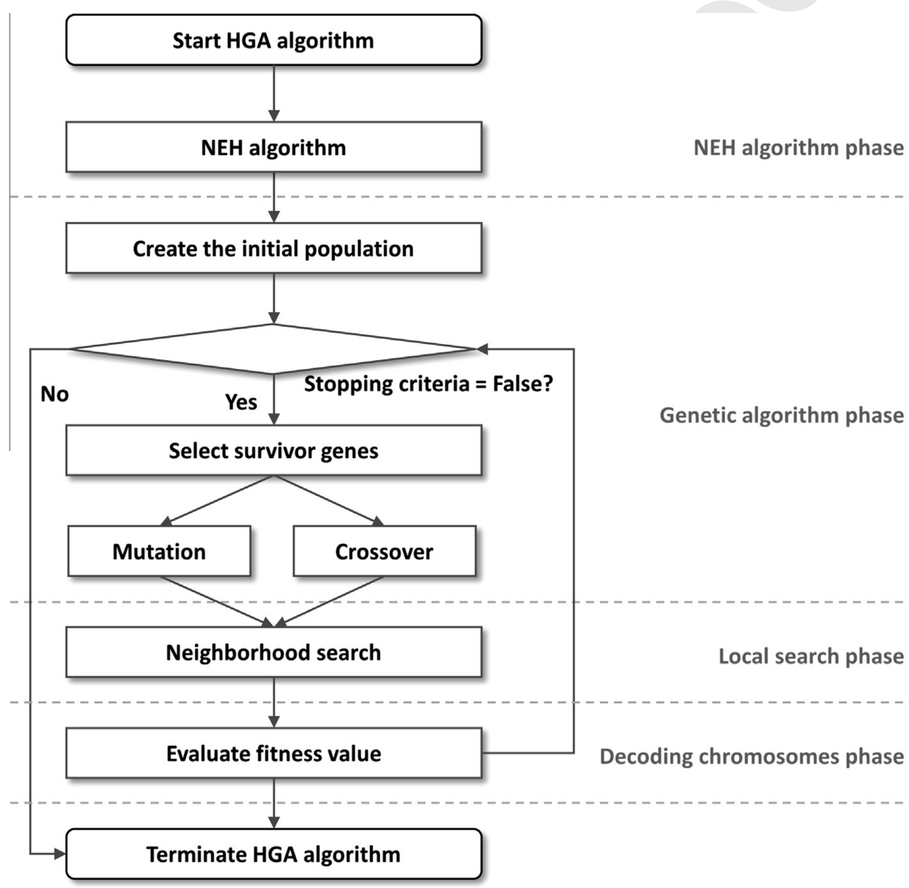

Fig. 7. Overall HGA framework.

4.4. Chromosome decoding phase

Based on the list of jobs at the entry point, the detailed job schedule, with start and finish times, is determined by the machine allocation rule.

4.4.1. Machine allocation rule

The proposed algorithm generates a sequence of jobs, but gives no information on the allocation of jobs to machines. Therefore, a machine allocation rule is needed to generate the detailed schedule from a chromosome representation. In this paper, we propose a machine score rule.

If the machines are not unrelated, i.e., jobs have processing times that depend on the machine used, each job should be assigned to the machine that will complete its process soonest. In this sense, the machine score rule tends to minimize the total tardiness and the number of tardy jobs, especially for an HFS with nighttime work and simultaneous work (Jun & Park, 2013). The machine score index is calculated as follows:

$$Machine\ Score_{k,s} = \alpha \times Uniqueness_{k,s} + (1-\alpha) \times Usefulness_{k,s}$$

$$Uniqueness_{k,s} = \frac{\sum_t (a_{k,s,t} \times \prod_{l \neq k}(1 - a_{l,s,t}))}{\sum_t a_{k,s,t}}$$

$$Usefulness_{k,s} = \frac{v_{k,s}}{max_g(v_{g,s})}$$

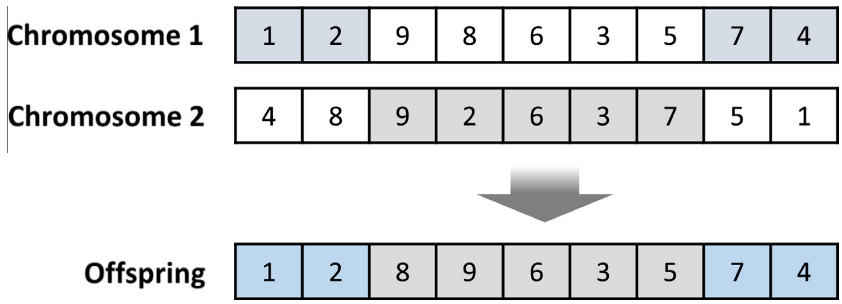

Fig. 8. Two-point crossover operator.

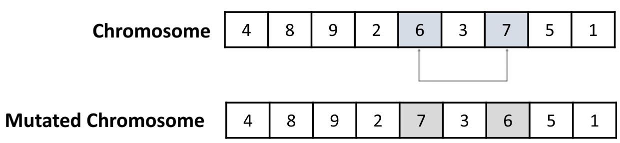

Fig. 9. Swap mutation operator.

In this rule, available machines are allocated according to the non-increasing order of the machine score index. At first, identical machines at the same stage are bracketed into a group. The machine score index is composed of the weighted sum of group k's uniqueness and usefulness in stage s. α is the uniqueness weight of a machine group, and t denotes the order type in terms of voltage and capacity. $v_{k,s}$ is the relative speed of machine group k in stage s. If machine group k can process order type t in stage s, $a_{k,s,t} = 1$; otherwise, $a_{k,s,t} = 0$.

4.4.2. Fitness function

The evaluation of a chromosome is determined by the fitness function. The fitness function plays an important role in selecting survivor genes for the next generation. In this study, the fitness function is defined as the total tardiness of the schedule.

4.4.3. Selection

The next survivor chromosomes are selected using the roulette wheel method (Gen & Cheng, 1997).

4.4.4. Stopping criterion

A maximum computation time is employed as a stopping criterion.

5. Experimental results

5.1. Experiment design

In the simulation experiment, we considered 30 machines in Stage 1, and 12 machines in Stage 2. There were seven different machine types. The processing time of each job on each machine was specified based on previous production data. The due date of each job was specified as follows:

$$d_j = r_j \times DT + \text{random}\left[LB(Voltage_j, Capacity_j), UB(Voltage_j, Capacity_j)\right]$$

where DT denotes the due date tightness parameter and r_j is the previous release time of job type j. $LB\,(Voltage_j,\ Capacity_j)$ and $UB\,(Voltage_j,\ Capacity_j)$ denote the lower and upper bounds, respectively, of job type j based on previous production data. All algorithms were coded in VB.Net and run on an Intel Core 2 Quad 2.4 GHz processor with 4 GB RAM.

Several pilot experimental tests were conducted to choose the best parameter values for the GA. The population size was set to 100, and the crossover rate and mutation ratio were set to 0.4 and 0.3. The test problems (N = 100, 250, 500) were generated from the distribution of previous orders to allow the practical evaluation of the algorithms. The detailed experimental parameters are summarized in Table 4.

The relative deviation index (RDI) is applied to compare the results (Akhshabi, Tavakkoli-Moghaddam, & Rahnamay-Roodposhti, 2014). RPD is calculated as follows:

$$RDI_k = \frac{F_k - Min_k}{Max_k - Min_k} \times 100$$

F_k is the total tardiness value obtained for the kth experiment, and Max_k and Min_k are the best and worst solutions in the kth experiment.

5.2. Experimental results

To evaluate the performance of probabilistic search methods, we repeated the simulation experiments 30 times. The average performance and relative deviation index of all algorithms over the 30 trials is compared in Tables 5–7. The average CPU time of each algorithm is also shown in Tables 5–7. The CPU time of HGA and a simple GA are the same, because these two algorithms used the same stopping condition. The performance of all algorithms is analyzed in terms of RDI using a one-way ANOVA and 95% confidence interval plots.

To compare the average performance of the algorithms statistically, we conducted a one-way ANOVA analysis of the RDIs (Rabiee, Rad, Mazinani, & Shafaei, 2014). The results of this ANOVA analysis for small-, medium-, and large-size problems are presented in Table 8. The results show that the P-value is close to zero in each case. Thus, the RDI of at least one algorithm is significantly different.

Additionally, to evaluate the significance of the results, the 95% confidence intervals are shown in Figs. 10–12. These figures indicate that there is a significant difference between the average

Table 4
Detailed experimental parameters.

Parameters	Small size	Medium size	Large size
Number of jobs	100	250	500
Number of machines	30 (Stage 1), 12 (Stage 2)		
Number of experiments	30	30	30
Maximum CPU time (s)	2000	6,000	50,000
Due date tightness	0.3	1.0	1.0
Population	100		
Genetic algorithm parameters	Mutation rate (0.3), Crossover rate (0.4)		
N_Threshold	8		

Table 5
Result summary (Small size).

Algorithm	Average objective function	Average RDI	Average computation time (s)
Genetic algorithm	1410.3	30.71377	2000
Proposed algorithm	1396.533	0.10929	2000
NEH algorithm	1396.8	1.039641	58.43179
EDD	1431.933	76.09123	0.304879
SLK	1428.933	68.64578	0.249955
CR	1428.6	65.62185	0.293273
COVERT	1427.033	68.87144	0.481561
MDD	1430.333	73.43126	0.503761

Table 6
Result summary (Medium size).

Algorithm	Average objective function	Average RDI	Average computation time (s)
Genetic algorithm	2076.7	4.968086	6000
Proposed algorithm	2073.133	2.732215	6000
NEH algorithm	2125.167	32.21516	3027.462
EDD	2204.433	76.75813	0.557202
SLK	2206.367	77.64346	0.827727
CR	2229.933	88.46668	0.611689
COVERT	2177.133	62.98672	1.018642
MDD	2200	74.86941	1.160462

Table 7
Result summary (Large size).

Algorithm	Average objective function	Average RDI	Average computation time (s)
Genetic algorithm	28120.6	37.55309	50000
Proposed algorithm	27637.8	0	50000
NEH algorithm	28603.3	75.0182	42333.95
EDD	28820.6	92.26373	13.78962
SLK	28895.73	98.11295	16.4751
CR	28780.73	89.57702	13.19096
COVERT	28201	44.208	19.51381
MDD	28302.63	51.76036	14.16533

Table 8
ANOVA table for RDI in small, medium, and large size problems.

Size	Source	df	SS	MS	F	*P* value
Small	Factor	6	200114.97	33352.495	65.132	0.000
	Error	203	103950.901	512.073		
	Total	209	304065.871			
Medium	Factor	6	231130.823	38521.804	128.3	0.000
	Error	203	60950.116	300.247		
	Total	209	292080.939			
Large	Factor	6	237214.294	39535.716	660.103	0.000
	Error	203	12158.334	59.893		
	Total	209	249372.628			

performance of the algorithms in terms of RDI. It is clear that the proposed algorithm outperforms the others, because the HGA produces a much smaller average RDI value, and the confidence interval of the proposed algorithm rarely overlaps with those of the other algorithms. On the contrary, NEH and SGA obtained inferior solutions to our algorithm for the large- and small-sized problems, respectively. Hence, the proposed HGA is superior to the other algorithms under the same parameters and computation time for all problem sizes.

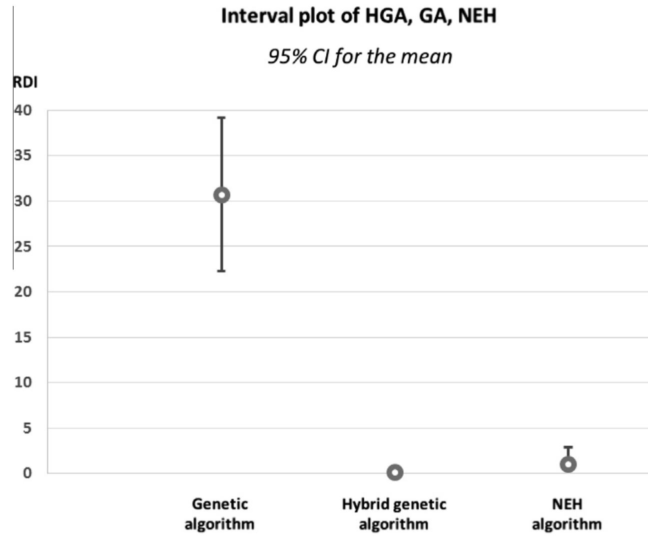

Fig. 10. Confidence interval 95% for RDI of algorithms (Small size).

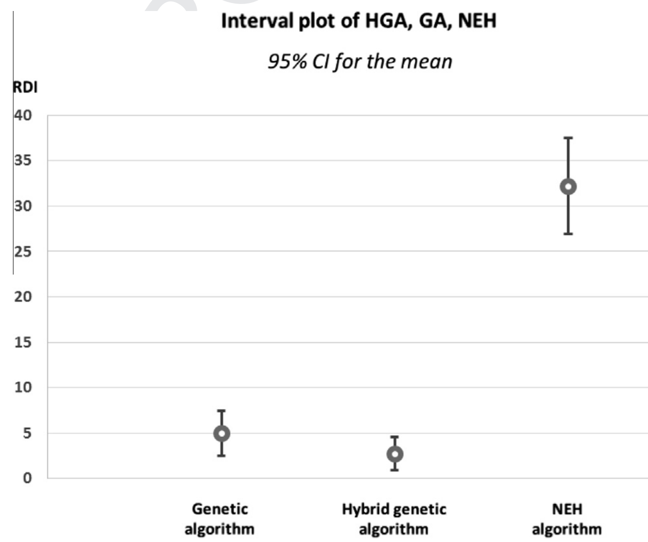

Fig. 11. Confidence interval 95% for RDI of algorithms (Medium size).

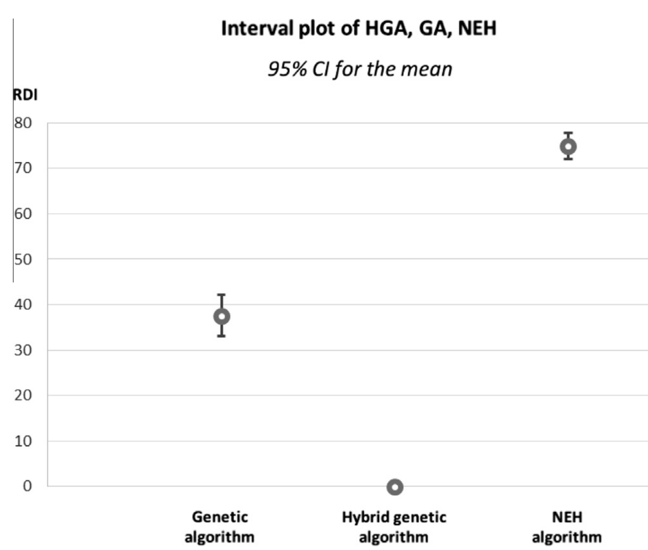

Fig. 12. Confidence interval 95% for RDI of algorithms (Large size).

Please cite this article in press as: Jun, S., & Park, J. A hybrid genetic algorithm for the hybrid flow shop scheduling problem with nighttime work and simultaneous work constraints: A case study from the transformer industry. *Expert Systems with Applications* (2015), http://dx.doi.org/10.1016/j.eswa.2015.03.012

6. Conclusion and future work

In this paper, we defined the HFSP with distinguishing real-world constraints of nighttime work and simultaneous work. To improve the exploration ability of the proposed algorithm, we incorporate heuristic algorithms to explore possible solutions effectively. We also developed a machine allocation rule that selects and assigns operations to machines. The objective of the proposed algorithm is to minimize the total tardiness performance measure. We presented a case study of a transformer production factory to evaluate the performance of the proposed algorithm. Simulation results demonstrate that the proposed algorithm outperforms the NEH algorithm, a simple GA, and various dispatching rules in terms of the total tardiness and robustness.

The main contribution of this paper is the development of a hybrid GA to effectively solve the HFSP. Because of the increased complexity of a hybrid flow shop with industry-specific constraints, an efficient and robust algorithm is of particular importance. In this context, the HFSP with nighttime work and simultaneous work constraints has a number of applications in various manufacturing and service systems. Furthermore, the proposed approach could be extended to other industries that employ skilled craft workers, machine operators, and assemblers.

Nevertheless, the proposed approach has some limitations. First, the initial solution plays an important role in determining the eventual outcome, because the performance of the proposed algorithm depends on the quality of the initial population. Additionally, mathematical algorithms for the HFSP are not considered in this paper. An implicit enumeration technique such as the branch-and-bound algorithm, integer programming, and a lower bound can be used to find the optimal solution.

Future work could take one of several directions. First, the proposed algorithm could be adapted to other scheduling problems and environments. For example, the proposed algorithm could handle a multi-objective optimization problem, such as an objective function that combines the total tardiness with the makespan. Consideration of other constraints, such as the learning ability of workers or the balance of the workload, are also interesting and worthy of future investigation.

Furthermore, data mining techniques such as support vector machines or decision tree algorithms could be applied to determine more sophisticated parameter values by analyzing the characteristics of resources, jobs, and orders. Further research based on data mining techniques and constraints may offer opportunities to develop an automated scheduling system to solve more complex scheduling problems as an alternative to a well-trained 'scheduler.'

Acknowledgments

This paper was supported in part by the Automation and Systems Research Institute at Seoul National University. We also would like to acknowledge the support of Hyundai Heavy Industry.

References

Akhshabi, M., Tavakkoli-Moghaddam, R., & Rahnamay-Roodposhti, F. (2014). A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time. *The International Journal of Advanced Manufacturing Technology, 70*(5–8), 1181–1188.

Arthanari, T., & Ramamurthy, K. (1971). An extension of two machines sequencing problem. *Opsearch, 8*(1), 10–22.

Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economics, 64*(1), 101–111.

Bożejko, W., Pempera, J., & Smutnicki, C. (2013). Parallel tabu search algorithm for the hybrid flow shop problem. *Computers & Industrial Engineering, 65*(3), 466–474.

Costa, A., Cappadonna, F. A., & Fichera, S. (2014). A novel genetic algorithm for the hybrid flow shop scheduling with parallel batching and eligibility constraints. *The International Journal of Advanced Manufacturing Technology, 75*(5–8), 833–847.

Engin, O., Ceran, G., & Yilmaz, M. K. (2011). An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. *Applied Soft Computing, 11*(3), 3056–3065.

Fattahi, P., Hosseini, S. M. H., Jolai, F., & Tavakkoli-Moghaddam, R. (2014). A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations. *Applied Mathematical Modelling, 38*(1), 119–134.

Feng, H., Lu, S., & Li, X. (2009). Genetic algorithm for hybrid flow-shop scheduling with parallel batch processors. Paper presented at the Information Engineering, 2009. ICIE'09. WASE International Conference on Information Engineering.

Gen, M., & Cheng, R. (1997). Genetic algorithms and engineering design. Wiley.

Gupta, J. N. (1988). Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society, 39*(4), 359–364.

Gupta, J., Hariri, A., & Potts, C. (1997). Scheduling a two-stage hybrid flow shop with parallel machines at the first stage. *Annals of Operations Research, 69*, 171–191.

Holland, J. H. (1975). Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.

Jun, S., & Park, J. (2013). The development of a heuristic algorithm for the transformer production scheduling. The Journal of Korea Society for Simulation Autumn Conference, 6–8. <http://simul.hosting.bizfree.kr/files/full_kss_2013autumn.pdf> (accessed April 10, 2014).

Kochhar, S., & Morris, R. J. (1987). Heuristic methods for flexible flow line scheduling. *Journal of Manufacturing Systems, 6*(4), 299–314.

Korytkowski, P., Wiśniewski, T., & Rymaszewski, S. (2013). An evolutionary simulation-based optimization approach for dispatching scheduling. *Simulation Modelling Practice and Theory, 35*, 69–85.

Li, J. Q., Pan, Q. K., & Wang, F. T. (2014). A hybrid variable neighborhood search for solving the hybrid flow shop scheduling problem. *Applied Soft Computing, 24*, 63–77.

Liao, C.-J., Tjandradjaja, E., & Chung, T.-P. (2012). An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem. *Applied Soft Computing, 12*(6), 1755–1764.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: a survey. *Computers & Industrial Engineering, 37*(1), 57–61.

Lopez, P., & Roubellat, F. (2008). Production scheduling. Wiley.

Moradinasab, N., Shafaei, R., Rabiee, M., & Ramezani, P. (2013). No-wait two stage hybrid flow shop scheduling with genetic and adaptive imperialist competitive algorithms. *Journal of Experimental & Theoretical Artificial Intelligence, 25*(2), 207–225.

Naderi, B., Gohari, S., & Yazdani, M. (2014). Hybrid flexible flowshop problems: Models and solution methods. *Applied Mathematical Modelling, 38*(24), 5767–5780.

Pinedo, M. (2008). Scheduling: Theory, algorithms, and systems (3rd ed.). Springer.

Rabiee, M., Rad, R. S., Mazinani, M., & Shafaei, R. (2014). An intelligent hybrid meta-heuristic for solving a case of no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machines. *The International Journal of Advanced Manufacturing Technology, 71*(5–8), 1229–1245.

Ribas, I., Leisten, R., & Framiñan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research, 37*(8), 1439–1454.

Rossi, A., Pandolfi, A., & Lanzetta, M. (2014). Dynamic set-up rules for hybrid flow shop scheduling with parallel batching machines. *International Journal of Production Research, 52*(13), 3842–3857.

Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research, 165*(2), 479–494.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*(1), 1–18.

Sun, L., & Yu, S. (2015). Scheduling a real-world hybrid flow shop with variable processing times using Lagrangian relaxation. *The International Journal of Advanced Manufacturing Technology*, 1–10.

Tari, F. G., & Olfat, L. (2013). Heuristic rules for tardiness problem in flow shop with intermediate due dates. *The International Journal of Advanced Manufacturing Technology*, 1–13.

Tsujimura, Y., & Gen, M. (1999). Parts loading scheduling in a flexible forging machine using an advanced genetic algorithm. *Journal of Intelligent Manufacturing, 10*(2), 149–159.

Vallada, E., Ruiz, R., & Minella, G. (2008). Minimising total tardiness in the *m*-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers & Operations Research, 35*(4), 1350–1373.

Please cite this article in press as: Jun, S., & Park, J. A hybrid genetic algorithm for the hybrid flow shop scheduling problem with nighttime work and simultaneous work constraints: A case study from the transformer industry. *Expert Systems with Applications* (2015), http://dx.doi.org/10.1016/j.eswa.2015.03.012