Thai Journal of Mathematics : 325-337
Special Issue : Annual Meeting in Mathematics 2019

http://thaijmath.in.cmu.ac.th
Online ISSN 1686-0209

Makespan Minimization for Parallel Machines Environment with Machine Dependent Processing Time by Using PBIL Combined with Local Search

Pensiri Sompong[†,1] **and Sungkom Srisomporn**[‡]

[†]Department of General Science, Faculty of Science and Engineering, Kasetsart University Chalermphrakiat Sakon Nakhon Province Campus, Sakon Nakhon, Thailand, 47000
e-mail : pensiri.so@ku.th

[‡]Department of Mechanical and Manufacturing Engineering, Faculty of Science and Engineering, Kasetsart University Chalermphrakiat Sakon Nakhon Province Campus, Sakon Nakhon, Thailand, 47000
e-mail : sungkom.s@ku.th

Abstract : Population-based incremental learning algorithm (PBIL) is proposed to solve parallel machines scheduling problem with machine dependent processing time. The initial population of proposed algorithm is created based on probability vector resulting from the solution obtained from applying shortest processing time (SPT) dispatching rule for parallel machines to represent the jobs assigned on the machines. Local search is performed during the process to move a job to an appropriate machine that makespan is minimized. The performance of the algorithm is illustrated by numerical examples. The solutions obtained from PBIL are compared to the solution from SPT. The results show that the assignment of jobs by using PBIL combined with local search can reduce makespan and it is suitable for solving parallel machines scheduling problem with machine dependent processing time.

Keywords : parallel machines; scheduling; population-based incremental learning; machine dependent processing time.

[1]Corresponding author.

2010 Mathematics Subject Classification : 90C59; 68M20.

1 Introduction

Parallel machines scheduling is an optimization problem associated with the assignment of n jobs to m parallel machines. In general, parallel machines environment is designed to reduce the maximum total completion time or makespan (C_{max}). Makespan is varied due to the different processing time of jobs on each machine. Processing time may depend on machines and sometimes, setup time to prepare for processing the next job may be included. By the literature on parallel machines scheduling problem with the objective to minimize makespan, there are many researches presenting the methods to assign the jobs on each machine and some procedures are inserted to improve the solution or enhance the performance. Arnaout et al. [1] introduced Ant Colony Optimization (ACO) algorithm for the non-preemptive unrelated parallel machines scheduling problem with machine-dependent and sequence-dependent setup times. Li et al. [2] considered the identical parallel machines scheduling problem in which processing time is linear decreasing function of the consumed resource. Simulated annealing algorithm was used to obtain the solution. Balin [3] proposed new crossover operator and optimality criterion to adapt genetic algorithm (GA) for solving non-identical parallel machines scheduling problem. Vallada and Ruiz [4] presented GA for the unrelated parallel machines scheduling in which machine and job sequence dependent setup times were considered. Local search was included to enhance crossover and mutation operators. Alcan and Başlıgil [5] studied the problem of non-identical parallel machines with triangular fuzzy processing times by using GA. Cappadonna et al. [6] addressed the unrelated parallel machine scheduling problem with limited human resources. Processing and setup times depend on the machine. Mixed integer linear programming (MILP) was provided for solving the problem. Parallel machines scheduling with learning effects and fuzzy processing time was addressed by Yeh et al. [7]. Two heuristic algorithms were proposed, simulated annealing algorithm and GA. Unrelated parallel machines scheduling problem with sequence and machine dependent setup times and machine dependent processing time was studied by Joo and Kim [8]. In their woik, mathematical model and hybrid GA with three dispatching rules were proposed to solve for optimal solution and large-sized problems, respectively. An immune-inspired algorithm was presented by Diana et al. [9] for unrelated parallel machines scheduling problem with sequence dependent setup time. A Variable Neighbourhood Descent (VND) was used as local search to guide the solution to a local optimum. Sels et al. [10] developed three heuristic approaches for solving the problem of unrelated parallel machines, GA, tabu search algorithm and hybridization of these two heuristics with a truncated branch and bound procedure. The hybridization was applied to accelerate the search process to near optimal solution. Kılıç and Yüzgeç [11] presented antlion optimization algorithm (ALO) inspired by nature and animals for solving unrelated parallel machine scheduling problem with setup

time. Machine dependent processing time was also considered.

Population-based incremental learning algorithm is an evaluation algorithm that combines the mechanism of genetic algorithm and competitive learning. PBIL works with probability vector used to create a new set of solutions called population through learning (Baluja [12], Folly [13]). Some applications of using PBIL for solving optimization problem can be found in the previous works. Pang et al. [14] proposed adaptive PBIL algorithm for solving flow shop and job shop scheduling problem. Wan and Qiu [15] solved vehicle routing optimization problem by using advanced PBIL algorithm with the objective to minimize the cost and meet the time restriction. Chen et al. [16] applied PBIL to cloud computing resource scheduling problem. Chen et al. [17] used an artificial immune system combined with PBIL and collaborative filtering to develop a classifier for network intrusion and anomaly detection in electronic commerce environments. Meng et al. [18] selected the cities with small penalty values in serial colored traveling salesman problem by applying PBIL.

In this paper, makespan minimization for parallel machines scheduling problem with machine dependent processing time is studied. PBIL combined with local search is proposed to solve the problem. The notations and problem descriptions are given in section 2. Solution representation and shortest processing time procedure adapted to parallel machines are also described in this section. Section 3 details the solution procedure associated with the applications of PBIL and local search. The performance and results obtained from using the proposed algorithm are discussed in section 4. Finally, the conclusion is presented in section 5.

2 Preliminaries

In this section, the notations and problem descriptions for parallel machines scheduling problem with machine dependent processing time are presented. To explain the job assignment on the machine, solution representation in the matrix form is proposed and the principle of shortest processing time are then described.

N : number of jobs.

M : number of machines.

p_{im} : processing time of job i on machine m where $i \in \{1, 2, ..., N\}$ and $m \in \{1, 2, ..., M\}$.

C_m : total completion time of machine m, where $m \in \{1, 2, ..., M\}$.

C_{max} : maximum completion time or makespan.

$\mathbf{P}$: probability vector.

$prob_k$: probability of k-th position in probability vector $\mathbf{P}$.

t : number of iterations

LR : learning rate.

The problem descriptions are as follows.

(1) All jobs are available to process at time zero.

(2) Job can be processed on any machine and job processing time is machine dependent.

(3) Job is completed by only one machine without interruption.

(4) Job can be started immediately after precedent job is completed.

2.1 Solution Representation

Because parallel machines scheduling problem is associated with the assignment of jobs to machines, an $M \times N$ matrix such that all elements are 0 or 1 is presented for scheduling problem with N jobs and M machines. The value of element $x(i, j)$ in the matrix indicates job j which will be processed by machine i. For example, if there are 8 jobs which are assigned to 3 machines, solution representation can be written in 3×8 matrix form. Suppose the values of all elements are given in matrix X_1 as follow.

$$X_1 = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

It means that machine 1 operates jobs 2, 4 and 7 while machine 2 operates jobs 1, 3 and 8. The remaining jobs are operated on machine 3. Because a job is completed by only one machine, there is only one element valued 1 in each column.

2.2 Shortest Processing Time

Shortest processing time (SPT) is a dispatching rule applied to single machine scheduling problem with the objective to minimize job tardiness. Job sequence of SPT algorithm depends on job processing time, job with smaller processing time is firstly chosen to schedule. In the case that the decrease of total completion time is of interest, parallel machines environment is applied and many job assignment procedures are used to assign jobs to machines. SPT is one of them that has been used for the job assignment. Due to machine dependent processing time, sum of job processing time on all machines is calculated. Job with sum the smaller is firstly selected to process on a machine having the least total completion time. The steps of SPT applied to parallel machines scheduling problem with machine dependent processing time are as follows.

Step 1 : Sort the jobs according to the sum of processing time of each job on all machines and put them to set S, $S = \{j_1, j_2, ..., j_N\}$

Step 2 : Let $S' = \emptyset$ and assign the first job in S to the machine with shortest processing time and put it to set S', $S' = S' \cup \{j_1\}$, and $S = S - \{j_1\}$.

Step 3 : Assign the next job j_k in S to all machines and calculate C_m, $m \in \{1, 2, ..., M\}$.

Step 4 : Choose the machine leading to minimum C_m and assign job j_k in step 3 to such a machine. If there are two or more values of C_m are equal, job j_k is scheduled to the machine with minimum index. Then, $S' = S' \cup \{j_k\}$ and $S = S - \{j_k\}$.

Step 5 : Update C_m and C_{max}. Go to step 3 and continue until $S = \emptyset$ and $S' = \{j_1, j_2, ..., j_N\}$.

Example 2.1. Suppose that there are 8 jobs to be scheduled on 3 machines and let p be 3×8 matrix of processing time as follow.

$$p = \begin{bmatrix} 7 & 6 & 4 & 9 & 5 & 8 & 6 & 5 \\ 5 & 4 & 6 & 3 & 7 & 4 & 7 & 8 \\ 4 & 5 & 3 & 6 & 6 & 2 & 5 & 9 \end{bmatrix}$$

By sorting the jobs according to the sum of processing time of each job on all machines, set of job order $S = \{3, 6, 2, 1, 4, 5, 7, 8\}$. Job schedule obtained from applying SPT procedure mentioned previously is shown in Fig.1 with $C_{max} = 15$.

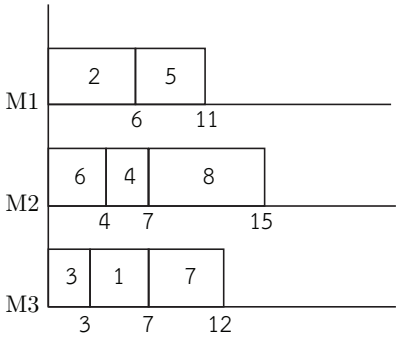

Figure 1: The assignment of 8 jobs to 3 machines by using SPT.

3 Solution Procedure

In general, parallel machines scheduling problem is divided into two parts, assignment of jobs to machines and improvement of schedule. In this study, PBIL is used to assign n independent jobs to m parallel machines and local search is performed to improve schedule. The procedure of PBIL and local search are presented in the following sections.

3.1 Population-based incremental learning algorithm

In order to represent the job assignment to machines which is the solution of scheduling problem, population-based incremental learning algorithm (PBIL) is used to created a set of solutions called population. As mentioned in section 2.1, the solutions are written in the matrix form and all elements are valued either 0 or 1 as seen from matrix X_1. In PBIL, these values can be derived from probability vector. Probability vector defines probability of each position in solution containing 0 or 1. To create a population based on probability vector, the procedure is explained in the following example.

Example 3.1. Suppose that $\mathbf{P}$ is a 1×8 probability vector such that

$$\mathbf{P} = \begin{bmatrix} 0.75 & 0.6 & 0.2 & 0.4 & 0.35 & 0.4 & 0.8 & 0.5 \end{bmatrix}.$$

Given 8 jobs to be assigned to 3 machines, a 3×8 matrix X_2 such that

$$X_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

is created according to the probability vector $\mathbf{P}$. For an element $\mathbf{P}(1,1) = 0.75$, it means that probability generating 1 for $X_2(i,1)$, $i = 1,2,3$, is 0.75 ($prob_1 = 0.75$). Simulataneously, the probability generating 0 for $X_2(i,1)$, $i = 1,2,3$, is 0.25 obtained by subtracting $prob_1$ from 1. Similarly, probabilities to generate 1 for $X_2(i,j)$, where $i = 1,2,3$ and $j = 2,3,...,8$ are $\mathbf{P}(1,j)$, $j = 2,3,...,8$ while probabilities to generate 0 for $X_2(i,j)$ are obtained by $1 - prob_j$. It can be seen that X_2 satisfies the property that $\sum_{i=1}^{M} x(i,j) = 1$ for $j = 1,2,...,N$ indicating a job is completed by only one machine.

In PBIL, probability vector will be improved for each iteration to create higher quality of solution with higher probability. The procedure of PBIL applied to scheduling problem is as follows.

Step 1 : set an initial $1 \times N$ probability vector $\mathbf{P}$.

Step 2 : create a set of solutions (population) according to probability vector $\mathbf{P}$.

Step 3 : evaluate makespan, C_{max}, of each solution and find the best solution from populations in step 2.

Step 4 : update $\mathbf{P}$ by using equation (3.1).

$$prob_k^{(t)} = prob_k^{(t-1)}(1 - LR) + Best_k^{(t-1)}(LR) \tag{3.1}$$

for $k = 1, 2, ..., N$ and $Best_k^{(t-1)}$ is binary integer (0 or 1) of the best solution at iteration $t - 1$.

Step 5 : do step 2 to 4 until the stopping criteria is satisfied. Stopping criteria used in this study is set to a maximum number of iterations, 500 iterations. After t iteration, values of $prob_k$ are approached to either 0 or 1.

3.2 Local search

The second part of parallel machines scheduling problem is to improve the solutions. Once the job assignment is carried out by using PBIL, local search is applied to move a job to a new machine with decreasing makespan. Vallada and Ruiz [4] presented the acceptance criterion of the job movement between pairs of machines by the comparison of the amount of completion time reduced on one machine, the amount of completion time increased on the other machine and the values of makspan before and after the movement. The movement is accepted if it satisfies one of the following cases.

(1) Completion time of both machines is reduced.

(2) Completion time of one machine is decreased while completion time of the other machine is increased. Simultaneously, The amount of time decreased is greater than the amount of time increased and the value of makspan is not increased.

In order to move the jobs in this study, the machines with maximum and minimum total completion times are considered. The condition used to decide whether a job should be moved or not is the makespan values before and after the movement. The steps of local search are as follows.

Step 1 : Let m and m' be the machines with maximum and minimum total completion time, respectively. Compute C_m, $C_{m'}$ and C_{max}.

Step 2 : Determine a job i with longest processing time on machine m. If there are jobs that processing times are equal, job with smaller processing time on machine m' is chosen. Then, job i is moved to machine m'.

Step 3 : Compute new makespan, C^*_{max}, and consider the following conditions.

(1) If $C^*_{max} < C_{max}$, update $C_{max} = C^*_{max}$ and C_k for $k = 1, ..., M$. Do step 1.

(2) If $C^*_{max} \geq C_{max}$, determine a new job j with shortest processing time on machine m. Then, job j is moved to machine m'. Compute new values of C^*_{max}. In the case that C^*_{max} is still greater than or equal to C_{max}, the algorithm is terminated.

The algorithm is continued from step 1 to step 3 until it is terminated.

Example 3.2. Given 8 jobs to be assigned to 3 machines. Suppose that the job assignment obtained from using PBIL is defined by matrix X_2 as shown in example 3.1. Thus, jobs 2 and 7 are processed on machine 1 while jobs 1,3 and 8 are processed on machine 2. The remaining jobs are processed on machine 3. By applying the matrix of processing time p in example 2.1 to X_2, the gantt chart for the job assignment is shown in Fig.2 with $C_{max} = 19$.

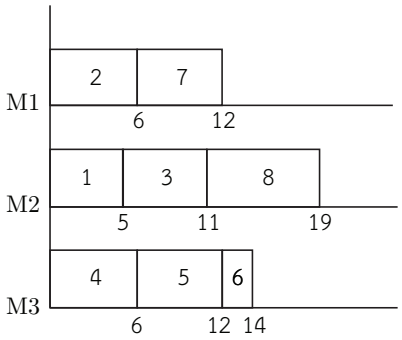

Figure 2: Gantt chart for the job assignment defined by matrix X_2.

Iteration 1 :

Step 1 : As seen from Fig.2, the machines with maximum and minimum total completion times are machine 2 and machine 1, respectively. Thus, a job on machine 2 is moved to machine 1. Now, $C_2 = 19$, $C_1 = 12$ and $C_{max} = 19$.

Step 2 : Determine a job with longest processing time on machine 2. Then, job 8 is selected for moving to machine 1.

Step 3 : Compute new makespan, $C^*_{max} = 17$. Because $C^*_{max} < C_{max}$, job 8 is processed on machine 1 and job schedule is presented in Fig.3(a). Update $C_1 = 17$, $C_2 = 11$, $C_3 = 14$ and $C_{max} = 17$.

Iteration 2 :

Step 1 : As seen from Fig.3(a), the machines with maximum and minimum total completion times are machine 1 and machine 2, respectively. Thus, a job on machine 1 is moved to machine 2. Now, $C_1 = 17$, $C_2 = 11$ and $C_{max} = 17$.

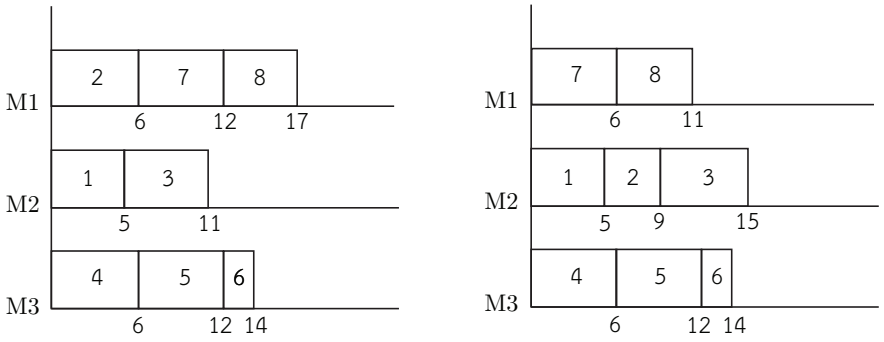

Figure 3: (a) Job schedule at iteration 1 with $C_{max} = 17$. (b) Job schedule at iteration 2 with $C_{max} = 15$.

Step 2 : Determine a job with longest processing time on machine 1. It can be seen that processing time of job 2 and job 7 are equal, job 2 is selected due to shortest processing time on machine 2.

Step 3 : Compute new makespan, $C^*_{max} = 15$. Because $C^*_{max} < C_{max}$, job 2 is processed on machine 2 and job schedule is presented in Fig.3(b). Update $C_1 = 11$, $C_2 = 15$, $C_3 = 14$ and $C_{max} = 15$.

Iteration 3 :

Step 1 : As seen from Fig.3(b), the machines with maximum and minimum total completion times are machine 2 and machine 1, respectively. Thus, a job on machine 2 is moved to machine 1. Now, $C_1 = 11$, $C_2 = 15$ and $C_{max} = 15$.

Step 2 : Determine a job with longest processing time on machine 1. Then, job 3 is selected to move.

Step 3 : Compute new makespan, $C^*_{max} = 15$. Because $C^*_{max} = C_{max}$, job 2 is selected instead for moving to machine 1 because of minimum processing time. Now, $C^*_{max} = 17$. It can be seen that C^*_{max} is greater than C_{max}. Thus, there is no job should be moved and job schedule at iteration 2 is final. The algorithm is terminated.

4 Experiments and Results

The performance and efficiency of the proposed PBIL algorithm combined with local search for the problem addressed previously are presented in this section. The problem instances consist of 20, 40, 60, 80 and 100 jobs. According to parallel machines and machine dependent processing time, the number of machines and job processing time are randomly generated as follows (Joo and Kim [8] and Hung et al. [19]).

(1) The number of machines is calculated by $M = \left[\frac{N}{\gamma} + 0.5\right]$, where M is varied by three values of parameter γ, $\gamma = 4, 5, 6$, and $[x]$ is the greatest integer less than x.

(2) The based processing time of job i and the machine adjusting factor of job i on machine m , b_i and δ_{im}, are randomly generated in the range of $[3, 25]$ and $[0.5, 1.5]$, respectively.

(3) Processing time of job i on machine m, p_{im}, is calculated by $b_i \times \delta_{im}$.

To validate and illustrate the performance of the proposed algorithm, 5 problems are randomly generated for each level of parameters and 5 run times are performed for each problem such that the least one is selected to be the best solution of such a problem. By the different levels of parameters used to generate the problems, a total of 375 instances are tested. In order to obtain job schedule of the problems, PBIL combined with local search addressed in section 3 is applied. Population size used in the study is 100 and the initial probabilities of all positions in probability vector is set based on the solution obtained from SPT algorithm in which local search is applied and such a solution is set to be one of population. Learning rate, $LR = 0.05$, is used to update the probability vector for the next iteration. The stopping criteria is set to a maximum number of iterations, 500 iterations. The results obtained from the proposed algorithm are compared to SPT stated in section 2.2 and the relative percentage deviation (RPD) of each problem can be calculated by equation (4.1).

$$\text{RPD} = \frac{\text{SPT}_{\text{sol}} - \text{PBIL}_{\text{sol}}}{\text{PBIL}_{\text{sol}}} \times 100 \tag{4.1}$$

where SPT_{sol} and PBIL_{sol} are solutions derived from SPT and PBIL combined with local search, respectively.

Table 1 is shown the average RPD between makespan obtained from SPT and the proposed algorithm for each level of parameters indicating the decrease of makespan. For $\gamma = 4$, the number of machines is more than the other cases resulting in the opportunity to move a job to a new machine in local search step and the average RPD is more than 20 % for all cases. The number of machines can be decreased by the increase of γ. It can be seen that the average RPD is decreased when the number of machines is decreased. By this situation, the result is shown that PBIL introduced in the study has performed better than SPT algorithm

Table 1: The average RPD between makespan of SPT and PBIL for each instance.

Number of jobs	γ	Number of machines	Average RPD
20	4	5	23.88
	5	4	24.93
	6	3	23.49
40	4	10	24.75
	5	8	24.29
	6	7	19.91
60	4	15	28.25
	5	12	23.92
	6	10	23.38
80	4	20	22.25
	5	16	18.40
	6	13	20.77
100	4	25	22.55
	5	20	18.19
	6	17	16.75

although the average RPD is decreased, as seen from average RPD which is more than 16%. Overall, the results shown in table 1 indicate good performance of proposed algorithm.

5 Conclusion

Minimization of makespan for parallel machines scheduling problem with machine dependent processing time is studied. The problem is divided into two parts, assignment of jobs to machines and improvement of schedule. PBIL is used to represent the jobs assignment on the machines. The initial population is created based on the solution obtained from SPT algorithm in which local search step is applied. To improve the solutions, local search is performed after job assignment is carried out by using PBIL to move a job to new machine with decreasing makespan. To show the efficiency of proposed algorithm, the average RPD between makespan obtained from SPT and PBIL combined with local search is presented. The results show that using the proposed algorithm can reduce makespan although the number of machines is decreased and it is suitable for solving parallel machines scheduling problem with machine dependent processing time.

Acknowledgement : I would like to thank the referee(s) for his comments and suggestions on the manuscript.

References

[1] J.P. Arnaout, G. Rabadi, R. Musa, A Two-stage Ant Colony Optimization Algorithm to Minimize the Makespan on Unrelated Parallel Machines with Sequence-dependent Setup Times, Journal of Intelligent Manufacturing 21 (2010) 693–701.

[2] K. Li, Y. Shi, S.L. Yang, B.Y. Cheng, Parallel Machine Scheduling Problem to Minimize the Makespan with Resource Dependent Processing Times, Applied Soft Computing 11 (2011) 5551–5557.

[3] S. Balin, Non-identical parallel machine scheduling using genetic algorithm, Expert Systems with Applications 38 (2011) 6814–6821.

[4] E. Vallada, R. Ruiz, A Genetic Algorithm for the Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times, European Journal of Operational Research 211 (2011) 612–622.

[5] P. Alcan, H. Başlıgil, A genetic algorithm application using fuzzy processing times in non-identical parallel machine scheduling problem, Advances in Engineering Software 45 (2012) 272–280.

[6] F.A. Cappadonna, A. Costa, S. Fichera, Makespan Minimization of Unrelated Parallel Machines with Limited Human Resources, Procedia CIRP 12 (2013) 450–455.

[7] W.C. Yeh, P.J. Lai, W.C. Lee, M.C. Chuang, Parallel-machine Scheduling to Minimize Makespan with Fuzzy Processing Times and Learning Effects, Information Sciences 269 (2014) 142–158.

[8] C.M. Joo, B.S. Kim, Hybrid Genetic Algorithms with Dispatching rules for Unrelated Parallel Machine Scheduling with Setup Time and Production Availability, Computer & Industrial Engineering 85 (2015) 102–109.

[9] R.O.M. Diana, M.F. de França Filho, S.R. de Souza, J.F. de Almeida Vitor, An Immune-inspired Algorithm for an Unrelated Parallel Machines' Scheduling Problem with Sequence and Machine Dependent Setup-times for Makespan Minimisation, Neurocomputing 163 (2015) 94–105.

[10] V. Sels, J. Coelho, A.M. Dias, M. Vanhouck, Hybrid Tabu Search and Truncated Branch and Bound for the Unrelated Parallel Machine Scheduling Problem, Computers & Operations Research 53 (2015) 107–117.

[11] H. Kılıç, U. Yüzgeç, Improved antlion optimization algorithm via tournament selection and its application to parallel machine scheduling, Computers & Industrial Engineering (2019) 166–186.

[12] S. Baluja, Population-Based Incremental Learning : A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, Technical Report CMU-CS-94-163, Carnegie Mellon University, USA, 1994.

[13] K.A. Folly, Performance evaluation of power system stabilizers based on Population-Based Incremental Learning (PBIL) algorithm, Electrical Power and Energy Systems 33 (2011) 1279–1287.

[14] H. Pang, K. Hu, Z. Hong, Adaptive PBIL Algorithm and Its Application to Solve Scheduling Problem, 2006 IEEE Conference on Computer Aided Control System Design (2006) 784–789.

[15] S. Wan, D. Qiu, Vehicle Routing Optimization Problem with Time Constraint Using Advanced PBIL Algorithm, 2008 IEEE International Conference on Service Operations and Logistics, and Informatics (2008) 1394–1398.

[16] N. Chen, S. Fang, X. Wang, A Cloud Computing Resource Scheduling Scheme Based on Estimation of Distribution Algorithm, The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014) (2014) 304–308.

[17] M.H. Chen, P.C. Chang, J.L. Wu, A population-based incremental learning approach with artificial immune system for network intrusion detection, Engineering Applications of Artificial Intelligence 51 (2016) 171–184.

[18] X. Meng, J. Li, MC. Zhou, X. Dai, J. Dou, Population-based Incremental Learning Algorithm for a Serial Colored Travelling Salesman Problem, IEEE Transaction on Systems, Man and Cybernetics: Systems 48 (2018) 277–288.

[19] Y.F. Hung, J.S. Bao, Y.E. Cheng, Minimizing Earliness and Tardiness Costs in Scheduling Jobs with Time Windows, Computers & Industrial Engineering 113 (2017) 871–890.

(Received 14 June 2019)
(Accepted 24 December 2019)