Journal of Optimization in Industrial Engineering, Vol.11, Issue 2, Summer and Autumn 2018, 63-76
DOI: 10.22094/joie.2018.792.1505

An Efficient Bi-Objective Genetic Algorithm for the Single Batch-Processing Machine Scheduling Problem with Sequence-Dependent Family Setup Time and Non-Identical Job Sizes

Javad Rezaeian[a,*] , Yaser Zarook[b]

[a]*Department of Industrial Engineering, Mazandaran University of Science and Technology*
Received 12 July2 017; Revised 30 November 2017; Accepted 19 February 2018

Abstract
This paper considers the problem of minimizing make-span and maximum tardiness simultaneously for scheduling jobs under non-identical job sizes, dynamic job arrivals, incompatible job families, and sequence-dependent family setup time on the single batch- processor, where split size of jobs is allowed between batches. At first, a new Mixed Integer Linear Programming (MILP) model is proposed for this problem; then, it is solved by ε-constraint method. Since this problem is NP-hard, a bi-objective genetic algorithm (BOGA) is offered for real-sized problems. The efficiency of the proposed BOGA is evaluated to be compared with many test problems by ε-constraint method based on performance measures. The results show that the proposed BOGA is found to be more efficient and faster than the ε-constraint method in generating Pareto fronts in most cases.

Keywords: Batch Processing; Incompatible Job Family; Release Date; Split Job Size; Family Setup Time.

1. Introduction & Literature Review

In recent decades, a batch-scheduling problem is a usual category of production planning in most industries. The main reasons of batch processing are avoidance of setup times and material handling costs (Baker 1943). Rui et al. (2012) divided batching into two types: serial batching and parallel batching. In a serial batching, the length of a batch is equivalent to the sum of the processing times of jobs in it. In a parallel batching, several jobs are processed in a batch simultaneously on a processor at the same time, such that all the jobs in the batch start and complete their process at the same time. The processing time of a batch is equivalent to the biggest processing time of jobs in it. The batching parallel machine (BPM) is able to process a group of jobs as long as the sum of job sizes in the batch is less than or equal to the capacity of the machine. Once a batch is processed, the BPM cannot be interrupted; no jobs can be removed from the machine until the process is completed. To date, BPM problem has attracted many investigators (Pinedo 2008; Rui et al., 2012;Dauzère-Pérès and Mönch 2013;Guo et al. 2013). They are commonly used to test electronic assemblies to detect early failures and burn in oven. This problem is motivated by the burn-in operation found in the final testing phase in semiconductor manufacturing. This problem is important because the scheduling of batching operations has a significant economic impact. These operations constitute a bottleneck in the final testing phase; consequently, efficient scheduling to maximize throughput is of great concern in productivity and on- time delivery management. On the other hand, optimizing a single-objective generally may lead to deterioration of other possible objectives. Many industries, such as semiconductor manufacturing, have trade-offs in their scheduling where multiple objectives need to be considered in order to optimize the overall performance of the system.Crauwelsand Potts (1996) studied the BPM problem. Perhaps, for the first time, they represented local search heuristics for single batch machine scheduling to minimize the number of late jobs. Considering jobs with different sizes, Wang & Uzsoy (2002) gave complexity results for $C_{\max}$ and ΣC_j criteria and provided some heuristics and a branch-and-bound algorithm. A branch-and-bound procedure for minimizing $C_{\max}$ was also developed by Dupont & Dhaenens (2002). Sung et al. (2002) minimized makespan on a single burn in oven with job family and dynamic job arrivals (Cheng et al., 2013). Rafiee et al. (2010) represented a branch and price algorithm to minimize make-span on a single batch-processing machine with non-identical job sizes. This paper considers non-identical job sizes that are allowed to split job size into the batches. Therefore, split job size will interrupt job sizes on the batches.

In most BPM studies, jobs are compatible for batching; however, considered jobs belong to incompatible job families, meaning that every job assigned to a batch must belong to the same family, and that jobs belonging to the same family share a common processing time, which is also the batch processing time.

*Corresponding author Email address: j.rezaeian@ustmb.ac.ir

In literature, researches considered the various aims; Malvea&Uzsoy (2007) studied BPM with different performance measures in the case of incompatible job families in which each job has unit size. They also presented exact and approximate solutions for single and parallel batch processing machines problems. Dupont & Dhaenens (2002) suggested an exact procedure for minimizing the makespan on a batch processing machine with non-identical job sizes. Yao & Jiang (2012) offered a branch and bound algorithm for minimizing total completion time on a single batch processing machine with incompatible job families and dynamic job arrivals. Perez et al. (2005) minimized total weighted tardiness on a single batch processing machine with incompatible job families. Most studies have considered single objective optimization. Sometimes, many industries such as semiconductor manufacturing consider both of their customer's requirements such as assuring on-time reception and their manufacture's requirements, such as reducing work-in-process inventory. For example, Liu et al. (2009) presented a bi-criterion scheduling problem with equal processing times on a batch processing machine. Xu et al.(2013) considered a bi-objective scheduling problem on batch machines recently. This paper considers a multi-objective optimization problem that aims to minimize makespan and maximum tardiness simultaneously. Hence, lower C_{max} leads to higher productivity and higher throughput level for the bottleneck operation. T_{max}, which represents most violations of due dates, is related to customer's on-time delivery. In the literature of multi-objective optimization problem, Husseinzadeh et al. (2010) presented a multi-objective genetic algorithm for minimizing makespan and maximum tardiness under situation of non-identical job size on a BPM problem; therefore, already, contradiction between two aims has been proven. According to the mentioned studies, to get closer batch processing problem to the real word, a common set of constraints has been removed in this paper. These constraints consists of non-identical job sizes where split size of jobs is allowed between batches, dynamic job arrivals, incompatible job families, and sequence-dependent family setup time on the single batch_ processor. In the literature of scheduling problem, including abbreviations $\alpha \mid \beta \mid \gamma$, α shows number of machines, β shows special conditions, and γ shows objective functions of the problem; therefore, this problem is presented as $1|r_j,s_j$, B, incompatible family, split job size$|C_{max}$, T_{max}, meaning that the single batch processing machine scheduling problem is allowed to operate under dynamic job arrivals, incompatible job families, and non-identical job sizes which split job size. The objective functions are minimization of C_{max} and T_{max}, respectively. Fig.1 shows a graphic illustration of the problem.

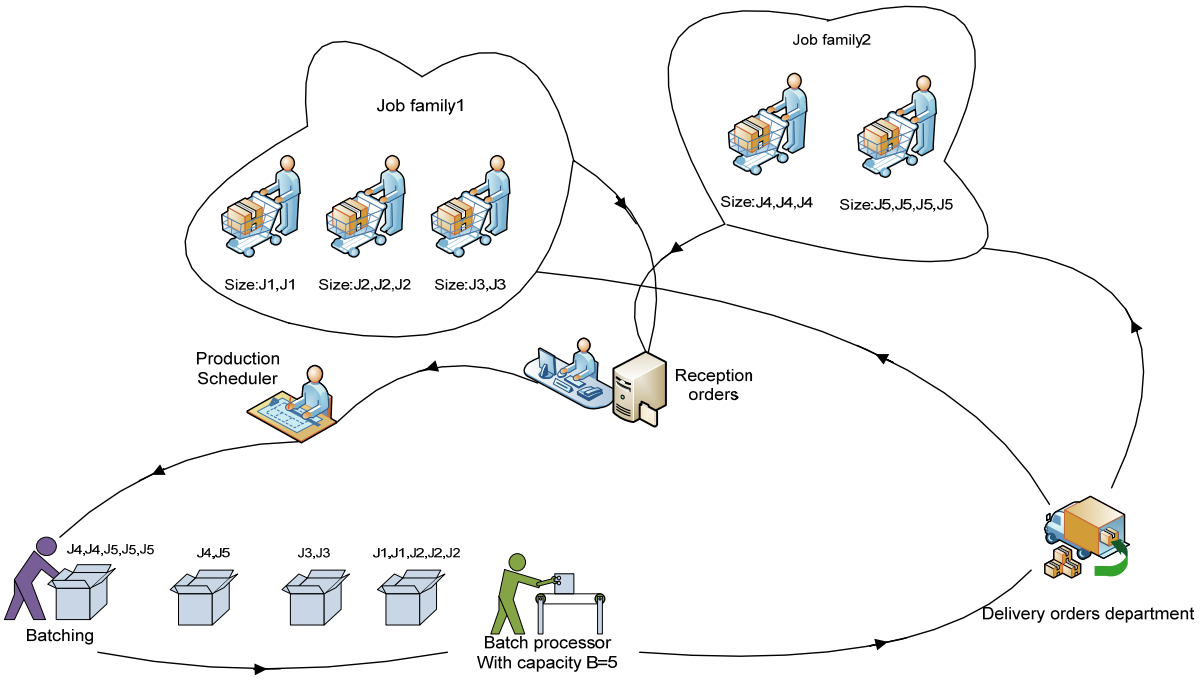

Fig.1. Illustration of the problem

Recently, solving the batch machine scheduling problems using meta-heuristic algorithms and approximate methods has become interesting by researchers. Melouk et al. (2004) represented a simulated annealing for minimizing make-span on single batch processing machine with non-identical job sizes. Damodaran et al. (2006) introduced a genetic algorithm for minimizing make-span on a batch-processing machine with non-identical job sizes. Malvea and Uzsoy (2007) presented a genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. Koh et al.(2005) proposed a

hybrid genetic algorithm based on a random key representation to minimize makespan and total weighted completion time on a BPM problem with incompatible job families and arbitrary jobs size. A hybrid genetic algorithm was presented on identical parallel patch-processing machines with non-identical job sizes by Husseinzadeh et al. (2008). Rui et al. (2012) presented an ant colony optimization and mixed integer programming (MIP) for minimizing make-span on a BPM problem with

The rest of the paper is organized as follows: in Section 2, a problem definition and a Mixed Integer Linear Programming (MILP) model are presented; then, to validate a new mathematical model and contradiction between the objectives, an example is solved by ε-constraint method in lingo9 software. In Section 3, the structure of a bi-objective genetic algorithm and the analysis and discussion of this proposed approach are presented. Some experimentations and comparison are shown in section 4. Finally, in section 5, conclusion and suggestions for future researches are presented.

2. Problem Formulation

2.1. Problem description

This paper focuses on a BPM problem with the following assumptions:

- There are *N* jobs with dynamic arrivals and predefined due dates that are agreeable.
- A single processor has a limited capacity *B*, and job *j* has a specified size S_j that is not more than machine capacity.
- Total job sizes of a batch should not be more than machine capacity *B*.
- Each job can be divided on the size in several batches.
- Each batch should be processed only for once; namely, it should not be interrupted.
- The machine only can process several jobs of a family in each batch simultaneously, and jobs of different families must be processed separately.
- The jobs of a family have equal processing times, and processing time of a batch is equal to processing time of the family assigned to the batch, and each opened batch belongs to only a family.
- There is setup time between two batches of different job families.
- The machine is assumed available continuously and breakdown is not allowed.
- The machine cannot be idle when at least a non-completed job exists.

2.2. Mathematical description of the problem

This section provides a brief description of the mathematics, a set of jobs $N= \{J_1, J_2, \dots, J_n\}$, a set of job family $M=\{F_1, F_2, \dots, F_m\}$, and a set of $K=\{B_1, B_2, \dots, B_k\}$. The following indices, parameters, and decision variables are used throughout the paper.

Indices:

j index for jobs $(j = 1, 2, \dots, n)$

i index for families $(i = 1, 2, \dots, m)$

k index for batches $(k = 1, 2, \dots, k)$

p index for sequence of batches $(p = 1, 2, \dots, p)$

Parameters:

s_j Size of job j

d_j Due date of job j

r_j Release date of job j

p_i Processing time of family i

B Capacity of a single processor

M Number of family

N_i Number of jobs of family i

F_i Set of job of family i

Matrix setup time of between job families

Decision variables

$\mathbf{X_{jikp}}$=1, if job j of family I is assigned to the batch k which is processed in position p on the machine; 0, otherwise

$\mathbf{Q_{jikp}}$ = size of job j of family i assigned to batch k which is processed in position p on the machine (integer)

$$\boldsymbol{Z_{jikpj'i'k'p-1}}=\begin{cases}1 & \text{, if } \mathbf{X_{jikp}} \times \mathbf{X_{j'i'k'p-1}} = 1 \\ 0 & \text{, otherwise}\end{cases}$$

$\boldsymbol{W_{jikp}} = X_{jikp} \times C_{ikp}$

$\mathbf{ST_{ikp}}$= start time of batch k of family I in position p on the machine

$\mathbf{C_{ikp}}$= completion time of batch k of family i in position p on the machine

$\mathbf{C_j}$= completion time job j

$\mathbf{T_j}$= tardiness of job j

$\mathbf{E_j}$= earliness of job j

2.3. The mathematical modeling

In this subsection, objective functions and constraints are formulated as follows:

$$min\, z_1 = C_{max} \quad (1)$$

$$min\, z_2 = T_{max} \quad (2)$$

st:

$$\sum_{i=1}^{m} X_{jikp} \leq 1 \quad \forall k, j, p \quad (3)$$

$$\sum_{j \in Fi} Q_{jikp} \leq B \forall k, i, p \quad (4)$$

$$\sum_{p=1}^{p} \sum_{k=1}^{k} Q_{jikp} = s_j \forall i, j \in Fi \quad (5)$$

$$\sum_{p=1}^{p} \sum_{k=1}^{k} X_{jikp} \leq s_j \forall i, j \in Fi \quad (6)$$

$$Q_{jikp} \leq X_{jikp} * s_j \forall i, j \in Fi, k, p \quad (7)$$

$$X_{jikp} \leq Q_{jikp} \forall i, j \in Fi, k, p \quad (8)$$

$$X_{jikp} * r_j \leq ST_{ikp} \forall i, j \in Fi, k, p \quad (9)$$

$$ST_{i'k'p-1} + P_{i'} + Z_{jikpj'i'k'p-1} * setup_{i',i} \leq ST_{ikp} \quad \forall p \geq 2, \forall i, i', j \in F_i, j' \in F_{i'}, k, k' \quad (10)$$

$$X_{jikp} + X_{j'i'k'p-1} \leq Z_{jikpj'i'k'p-1} + 1 \forall p \geq 2, \forall i, i', j \in F_i, j' \in F_{i'}, k, k' \quad (11)$$

$$2 * Z_{jikpj'i'k'p-1} \leq X_{jikp} + X_{j'i'k'p-1} \forall p \geq 2, \forall i, i', j \in F_i, j' \in F_{i'}, k, k' \quad (12)$$

$$X_{j'i'k'p+1} \leq X_{jikp} \forall i, i', j \in F_i, j' \in F_{i'}, k, k', p \quad (13)$$

$$C_{ikp} = ST_{ikp} + P_i \forall k, i, p \quad (14)$$

$$C_j \geq W_{jikp} \forall i, j \in F_i, k, p \quad (15)$$

$$W_{jikp} \leq M.X_{jikp} \;\; , W_{jikp} \leq C_{ikp} \;\; , W_{jikp} \geq C_{ikp} - \left(1 - X_{jikp}\right).M \;\; , W_{jikp} \geq 0 \quad \forall i, j \in F_i, k, p \quad (16)$$

$$C_j + E_j - T_j = d_j \quad (17)$$

$$C_{max} \geq C_j \forall j \quad (18)$$

$$T_{max} \geq T_j \forall j \quad (19)$$

$$X_{jikp} = 0 \;\; or \;\; 1 \;\; , \qquad Q_{jikp} \geq 0 \;\&integer \quad (20)$$

Equations (1) and (2) show the objective functions, the makespan, and the maximum tardiness, respectively. The makespan of BPM problems is equal to the completion time of the last batch and maximum tardiness is equal to maximum deviation of the job's due date. Eq. (3) ensures that each batch only belongs to a family. Eq. (4) ensures that the sum of the size of jobs assigned to a batch do not exceed machine's capacity. Eq. (5) guarantees the size of each job. Eq. (6) ensures that job j can be in maximum number of batch equivalent to s_j. Eqs. (7) and (8) ensure

the relation between decision variables. Constraint sets (9, 10) represent the starting time of batches. Constraint sets (11, 12) are determined for linearization of Eq. (10). Eq. (13) guarantees sequence of processing batches on the machine. Eq. (14) calculates the completion time of each batch. Constraint sets (15-16) calculate completion time of each job linearly. Eq. (17) determines earliness and tardiness of each job. Constraint sets (18) and (19) determine $C_{max}$and T_{max}, respectively. Eq. (20) represents binary restriction and integer restriction on the decision variables.

To validate the proposed mathematical model and inconsistency between objectives, a small test problem is provided with the following data: $N=5$, $N_F=2$,$B=4$, $M_1=\{1,2,5\}$, $M_2=\{3,4\}$, $p_i=\{3,2\}$, $r_j=\{3,8,5,5,5\}$, $s_j=\{2,2,2,2,2\}$, $d_j=\{7,13,11,8,8\}$, family setup time=$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$.

Using ε-constraint method in lingo 9, we could get Pareto frontier as in Fig.3 where batch and sequence are shown in Fig.2:

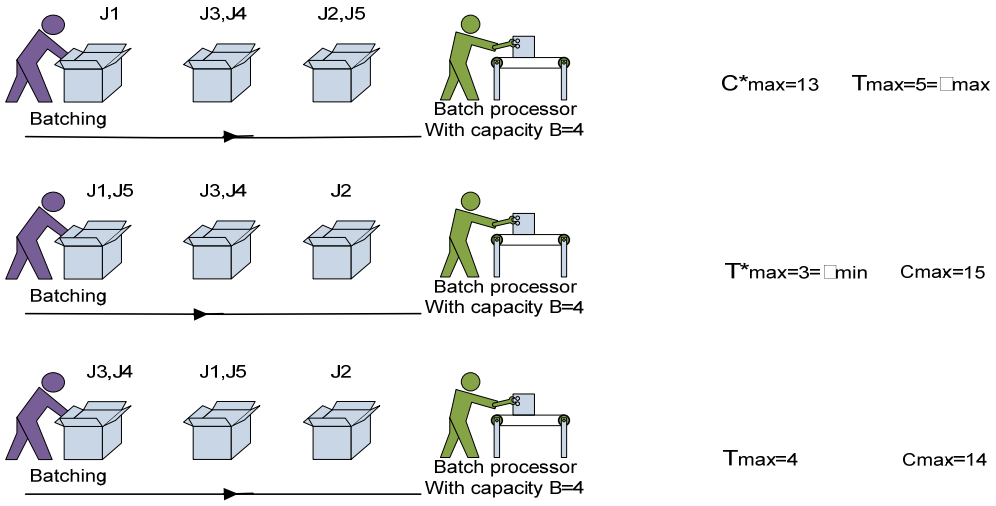

Fig.2. Non-dominated solution of constraint method

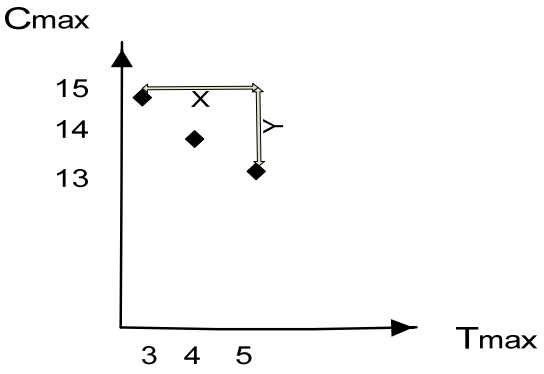

Fig.3. Pareto optimal front

3. Proposed Bi-Objective Genetic Algorithm

There are different methods to solve multi-objective optimization problems such as weighted average of the objectives, goal programming, ε-constraint method (Gendreau 2009; Mavrotas 2009), and many versions of multi-objective evolutionary algorithms (MOEAs) (Arabani et al., 2011; Hui et al., 2013; Rezaeian et al., 2013). Herein, by using the structure of non-dominated sorting genetic algorithm (NSGAII) presented by Deb et al. (2002), a bi-objective genetic algorithm (BOGA) is represented for this problem. Fig.4 shows diagram of the proposed BOGA.

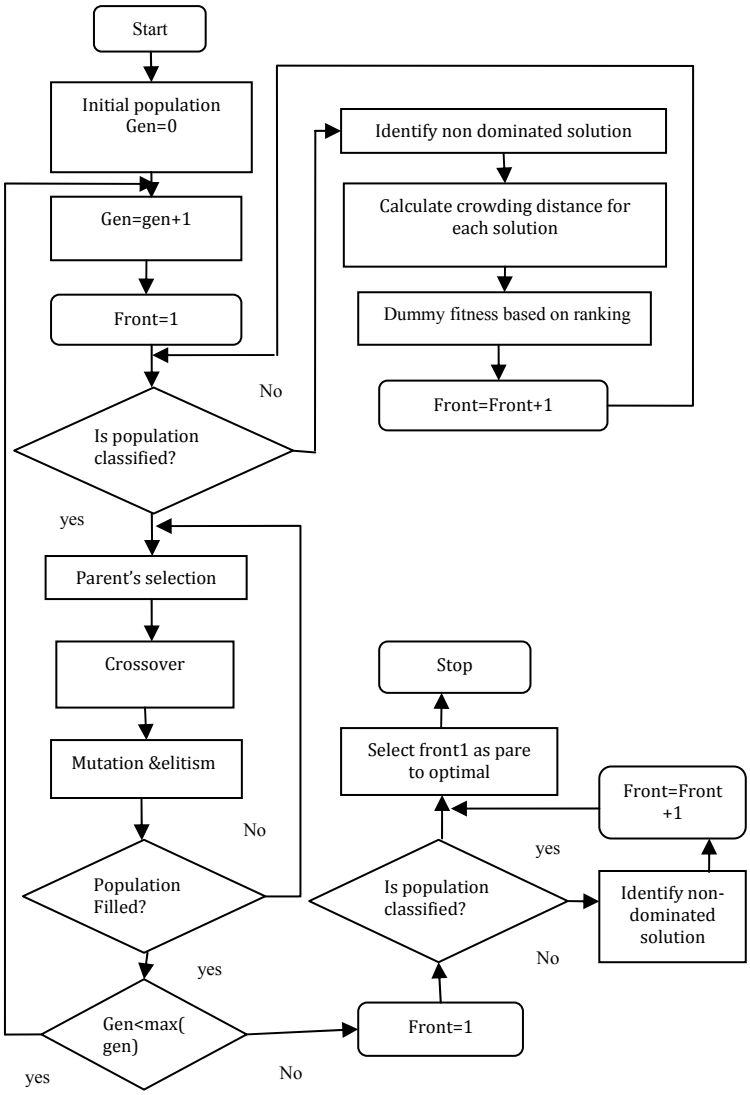

Fig.4. Structure of the proposed BOGA

3.1. Solution coding and initialization

Initialization is one of the main steps of GA which randomly generates initial population of individuals (P_t). Figures 5 and 6 show coding and decoding of a chromosome for a given example in section 2 respectively. Each chromosome consists of N batches where N random numbers from [0, 1] are used as keys to represent a sequence of batches, and there are $B.N$ gens where the values of gens are generated by the following pseudocode:

Pseudo code 1. Solution coding

For $i=1, 2\ldots, \boldsymbol{M}$

Number of $\sum_{j\in F_1,\ldots,F_{i-1}} S_j, \ldots, \sum_{j\in F_1,\ldots,F_i} S_j$ will be assigned to the gens of $\boldsymbol{B.N_{i-1}}, \ldots, \boldsymbol{B(N_{i-1}+N_i)}$ Randomly.

End

Batches belong to family1 $\{J_1, J_2, J_5\}$ Batches belong to family2 $\{J_3, J_4\}$

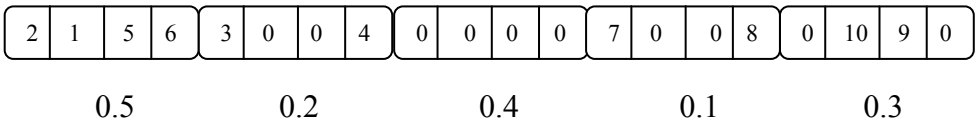

Fig. 5. Chromosome structure

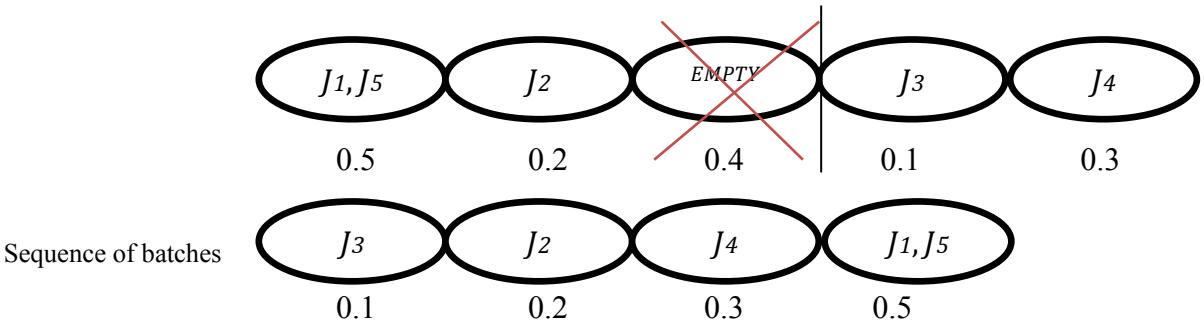

Fig. 6. Decode of the chromosome

3.2. Evaluation & selection

The solutions are arranged in different fronts by non-dominated sorting. For each solution of the i^{th} front, a dummy fitness value ($rank_i$) will be assigned whose value of Pareto front is lower than other fronts. Based on the fitness values, two parents are selected by the roulette wheel technique where chromosomes with higher fitness are most likely to be selected for crossover. The fitness value is calculated based on a geometric distribution as follows:

$$Fitness_i = (1/\text{population size}) * (1 - 1/\text{populationsize})^{rank_i}$$

3.3. Generation

By using the following operators, the new generation will be produced as follows:

Step1: some of elite solutions will be moved to the next generation according to elitism operator1 presented in subsection 3.3.4.

Step2: two selected parents by roulette wheel will be crossed by cross operator1 and cross operator2 with the same probability.

Step3: the generated offspring from step2 will be mutated.

Step4: the generated offspring of steps 2 and 3 will be checked by elitism operator2.

Step5: Repeat steps 2 and 4 until the next generation is completed.

3.3.1. Crossover operator1

This operator will combine two random parents for generating new offspring, as shown in Fig.7. The performance of this operator is illustrated according to pseudo code 2:

V_{l1i} The value of the l^{th} gen of the i^{th} family of 1^{th} parent.

V_{l2i} The value of the l^{th} gen of the i^{th} family of 2^{th} offspring.

V_{li} The value of the l^{th} gen of the i^{th} family of offspring.

V'_{li} The value of the l^{th} gen of the i^{th} family of modified offspring.

The values of gens of each offspring will be calculated as follows:

$$V_{li} = V_{l1i} + V_{l2i} \quad ; \quad \forall i = 1,..,N_f \ , \ \forall l = 1,..,N_i.B$$

In addition, the values of gens of modified offspring will be calculated by the following pseudocode:

Pseudo code 2. Crossover operator 1

$\boldsymbol{V'_{li} = [0]_{1\times N.B}}$
For $\forall \boldsymbol{i = 1, 2, \dots, M}$
For$\boldsymbol{z = 1, 2, \dots, \sum_{j\in F_i} S_j}$
$\boldsymbol{Max\{V_{li}\} = V_{l^*i}} \quad , \quad \boldsymbol{\forall l = 1, \dots, B.N_i}$
$\boldsymbol{V'_{l^*i} = \sum_{j\in F_{1,\dots,F_{i-1}}} S_j + z}$
$\boldsymbol{V_{l^*i}}$=0
End
End

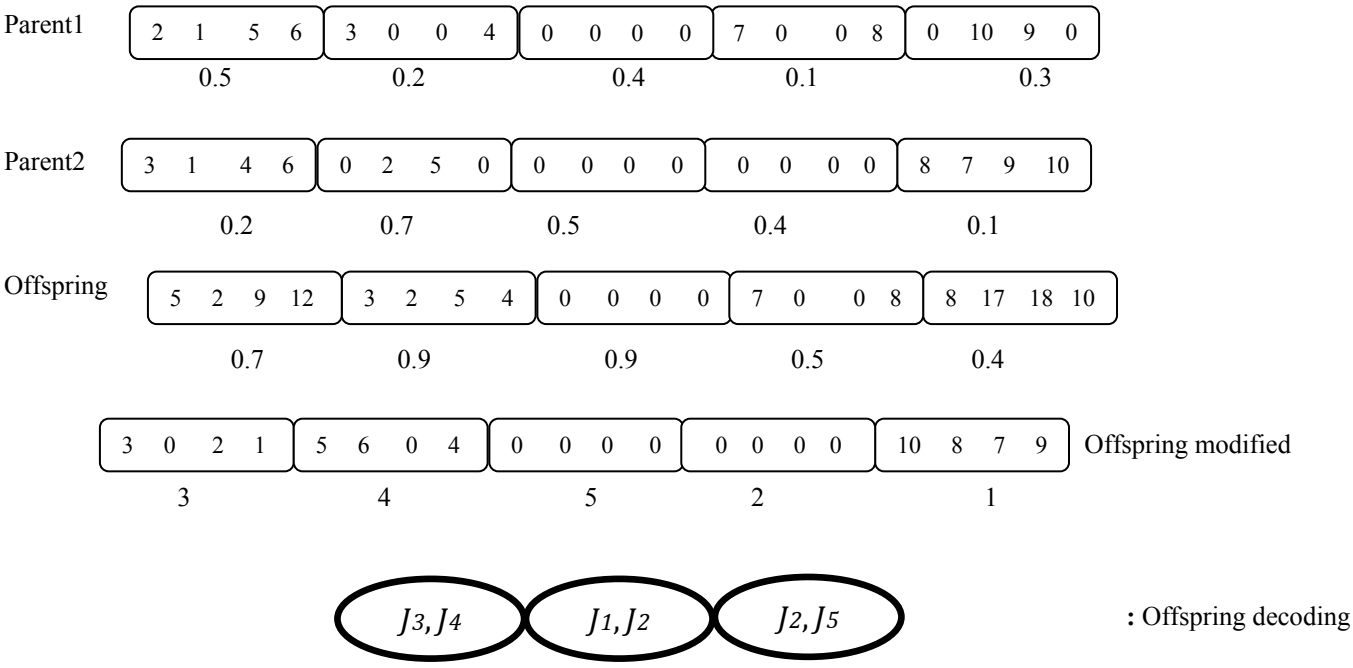

Fig. 7. Crossover operator1

3.3.2. Crossover operator 2

In this operator, two parents are combined to produce two offspring. At first, a random integer number between [1, N_f-1] will be produced and, then, the batches of two selected parents will be changed together from cut-point. This operator improves the sequence of job families as shown in Fig.8.

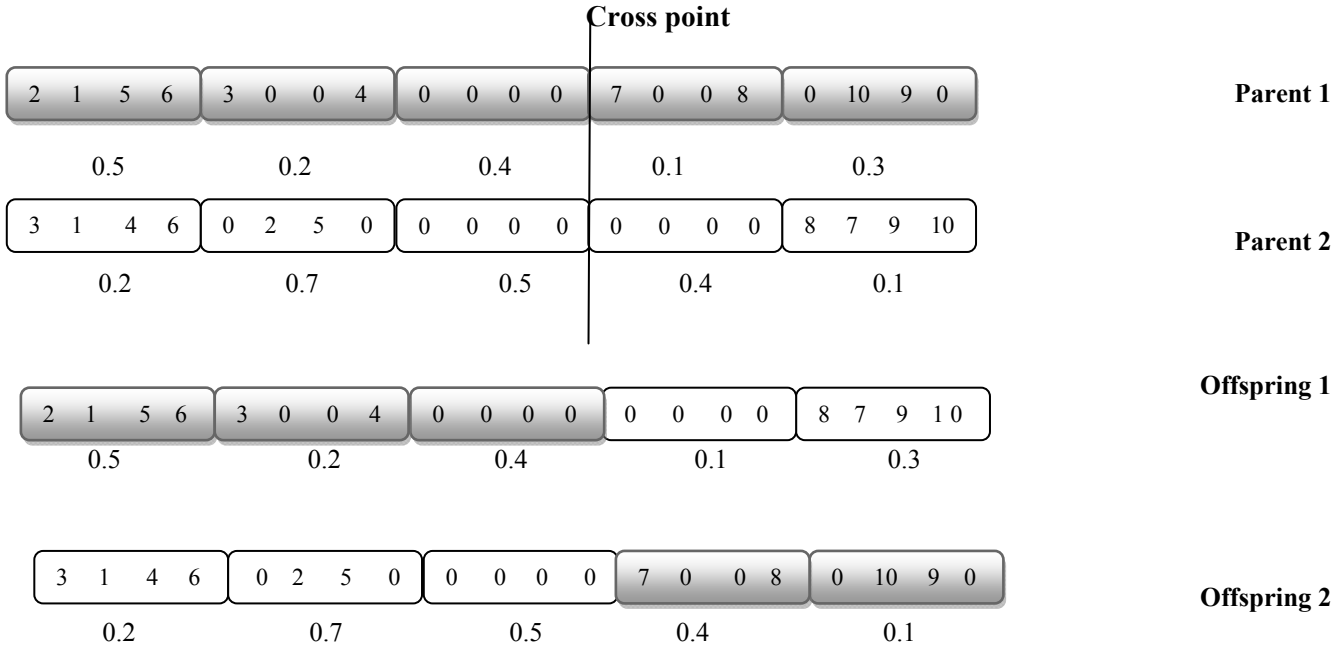

Fig. 8. Crossover operator 2

3.3.3. Mutation

To maintain local optimum, the multi-swapping mutation is used as a mutation operator. Herein, the contents of two

random genes of each family are swapped. Fig.9 shows the performance of this operator:

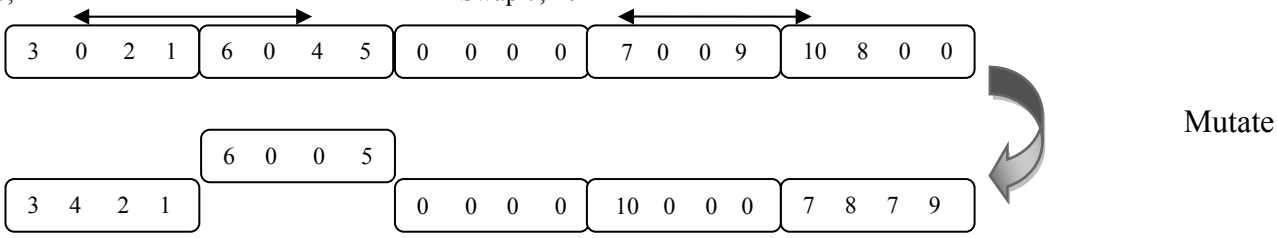

Fig. 9. Multiswapping mutation operator

3.3.4. Elitism operators

Herein, two elitism operators are applied as follows: at first, a number of the best solutions of the last generation are transferred to the next generation. Then, generated offspring must be compared with solutions of the worst frontier of the last generation; if it dominated all of them, then it will enter the next generation.

3.4. Parameters setting

The proposed algorithm uses parameters as number of generation, population size, crossover rate, or mutation rate. The performance of an Evaluation Algorithm (EA) is generally sensitive to the setting of the parameters influencing the search behavior and the quality of convergence. It is highly desirable to set these parameters to the levels that produce high-quality solutions. Herein, the parameters are set by test problems in subsection 4.1, and the results are given in Table 1.

Table 1
Parameters setting

Problem size	Cross over rate (1)	Cross over rate (2)	Mutation rate	Elitism	Population size	Number of generation
S	0.425	0.425	0.05	0.1	50	30
M	0.42	0.42	0.06	0.1	150	100
L	0.415	0.415	0.07	0.1	200	150

4. Computational Experiments

4.1. Test problem instances

To evaluate the effectiveness of the proposed BOGA, 18 numerical examples (small, medium, and large-scale sizes) are generated in which their data are produced randomly. To cover various types of problems, some factors are identified: number of jobs and number of family jobs. Also, job sizes, release dates, processing times, and due dates are generated from discrete uniform distribution. The parameters and their levels are shown in Table 2.

Table 2
Random test problem instances

Problem size	Number job	Number family	Parameters from discrete uniform			
S	4-5-6-7	2-3	s_j	r_j	p_j	d_j
M	8-10	2-3				
M	15-20-30	3	[1,5]	[1,15]	[1,10]	r_j+p_j+[1,20]
L	50-70-100	5				

4.2. Performance measures

In the literature of multi-objective optimization like those of Zitzler et al.(2003), Paquete (2005), Knowles et al. (2006),Favuzza et al. (2006), and Minella et al. (2008), or more recently, Rezaeian et al. (2013) are examples of the enormous efforts made to provide the necessary tools for a better evaluation and comparison of multi-objective algorithms.

According to the above studies, this paper considers four performance measures: the first performance measure is the diversification of solutions that is achieved by considering the Crowding Distance (CD) proposed by Favuzza et al. (2006) as follows:

$$\text{Diversification} = \sum_{\text{i}} \frac{\text{CD}_{\text{i}}}{\text{NNS}}$$

The second performance measure is the Domain Of points of each Solution (DOS) that is achieved by using the following equation:

$$DOS = X + Y$$

In the above equation, X and Y are maximum distances between non-dominated solution along the axis of $T_{max}$and C_{max} , as shown in Fig.3.

The third performance measure is the computational time, and the last performance measure is the number of non-dominated solutions (NNS), where these performance measures have been used to compare the proposed multi-objective algorithms by Favuzza et al. (2006) and Rezaeian et al. (2013).

4.3. Computational results and comparisons

To evaluate the proposed algorithms, ε-constraint and BOGA are coded in lingo9 and Matlab7, respectively. Each test problem is solved ten times through an HP 4520s laptop with 4GB of RAM and a 3GHz processor running in windows7. In Table 3, the average of results is reported under four considered performance measures.

Table 3
Computational results (S=small, M=medium, L=large)

Problem	Size problem			$\varepsilon-$ constraint				BOGA			
Number	N	**M**	Size	$Time_{(S)}$	NNS	DOS	Diversity	$Time_{(S)}$	NNS	DOS	Diversity
1	4	2	S	714	2	3	10	2	2	3	10
2	4	3	S	603	2	17	22	5	3	14	19
3	5	2	S	698	3	7	11	10	4	9	15
4	5	3	S	786	2	10	15	5	2	16	18
5	6	2	S	965	3	11	18	20	5	22	25
6	6	3	S	851	2	7	8	4	3	6	9
7	7	2	S	928	4	7.1	10	14	8	7.5	8
8	7	3	S	1949	5	10.8	12.8	25	10	9	10.5
9	8	2	M	2920	4	17	15.4	26	8	26	25.3
10	8	3	M	5015	4	15	20	52	6	17	23
11	10	2	M	3060	3	10.8	18.5	36	9	14	24.6
12	10	3	M	3600	4	6	7	80	6	6.8	7.9
13	15	3	M	3851	4	13	20	23	7	17	25
14	20	3	M	3625	5	14.5	26	12	11	20.1	32.1
15	30	3	M	3967	4	12	19	67	12	15	24
16	50	5	L	4236	6	9	17	86	18	19	45
17	70	5	L	5469	9	12	19	187	21	14	42.6
18	100	5	L	5224	7	12.5	22.5	500	13	15	45.8

The relative variations of criteria for algorithms are calculated by the following relation and are shown in Table 4. In this table, negative values illustrate the smaller value of BOGA algorithm than $\varepsilon-$constraint method, and vice versa.

$$(BOGA_{\,i} - \varepsilon\_constraint_i)/\varepsilon\_constraint_i * 100$$

Table 4
Variation percent of the performance measures

Test problem	Variation percent Time	Variation percent NDS	Variation percent DOS	Variation percent Diversity
1	-99.7	0.0	0.0	0.0
2	-99.1	50.0	-17.6	-13.6
3	- 98.5	33.3	28.5	36.3
4	-99.3	0.0	60.0	20.0
5	-97.9	66.6	100.0	38.8
6	-99.5	50.0	-14.2	12.5
7	-98.4	100.0	5.6	-20.0
8	-98.7	100.0	-16.6	-17.9
9	-99.1	100.0	52.9	64.2
10	-98.9	50.0	13.3	15.0
11	-98.8	200.0	29.6	32.9
12	-97.7	50.0	13.3	12.8
13	-99.4	75.0	30.7	25.0
14	-99.6	120.0	38.6	23.4
15	-98.3	200.0	25.0	26.3
16	-97.9	200.0	111.1	164.7
17	-96.5	133.3	16.6	124.2
18	-87.8	85.7	20.0	103.5

Figs.11-14 are extracted from Table 4. It is seen that there is a significant difference between the proposed BOGA and $\varepsilon-$ constraint method; Fig.11 shows the CPU time of the BOGA which is smaller than $\varepsilon-$ constraint method in all of the instances; the proposed BOGA produces more points than the $\varepsilon-$ constraint method (Fig.12), except two cases (i.e., test numbers 1 and 4) where both algorithms are equal. In addition, other performance measures for the proposed BOGA work better than $\varepsilon-$ constraint method (Fig.13-14). In other words, the proposed BOGA produces more solutions in a wide range of different values, providing a higher degree of diversification of solutions. This does not mean that the quality of BOGA is better than $\varepsilon-$ constraint method. The difference between these algorithms for test number 13 is shown in Fig. 10. For this case, $\varepsilon-$ constraint method generates non-dominated ***optimal*** solutions in an unreasonable time, while BOGA obtains many diverse solutions in a reasonable time. Table 3 clarifies that the proposed BOGA performs better than $\varepsilon-$ constraint method in most cases according to 4 defined criteria. In few cases, $\varepsilon-$ constraint method performs better than the proposed BOGA (i.e., test numbers 2 and 8). Some performance measures of $\varepsilon-$ constraint method are carred out better than the proposed BOGA; differences are inconsiderable. For example, in test number 8,$\varepsilon-$ constraint method has produced a domain and diversification value of 10.8 and 12.8, respectively, where both of them are slightly better than the corresponding value for the proposed BOGA; however, the proposed BOGA is strongly better in the NNS measure and CPU time than $\varepsilon-$ constraint method. Finally, this paper considered $\varepsilon-$ constraint method as superior algorithm to generate non-dominated ***optimal*** solutions in an unreasonable time, while BOGA is identified as superior algorithm to obtain many diverse solutions in a reasonable time.

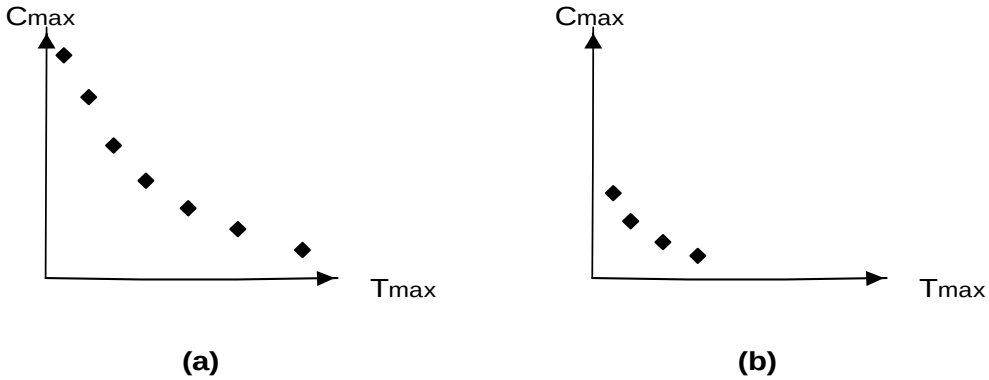

Fig.10. Comparision of (a) BOG and (b) constraint method

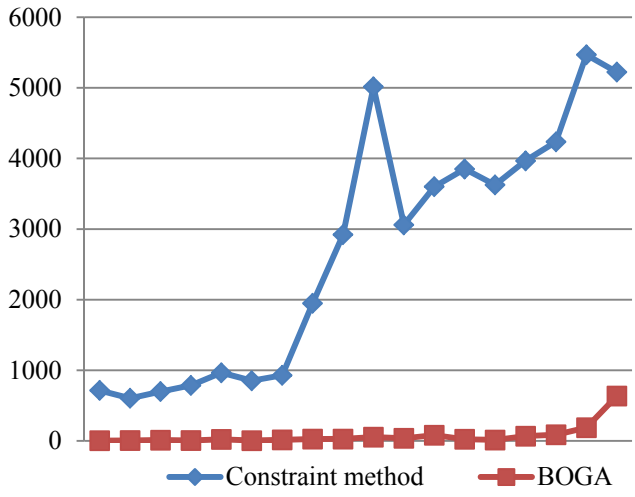

Fig.11. CPU time of proposed algorithms

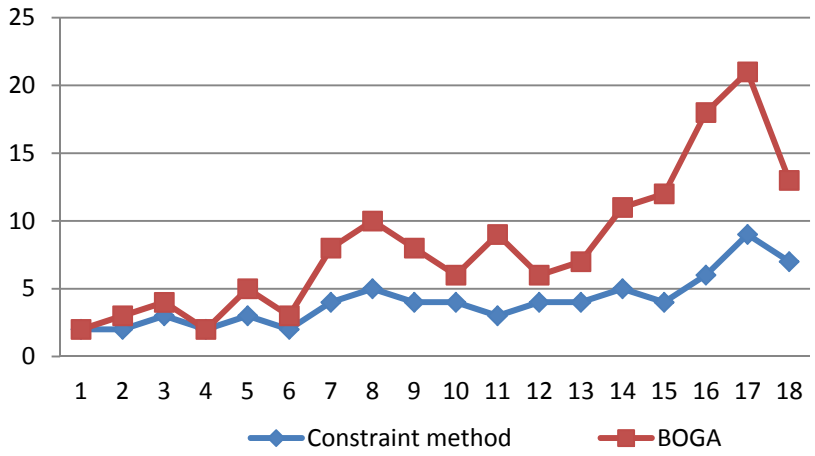

Fig.12: NNS mesure of proposed algorithms

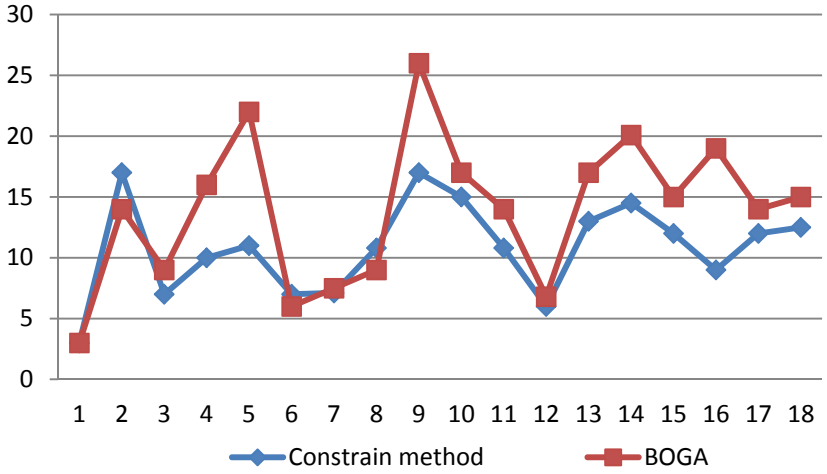

Fig.13: DOS of proposed algorithms

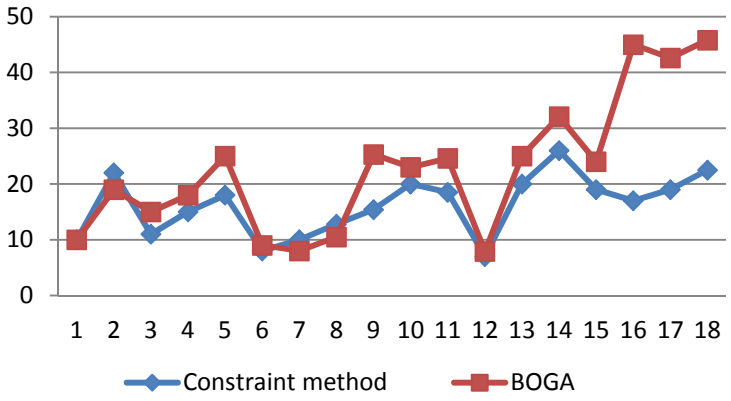

Fig.14: Diversity of proposed algorithms

5. Conclusions and Future Research

In this paper, a new bi objective mixed integer linear programming (MILP)model was proposed for a single batch processing machine scheduling problem. The considered assumptions are as follows: non-identical job sizes, incompatible job family, family setup time, split job size, and dynamic job arrivals. At first, $\varepsilon-$ constraint method was applied to solve the problems, since the problem is NP-hard, then a BOGA algorithm was also presented to solve the problems in real size. According to the considered performance measures, two algorithms were compared. The comparison results demonstrated the effectiveness of the proposed BOGA. Some directions that lead to the ability to discover better solutions can be summarized as follows: using a mechanism for generating the better initial population, localizing the search in crossover and mutation stages using heuristics and using an effective local search heuristic, which has the ability of steering the search quickly towards the Pareto- optimal solutions. Moreover, extension of the proposed model to the case of scheduling other shop systems, e.g., flow shop and parallel batch processing machines are encouraged. Developing heuristic algorithms to minimize maximum tardiness on a batching machine and providing efficient lower bounds for the evaluation of algorithms would be interesting directions.

References

Arabani, A., Zandieh, M., and Fatemi Ghomi, S. M. T. (2011). Multi-objective genetic-based algorithms for a cross-docking scheduling problem. *Applied Soft Computing*, 11(8), 4954-4970.

Baker, R. (1943). Principles of sequencing and scheduling. *Wiley, New Jersey*.

Cheng, B. , Wang, Q., Yang, Sh., and Hu, X. (2013).An improved ant colony optimization for scheduling identical parallel batching machines with arbitrary job sizes. *Applied Soft Computing*, 13(2), 765-772.

Crauwels, H. A. J., and Potts, C. N. (1996). Local search heuristics for single-machine scheduling with batching to minimize the number of late jobs. *European Journal of Operational Research*, 90(2), 200-213.

Damodaran, P., Manjeshwar, P. K., and Srihari, K. (2006). Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms. *International Journal of Production Economics*, 103(2), 882–891.

Dauzère-Pérès, S., and Mönch, L. (2013). Scheduling jobs on a single batch processing machine with incompatible job families and weighted number of tardy jobs objective. *Computers& Operations Research*, 40(5), 1224-1233.

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T. (2002). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. *Lecture notes in computer science* 1917 (2000): 849-858.

Dupont, L., and Dhaenens, C. (2002). Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure. *Computers & Operations Research*, 29(7), 807-819.

Favuzza, S., Ippolito, M. G., Riva Sanseverino, E. (2006). Crowded comparison operators for constraints handling in NSGA-II for optimal design of the compensation system in electrical distribution networks. *Advanced Engineering Informatics*, 20(2), 201–211.

Gendreau, M. (2009). An exact -constraint method for bi objective combinatorial optimization problems. *European Journal of Operational Research*, 194(1), 39-50.

Guo, Z. X., Wong, W. K., and Leung, S. Y. S. (2013). A hybrid intelligentmodel for order allocation planning in make-to-order manufacturing.*Applied Soft Computing,* 13(3), 1376-1390.

Hui, Lu., Ruiyao,Niu., Jing, Liu., Zheng, Zhu. (2013). A chaotic non-dominated sorting genetic algorithm for the multi-objective automatic test task scheduling problem. *Applied Soft Computing*, 13(5), 2790-2802.

HusseinzadehKashan, A., Karimi, B., and Jolai, F. (2010). An effective hybrid multi-objective genetic algorithm for bi-criteria scheduling on a single batch processing machine with non-identical job sizes. *Engineering Applications of Artificial Intelligence*, 23(6), 911–922.

Husseinzadeh Kashan, A. , Karimi, B., and Jenabi, M. (2008). A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. *Computers & Operations Research*, 35(4), 1084–1098.

Knowles, J., L. Thiele, E. Zitzler. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report 214, *Computer Engineering and Networks Laboratory* (TIK), ETH Zurich. [Revised version.]

Koh, Sh., Koo, P. H., Kim, D. Ch., and Hur, W. Su. (2005). Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. *International Journal of Production Economics*, 98(1), 81–96.

Liu, L. L., Ng, C. T., and Chen, T. C. E. (2009). Bi-criterion scheduling with equal processing times on a batch processing machine. *Computers & Operations Research*, 36(2), 110–118.

Malvea, S., and Uzsoy, R. (2007). A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. *Computers & Operations Research*, 34(10), 3016 – 3028.

Mavrotas, G. (2009). Effective implementation of the -constraint method in multi objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2), 455-465.

Melouk, Sh., Damodaran, P., and Chang, P. Y. (2004). Minimizing makespan for single machine batch processing with non-identical job sizes using

simulated annealing. *International Journal of Production Economics*, 87(2), 141–147.

Minella et al. (2008). A Review and Evaluation of Multi-objective Algorithms for the Flow-shop Scheduling Problem, *Journal on Computing*, 20(3),451-471.

Paquete, L. F. (2005). Stochastic local search algorithms for multi-objective combinatorial optimization: Method and analysis. Ph.D.thesis, *Computer Science Department, Darmstadt Universityof Technology, Darmstadt, Germany.*

Perez, I., Fowler, W., and Matthew, W. (2005). Minimizing total weighted tardiness on a single batch process machine with incompatible job families. *Computers & Operations Research*, 32(2), 327–341.

Pinedo, M. L. (2008). Scheduling Theory, algorithms, and systems. *3ed, Springer, New York.*

Rafiee, N., Karimi, B., and HusseinzadehKashan, A. (2010). A branch and price algorithm to minimize makespan on a single batch processing machine with non-identical job sizes. *Computers & Operations Research*, 37(10), 1720–1730.

Rezaeian, J., Javadian, N., Tavakkoli-Moghaddam, R., and Jolai F. (2013). A hybrid multi-objective approach based on the genetic algorithm and neural network to design an incremental cellular manufacturing system. *Computers & Industrial Engineering*, 66(4), 1004-1014.

Rui, X., Huaping, Ch., and Xueping, Li. (2012). Makespan minimization on single batch-processing machine via ant colony optimization. *Computers & Operations Research*, 39(3), 582–593.

Sung, C. S., Choung, Y. I., Hong, J. M., and Kim, Y. H. (2002). Minimizing makespan on a single burn-in oven with job families and dynamic job arrivals. *Computers & Operations Research*, 29(8), 995-1007.

Wang, Ch. Sh., and Uzsoy, R. (2002). A genetic algorithm to minimize maximum lateness on a batch processing machine. *Computers & Operations Research*, 29(12), 1621-1640.

Xu, R., Chen, H., and Li, X. (2013). A bi-objective scheduling problem on batch machines via a Pareto-based ant colony system. *International Journal of Production Economics*, 145(1), 371-386.

Yao, Sh., and Jiang, Z. (2012). A branch and bound algorithm for minimizing total completion time on a single batch machine with incompatible job families and dynamic arrivals. *Computers & Operations Research*, 39(5), 939–951.

Zitzler, E., L. Thiele, M. Laumanns, C. M. Fonseca, V. G. da Fonseca. (2003). Performance assessment of multi-objective optimizers:An analysis and review. *IEEE Trans. Evolutionary Comput.* (7) 117–132.

This article can be cited: Rezaeian, J.&, Zarook, Y.(2018).An Efficient Bi-Objective Genetic Algorithm for the Single Batch-Processing Machine Scheduling Problem with Sequence-Dependent Family Setup Time and Non-Identical Job Sizes.*journal of Optimization in Industrial Engineering*. 11(2),2018, 63-76.

URL:http://qjie.ir/article_538503.html
DOI: 10.22094/joie.2018.792.1505