The 9th International Conference on Modelling, Identification and Control
(ICMIC 2017), Kunming, China, July 10-12, 2017

Improved Imperialist Competitive Algorithm for Flexible Flow Shop Scheduling

SUN Yue, LIN Shuo, LI Tan, and MA Xiaofu

***Abstract*—In this paper, we investigate the flexible flow-shop scheduling problem (FFSP). The mathematical model is established to minimize the makespan. Improved Imperialist Competitive Algorithm (IICA) is proposed as the global searching mechanism for this optimization problem. Based on the standard imperialist competitive algorithm, two improvements, reform operation and elite individual retention strategy have been introduced. At the same time, to improve the diversity of solutions, the Hamming distance has also been introduced to select the individuals. Through the standard instance tests, the proposed IICA outperforms the genetic algorithm and the standard imperialist competitive algorithm in terms of the convergence speed and global optimum searching for the flexible flow-shop scheduling problem[1].**

***Key words*— Flexible flow-shop problem, imperialist Competitive Algorithm, Hamming distance, makespan.**

I. Introduction

The classical flow-shop scheduling problem can be extended to many derived problems. One of them is the flexible flow-shop scheduling problem, which has been proved to be a NP difficult [1] [2]. The research of FFSP is very theoretical and challenging, but the effective algorithm design is significant for the practical applications. One nature of FFSP is to allow the existence of parallel machines in certain stages, and thus it is widely used in the actual production process for the steel, chemical and pharmaceutical fields. FFSP is normally described as follows. Multiple jobs are needed to process, and each job has the same processing route and contains at least 2 stages. Multi-parallel machines exist in at least one of the stages. Usually, the processing time of each job in each stage has a known value. Then, the work station distribution of the parallel machines and the processing order of the job on the same machine should be selected in a smart way in order to minimize the makespan as the optimization goal for the flexible flow-shop scheduling problem.

Currently, heuristic algorithms as well as branch and bound methods have been proposed for solving FFSP. However, most of the solutions is limited either because they can only solve the production scheduling problems in the small scale, or because their coding methods are imperfect that can result in illegal solutions. To overcome these issues, in this paper, we adopt and improve the Imperialist Competitive Algorithm (ICA) for FFSP. ICA has been initially proposed by Atashpaz-Gargari and Lucas [3] in 2007. Its idea is based on the simulating the colonial competition process of human society, and in nature is a novel meta-heuristic algorithm. ICA's advantage is that it is simple but time-saving. In other words, this optimization algorithm is extremely efficient and easy to use. In addition, ICA can save memory as well as the searching time, and thus it can quickly converge to the optimal solution in the search space. ICA has been more and more widely used during recent years. For example, Mohammad Shahrazad et al [4] have applied ICA to the electric power industry [5]; Shuhui Xu et al [6] have applied ICA into solving the traveling salesman problem (TSP); Mojtaba Ghasemia et al [7] have proposed the application of ICA in the problem of multi-objective optimized energy flow. MH Fazel Zarandia et al [8] have applied ICA to the stock price forecasting problem. In all these applications, ICA shows excellent performance in terms of convergence rate and global optimum searching capability. In the literature, there are very few efforts of applying ICA to the scheduling optimization problems. Therefore, this paper applies and customize the ICA to solve FFSP, and finally verify the performance of the proposed algorithm.

II. Problem Description

In this paper, FFSP [9]is described as follows: n jobs will be processed in m operations following the shop orders; Among the m stages, at least one stage includes several paralleled work stations, the number of the paralleled work stations in the j th stage is M_j , each stage contains at least one work station, and the same job has the same processing time in any paralleled work station of the same stage.

A. Parameter Setting

n represents the total number of the processed jobs;

J_i represents job i , where $i \in \{1,...,n\}$;

m represents the total number of the processing stages;

[1] Manuscript received March 17, 2017. This work was supported in part by the Natural Science Foundation of Liaoning Province of China (201602615), Sciences and Discipline Content Education Project(XKHY2-62) and the Liaoning Provincial Social Planning Fund (L15BGL017)

Sun Yue is a Master Students in Shenyang Jianzhu University, Shenyang, China (corresponding author to provide phone: 13897982683; e-mail: SUN_YUE_1212@163.com).

Lin Shuo is an Associate Professor of Faculty of Information and Control Engineering, Shenyang Jianzhu University. He graduated from the Chinese Academy of Sciences in Shenyang Institute of Automation of PhD, Shenyang, China (e-mail: farewell_lin@163.com).

Li Tan is a teacher of Municipal and Environmental Engineering Institute in Shenyang Jianzhu University and he is a Master graduated from Shenyang Jianzhu University (e-mail: 110713763@qq.com)

Ma Xiaofu is a PhD student in the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA,24060 (email: xfma@vt.edu).

M_j represents the number of parallel work stations in $Stage_j$, $j \in \{1,...,m\}$;

M represents the total number of parallel work stations;

$WS_{j,k}$ represents the k th work station in $Stage_j$, $j \in \{1,...,m\}$, $k \in \{1,...,M_j\}$;

B. Constraint Conditions

(1) Each job J_i in each stage chooses any one paralleled work station for processing. Once job J_i starts processing, it cannot be interrupted.

(2) Each paralleled work station can only process one job J_i at the same time, and each job J_i can only be processed by one paralleled work station.

(3) The processing time of jobs contains the ready time and the moving time.

(4) Job waiting is allowed between two adjacent stages, and the work station is allowed to be in idle state if no job on it.

The optimization goal of FFSP is to find out the processing order of these n jobs in order to minimize the makespan. Equation (1) shows the calculation for the total completion time:

$$C_{max} = max\left\{C_{max}^i \mid i = 1,2,...M\right\} \tag{1}$$

In this equation, C_{max}^i represents the completion time of the final processed job on the i th machine, C_{max} represents the completion time of the last processed job.

The relationship between the start time of job and the completion time of job in each stage is given by equation (2). The processing order relationship between each job in the adjacent stage is given by equation (3).

$$C_{i,j} = S_{i,j} + T_{i,j} \quad i \in \{1,...,n\}, j \in \{1,...,m\} \tag{2}$$

$$C_{i,j-1} \le S_{i,j} \quad i \in \{1,...,n\}, j \in \{1,...,m\} \tag{3}$$

III. Improved Imperialist Competitive Algorithm

We introduce two improvements in ICA because the standard ICA can become premature convergence and slide into the local extreme. We propose IICA, and the algorithm flow chart of IICA is shown in Figure 1.

A. Algorithm Improvements

(1) Reform operation: After the imperialist competition, the objective function value of empire group's weakest colony is the worst. The colony reform operation is increased to improve the global searching ability of the algorithm group, and the weakest colonies in each empire will be replaced by a random solution to improve the optimization performance of the algorithm.

(2) Elite individual retention strategy: In the empire elimination part of the standard ICA, the empire is usually eliminated when the imperialist itself only exists in empire group. But the imperialist individual is excellent, which is considered to be the elite individual and conducive to the convergence of the algorithm. Therefore, we retain the elite individual so that a better solution can be found in the evolutionary process.

B. Algorithm Description

(1) Empire initialization: In IICA, each country represents a solution to a particular optimization problem. Each country is represented by a real number array or a vector. For a N_{var} -dimension optimization problem, the array is encoded as follows:

$$country = [p_1, p_2, ..., p_{N_{var}}]$$

The size of the power of each country is measured by the cost function. The smaller the cost function value , the greater the power of the country. The inverse relationship is between the two.

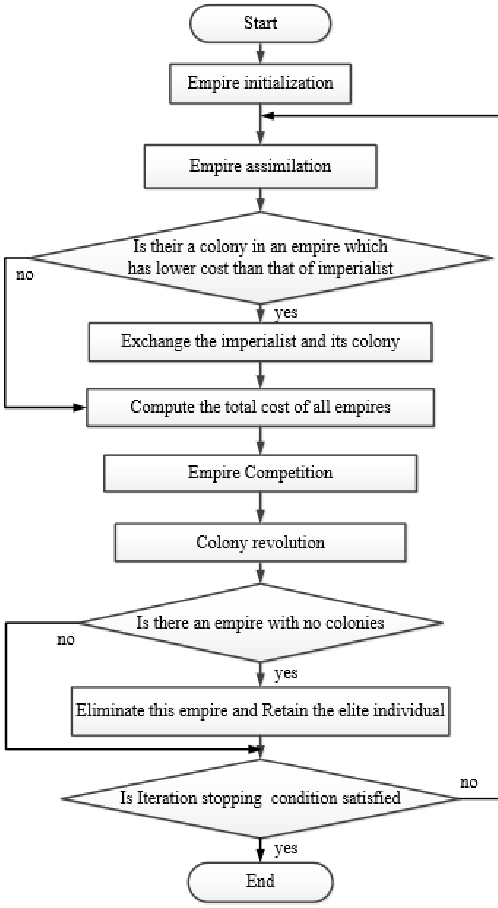
Start
Empire initialization
Empire assimilation
Is their a colony in an empire which has lower cost than that of imperialist
no
yes
Exchange the imperialist and its colony
Compute the total cost of all empires
Empire Competition
Colony revolution
Is there an empire with no colonies
no
yes
Eliminate this empire and Retain the elite individual
Is Iteration stopping condition satisfied
no
yes
End

Fig.1 Flow chart of improved imperialist competitive algorithm

First, calculate the N_{pop} countries' cost function value, which is used to select the larger N_{imp} countries as the imperialists, and the rest N_{col} countries become the colonies of these imperialists. The corresponding imperialist and colonies form an empire group. In order to form the original empire group, the number of colonies that is owned by each imperialist is distributed according to the power of imperialists. The number of colonies in each imperialist is calculated according to equation (4) ~ (6)

$$Cost\_imp_n = \max\{\cos t\_imp_i\} - \cos t\_imp_n \quad i \in \{1,\dots,N_{imp}\} \tag{4}$$

$$p_n = \frac{Cost\_imp_n}{\sum_{i=1}^{N_{imp}} Cost\_imp_i} \tag{5}$$

$$N_{col}^n = round(p_n \cdot N_{col}) \tag{6}$$

In these equations: $cost\_imp_n$ is the cost function value of the n th imperialist, $Cost\_imp_n$ is the standard cost function value of the n th imperialist, p_n is its standard power, N_{col}^n is the number of the colonies of the n th imperialist owns.

(2) Empire assimilation: FFSP is a discrete combinatorial optimization problem, the assimilation process achieves the movement of the colonies to the empire and the migration [12]in the process of movement by referring to the crossover and variation operations [10] [11] in the genetic algorithm. Article [13] uses the two-point crossover method. Figure 2 shows the cross process. Figure 3 shows the variation process in which two gene fragments are selected randomly and their positions are exchanged.

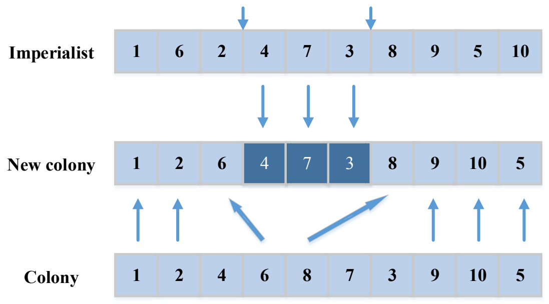

Fig. 2 The cross process of the imperialist and colony

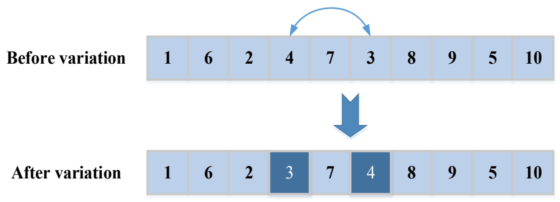

Fig. 3 The variation process

(3) Exchange the position of imperialist and colony: The power of the colony may be greater than its imperialist's in the process when the colony moved to the empire. In this case, the positions of the imperialist and the colony are exchanged. After that, the colony will join the competition as a new empire

(4) Empire competition: First, empire competition needs to calculate the total cost function value for each empire group, which includes two parts, the cost function value of the imperialist and the colony owned by the imperialist. The smaller the total cost function of an empire group, the greater the power of the empire group. Its definition is shown in equation (7):

$$T.C_n = cost\_imp_n + \xi\left(\frac{\sum_{j=1}^{N_{col}^n} cost\_col_j}{N_{col}^n}\right) \tag{7}$$

$T.C_n$ is the total cost function value of the n th empire group. ξ is a positive real number less than 1. The simulation results show that the performance is the best when ξ equals 0.15 in solving FFSP.

The process of empire competition will enable more powerful empires occupy more and more colonies, while the weak empire will lose the colonies gradually. Different empires have a certain probability of the possession of the colonies. The size of this possibility is defined by equation (8) and (9):

$$N\_T\_C_n = max\{T.C_i\} - T.C_n \tag{8}$$

$$Pp_n = \frac{N\_T\_C_n}{\sum_{i=1}^{N_{imp}} N\_T\_C_i} \tag{9}$$

In these equations, N_{imp} is the number of imperialists, $N\_T\_C_n$ is the total relative cost function value of the n th empire group, Pp_n is the probability of the possession of the colonies of the n th empire.

In order to assign free colonies to the imperialist, the following vector is constructed:

$$P = [P_{p_1}, P_{p_2}, P_{p_3}, \dots, P_{p_n}]$$

Then a vector R is conducted, and it is of the same size as P and constituted by uniform distribution random numbers.

$$R = [r_1, r_2, r_3, \dots, r_n] \quad r_1, r_2, r_3, \dots, r_n \in U(0,1)$$

The vector D is gained by the following calculation:

$$D = P - R = [D_1, D_2, D_3, \dots, D_n] =$$
$$[P_{p_1} - r_1, P_{p_2} - r_2, P_{p_3} - r_3, \dots, P_{p_n} - r_n]$$

The greatest element in vector D corresponding to the empire group will occupy the weakest colony of the above.

(5) Colony Revolution [14] To increase the global searching ability of the algorithm group, the weakest colony in each empire is replaced by a random solution. In order to enhance the information exchange between individuals effectively, the concept of Hamming distance [15] [16] [17] in the coding theory is introduced.

Hamming distance (HD) is a basic method used to describe the distance between two n -word-length code $x = (x_1, x_2, \dots, x_n)$, $y = (y_1, y_2, \dots, y_n)$. The distance equation of the two n -word-length code is as follows:

$$D(x,y) = \sum_{k=1}^{n} x_k \oplus y_k \tag{10}$$

In equation (10), $x_k \in \{0,1\}, y_k \in \{0,1\}$, $\oplus$ represents nonequivalence operation. $D(x,y)$ represents the number of different values at the same location in two identical dimensional vectors.

Then, we define the similarity s between individuals, that is, the proportion of the same number of genes in the two individuals in the total number of genes. And the similarity threshold Rt is compared with s to determine whether the individuals are similar.

$$s = N_D / N_{Gene} \tag{11}$$

In equation (11), s is the ratio of the number of identical gene fragments of two individuals N_D and the total number of genes of the individual N_{Gene}. If it is larger than Rt, we say the two individuals are similar.

(6) Empire Elimination and Algorithm Termination Condition: For the competition between empires, the empires with the larger power become increasingly strong through occupying the colonies of other empires, while the number of colonies of empires with the smaller power continues to decline. When an empire loses all its colonies, the empire is eliminated. But if the empire is an excellent individual, which is considered as an elite individual and conducive to the convergence of the algorithm, it will be selected and will belong to the empire who occupied its last colony in order to find a better solution in the evolution process. The algorithm continues to iterate, and when there is only one empire or the number of iterations is reached, the algorithm ends.

IV. Examples Analysis

We implement IICA using MATLAB2012b and run tests on the PC with Window 7 operating system and Core i7 CPU and 8GB memory. The test cases of standard flexible flow shop in article [18] are used to test the performance of IICA for solving the FFSP.

A. Analysis of Algorithm Parameters

The parameter value selection of the algorithm has a big impact on the optimization performance of the algorithm. The parameter of the standard ICA algorithm (colonial influence factor: K) is discussed and analyzed to determine the optimal parameter values. The instance j15c10a1 is selected to test the parameter which has four levels, and the algorithm runs 20 times in each level. The average makespan $\overline{C_{max}}$ is considered as the evaluation index and the smaller $\overline{C_{max}}$, the better the performance of algorithm optimization. The test results are shown in Table 1.

The experimental results show that the influence trend of the parameter values on the performance of the algorithm is shown in Figure 4. When K is 0.15, the standard ICA algorithm gets the minimum value, that is, the optimization is best, so K is set to 0.15.

TABEL1
PARAMETER TEST RESULTS

Parameter	Level			
	1	2	3	4
K	0.05	0.10	0.15	0.20

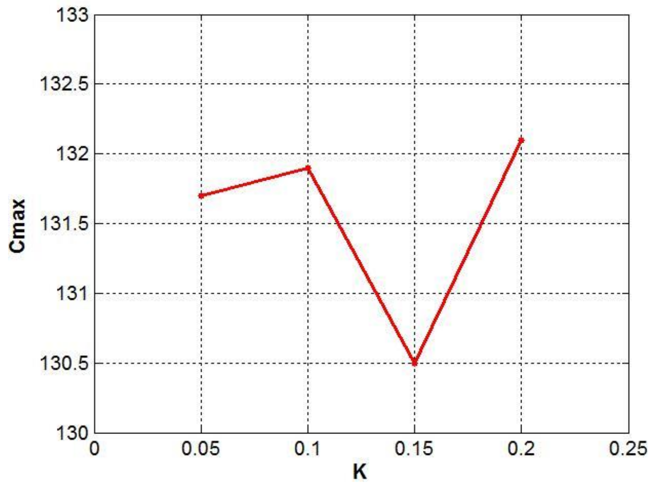

Fig.4 The influence trends of parameter K on algorithm performance

B. Algorithm Performance Test

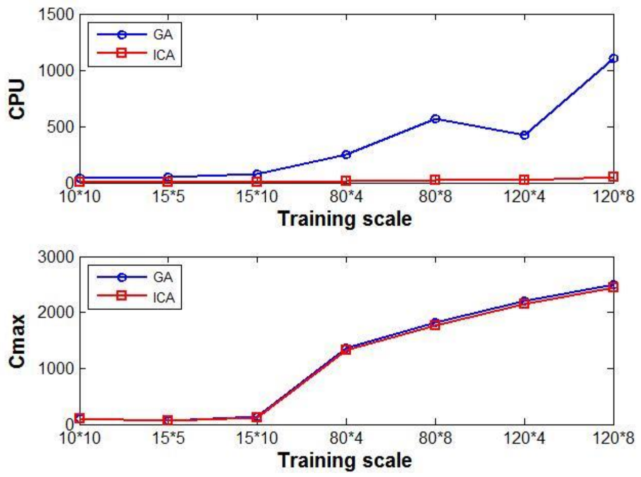

Fig.5 The curve of GA and ICA corresponding to 7 kinds of scale examples

The curve of GA and ICA corresponding to 7 kinds of scale examples is shown in Figure 5. The average makespan $\overline{C_{max}}$ of ICA is slightly better than that of GA. With the increase of the training scale, the optimization effect is gradually increased. For the average evolution time $\overline{T_{cpu}}$, the optimization efficiency of ICA is better than that of GA. $\overline{T_{cpu}}$ of ICA is 95.7% lower than that of GA for the instance j120c8.So in terms of convergence speed, ICA algorithm has obvious advantages. However, the convergence speed of ICA algorithm in the evolution is faster, but also faces with the problems of the precocious convergence and getting stuck into local extreme value. Therefore, in order to prevent premature convergence, and to expand the search range, IICA is introduced to ensure the continuous evolution ability of ICA.

One test case is for the size of 15 jobs, 80 jobs and 120 jobs. The genetic algorithm (GA) (the value of the crossover operator is 0.7, the value of variation operator is 0.8, the value of population size is 30) and ICA (the number of imperialist is 5, the number of colony is 25, colony influence factor ξ is 0.15, Rt is 0.3) are selected as the methods for comparison. The average makespan $\overline{C_{max}}$ and the average evolution time $\overline{T_{cpu}}$ are considered as the evaluation index. The maximum evolutionary generation of each algorithm is set to 100 generations. Using this test, the notation of j120c8a1 represents class a problem 1 which contains 120 jobs and 8 stages, each stage has three parallel machines. The test results are shown in Table 2.

According to the data in Table 2, for the instance j15c10a2, $\overline{C_{max}}$ of IICA is 0.81% lower than that of the ICA, and 3.1% less than that of GA. $\overline{T_{cpu}}$ of the IICA is 65.2% higher than that of ICA, and 67.9% less than that of GA. For the instance j120c4a1, $\overline{C_{max}}$ of IICA is 1.2% less than that of ICA, and 2.3% less than that of GA, and $\overline{T_{cpu}}$ of IICA is 55.8% higher than that of the ICA and 72.5% less than that of GA. As we can see, regardless of large-scale or small-scale instances, the optimal performance difference between the three algorithms is not very obviously, but IICA can still reflect the gradual optimization effect. However, for the evolutionary time, the global search range is expanded in IICA, and the time to search for the optimal solution also increases. Nevertheless, it is still within the acceptable range and is obviously better than GA.

TABEL2
AVERAGE VALUES OF OPTIMIZING INDEX OF THE THREE ALGORITHMS IN 12 KINDS OF SCALE INSTANCE TESTS.

Instance number	GA		ICA		IICA	
	$\overline{C_{max}}$	$\overline{T_{cpu}}$	$\overline{C_{max}}$	$\overline{T_{cpu}}$	$\overline{C_{max}}$	$\overline{T_{cpu}}$
j15c5a1	85.47	5.20	85.02	0.95	84.82	2.06
j15c5a2	80.53	4.28	80.23	1.28	79.17	1.78
j15c10a1	138.93	12.76	138.14	2.03	137.79	4.44
j15c10a2	137.84	11.18	134.81	1.98	133.73	3.27
j80c4a1	1549.74	29.21	1507.43	3.15	1501.41	7.47
j80c4a2	1418.38	27.78	1397.25	3.54	1387.28	5.36
j80c8a1	1907.46	51.42	1852.17	6.07	1835.16	13.35
j80c8a2	1927.32	57.02	1868.47	7.02	1849.62	13.15
j120c4a1	2196.76	31.83	2173.62	6.81	2146.70	9.04
j120c4a2	2307.96	34.67	2274.45	7.98	2263.24	8.07
j120c8a1	2619.28	77.15	2572.60	9.53	2558.83	19.67
j120c8a2	2553.81	87.92	2513.47	10.90	2495.13	20.53

Figure 6 shows the relationship between the makespan and CPU time under three algorithms of j80c4a1. It can be seen that the longer the iteration time, the better the fitness values. ICA is prematurely convergent, and stagnates at 1.9s and then slides into the local extremum. IICA has a better capability to improve than ICA, which stagnates at 9.7 s, with the fitness value of 1943. Therefore, IICA has a better optimization results than GA and ICA.

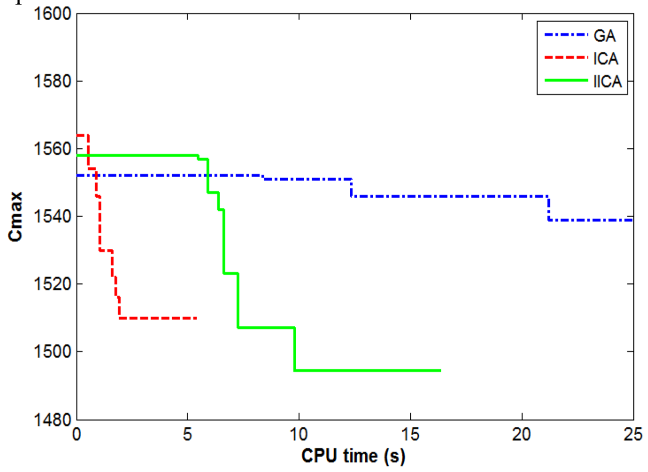

Fig.6 The relationship graph for the maximum completion time and CPU time of each algorithm

V. CONCLUSION

In this paper, we studied the FFSP, and proposed a novel swarm intelligence algorithm, IICA. IICA is adopted as the global optimization method for minimizing the makespan as the optimization goal. In nature, IICA uses the breakthrough point of enhancing the ability of algorithm to jump out of the local minimum and improving the evolution of the algorithm. We introduce the reform operation and the elite individual retention strategy as well as the individual selection mechanism based on the Hamming distance. ICA is tested by standard example. The test results show that, compared with GA and standard ICA, IICA has a better performance specially for FFSP. By numerical analysis, we verified the feasibility and effectiveness of IICA in solving the FFSP, and demonstrate its potentials for other practical applications.

As future works, we aim at investigating whether IICA can achieve better searching performance when increasing the evolutionary generation. We are also working on future improve the converging time of the proposed IICA.

References

[1] Z. H. Han, Y. H. Zhu and X. F. Ma, "Multiple rules with game theoretic analysis for flexible flow shop scheduling problem with component altering times," *International Journal of Modelling Identification & Control.*, vol.26, no.1, pp.1-18, 2016.

[2] Z. H. Han, X. F. Ma and L. L. Yao, "Cost Optimization Problem of Hybrid Flow-Shop Based on PSO Algorithm," *Advanced Materials Research.*, vol.532-533, pp.1616-1620, 2012.

[3] A. Gargarie and Lucacs, "Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition," *Proceedings of the 2007 IEEE Congress on Evolutionary Computation. Piscataway: IEEE Press.*, pp.4661-4667, 2007.

[4] M. Shahrazad, A. Hessam and A. Ahmed, "Application of imperialist competitive optimization algorithm in power industry," *International Journal of Industrial Engineering Computations*., vol.6, no.1, pp.43-58, 2015.

[5] Y. X. Shen and F. Li, "A Survey of Diagnosis Methods for Wind Power System," *Control Engineering of China.*, vol.20, no.5, pp.785-795, 2013.

[6] S. H. Xu, Y. Wang and A. Q. Huang, "Application of Imperialist Competitive Algorithm on Solving the Traveling Salesman Problem," *Algorithms.*, vol.7, no.2, pp.229-242, 2014.

[7] M. Ghasemia, S. Ghavidela and M. M. Ghanbarianb, "Application of imperialist competitive algorithm with its modified techniques for multi-objective optimal power flow problem," *Inform. Sci.*, vol.281, pp.225-247, 2014.

[8] M. H. F. Zarandia, M. Zarinbala and N. Ghanbaria, "A new fuzzy functions model tuned by hybridizing imperialist competitive algorithm and simulated annealing application," *Information Sciences*., vol.222, pp.213-228, 2013.

[9] Z. H. Han, Y. H. Zhu and H. B. Shi, "A co-evolution CGA solution for the flexible flow shop scheduling problem," *CAAI Transactions on Intelligent Systems.*, vol.10, no.4, pp.562-568, 2015.

[10] S. Manaseer, W. Manasir and M. Alshraideh, "Automatic Test Data Generation for Java Card Applications Using Genetic Algorithm," *Journal of Software Engineering and Applications.*, vol.8, no.12, pp. 603-616, 2015.

[11] K. Kim, I. J. Jeong, "Flow shop scheduling with no-wait flexible lot streaming using an adaptive genetic algorithm," *The International Journal of Advanced Manufacturing Technology*., vol.44, pp. 1180-1190, 2009.

[12] K. Lian, C. Zhang and L. Gao, "A modified colonial competitive algorithm for the mixed-model U-line balancing and sequencing problem," I*nternational Journal of Production Research*., vol.50, no.18, pp.5117-5131, 2012.

[13] F. Jolaia, M. Rabiee and H. Asefi, "A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times," *International Journal of Production Research*., vol.50, no.24, pp.7447-7466, 2012.

[14] E. Shokrolahpour, M. Zandieh and B. Dorri, "A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flow-shop problem," *International Journal of Production Research.*, vol.49, no.11, pp.3087-3103, 2010.

[15] H. J. Zhang, G. S. Wang and Y. X. Zhong, "Text Similarity Computing Based on Hamming Distance," *Computer Engineering and Applications*., vol.37, no.19, pp.21-22, 2001.

[16] S. Dolev, S. Frenkel and E. Dan, "Preserving Hamming Distance in Arithmetic and Logical Operations," *Journal of Electronic Testing*., vol.29, no.6, pp.903-907, 2013.

[17] L. C. Liu, Y. Yao, "Weighted inverse maximum perfect matching problems under the Hamming distance," *Journal of Global Optimization*., vol.5, no.3, pp.549-557, 2013.

[18] J. Carlier and E. Neron, "An exact method for solving the multi-processor flow-shop," *RAIRO-Operations Research*., vol.34, no.1, pp.1-25, 2000.