Efficient Heuristic Solution Methodologies for Scheduling Batch Processor with Incompatible Job-Families, Non-identical Job-Sizes and Non-identical Job-Dimensions

M. Mathirajan[1(✉)] and M. Ramasubramanian[2]

[1] Department of Management Studies, Indian Institute of Science, Bangalore 560012, India
iiscmathi@gmail.com
[2] Loyola Institute of Business Administration, Chennai 600034, India
rams.m@liba.edu

Abstract. Efficient scheduling of heat-treatment furnace (HTF), a batch processor (BP), is very important to meet both throughput benefits as well as the committed due date to the customer, as the heat-treatment operations require very long processing time in the entire steel casting manufacturing process and accounts for large part of the total casting processing time required. In the recent time, there are good number of studies reported in the literature related to scheduling of BP associated with many discrete parts manufacturing. However, still there is very scant treatment has been given in scheduling of HTF problem, which has one of the unique job-characteristic: non-identical job-dimensions. This characteristic differentiates most of the other BP problems reported in the literature. Thus, this study considers a scheduling HTF, close to real-life problem characteristics, and proposes efficient heuristic solution methodologies.

Keywords: Heat-treatment furnace · Non-identical job-dimensions · Lower bound · Greedy heuristic algorithm · Genetic algorithm

1 Introduction

This study is motivated from steel-casting industries, particularly scheduling of heat-treatment furnaces (HTF), a batch processor (BP). Heat-treatment operation is one of the most important operations as it determines the final properties that enable components to perform under demanding service conditions such as large mechanical load, high temperature and anti-corrosive processing. The jobs (castings) in work-in-process (WIP) inventory, available in front of a HTF, vary widely in sizes and dimensions. Furthermore, jobs are primarily classified into several job families based on the alloy type. These job families are incompatible since the temperature requirement for low alloy and high alloy vary for similar type of heat-treatment operations required. These job families are further classified into various sub-families based on the type of heat treatment operations they required. These sub-families are also incompatible as each of these sub-families requires different combination of heat-treatment operations.

Published by Springer Nature Switzerland AG 2019
F. Ameri et al. (Eds.): APMS 2019, IFIP AICT 567, pp. 212–222, 2019.
https://doi.org/10.1007/978-3-030-29996-5_25

The widely varying job sizes, job dimensions and multiple incompatible job family characteristics, which differentiate other batch processing operations across many other industries, create complexity in scheduling HTF.

It is also important to note that the heat-treatment operation requires lengthiest processing time (say between 6 h to 48 h) taking up a large part of total processing time. Because of these, the heat-treatment operation is a major bottleneck operation in the entire steel casting process. Considering the complexity and bottleneck nature of the heat treatment process the efficient scheduling of this operation to maximize throughput, in order to enhance productivity of the entire steel casting manufacturing process, is of importance to the firms. The concerns of the management in increasing the throughput of the bottleneck machine, thereby increasing productivity, motivated us to adopt the scheduling objective of minimizing makespan, as this objective can be a surrogate measure for maximizing throughput of the entire steel casting manufacturing process.

Accordingly, this study specifically focuses on developing a few simple Greedy Heuristic Algorithms and meta heuristic: Genetic Algorithm for the research problem of minimizing makespan (Cmax) on a batch processor (BP) with multiple incompatible job families (MIJF), where all jobs of the same family have identical processing times and jobs of different families cannot be processed together, non-identical job sizes (NIJS), non-identical job dimensions (NIJD), and non-agreeable release times and due dates (NARD).

The rest of this paper is organized as follows. Closely related literature review is presented in Sect. 2. In Sect. 3 several Greedy Heuristic Algorithms (GHA) and meta-heuristic algorithm: Genetic Algorithm (GA) are proposed to solve the problem. A lower bound procedure, which is considered as benchmark procedure, proposed for the research problem is discussed in Sect. 4. The computational experiment for evaluating the proposed heuristic algorithms is discussed in Sect. 5. In Sect. 6 performance evaluation of the proposed heuristic algorithms is analysed and presented. Finally, conclusions and future research areas are discussed in Sect. 6.

2 Related Literature Review

Scheduling BP in discrete parts manufacturing has been widely studied in the literature [10, 12, 13], due to its intensive varied applications in the real-world industry. This research is motivated by heat-treatment operations, which is performed in heat-treatment furnace, in steel casting industries. One of the job's (casting's) characteristics on non-identical job dimension, which is not applicable in most of the applications of BP, is uniquely differentiate almost all the applications of BP listed in the literature. Furthermore, the non-identical job size and non-identical job dimension characteristics together are not addressed [1, 3, 4, 11].

[14, 15] considered the scheduling of an HTF with a single job-family. Subsequently, [16] extended their earlier studies, by considering the important additional real-life characteristics on multiple and incompatible job families and proposed a MILP model and demonstrated its computational intractability. Due to the computational intractability of scheduling of BP with incompatible job-families problems, recently, researchers have started to propose meta heuristic algorithms, in addition to simple greedy heuristic algorithms for scheduling of BP [5, 6, 17]. In the similar line, this

study is proposing (a) a lower bound procedure and (b) a few greedy heuristics and meta heuristic: Genetic Algorithms for obtaining efficient HTF-schedule for the problem considered in this study.

3 Solution Methodologies

The development of nine variants of simple greedy heuristic algorithms (GHA) and meta heuristic: Genetic Algorithm (GA) to address the research problem considered in this study are discussed in this section. In the proposed GHA, first we sort the jobs by a criterion and then construct a set of batches (this process is called as Sort by Criteria and Construct Batch) by picking jobs from the sorted list. Finally, the set of batches constructed will be sequenced. Accordingly, the step-by-step procedure of the proposed GHA is as follows:

3.1 Greedy Heuristic Algorithm (GHA)

Step 1: Group the jobs and form different family based on the family identification data. For each family formed, sort the list of jobs, which are to be processed in HTF, using a specific sorting criterion.
Step 2: Select family F and b=1
Step 3: Select a set of feasible jobs, which are yet to be considered in forming a batch as well as satisfying capacity restrictions on size, dimension and release time constraint, from the selected job family, sequentially from the sorted list of jobs and construct batch 'b'. Also compute EBAT(b, F) as follows:

$$EBAT(b,F) = \text{maximum } \{\text{release time of all jobs in the batch 'b', of family 'F'}\}$$

Step 4: If the job-list is not empty for the selected family 'F', then form next batch from the selected family. That is, b=b+1.
Step 5: Repeat Steps 3 to 4 until the job-list for the selected family 'F' is empty. Otherwise, GOTO Step 6.
Step 6: F = F+1. If family 'F' is last family for the considered problem, then GOTO Step 7; Otherwise GOTO Step 2.
Step 7: Sort the set of batches formed across the family 'F' with respect to minimum-to-maximum criterion considering the value of EBAT(b,F). As per the sorting order of the list of batches, compute the completion time of batch b of family F (= C (b,F)), as follows:

$$C(b,F) = EBAT(b,F) + \text{Processing Time of Family 'F'}$$

Step 8: Calculate the makespan for the numerical problem using the following equation:

C_{max} = *Max {C(b, F), b = 1, 2,..... maximum number of batches formed for family F for the given problem, and F = 1,2,, ... maximum number of families grouped for the given problem}*

The proposed GHA, presented here can be varied by introducing the nine different sorting criterions, as given in Table 1, and with that we have nine proposed variants of GHA for scheduling of HTF, considered in this study to minimize makespan.

Table 1. Sorting criterion considered for the proposed variants of GHA

Code	Sorting criterion	Proposed variants of GHA
SL	**S**ort by **L**ength and construct **B**atches	SLB
SW	**S**ort by **W**idth and construct **B**atches	SWB
SH	**S**ort by **H**eight and construct **B**atches	SHB
SV	**S**ort by **V**olume and construct **B**atches	SVB
SS	**S**ort by **S**ize and construct **B**atches	SSB
SD	**S**ort by **D**ue date and construct **B**atches	SDB
SVD	**S**ort by (**V**olume/**D**ue date) and construct **B**atches	SVDB
SSD	**S**ort by (**S**ize/**D**ue date) and construct **B**atches	SSDB
SR	**S**ort by **R**elease time and construct **B**atches	SRB

3.2 Genetic Algorithm (GA)

GA has been applied to scheduling BP, related to semiconductor industry [2, 8, 18]. However, to the best of our knowledge, there are no studies considering *GA* for the problem configurations defined in this study. Accordingly, this study considers GA and use random key based representation for *GA*. Further, in order to fix value for GA-parameters *(*such as: number of generations, population size, crossover percentage, crossover probability, migration percentage and mutation), which are problem specific, we conduct a computational experiment. Based on the computational experiment conducted, the values of the *GA*-parameters are fixed as in Table 2.

Table 2. GA-Parameter and its value for GA-Implementation

GA-parameter	Values of the parameter
Number of generations	200
Population size	25
Migration percentage	20%
Crossover percentage	50%
Mutation percentage	30%
Crossover probability	0.6

Considering the value of each of the *GA*-parameters given in Table 2, the step-by-step procedure of the proposed *GA* for scheduling a BP with MIJF, NIJS and NIJD is as follows:

Step 1. Generate a population of permutation sequences with size 25. This sequence is essentially job indices numbered from 1 to n.
Step 2. Sort the first sequence based on the job length using criterion SL mentioned in Table 1 with the maximum length job being first and the minimum length job being last.
Step 3. Repeat step 2 for the next eight sequences using criteria: SW, SH, SV, SS, SD, SVD, SSD, SR mentioned in Table 1.
Step 4. Generate random permutation sequences for the rest of the population.
Step 5. If the number of generations is less than or equal to 200, construct and schedule batches using the GHA mentioned in Sect. 3.1 for each of the sequence. Else, go to step 16.
Step 6. Calculate the fitness value and the makespan of the schedule for each of the sequences.
Step 7. Sort the population of sequences based on the fitness value with lowest value on the top to highest on the bottom.
Step 8. Encode the permutation sequences using random keys.
Step 9. Keep top 20% of the population for migration to next generation.
Step 10. Select two chromosomes randomly from the population and apply crossover operation.
Step 11. Repeat step 10 to generate a total of 50% of population sequences which are the new set of off springs.
Step 12. Replace the middle 50% of the population with the new off springs.
Step 13. Generate rest 30% of the chromosome population randomly and replace the lowest 30% of the population.
Step 14. Sort each chromosome by random keys which are generated using crossover and mutation in descending order.
Step 15. Decode the chromosomes to get a new set of permutation sequences. Go to step 5.
Step 16. Output the best makespan value from the last generation.

Each of the proposed heuristic algorithms, presented here, is implemented in Turbo C++ language on a system with 1 GB RAM, 2.4 Ghz processor with Windows XP operating system.

4 Computational Experiments

The following proposed (a) benchmark procedure, (b) experimental design, and (c) performance measure are considered for empirical evaluation of the proposed heuristic algorithms.

4.1 Benchmark Procedure: Proposed Lower Bound (LB) Procedure

In BP literature, researchers have proposed LB procedures for evaluating the heuristic algorithms (Example: [7]). However, these studies do not consider NIJS and NIJD characteristics explicitly while developing LB procedure in scheduling a BP. Further, there are LB procedures proposed for three-dimensional bin packing problem [9] which consider only NIJD characteristic for obtaining LB. To the best of our knowledge, there are no studies which consider both NIJD and NIJS to obtain LB while scheduling a BP. This motivated us to develop an LB procedure for the research problem, considered in this study. Since our research problem is an extension of the three-dimensional bin packing problem, the LB procedure proposed by [9] is used with appropriate modification by considering both NIJD and NIJS characteristics to obtain LB on makespan. The proposed LB procedure in calculating LB on makespan for the research problem is as follows:

Step 1: Index the jobs in the non-decreasing order of the release times r_j.
Step 2: Consider all the jobs to be available at time zero.
Step 3: Now calculate $C_{max}(f, j, n)$ for jobs $j, \ldots, n$ belonging to the first job family 'f' using the following sub-steps, taking appropriate processing time for the family.
Step 3(a): Calculate LB on makespan by considering only NIJD called as C_{max}^{NIJD} using the LB procedure proposed in [9] with suitable modification.
Step 3(b): Calculate LB on makespan by considering only NIJS characteristic called as C_{max}^{NIJS},
Step 3(c): Calculate LB on makespan by considering both NIJD and NIJS as

$$LBC_{max} = max\left\{C_{max}^{NIJD}, C_{max}^{NIJS}\right\}.$$

Step 4: Repeat step 3 for other job families in the list with their appropriate processing times.
Step 5: Add all $C_{max}(f, j, n)$ and finally, add the release time of the job at head of the list with $C_{max}(f, j, n)$.
Step 6: Store the value $(r_j + C_{max}(f, j, n))$ in a separate list.
Step 7: Remove the job at the head of the list.
Step 8: Repeat steps 2–7 till the list is empty.
Step 9: Then LB for the research problem, LBC_{max} is maximum of all the values stored in the separate list. i.e. $LBC_{max} = \max_{j=1,2,\ldots,n}\left\{r_j + \sum_{f=1}^{p} Cmax(f, j, n)\right\}$;
Where, p is number of job families.

The procedure to obtain LB on makespan for scheduling a BP with MIJF, NIJS and NIJD is implemented in Turbo C++ language.

4.2 Experimental Design

Based on the problem configurations considered in this study, the parameters: Number of Jobs (n), Job size (s_i), Job Dimension ((Length (l_i), Width (w_i) and Height (h_i)), Number of Incompatible Job families (f_i), Release time (ri) and Due date (di) are required data. Accordingly, an experimental design is developed (Table 3) to generate suitable test data. The range of intervals used in each uniform distribution is based on the observation from the user industries. The proposed experimental design for generating test problems is implemented in Turbo C++ language.

Table 3. A summary of experimental design for the research problem

Problem parameters		No. of levels	Values
No. of jobs (n)		6	25, 50, 75, 100, 125, 150
Job parameters	No. of incompatible job families (f_i)	2	4, 6
	Release time (ri)	2	U[0,84], U[0,42]
	Due date (di)	1	r_i +U[168,240]
	Job size (s_i)	2	U[1, S/2], U[1, S/4]
	Job length (w_i)	2	U[1, W], U[1, W/2]
	Job height (h_i)	2	U[1, H], U[1, H/2]
	Job length (l_i)	2	U[1, L], U[1, L/2]
Number of configurations			$6 \times 2 \times 2 \times 1 \times 2 \times 2 \times 2 \times 2 = 384$
Number of instances per configuration			10
Total number of instances			3840

In addition to the input provided for the problem parameters, mentioned in Table 3, we assume that there is only one BP which has BP-Size as S = 2500 kg; and BP-Dimension as L = 2500 mm, W = 1000 mm, and H = 1250 mm. Also, we assume the processing time of the job families as (a) 13, 15, 12, 10 respectively when number of job families are 4, and (b) 13, 15, 12, 10, 22, 18 respectively when number of families are 6.

4.3 Measures of Effectiveness

The performance of the proposed heuristic algorithms is compared using the performance measure: Average Relative Percentage Deviation (ARPD), indicating the average performance of proposed heuristic algorithms. The details of the calculation of the performance measures ARPD is as follows.

Let C_H be the makespan given by the proposed heuristic algorithm "H". Let C_{LB} be the Lower Bound on C_{max} given by the proposed LB procedure. Then, the Relative Percentage Deviation (RPD) on instance '*i*' for the proposed heuristic algorithm "H" is $RPD_H(i)$ and computed as follows:

$$RPD_H(i) = \left(\frac{C_H(i) - C_{LB}(i)}{C_{LB}(i)} \right) * 100 \quad \text{(A)}$$

The average RPD (i.e, ARPD) is calculated as follows:

$$ARPD_H(p) = \frac{\sum_{i=1}^{N} RPD_H(i)}{N} \quad \text{(B)}$$

Where, $ARPD_H(p)$ = ARPD of proposed heuristic algorithm 'H' for problem configuration p over N instances of planned configuration p.

5 Performance Evaluation of the Proposed Heuristic Algorithms

To understand the performances of the proposed heuristic algorithms in comparison with the proposed LB, the randomly generated problem instances (3840 instances) is used. First, the proposed heuristic algorithms are run through each of the 3840 instances and the makespan values are recorded. Then, the LB-procedure is also run through each of these 3840 instances and the LB on makespan is obtained. With these values, for each problem instance the RPD value is computed using equation (A). Furthermore, Problem instance wise and the proposed heuristic algorithm wise, the *computed* $RPD_H(i)$ is used for calculating the scores: ARPD using equation (B). The computed ARPD score is presented in Table 4.

From Table 4, we can observe that (a) the proposed GA outperforms the other proposed nine variants of GHA, and (b) within the proposed variants of the greedy heuristic algorithms: the variant of the GHA: *SWB* performs relatively better and the variant of the GHA: *SVDB* becomes relatively the second best one.

Finally, in order to check for the influence of individual problem parameters on the performance of the proposed heuristic algorithms in comparison with LB on makespan, the computed RPD score is used for statistical test. For this purpose, a statistical test: multi-factor (***heuristic algorithm***, ***f***, ***n***, ***r***, ***s***, ***w***, ***h***, ***l***) ANOVA is used on the score: RPD. Since the normality and equal variance assumption failed for our data, we used *Mann-Whitney* non-parametric test for factors with two groups (***f, r, s, h, w, l***) and *Kruskal-Wallis*non-parametric test for factors with many groups (***heuristic algorithm***, ***n***). The results of this analysis show that the there is an influence of all problem parameters, considered in this study, on the performance of the proposed heuristic algorithms.

Table 4. Performance of the proposed heuristic algorithm w.r.t. the score: ARPD based on LB

Problem configuration	No. of instances	Proposed variants of GHA									Proposed GA
		SLB	SWB	SHB	SVB	SSB	SDB	SVDB	SSDB	SRB	
(***n* = *1***, *, *, *, *, *, *)	640	18.9	17.5	18.4	17.5	19.5	22.9	17.8	20.6	26.3	4.4
(***n* = *2***, *, *, *, *, *, *)	640	17	15.8	16.9	16.2	16.5	20.3	16.6	17.6	21.4	3.9
(***n* = *3***, *, *, *, *, *, *)	640	24.9	20.8	22.3	21.1	28.4	29.3	21.5	29.2	29.5	9.2
(***n* = *4***, *, *, *, *, *, *)	640	15.3	13.7	14.2	13.5	14.5	17.1	13.9	15.1	17.2	4.9
(***n* = *5***, *, *, *, *, *, *)	640	12.9	11.4	12.6	11.6	12.4	14.2	11.8	12.6	14.6	4.3
(***n* = *6***, *, *, *, *, *, *)	640	18.2	13.3	15.4	14.2	23.8	20.8	14.4	23.4	20.6	7.9
(*, ***f* = *1***, *, *, *, *, *)	1920	19.1	16.4	17.7	16.7	20.4	21.5	17	20.8	22.3	6.2
(*, ***f* = *2***, *, *, *, *, *)	1920	16.7	14.4	15.5	14.6	18	20	15	18.7	20.9	5.3
(*, *, ***r* = *1***, *, *, *, *)	1920	19.9	17.8	18.9	17.8	21.4	23.4	18.2	22.2	24.5	6.6
(*, *, ***r* = *2***, *, *, *, *)	1920	15.9	13.1	14.4	13.5	17	18.1	13.8	17.3	18.7	4.9
(*, *, *, ***s* = *1***, *, *, *)	1920	10.6	9.9	10.3	9.9	9	12.4	10	9.6	13.3	2.5
(*, *, *, ***s* = *2***, *, *, *)	1920	25.2	20.9	23	21.5	29.4	29.2	22	29.9	29.9	9
(*, *, *, *, ***h* = *1***, *, *)	1920	22.7	18	20.2	19	25.3	26	19.2	25.9	26.9	9
(*, *, *, *, ***h* = *2***, *, *)	1920	13	12.8	13	12.4	13.1	15.5	12.8	13.6	16.3	2.5
(*, *, *, *, *, ***w* = *1***, *)	1920	22.5	18	20.1	18.6	24.9	26.2	19	25.7	27	8.9
(*, *, *, *, *, ***w* = *2***, *)	1920	13.2	12.8	13.2	12.8	13.4	15.4	13	13.8	16.2	2.6
(*, *, *, *, *, *, ***l* = *1***)	1920	21	17.3	19.6	17.8	23.4	24.6	18	24	25.4	8.1
(*, *, *, *, *, *, ***l* = *2***)	1920	14.7	13.5	13.7	13.6	15	16.9	14	15.5	17.8	3.4

6 Conclusion

This study addresses new problem configurations, close to real-life situations, while considering the scheduling of HTF. Due to the computational difficulty in obtaining optimal solution for the research problem defined in the study, nine variants of simple greedy heuristic algorithms and meta heuristic: Genetic Algorithm are proposed to obtain efficient scheduling. To understand the efficiency of the proposed heuristic algorithms, a LB-procedure is developed to obtain LB.

Based on the series of computational experiments conducted, considering 3840 randomly generated problem instances (representing 384 problem configurations), we observe that (a) the problem parameters considered in this study has influence on the performance of the heuristic algorithms, (b) the proposed LB-procedure is found to be efficient, and (c) the proposed GA outperforms among the proposed heuristic algorithms. However, the computational time required for GA increases as the problem size keeps increasing. Furthermore, in case the decision maker wants to choose a heuristic algorithm which is computationally advantageous among the proposed algorithms, the variants of greedy heuristic algorithm: *sort by width and construct batch* (*SWB)* is relatively better algorithm for the research problem considered.

There are several interesting future research directions. For example: (i) Appropriate modification and/or extension of LP-procedure and heuristic algorithms to address the situation on the availability of more than one non-identical BPs; (ii) It would be interesting to study with the objective of minimizing other completion time-based objectives such as total completion time, total flow time, etc.; and (iii) Considering due date-based objectives such as minimizing number of tardy jobs, maximum lateness, total tardiness, total weighted tardiness are possible future research directions for the research problem considered in this study.

References

1. Baykasoglu, A., Ozsoydan, F.B.: Dynamic scheduling of parallel heat treatment furnaces: a case study at a manufacturing system. J. Manuf. Syst. **46**, 152–162 (2018)
2. Cheragh, S.H., Vishwaram, V., Krishnan, K.K.: Scheduling a single batch processing machine with disagreeable ready times and due dates. Int. J. Ind. Eng.-Theor. Appl. Pract. **10**(2), 175–187 (2003)
3. Gokhale, R., Mathirajan, M.: Minimizing total weighted tardiness on heterogeneous batch processors with incompatible job families. Int. J. Adv. Manuf. Technol. **70**(9–12), 1563–1578 (2014)
4. Huang, J., Liu, J.J.: Hierarchical production planning and real-time control for parallel batch machines in a flow shop with incompatible jobs. Mathematical Problems in Engineering (2018). https://www.hindawi.com/journals/mpe/2018/7268578/abs/. Accessed on 24 May 2019
5. Hulett, M., Damodaran, P., Amouie, M.: scheduling non-identical parallel batch processing machines to minimize total weighted tardiness using particle swarm optimization. Comput. Ind. Eng. **113**, 425–436 (2017)

6. Jia, Z., Wang, C., Leung, J.Y.: An ACO algorithm for makespan minimization in parallel batch machines with non-identical job sizes and incompatible job families. Appl. Soft Comput. **38**, 395–404 (2016)
7. Koh, S.G., Koo, P.H., Kim, D.C., Hur, W.S.: Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. Int. J. Prod. Econ. **98**(1), 81–96 (2005)
8. Malve, S., Uzsoy, R.: A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. Comput. Oper. Res. **34**(10), 3016–3028 (2007)
9. Martello, S., Pisinger, D., Vigo, D.: The three-dimensional bin packing problem. Oper. Res. **48**(2), 256–267 (2000)
10. Mathirajan, M., Gokhale, R., Ramasubramaniam, M.: Modeling of scheduling batch processor in discrete parts manufacturing. In: Ramanathan, U., Ramanathan, R. (eds.) Supply Chain Strategies, Issues and Models. Springer, London (2014). https://doi.org/10.1007/978-1-4471-5352-8_7
11. Mathirajan, M., Sivakumar, A.I.: Minimizing total weighted tardiness on heterogeneous batch processing machines with incompatible job families. Int. J. Adv. Manuf. Technol. **28**(9), 1038–1047 (2006)
12. Mathirajan, M., Sivakumar, A.I.: A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. Int. J. Adv. Manuf. Technol. **29**(9–10), 990–1001 (2006)
13. Monch, L., Fowler, J.W., Mason, S.J., Dauzere-Peres, S.: A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. J. Sched. **14**(6), 583–599 (2011)
14. Ramasubramanian, M., Mathirajan, M., Ramachandran, V.: Minimizing makespan on a single heat-treatment furnace in steel casting industry. Int. J. Serv. Oper. Manag. **7**, 112–142 (2010)
15. Ramasubramanian, M., Mathirajan, M.: Heuristic algorithms for scheduling heat-treatment furnace of steel-casting foundry manufacturing. Int. J. Adv. Oper. Manag. **3**, 271–289 (2011)
16. Ramasubramaniam, M., Mathirajan, M.: A mathematical model for scheduling a batch processing machine with multiple incompatible job families, non-identical job dimensions, non-identical job sizes, non-agreeable release times and due dates. In: 2013 International Conference on Manufacturing, Optimization, Industrial and Material Engineering (MOIME 2013) (2013)
17. Su, L., Qi, Y., Jin, L.: Integrated batch planning optimization based on fuzzy genetic and constraint satisfaction for steel production. Int. J. Simul. Modell. **15**(1), 133–143 (2016)
18. Wang, C.S., Uzsoy, R.: A genetic algorithm to minimize maximum lateness on a batch processing machine. Comput. Oper. Res. **29**(12), 1621–1640 (2002)