A Novel Imperialist Competitive Algorithm to Solve Flexible Flow Shop Scheduling Problem in Order to Minimize Maximum Completion Time

S.F.Attar
Department of Industrial Engineering, Faculty of Engineering, Tarbiat Moallem University, Karaj, Iran

M.Mohammadi
Department of Industrial Engineering, Faculty of Engineering, Tarbiat Moallem University, Karaj, Iran

R.Tavakkoli-Moghaddam
Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran

ABSTRACT
This paper demonstrates solving the flexible flow shop scheduling problem (FFSP) with considering limited waiting time constraint, sequence dependent setup times and different ready time to minimize maximum completion time (i.e. makespan). Since the problem studied is NP-hard, metaheuristic algorithms are proper to solve this class of problems. Hence, in this paper, a novel imperialist competitive algorithm (ICA) is proposed to tackle of addressed problem. In order to achieve the reliable results in our proposed algorithm, a comprehensive tuning is performed using Taguchi method. to validate this proposed algorithm, the other popular algorithm namely simulated annealing is developed for this goal. Simulation results indicated that ICA is superior to SA.

General Terms
Algorithms, Scheduling.

Keywords
ICA, Flexible flow shop, Limited waiting time, Sequence dependent setup times, Ready time

1. INTRODUCTION
In this investigation, we consider a flexible flow shop scheduling (FFS) problem, also called flow shop with multiple processors. A FFS consists of multiple stages with each stage contains parallel machines. There are multiple jobs immediately available for scheduling. Each job consists of a chain of operations. Each job must be processed by only one machine in each stage and it must go through all stages on this order. Each operation takes a specific setup time and processing time on a stage. Preemption is usually not allowed. Two types of decisions must be made for FFS. First is assignment of each job to a specific machine at each stage and the other is sequencing of jobs on each machine [1]. Arthanari and Ramamurthy [2] and Salvador [3] were among the initial whom identified this problem. FFS is a combination of two scheduling problems: the flow shop scheduling problem and the parallel machines scheduling problem. In flow shop scheduling problems, a series of different machines is arranged in multiple stages with only one machine at each stage. In parallel machines scheduling problems, there is a series of identical machines and they are all at the same stage.

This kind of manufacturing environment is fairly prevalent in the chemical processing and petroleum industries, flexible manufacturing and assembly environments, and in packaging industries. Salvador [3] had identified the FFS problem in the polymer, chemical, and petrochemical industries. In these environments, there is more than one parallel plant which can be considered as flow shops and the jobs can be easily processed through any one of the plants at each processing stage. A flow shop in which parallel machines are added at one or more stages to reduce the on bottle-neck facilities or to increase the production capacities can be also viewed as the application of the FFS problem. Applications of the FFS problems are also found in computer systems and telecommunication networks [4].

Botta-Genoulaz [5] presented six heuristic algorithms to minimize the maximum tardiness in a flexible flow-shop problem with different due dates. Kochhar et al. [6] provide a local search approach to a realistic flexible flow line problem with setups, buffer capacities, blocking, breakdowns and downtimes. Wittrock [7] developed heuristics to minimize makespan in the k-stage hybrid flow shop (HFS) with identical machines in each stage. Brah and Hunsucker [8] developed a branch and bound for a k-stage hybrid flow shop to minimize makespan. Rajendran and Chaudhuri [9] developed branch and bound algorithms to minimize makespan and total flow time, respectively, for a k-stage problem. Vignier et al. [10] studied a k-stage HFS to minimize the total completion time.

Moursli and Pochet [11] presented a branch-and-bound algorithm to minimize the objective of the makespan. Og˘uz et al. [12] developed nine heuristic algorithms for solving the two-stage HFS problem of minimizing makespan. Gupta et al. [13] also proposed heuristics to minimize makespan in a two stage HFS with parallel identical machines at the first stage. Ruiz and Maroto [14] proposed some genetic algorithms for a HFS problem with unrelated parallel machines per stage, sequence-dependent setup times and machine eligibility. The researchers conduct several experiments with a set of random instances as well as with real data taken from companies of the ceramic tile manufacturing sector. Although many realistic considerations and constraints are addressed in several papers in literature, very few papers considered such realistic constraints jointly. Moreover, majority of researches in scheduling problems area have assumed that the waiting time for each job between every two successive stages is infinite or processing should be carry

out without waiting time (i.e. no-wait) [15-20]. However, in real manufacturing systems, such as steelmaking process, the waiting time in buffers are limited to keep the steel liquid hot enough for processing [21]. Su [22] considered a hybrid two-stage flow shop with limited waiting time constraint. This paper presented a heuristic algorithm and a mixed integer program to minimize the makespan. Behmanian and Zandieh [23] suggested an imperialist competitive algorithm for minimizing summation of earliness and quadratic tardiness in hybrid flow shop scheduling problem. Results of their study indicated that ICA is superior that the other algorithms which were applied. With respect to literature, it could be seen there is not any study that mentioned flexible flow shop scheduling problem with all of following characteristics: Considering different ready times for jobs, limited waiting time constraint, sequence dependent setup times and makespan criterion. Our goal in this paper is to develop an efficient metaheuristic to solve the flexible flow shop scheduling problem with sequence dependent setup time ,limited waiting time and different ready time to minimize maximum completion time. The paper has the following structure: Framework of the problem that is studied in this paper is presented in Section 2. Section 3 introduces the proposed algorithm. Section 4 presents the experimental design and computational results. Finally, Section 5 is devoted to conclusions and future works.

2. Problem description

To define the hybrid flow shop scheduling problem, assume that a set $N = \{1, 2, .., n\}$ of n jobs which are available in different times (i.e. $R = \{r_1, r_2, .., r_n\}$) must be sequentially processed on a set of s stages $S = (1, 2, ..., s)$. Each job is processed first at stage 1, then at stage 2,…, and finally at stage s. at stage *i*, m_i identical parallel machines are available. Each job $i \in N$ could be only processed on one machine at a time and consists of s operation $(O_{1j}, O_{2j}, ..., O_{sj})$. An operation O_{ij} has a processing time p_{ij} and has to be processed without preemption on only one of the machines at stage *i*. moreover, the setup times for job *j* after job *k* at stage l is defines with $sdst_{kjl}$ and limited waiting time for job *j* between stage *i* and *i+1* is illustrated with $lwt_{i,i+1,j}$.

Since a relatively simple HFS, such as a two-stage HFS with limited waiting time, is NP-hard in the strong sense [22], our problem at least has same difficulty. So, a novel metaheuristic algorithm is applied to solve addressed problem. The structure of imperialist competitive algorithm is shown in next section.

3. Imperialist competitive algorithm

The optimization algorithms mainly are inspired from nature procedures or life of animals. In these algorithms, socio-political and cultural concepts are not considered. Recently, a new optimization methodology namely imperialist competitive algorithm originates the socio-political evolution. ICA has been modeled mathematically by atashpaz-gargary [24] which is utilizing this historical phenomenon as powerful tool for solving the optimization problems. This algorithm recently has been attracted to many researchers to tackle of optimization problems such as scheduling problem [23-25-26]. Briefly, this algorithm starts with initial solutions which are called initial countries that are similar to chromosome in genetic algorithm and particle in particle swarm optimization algorithm. These countries are divided into two groups. First group is belonged to imperialist countries and second group is formed with membership of colonies countries as shown in the figure 1. Imperialist countries with applying assimilation strategy, try to decrease the gaps between colonies and them. Imperialistic competition beside the assimilation and revolution form the main core of ICA that make to reach the reliable and efficient solutions. Stages of our proposed algorithm are explained as follows: generating initial empires, assimilation, revolution, exchange between the best colony and imperialist, Imperialistic competition, elimination of powerless empire.

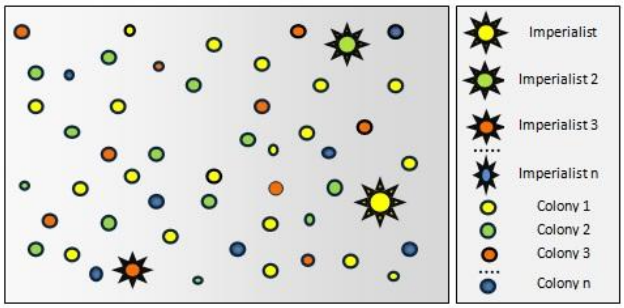

Fig 1: Generating the initial countries.

3.1 Generating initial empires

In ICA each solution is shown by an array. Each array be composed of amounts of variables to be optimized. These values are defined with characteristics of each specific problem. In GA terminology, this array is called "chromosome," while in ICA, "country" plays same role. In an *N* dimensional optimization problem, a country is a $1 \times N$ array. This array is defined with:

$$country = [v_1, v_2, v_3, ..., v_N] \quad (1)$$

Where v_i is the variable that to be optimized (i.e. v_i is similar to gen in GA). Each variable in a country denotes a socio-political characteristic in that country such as culture, language, business, economical policy and etc.

In proposed ICA, each solution represents sequence of jobs which to be assigned to earliest available machines. In order to reach this sequence, firstly, values of variables are generated randomly by uniform distribution function in range between zero and one. Secondly, these values are interpreted by sorting of them. Fig 2 indicates the initial structure (Fig 2(a)) and decoded structure (Fig 2(b)) for a problem with five jobs.

(a) Initial structure	0.87	0.45	0.23	0.47	0.64
(b) decoded structure	5	2	1	3	4

Fig 2: The structure of one solution for a problem with five jobs in ICA

The fitness function of each country is calculated using function f at the variables $(v_1, v_2, ..., v_N)$ as follow:

$$c_i = f(country_i) = f(v_{i1}, v_{i2}, ..., v_{iN}) \quad (2)$$

At first, algorithm generates initial countries randomly in number of population size then the most powerful countries are selected in number of N_{imp}. Remaining countries will form colonies. The colonies are randomly distributed among imperialists based on imperialist's power. For calculating the power of imperialists, first, the normalized cost of an imperialist is applied based on Eq.3.

$$C_n = \max c_i - c_n \qquad i = 1, 2, ..., N_{imp} \quad (3)$$

Where, c_n is the cost of nth imperialist and C_n is its normalized cost which is equal to the deviation of the maximum total completion time from the nth imperialist cost. Then the power of each imperialist is calculated according to Equation 4.

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \right|, \quad \sum_{i=1}^{N_{imp}} p_i = 1 \quad (4)$$

By attention to imperialist's power, the colonies are distributed among the imperialist. In addition, the initial number of colonies of an imperialist is calculated as follow:

$$NC_n = round \quad P_n . N_{col} \quad (5)$$

Where, NC_n is the initial number of colonies of nth empire and N_{col} is the number of all colonies. We randomly select NC_n of colonies and designate them for nth imperialist. As shown in this figure 3 empire with the bigger power has a larger number of colonies while empire with weaker power has less. Fig 3 shows the initial population of each empire.

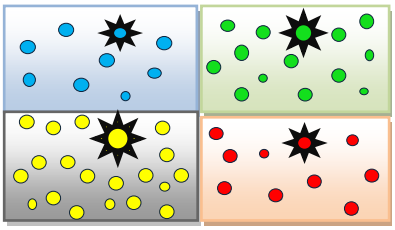

Fig 3: Generating the initial empires.

3.2 Assimilation

Imperialists try to improve all of their colonies. The aim of assimilation procedure is to assimilate the colonies's characteristic toward their imperialist such as culture, social structure, language and etc. As shown in Fig 4 each colony moves toward the imperialist by x units. x is a random number with uniform distribution. $x \sim U(o, \beta \times d)$, $\beta > 1$ Where β is a number greater than 1 and d is distance among colony and imperialist which is the vector of movement for colony toward imperialist. Parameter β causes the colony to get closer to imperialist from both sides.

To intensify property of this method and to search wider area around current solution we add a random amount of deviation θ to the direction of movement. θ is number with uniform distribution. $\theta \sim U(-\gamma, \gamma)$ where γ is a parameter that adjusts the deviation from the original path.

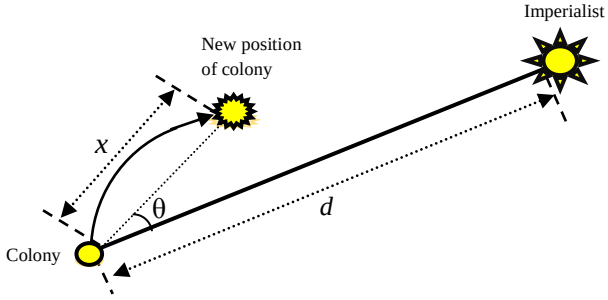

Fig 4: Moving colonies direction.

3.3 Revolution

This mechanism is similar to mutation process in genetic algorithm for creating diversification in solutions. In each iteration, for every colony a random number which is varying between zero and one is generated, then this value is compared with probability of revolution (i.e. P_R). If random number is lower than P_R, the procedure of revolution is performed. For conducting the revolution procedure, at first, the number of variables which should be changed are determined based on revolution ratio (R_R). In other words, R_R multiplies in number of jobs. After determining the number of elements for revolution, these elements are selected randomly. Then values of selected elements are changed randomly. The new colony will replace with previous colony while its cost is improved.

3.4 Exchanging positions of the imperialist and a colony

After assimilation for all colonies and revolution for a percentage of them, the best colony in each empire is compared with its imperialist. If the best colony is better than its imperialist, then the position of best colony and imperialist are exchanged.

3.5 Total power of an empire

The total power of an empire is calculated to apply in the imperialistic competition section. It is clear that the power of an empire is including the imperialist power and their colonies. Moreover, obviously the power of imperialist has main effect on total power of an empire while colonies power has lower impact. Hence, the equation of the total cost is defined as follow:

$$TC_n = \cos t(imperialist_n) + \xi mean\{\cos t(colonies \ of \ empire_n)\} \quad (6)$$

Where TC_n is the total cost of the nth empire and zeta (ξ) is a positive number which is considered to be less than 1. The total power of the empire will be determined by just the imperialist when the value of ξ is small. The role of the colonies, which determines the total power of an empire, becomes more important as the value of ξ increases.

3.6 Imperialistic competition

Imperialists try to increase their power by possessing and control the colonies of other empires. To apply this concept in our proposed algorithm, in each iteration, at first the weakest colony of weakest empire is determined. Then this colony is given to the other empires which depend on their total power. For this purpose we should calculate the possession probability of each empire, first the normalized total cost is calculated as follows:

$$NTC_n = \max\ TC_i\ - TC_n\ ,\ \ i = 1,2,..,N_{imp} \tag{7}$$

Where, NTC_n is the normalized total cost of nth empire and TC_n is the total cost of *n*th empire. By having the normalized total cost, the possession probability of each empire is calculated as below:

$$p_{emp_n} = \left| \frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i} \right| \tag{8}$$

We use Roulette wheel method for assigning the mentioned colony to the empires.

3.7 Eliminating the powerless empires

When each empire loses all of colonies this empire will collapse and its imperialist is considered as a colony and is assigned to other empires.

3.8 . Stopping criteria

Expiry criterion in our proposed algorithm is to get the maximum decades which is defined by user.

4. EXPERIMENTAL EVALUATION

In this section, in order to evaluating the performance of proposed algorithm against simulated annealing, at first some problems are generated randomly, then the results are obtained by these algorithms is converted to relative percentage deviation. The formulation of this transformation is as bellow:

$$RPD = \frac{|Method_{sol} - Best_{sol}|}{Best_{sol}} \times 100 \tag{9}$$

Also, $\overline{RPD}$ is defined according to below Equation.

$$ARPD = \overline{RPD} = \frac{\sum_{i=1}^{number\ of\ run} RPD}{number\ of\ run} \tag{10}$$

For conducting of experiments, algorithms are implemented in MATLAB 2009b and run in a personal computer with 2.66 GHz and 4 GB of RAM memory.

4.1 Data generation

For generating the random problems five parameters are considered. These parameters are number of jobs, number of stage, number of machines in each stage, SDST distribution function, processing time distribution function, LWT distribution function and ready time distribution function. The parameters and their levels are demonstrated in Table 1.

Table 1. Factors and their levels.

Factors	Levels
Number of jobs	6,30,100
Number of stages	2,4,8
Machine	constant (4) , variable (U(1,10))
Processing time	U(20,100)
SDST	U(10,40)
LWT	U(0,20)
Ready time	U(0,50)

4.2 Parameter tuning

One of important component of metaheuristic algorithm is calibration of parameters which impresses on performance of algorithm. In this investigation we employed the Taguchi method for this goal.

Taguchi [27] developed a family of FFE matrices which eventually reduce the number of experiments, but still provide sufficient information. In Taguchi method, orthogonal arrays are used to study a large number of decision variables with a small number of experiments. In Taguchi Method, the word "optimization" implies "determination of best levels of control factors". In turn, the best levels of control factors are those that maximize the Signal-to-Noise ratios. The Signal-to-Noise ratios are log functions of desired output characteristics. The experiments that are conducted to determine the best levels, are based on "Orthogonal Arrays", are balanced with respect to all control factors and yet are minimum in number. This in turn implies that the resources (materials and time) required for the experiments are also minimum.

Taguchi has created a transformation of the repetition data to another value which is the measure of variation. The transformation is the signal-to-noise (S/N) ratio which explains why this type of parameter design is called robust design [28, 29]. Here, the term ‘‘signal’’ denotes the desirable value (mean response variable) and ‘‘noise’’ denotes the undesirable value (standard deviation). So the S/N ratio indicates the amount of variation presents in the response variable. The aim is to maximize the signal-to-noise ratio.

Taguchi classifies objective functions into three categories: the smaller-the-better type, the larger-the-better type, and nominal-is best type. Since almost all objective functions in scheduling are classified in the smaller-the-better type, their corresponding S/N ratio [28] is:

$$S/N\ ratio = -\log_{10}(objective\ function)^2 \tag{11}$$

The parameters in our proposed calibration of parameters and their levels are shown in Table 2. After experimental design for mentioned problem, the results obtained by Taguchi method indicated that A (1), B (2), C (3) and D (1) is the best combination of parameter’s values (see Fig 5).

Table 2. Parameters and their levels

	Parameters			
	A	**B**	**C**	**D**
Level	**(Max_{DC},PopSize)**	**N_{imp}**	**ξ**	**P_R**
1	(200,50)	3	0.1	0.2
2	(100,100)	4	0.15	0.3
3	(50,200)	5	0.2	0.4

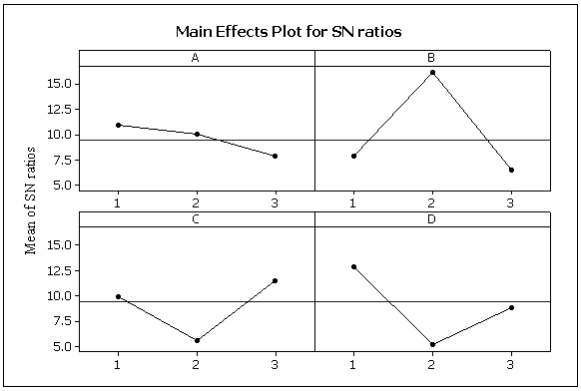

Fig 5: The mean S/N ratio plot for each level of the factors.

4.3 Performance analysis

After running the algorithms in random generated problems, results are obtained by algorithms are converted to RPD for analyzing the performance of algorithms. For this aim, Tuckey 95% confidence intervals of algorithms are employed. This analyze which are illustrated in Fig 6, indicated that ICA statistically outperformed SA. Furthermore, two sensitive analysis for evaluating the variation of job number and machine distribution on performance of algorithms are developed. As seen in Fig 7 and Fig 8, both of them have not specific trend versus variation of job numbers and machine distribution, but it could be said, ICA in most of situations obtains better solutions against SA. So, for the problem that has been studied in this paper, ICA as an efficient metaheuristic algorithm is recommended to researchers and practitioners.

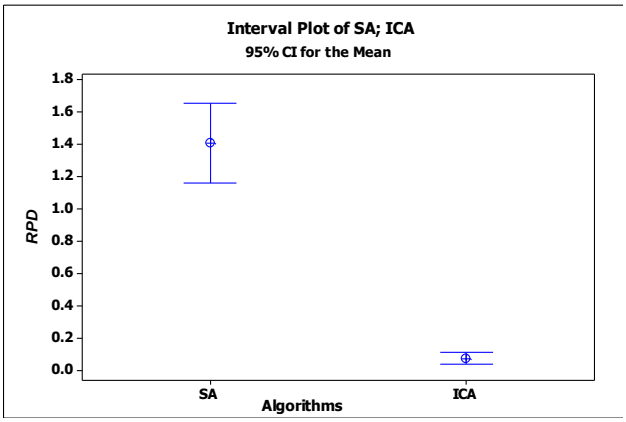

Fig 6: Means plot and Tukey intervals (at the 95% confidence level) for the type of algorithm factor.

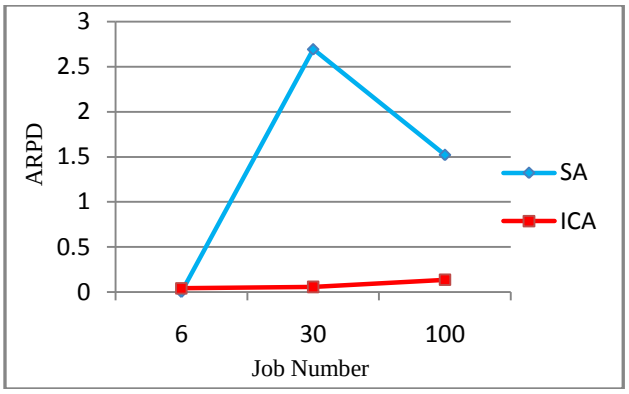

Fig 7: Interaction between job number and types of algorithms in terms of ARPD.

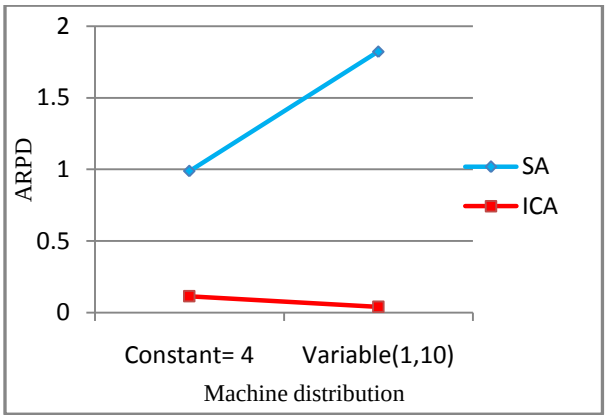

Fig 8: Interaction between machine distribution and types of algorithms in terms of ARPD.

5. CONCLUSION AND FURTHER RESEARCH

Flexible flow shop scheduling problem is very important in both fields of combinatorial optimization and engineering management. Most literature focused on solving this problem in unlimited waiting time or no-wait. Hence, in this paper we considered this problem with sequence dependent setup times, limited waiting time and different ready time. For tackle of addressed problem, a novel metaheuristic algorithm namely imperialist competitive algorithm is proposed. Furthermore, to reach the more reliable and accurate results a comeprhensive calibration methodology namely Taguchi method has been employed for this purpose. Computational experiments illustrated that ICA statistically outperforms SA. As a direction for future studies, it could be interesting to work on hybridization of ICA and SA for utilizing both capabilities. In addition, considering some practical assumptions for this problem could be regarded as an impressive research.

6. REFERENCES

[1] Lee GC, Kim YD. 2004. A brach-and-bound algorithm for a two-stage hybrid flow shop scheduling problem minimizing total tardiness. International Journal of Production Research, 42, 4731–43.

[2] Arthanary, T.S. and Ramamurthy, K.G. 1971. An extension of two machines sequencing problem, Opsearch, 8, 10–22.

[3] Salvador, M.S. 1973. A solution to a special case of flowshop scheduling problems", in: S.E. Elmaghraby (ed.), Symposium of the Theory of Scheduling and Applications, Springer-Verlag, New York, 83-91.

[4] Brah, S.A. 1988. Scheduling in a flow shop with multiple processors", Unpublished Ph.D. Dissertation, University of Houston, Houston, TX.

[5] Botta-Genoulaz, V. 2000. Hybrid flow shop scheduling with precedence constrains and time legs to minimize maximum lateness. International Journal of Production Economics, 64, 101–111.

[6] Kochhar, S., Morris, R. and Wong, W. 1988. The local search approach to flexible flow line scheduling. Engineering Costs and Production Economics,14(1):25–37.

[7] Wittrock, R. J. 1988. An adaptable scheduling algorithm for flexible flow lines. Journal of Operational Research, 36, 445–453.

[8] Brah, S.A. and Hunsucker, J.L. 1991. Branch and bound algorithm for the flow shop with multiple processors. European Journal of Operational Research, 51, 88–99.

[9] Rajendran, C. and Chaudhuri, D. 1992. Scheduling in n-job, m-stage flowshop with parallel processors to minimize makespan. International Journal of Production Economics, 27, 137–143.

[10] Vignier, A., Dardilrac, D., Dezalay, D., and Proust, C. 1996. A branch and bound approach to minimize the total completion time in a k-stage hybrid flow shop. Proceedings of the 1996 IEEE conference on emerging technologies and factory automation. (Vol. 1) (pp. 215–220).

[11] Moursli, O. and Pochet, Y. 2000. A branch-and-bound algorithm for the hybrid flow shop. International Journal of Production Economics, 64,113–125.

[12] Og˘uz, C., Ercan, M., Edwin Cheng, T. C., and Fung, Y. F. 2003. Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. European Journal of Operational Research, 149, 390–403.

[13] Gupta, J. N. D., Hariri, A. M. A., and Potts, C. N. 1997. Scheduling a two-stage hybrid flow shop with parallel machines at the first stage. Annals of Operations Research, 69, 171–191.

[14] Ruiz, R. and Maroto, C. 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research, 169(3):781–800.

[15] Azizoglu, M., Cakmak, E. and Kondakci, S. 2001. A flexible flowshop problem with total flow time minimization. European Journal of Operational Research, 132,528–538.

[16] Kurz, M. E., & Askin, R. G. (2004). Scheduling flexible flow lines with sequence- dependent setup times. European Journal of Operational Research, 159, 66–82.

[17] Zandieh, M., Fatemi Ghomi, S. M. T., and Moattar Husseini, S. M. 2006. An immune algorithm approach to hybrid flowshops scheduling with sequence-dependent setup times. Applied Mathematics and Computation, 180, 111–127.

[18] Jolai, F., Sheykh, S., Torabi, A. and Karimi, B. 2009. A genetic algorithm for solving no-wait flexible flow lines with due window and job rejection. International Journal of Advanced Manufacturing Technology, 42,523–532.

[19] Aldowaisan, T. and Allahverdi, A. 2004. A New heuristics for m-machine no-wait flowshop to minimize total completion time. Omega, 32, 345–352.

[20] Ruiz, R. and Allahverdi, A., 2009. New heuristics for no-wait flow shops with a linear combination of makespan and maximum lateness. International Journal of Production Research 47(20), 5717-5738.

[21] Sawik, T. 2000. Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. Mathematical and Computer Modeling, 31(13), 39–52.

[22] Su, L-H. 2003. A hybrid two-stage flow-shop with limited waiting time constraints. Computers and Industrial Engineer, 44, 409–424.

[23] Behnamian, J., & Zandieh, M. 2011. A discrete colonial competitive algorithm for hybrid flowshop scheduling to minimize earliness and quadratic tardiness penalties. Expert Systems with Applications, doi:10.1016/j.eswa.2011.04.241

[24] Atashpaz-Gargari and Lucas, E.C. 2007. Imperialist Competitive Algorithm: An algorithm for optimization inspired by imperialist competitive. IEEE Congress on Evolutionary computation, Singapore.

[25] Bagher, M. Zandieh, M. Farsijani, H. 2010. Balancing of stochastic U-type assembly lines: an imperialist competitive algorithm. International Journal of Advanced Manufacturing Technology. DOI 10.1007/s00170-010-2937-3

[26] Forouharfard, S. Zandieh, M., 2010. An imperialist competitive algorithm to schedule of receiving and shipping trucks in cross-docking systems. International Journal of Advanced Manufacturing Technology, 51:1179–1193.

[27] Ross, R.J., Taguchi Techniques for Quality Engineering, McGraw-Hill, USA, 1989.

[28] Phadke, M.S., Quality Engineering Using Robust Design, Prentice-Hall, USA, 1989.

[29] Al-Aomar, R., 2006. Incorporating robustness into genetic algorithm search of stochastic simulation outputs, Simulation Modeling Practice and Theory 14 , 201–223.