A Metaheuristic approach for Batch Sizing and Scheduling Problem in Flexible Flow Shop with Unrelated Parallel Machines

Ebrahim Asadi Gangraj
Assistant professor of industrial engineering
Babol Nooshirvani University of Technology, Babol, Iran

Nasim nahavandi
Associate professor of industrial engineering
Tarbiat Modares University, Tehran, Iran

ABSTRACT
This article considers a makespan minimization batch sizing and scheduling problem in a flexible flow shop scheduling problem with unrelated parallel machines and sequence dependent setup time. Because of NP-completeness of this problem, it is necessary to use the heuristics method. Therefore, this article presents a new mixed simulated-genetic algorithm (MSGA) to tackle this problem. In the comparison, this research reports optimality gaps which are calculated with respect to MSGA method and optimal solution for small instances and the average objective function for large instances. Computational studies indicate that the MSGA is computationally efficient and effective even for small and large instances.

Keywords
Batch sizing, flexible flow shop, metaheuristic method, scheduling.

1. INTRODUCTION
In the literature, scheduling problems have entailed the sequencing of orders (jobs) with fixed processing and set-up times. But in the real world, an order actually consists of several pieces or items. In this situation, order processing times become a decision variable in batch sizing problem. Therefore, the term "batch" is used in this research. In batch sizing problems, orders can be divided and a setup time is incurred when processing a new batch. For this problem, there is a trade-off between minimizing the total setup time (by reduces the number of batches or, equivalently, by increases the size of the batches), and order waiting time minimization (by increases the number of batches, or by reduces the batch sizes) [1].

This study concentrates on a manufacturing system with sequence dependent setup time between the batch processing. Because of equipment limitation, the customer orders must be split to batches with minimum and maximum bound for each batch quantity. This research focuses on a flexible flow shop environment with unrelated parallel machines, with makespan minimization. Therefore these questions are answered in this research: How the orders must be partitioned to batches and processed so that makepan is minimized?

The scheduling of n jobs through m stages where, at any stage, there exists one or more unrelated processors, is termed as flexible flow shop (FFS), flexible flow line (FFL), hybrid flow shop (HFS), or a flow shop with multiple processor (FSMP) scheduling problem with unrelated parallel machines. The FFS exists in many real world manufacturing problems, such as semiconductor assembly facilities [2], packaging industries [3], steal manufacturing [4], electronics manufacturing [5], glass container fabrication [6], automobile assembly [7], printed circuit board assembly [8,9], printed circuit board fabrication [10], ceramic tile manufacturing [11], and lead frame manufacturing [12].

There is a considerable amount of research available for the FFS. Jungwattanakit et al. [13] focused on bi-objective scheduling problem for FFS problems with unrelated parallel machines and setup times. Sheikh [14] presented a multi-objective FFS problem with limited time lag between stages. They presented a MILP with the objectives of maximizing the total profit from scheduled jobs and minimizing deviation from the due date. Huang et al., [15] dealt with permutation flow-shop scheduling problem with the minimizing makespan measure. They proposed a two-phase hybrid particle swarm optimization algorithm to tackle this problem. Rosas-gonzález et al. [16] studied a Genetic Algorithm (GA) to solve the N-Jobs M-Machines Permutation Flow-Shop Scheduling Problem with Break-down times and makespan minimization.

Batch sizing problems with setup times and costs have been considered by various researchers [17]. In some scheduling problems, either job processing times are usually assumed to be fixed or controllable. When job splitting is allowed, batch sizes are decision variables, and job processing times is controllable. Prasad and Maravelias [18] considered a simultaneous scheduling and batch sizing problem for multi-product, multi-stage environment. They developed a new mixed integer linear programming formulation that contains three levels of decisions: batches selection, assignment of batches to units and sequencing of batches in each unit. Chrétienne et al. [19] considered the integrated batch sizing and scheduling problem in single machine environment with tardiness and setup costs minimization. Wang and Guignard [20] had been developed a new hybrid method called “partial parameter uniformization” for batch sizing and scheduling in single machine environment. Me´ndez et al. [21] presented a new MILP for flow shop batch scheduling. The proposed model was based on a continuous time representation with sequence dependent setup times.

This paper simultaneously considers batch sizing and scheduling problem in flexible flow shop environment with sequence dependent setup time. The contributions of this article are twofold: for the first time, the batch sizing and scheduling problem has been considered in flexible flow shop environment with unrelated parallel machines and a mathematical model is proposed for this problem. Also a metaheuristic method, mixed simulated-genetic algorithm, hereafter MSGA, is proposed for this problem in FFS.

The remainder of this paper is organized as follows. Section 2 describes mathematical model for batch sizing and scheduling problem with unrelated parallel machines in FFS environment. Section 3 is dedicated to MSGA metaheuristic

method. In section 4, an experimental study is presented to evaluate the proposed method according to some experimental factors. Finally, section 5 is devoted to the main finding of this paper and suggestions for conducting some future researches.

2. MATHEMATICAL MODEL

The problem addressed in this research can be expressed formally as an integer programming. As mentioned above, the selected objective function is makespan minimization. The sets, parameters, decision variables, and the mathematical model are as follows:

H : Large Number

m : Number of orders

M : Set of order indices ($M = 1,2,...,m$)

l : Number of stages

L : Set of stage indices ($L = 1,2,...,l$)

n_i : Number of batch for order i ($n_i = \left\lceil \frac{r_i}{b_i} \right\rceil$)

N : Set of Batch indices ($N = 1,2,...,n_i$)

Q : Set of machine indices in stage L ($Q = 1,2,..,S_L$)

R : Set of order indices necessary to define decision variables ($R = \{j, j', i, i' | (j < j') \cup (j = j') \cap (i < i')\}$)

b_i : Minimum batch size of order i ($i \in M$)

B_i : Maximum batch size of order i ($i \in M$)

r_i : Demand of order i ($i \in M$)

p_{ikt} : Processing time of order i on machine k at stage t ($i \in M, k \in Q, t \in L$)

$sdt_{i\,i'}$: Sequence dependent setup time if order i' is processed before order i ($i, i' \in M$)

C_{ijt} : Completion time of batch j of order i at stage t ($i \in M, j \in n_i, t \in L$)

q_{ij} : Units number of order i in batch j ($i \in M, j \in n_i$)

X_{ijkt} : 1 if batch *j* of order *i* is processed on machine *k* at stage *t*, 0 otherwise ($i \in M, j \in n_i, k \in Q, t \in L$)

$y_{ii'jj't}$: 1 if batch *j* of order *i* is processed earlier than batch j' of order i' at stage *t* ($i, i' \in M, j, j' \in n_i, t \in L$)

$w_{ii'jj't}$: 1 if batch *j* of order *i* and batch j' of order i' are on same machine at stage *t* ($i, i' \in M, j, j' \in n_i, t \in L$)

If the decision variables are defined regarding the range of their indices, the number of decision variables increases to a huge number and the efficiency of the mathematical model will decrease drastically. Thus, the unnecessary decision variables are omitted by defining set R [22]. For instance, if $y_{ii'jj't}$ is equal to 1, the value of $y_{i'ij'jt}$ is 0 and vice versa. Thus, only one of them is used in the mathematical model.

Therefore the mathematical model of this problem in the FFS environment with unrelated parallel machines can be formulated as follows:

$$\text{Min Z} = \max_{i,j}\{C_{ijm}\} \tag{1}$$

$$\sum_{k=1}^{S_l} X_{ijkt} = 1,\ i \in M, j \in n_i, t \in L \tag{2}$$

$$q_{ij} \le H \sum_{k=1}^{S_t} X_{ijkt}\ \ i \in M, j \in n_i, t \in L \tag{3}$$

$$q_{ij} \ge \sum_{k=1}^{S_t} X_{ijkt}\ \ i \in M, j \in n_i, t \in L \tag{4}$$

$$C_{ij1} \ge \sum_{k=1}^{S_1} p_{ik1} X_{ijk1} q_{ij}\ \ i \in M, j \in n_i \tag{5}$$

$$\text{C}_{\text{ijt}} \ge C_{ij,t-1} + \sum_{k=1}^{S_t} p_{ikt} X_{ijkt} q_{ij} \quad i \in M, j \in n_i, t \in L \tag{6}$$

$$\text{C}_{\text{ijt}} + H\left(1 - y_{ii'jj't}\right) \ge C_{i'j't} + p_{ikt} X_{ijkt} q_{ij} + sdt_{i'i} \quad i, i' \in M, j, j' \in n_i, k \in Q, t \in L \tag{7}$$

$$C_{i'j't} + H\left(1 - w_{ii'jj't} + y_{ii'jj't}\right) \ge \text{C}_{\text{ijt}} + p_{i'kt} X_{i'j'kt} q_{ij} + sdt_{ii'} \quad i, i' \in M, j, j' \in n_i, k \in Q, t \in L \tag{8}$$

$$w_{ii'jj't} \ge X_{ijkt} + X_{i'j'kt} - 1 \quad i, i' \in M, j, j' \in n_i, k \in Q, t \in L \tag{9}$$

$$\sum_{j=1}^{n_i} q_{ij} = r_i\ \ i \in M \tag{10}$$

$$b_i \sum_{k=1}^{S_l} X_{ijkt} \le q_{ij} \le B_i \sum_{k=1}^{S_l} X_{ijkt}\ \ i \in M, j \in n_i, t \in L \tag{11}$$

$$X_{ijkt}, y_{ii'jj't}, w_{ii'jj't} \in \{0,1\} \quad i, i' \in M, j, j' \in n_i, k \in Q, t \in L \tag{12}$$

The objective function, as presented in Eq. (1), minimizes makespan. Constraint sets (2) indicate that each batch must be assigned to one machine at each stage. Constraint sets (3) and (4) ensure that each established batch must be processed on one machine at each stage. Constraint sets (5) show that completion time of each batch in the first stage is greater than or equal to its processing time in this stage. Constraint sets (7) and (8) preclude the interference between the processing operations of any two batches on a machine at any stage. At

most, one of these two constraint sets is active for each pair of batches. Constraint sets (9) determines the batches which are processed on the same machine in stage L. Constraint sets (10) impose that the demand must be fully satisfied for each order. Constraint sets (11) ensure the batch size constraints must be met. Finally constraints (12) force variables to assume binary values 0 or 1.

2.1 Model Linearization

As can be seen, the above model is obviously nonlinear (because of term $X_{ijkl}.q_{ij}$ in constraints (5), (6), (7), (8)); therefore it causes the long computational runtime. On the other side, this term can be linearized by introducing new auxiliary binary integer variable as follows:

$$W_{ijkt} = \begin{cases} q_{ij} & if\ X_{ijkt}\ equals 1 \\ 0 & otherwise \end{cases}$$

The required new constraint can be defined as follows:

$$W_{ijkt} = X_{ijkt}.q_{ij} \tag{13}$$

By considering the above equation, following constraints must be added to the mathematical model simultaneously:

$$W_{ijkt} \le q_{ij}\ \ i \in M,\ j \in n_i, k \in Q, t \in L \tag{14}$$

$$W_{ijkt} \le HX_{ijkl}\ \ i \in M,\ j \in n_i, k \in Q, t \in L \tag{15}$$

$$W_{ijkt} \ge H(X_{ijkt} - 1) + q_{ij}$$
$$i \in M,\ j \in n_i, k \in Q, t \in L \tag{16}$$

3. MIXED SIMULATED GENETIC ALGORITHM (MSGA)

After many reported experiments in the literature, genetic algorithms have been found to be efficient, effective and robust algorithm for complicated problems. Nevertheless, genetic algorithms also have their shortcomings. In fact, if the worst members are discarded after each generation, the population will tend to become homogeneous quickly. And then the crossovers and mutation may not produce offspring of large variation. For this reasons, some authors have suggested inserting another operator, namely, a Boltzmann-type operator, after the crossover and mutation operations. Therefore, it is called this new metaheuristic method as mixed simulated genetic algorithm (MSGA). In MSGA, new chromosomes are chosen to produce the next generation from parents and offspring according to Boltzeman function. The selection criterion is based on the fitness values of parents and offspring. Chromosomes with higher fitness values have a greater probability of surviving into the next generation. Those with less fitness values are not necessarily discarded.

Based on above discussion, the MSGA is now described as follows:

1. ***Random chromosome generation***: in the MSGA, number of chromosome in each generation equals a coefficient of number of orders (Poprate). Then, the orders are randomly sequenced and formed a chromosome.
2. ***Fitness function***: for calculating the fitness function, for each individual in current generation, it must be determined batch size and batch sequence. Batch sizing and batch sequencing is now described as follows:
   - Determine the maximum and minimum number of batches based on following equations:

$$MaNB_i = \frac{r_i}{b_i}\ i \in M \tag{17}$$

$$MiNB_i = \frac{r_i}{B_i}\ i \in M \tag{18}$$

   - Generate a random integer value in interval $MiNB_i \le NB_i \le MaNB$
   - Calculate the batch size for each order based on $FBS_i = round\ down\left\{\frac{r_i}{NB_i}\right\}$
   - Calculate number of unassigned quantity for each order as $RM_i = r_i - FBS_i.NB_i$
   - Append the unassigned quantity to each batch so that difference between the batch quantities must not greater than 1.
   - Finding the bottleneck stage as $FR_l = \frac{\sum_{k=1}^{S_l}\sum_{i=1}^{m}\sum_{j=1}^{n_i} P_{ijk}}{S_l}$
   - Sequence batches based on shortest average processing time at bottleneck stage.
   - Select machines at each stage based on following rule:
     - ✓ Batch is assigned to all machines (available and unavailable) at any stage and selects a machine that has the earliest completion time.
   - Finally calculate the maximum processing time as makespan.
3. ***Elitism*****:** to prevent the elimination of good solution, in each generation, some good chromosomes are migrated to next generation without fluctuation. In the MSGA, number of chromosome, for elitism, equals a coefficient of number of orders (Elitrate).
4. ***Crossover***: The basic operator for producing new individuals in GA is crossover. Crossover may produce better individuals that have some genetic material of both parents. In the MSGA, number of chromosome, for crossover, equals a coefficient of number of orders (crossrate). The conventional crossover operator combines sub-strings belonging to their parents. In this research, two chromosomes are selected and a two points cut method is applied to crossover.
5. ***Mutation***: another basic operation to generate the individual is mutation. This operator schemes allow jumps to different areas of the solution space. In this research, two mutation operators are used; two points swap and three points swap. In first (second) method,

two (three) points are randomly selected and the corresponding genes are changed. In these methods, number of chromosome, for mutation, equals a coefficient of number of orders (mutrate).

6. ***SA operator***: the SA operator is used to produce the new generation. At first, the chromosomes of current generation is sorted in ascending order of fitness value and the chromosomes of last generation is sorted in descending order of fitness value and then, they are compared one-to-one. If the fitness value of current generation is better than the last generation, the chromosome of current generation is selected for next generation. On the other side, if the fitness function of current generation has worse performance, the SA operator is used. In this operator, Boltzeman function is applied to select between chromosomes as follows:

$$P = \exp\left(-\frac{C(x_{i+1}) - C(x_i)}{T}\right) \quad (19)$$

In equation (19), T is temperature of current generation, $C(x_{i+1})$ fitness function of current solution (from current generation), $C(x_i)$ fitness function of last solution (from last generation).

Temperature is function of three operators: initial temperature, final temperature and cool rate. In each stage the following function is applied to calculate the current temperature:

$$T = T \cdot coolrate \quad (20)$$

4. COMPUTATIONAL STUDY

This section is devoted to consider the performance of proposed method. The best parameters for MSGA method firstly is selected, and then, compare the performance of MSGA method with optimal solution and GA.

Table 1: Set parameters for MSGA

Parameter	Value		
Poprate	1	2	5
Crossrate	0.8	0.6	0.5
Mutrate	0.1	0.2	0.3
Elitrate	0.1	0.2	0.2
Initemp	100	500	1000

For the selection of best value for each parameter, 60 experimental problems are generated for each state and average of objective function, standard deviation and run time is summarized in the table 2:

Table 2: average of objective function, standard deviation and run time for set parameters

State	Poprate	Poprate	Crossrate	Mutrate	Elitrate	Initemp	Average	S. Deviation	Time
1	1	0.8	0.1	0.1	500	0.001	0.8	42.45	9.65
2	2	0.8	0.1	0.1	500	0.001	0.8	47.12	11.02
3	5	0.8	0.1	0.1	500	0.001	0.8	71.18	13.60
4	2	0.8	0.1	0.1	500	0.001	0.8	39.40	4.03
5	2	0.6	0.2	0.2	500	0.001	0.8	60.34	4.41
6	2	0.5	0.3	0.2	500	0.001	0.8	73.41	3.54
7	2	0.8	0.1	0.1	100	0.1	0.8	82.94	4.76
8	2	0.8	0.1	0.1	500	0.01	0.8	62.75	4.03
9	2	0.8	0.1	0.1	1000	0.001	0.8	56.92	3.48
10	2	0.8	0.1	0.1	1000	0.01	0.8	47.06	2.60
11	2	0.8	0.1	0.1	1000	0.01	0.9	62.00	2.44
12	2	0.8	0.1	0.1	1000	0.01	0.95	66.77	3.08

regarding table 2, state 1, 2 and 3 is used to select best value for poprate, state 4, 5 and 6 is for crossrate, mutrate and elitrate, state 7, 8 and 9 is for initemp and fintemp. At last, in states 10, 11 and 12, the best value for coolrate is selected. According to average, standard deviation and run time, the best set parameters for MSGA is shown in table 3:

Table 3: best value for MSGA parameters

Parameter	Value
Poprate	2
Crossrate	0.8
Mutrate	0.1
Elitrate	0.1
Initemp	1000
Fintemp	0.01
coolrate	0.9

4.1 MSGA performance evaluation

In order to performance evaluation, two series of experiment were conducted; For this purpose, the first series compare proposed algorithms with the optimal solution for the small size problem and the second series compare the proposed algorithms with GA metaheuristic for the medium and large instances. All the heuristics are coded using the MATLAB software and the entire experiments are performed on a PC with Intel Core 2 Dou 2.2 GHz CPU and 2 GB RAM. All the optimal solutions are obtained by Lingo 9.0 software.

Because of NP-completeness of batch sizing and scheduling problem in FFS, it is very expensive to achieve the optimal solution for the medium and large instances. Therefore test problems in table 4 are limited to the small size problem. For this purpose, 20 instances are generated and Table 4, presents the comparison of exact method and MSGA:

Table 4: comparison of MSGA and exact method

Test problem	Exact	MSGA	Optimal Gap
1	136	140	2.9%
2	114	120	5.3%
3	140	147	5.0%
4	128	128	0.0%
5	137	146	6.6%
6	162	166	2.5%
7	104	104	0.0%
8	136	144	5.9%
9	152	152	0.0%
10	202	205	1.5%
11	133	142	6.8%
12	254	258	1.6%
13	154	158	2.6%
14	160	162	1.3%
15	200	204	2.0%
16	103	106	2.9%
17	138	146	5.8%
18	110	110	0.0%
19	188	196	4.3%
20	136	152	11.8%
Average	149.4	154.3	3.4%

According to table 4, it can be concluded, the MSGA is competitive with optimal solution for small instances, as can be observed from the average objective function (last row). Based on the results shown in this table, the average difference between optimal solution and metaheuristic (optimal gap) is 3.4%.

Unfortunately, since this is the first time that this problem is considered in the literature, we do not have the results of applying other algorithms to solve the problem to compare with the proposed methods. However, the goal of this research was to obtain the most efficient way of implementing MSGA algorithm to solve this problem. In fact, this article is compared proposed algorithm with the pure GA which is introduced and broadly used in the literature before. For this purpose, some computational experiments are conducted to compare the MSGA with GA. For comparison purpose, we generate some test problems based on table 5:

Table 5: experimental factors

Experimental factor	Level
Number of order	3 levels: 10, 20, 50
Number of stage	2 levels: 6, 12
order size[1]	U[20,50]
Sequence Dependent Setup Time	U[5,20]
Number of machines in each stage	3 levels: 3, 5, 8

[1]The processing time of each item in the order is assumed 1.

According to table 5, it will has 18 different test problems. Each test problem was run ten times and its performance, including the best Cmax value, was recorded in table 6. Other necessary data for each problem, such as maximum and minimum batch size, are randomly generated.

Table 6: Evaluating the quality of solution of the MSGA in comparison of GA

Problem	MSGA	GA	% Deviation
O10S6M3	333	340	2%
O10S6M5	272	275	1%
O10S6M8	305	314	3%
O10S12M3	522	559	7%
O10S12M5	401	417	4%
O10S12M8	344	347	1%
O20S6M3	348	351	1%
O20S6M5	287	296	3%
O20S6M8	201	215	7%
O20S12M3	694	729	5%
O20S12M5	481	491	2%
O20S12M8	394	410	4%
O50S6M3	370	377	2%
O50S6M5	415	432	4%
O50S6M8	481	510	6%
O50S12M3	717	746	4%
O50S12M5	564	598	6%
O50S12M8	654	700	7%

The three characteristics that typify the problem are the number of order, number of stages and number of unrelated parallel machines at each stage. For example, the notation *O10S12M5* means a 10-order, 12-stage problem and 5 unrelated parallel machines in each stage. The letters *J, C and M* are abbreviations for order, stage and machine, respectively.

The computational results are summarized in Table 6, in which the "% deviation" columns show the performance comparison among two algorithms. According to table 9, the MSGA is outperformed GA metaheuristic in medium and large size problems.

5. CONCLUSION

In this paper, we studied the simultaneous batch sizing and scheduling decisions in a flexible flow shop environment with unrelated parallel machines and setup time. The first contribution of the paper is to provide a mathematical model to minimize the makespan for batch sizing and scheduling problems. Also, this research presents a mixed simulated genetic algorithm (MSGA) for this problem. The results show methaheuristic has good performance to reach the optimal solution.

Future works can consider other environments, such as job shop and open shop. Other metaheuristics algorithms (SA, TS

or PSO) can also be applied for this problem. Afterwards, considering other type of objective function such as, tardiness, number of tardy job, flow time is opened for other research.

6. REFERENCES

[1] Coffman, E. G. and Yannakakis, M. 1990. batch sizing and sequencing on a single machine. Ann. Oper. Res. 26, 135-147.

[2] Quadt, D. and Kuhn, H. 2005. Conceptual framework for lot-sizing and scheduling of flexible flow lines. Int. J. of Prod. Res. 43(11), 2291-308.

[3] Adler, L., Fraiman, N., Kobacker, E., Pinedo, M., Plotnicoff, J. C. and Wu, T. P. 1993. BPSS: a scheduling support system for the packaging industry. Oper. Res. 41, 641-648.

[4] Voss, S. and Witt, A. 2007. Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: a real-world application. Inter. J. of Prod. Econ. 105(2), 445-458.

[5] Wittrock, R. J. 1988. An adaptable scheduling algorithm for flexible flow lines. Oper. Res. 36, 445-453.

[6] Leon, V. J. and Ramamoorthy, B. 1997. An adaptable problem space-based search method for flexible flow line scheduling. IIE Transactions. 29, 115-125.

[7] Agnetis, A., Pacifici, A., Rossi, F., Lucertini, M., Nicoletti, S. and Nicolo, F. 1997. Scheduling of flexible flow shop in an automobile assembly plant. Eur. J. of Oper. Res. 97(2), 348-362.

[8] Jin, Z.H.; Ohno, K.; Ito, T.; Elmaghraby, S.E.: Scheduling hybrid flowshops in printed circuit board assembly lines. Prod. and Oper. Manag. 11(2), 216-230, (2002)

[9] Hayrinen, T., Johnsson, M., Johtela, T., Smed, J. and Nevalainen, O. 2000. Scheduling algorithms for computer-aided line balancing in printed circuit board assembly. Prod. Planin. Con. 11(5), 497-510.

[10] Alisantoso, D., Khoo, L. P. and Jiang, P. Y. 2003. An immune algorithm approach to the scheduling of a flexible PCB flow shop. Inter. J. of Adv. Manuf. Technol. 22(11), 819-827.

[11] Ruiz, R. and Maroto, C. 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. Eur. J. of Oper. Res. 169, 781-800.

[12] Lee, G. C., Kim, Y. D. and Choi, S. W.. 2004. Bottleneck-focused scheduling for a hybrid flowshop. Int. J. of Prod. Res. 42, 165-181.

[13] Jungwattanakit, J. and Reodecha, M. Chaovalitwongse, P. and Werner, F. 2008. Algorithms for flexible flow shop problems with unrelated parallel machines, setup times and dual criteria. Int. J. of Adv. Manuf. Technol. 37, 354-370.

[14] Sheikh, S. 2012. Multi-objective flexible flow lines with due window, time lag, and job rejection. Int. J. of Adv. Manuf. Technol. doi: 10.1007/s00170-012-4112-5.

[15] Rosas-gonzález, A., Clemente-guerrero, D., Caballero-morales, S. and Flores-juan, J. 2013. An Evolutionary Approach for Solving the N-Jobs M-Machines Permutation Flow-Shop Scheduling Problem with Break-Down Times. Int. J. of Comput. Appl. 83(1); 1-6.

[16] Huang, K. Yang, C. and Tsai, C. 2012. A Two-Phase Hybrid Particle Swarm Optimization Algorithm for Solving Permutation Flow-Shop Scheduling Problem; Int. J. of Comput. Appl. 48(1),11-18

[17] Allahverdi, A., NG, C. T. and Cheng, T. C. E., and Kovalyov, M. Y. 2008. A survey of scheduling problems with setup times or costs. Eur. J. of Oper. Res. 187(3), 985-1032.

[18] Prasad, P. and Maravelias, C. T. 2008. Batch selection, assignment and sequencing in multi-stage multi-product processes. Comput. and Chem. Eng. 32, 1106-1119.

[19] Wang, S. and Guignard, M. 2006. Hybridizing discrete- and continuous-time models for batch sizing and scheduling problems. Comput. and Oper. Res. 33, 971-993.

[20] Chrétienne, P., Hazır, Ö. and Kedad-Sidhoum, S. 2011. Integrated batch sizing and scheduling on a single machine. J. of Sched. 14, 541-555.

[21] Me´ndez, C. A., Henning, G. P. and Cerda, J. 2011. An MILP continuous-time approach to short term scheduling of resource-constrained multistage flowshop batch facilities. Comput. and Oper. Res. 25, 701-711.

[22] Tadayon, B. and Salmasi, N. 2013. A two-criteria objective function flexible flowshop scheduling problem with machine eligibility constraint. Int. J. of Adv. Manuf. Technol. 64, 1001-1015.