A Survey on Multiprocessor Scheduling Using Evolutionary Technique

Annu Priya and Sudip Kumar Sahana

Abstract In this paper, various conventional approaches are studied for the task scheduling, precedence–resource constrained, load balancing, and multiprocessor scheduling problems. In parallel machines the sequence of dependent execution setup time for the minimization of makespan in scheduling problems and prepared a concise review. Multiprocessor scheduling is an NP-hard problem, whereas scheduling algorithm schedules the tasks which may or may not be dependent on each other. There are several traditional approaches existing for processor scheduling such as modified critical path (MCP), dominant sequence clustering (DSC), and priority-based multichromosome (PMC). While using these approaches, we achieve partial solutions in less than the minimum computing time. In this paper, an innovative multiprocessor scheduling technique that is inspired by evolutionary techniques has been embodied.

Keywords DAG · Genetic algorithm · Ant colony optimization

1 Introduction

Scheduling algorithm allows us to manage the sequencing of tasks, confine the scheduled task, time management, and optimizing effort and assignments in any production or manufacturing process. Scheduling is used to allocate machinery resources to tasks. Processor scheduling is required for the high performance of computing systems. There are different heuristic methods existing for processor scheduling. Multiprocessor scheduling is considered as NP-hard problems, which are very difficult to solve with conventional techniques. Processor scheduling is

A. Priya (✉) · S. K. Sahana
Department of Computer Science Engineering, Birla Institute of Technology, Mesra, India
e-mail: annu.priya12@yahoo.com

S. K. Sahana
e-mail: sudipsahana@bitmesra.ac.in

V. Nath and J. K. Mandal (eds.), *Nanoelectronics, Circuits and Communication Systems*, Lecture Notes in Electrical Engineering 511,
https://doi.org/10.1007/978-981-13-0776-8_14

classified into two categories: local processor scheduling and global processor scheduling. Conventional techniques mostly use local processor scheduling. Global processor scheduling is grouped into several families depending on scheduling processor techniques. These heuristic scheduling methods are biologically inspired and can be further grouped into several classes of heuristic approaches like an ant colony, bee colony, genetic algorithm, particle swarm optimization, etc. Local processor scheduling is scheduled upon the single-processor platform and also uses list scheduling to construct a schedule, one cycle at a time, whereas in global scheduling it uses the multiprocessor environment. In real-time task scheduling problem on multiprocessor, it has goal of meeting the deadline, and infrequent arrival of task has implicit deadline and such type of scheduling problems is handled by global static processor scheduling techniques. Static processor scheduling is deterministic scheduling policy. It is classified into two subclasses: optimal processor scheduling and suboptimal processor scheduling. Optimal processor scheduling is used when the job executed at that scheduler should know information about the requirement of processor and state of processors. Optimal scheduling reached NP-complete problem, so researchers focused on suboptimal processor scheduling that is classified as approximation and heuristic algorithm. Approximation scheduling uses the computational techniques for searching of entire solution space for an optimal solution. Heuristic approaches such as ant colony optimization, genetic algorithm, etc. provide feasible solutions to a scheduling problem which cannot give the optimal solution. Global dynamic scheduling had better to physically be inherent in a single processor. In dynamic processor scheduling techniques, a number of processors vary during execution. Dynamic processor scheduling leads to degradation of performance because of high run-time overhead. It has two subclasses: physically non-distributed processor scheduling and physically distributed processor scheduling. Physically non-distributed system includes the responsibility for the task scheduling, whereas in physically distributed system the work involved in decision-making should be physically spread among the various processors. Physically distributed processor scheduling is divided into two classes: (i) cooperative processor scheduling and (ii) non-cooperative processor scheduling. In non-cooperative processor scheduling, each separate processor acts as autonomous entities and it works as a decision-making for various resources running independently which affects the decision of rest system. In cooperative system, the goal of each processor is to carry its own part of scheduling and its work toward a common system. In the case of static system, the decision is taken for the taxonomy tree which reaches to the optimal, and for the consideration of the system the suboptimal heuristic solution is to be taken. Non-cooperative processor scheduling is classified into two categories: (i) approximation processor scheduling and (ii) heuristic processor scheduling. In the case of approximate processor scheduling, it is presented for the independent tasks and runs on different processors with different speeds in a multiprocessor environment. Approximation scheduling provides the low polynomial time complexity to the problems. These types of scheduling processes are secure by obtaining the solutions which are closest to the optimal solution. Heuristic processor scheduling approaches are used to find the suboptimal solution in considerable computation time. Heuristic-based method provide nearest optimal solution, and as a result of its appearance, computation time in GA is very less. The rest of this paper is organized as follows: Sect. 2 included processor scheduling techniques. Section 3 contains the literature survey of existing methods for processor scheduling. Section 4 contains problem statement. Section 5 concludes the discussion of comparison of different algorithms. Section 6 concludes about the future implementation and outlines the future scope of this work.

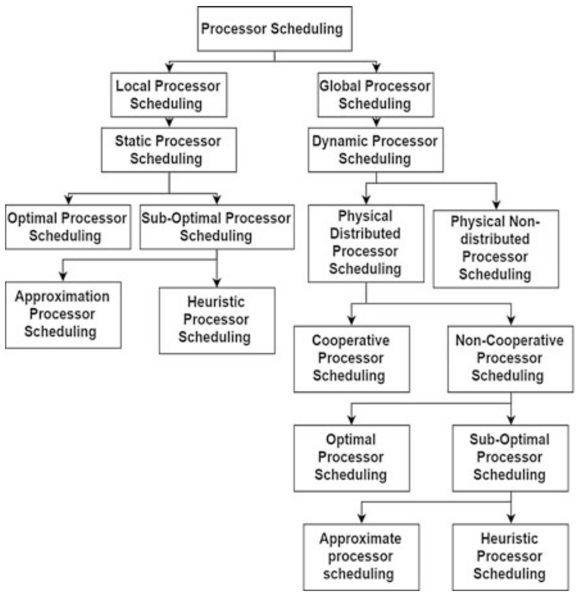
2 Processor Scheduling

Processor scheduling is classified into two categories: local processor scheduling and global processor scheduling. Conventional techniques mostly use local processor scheduling. Figure 1 shows the classification of process scheduling.

2.1 *Global Processor Scheduling*

Global processor scheduling is grouped into several families depending on scheduling processor techniques. These heuristic scheduling methods are

Fig. 1 Classification of processor scheduling


Processor Scheduling
Local Processor Scheduling
Global Processor Scheduling
Static Processor Scheduling
Dynamic Processor Scheduling
Optimal Processor Scheduling
Sub-Optimal Processor Scheduling
Physical Distributed Processor Scheduling
Physical Non-distributed Processor Scheduling
Approximation Processor Scheduling
Heuristic Processor Scheduling
Cooperative Processor Scheduling
Non-Cooperative Processor Scheduling
Optimal Processor Scheduling
Sub-Optimal Processor Scheduling
Approximate processor scheduling
Heuristic Processor Scheduling

biologically inspired and can be further grouped into several classes of heuristic approaches like an ant colony, bee colony, genetic algorithm, particle swarm optimization, etc.

2.2 *Local Processor Scheduling*

Local processor scheduling is scheduled upon the single-processor platform and also uses list scheduling to construct a schedule, one cycle at a time, whereas global scheduling processor is scheduled upon multiprocessor platform. In real-time task scheduling problem, the goal of multiprocessor is to meet deadlines, whereas the arrival of tasks is infrequent which has implicit deadlines. To deal with this type of scheduling problem, a global static processor scheduling technique is used.

2.3 *Static Processor Scheduling*

Static processor scheduling [1, 2] is deterministic scheduling policy. Optimal processor scheduling is used when the job executed at that scheduler should know information about the requirement of the processor and state of processors. The static processor scheduling is divided into two categories: (i) optimal scheduling and (ii) suboptimal processor scheduling. Optimal scheduling reached NP-complete problem, and researchers focused on suboptimal processor scheduling that is classified as approximation and heuristic algorithm. Approximation scheduling technique uses the computational methods for searching of entire solution space for an optimal solution. Heuristic approaches such as ant colony optimization, genetic algorithm, etc. provide feasible solutions to a scheduling problem which cannot give the optimal solution. Suboptimal processor scheduling has two sub-branches: (i) approximate processor scheduling and (ii) heuristic processor scheduling. In multiprocessor environment, the approximate processor scheduling is used to plan the independent tasks where the speed of the processors varies. Approximation scheduling provides the low polynomial complexity to the problems. By using approximation scheduling, we are assured to achieve the optimal solutions. Heuristic processor scheduling approaches find the suboptimal solution in significant time. In heuristic-based approaches, the results of computational time are very optimal and the initial population size is considerably nearby to the solution.

2.4 *Dynamic Processor Scheduling*

In dynamic processor scheduling techniques [3, 4], the allocated number of processors varies during execution. Dynamic processor scheduling leads to the degradation of performance because of high run-time overhead. Dynamic processor scheduling is divided into two categories: (i) physical distributed processor scheduling and (ii) physical non-distributed processor scheduling. For responsible task scheduling, the physically non-distributed processor scheduling is required, whereas, while using the global dynamic scheduling which is inherent in a single processor and also known as a physically non-distributed, the work involved is to make a decision which should be physically distributed among the processor. The physical distributed processor scheduling is divided into two subcategories: (i) non-cooperative processor scheduling (ii) cooperative processor scheduling. In non-cooperative processor scheduling the individual processor acts as autonomous entities, arrives at decision-making, and makes a choice to select the resources which are independent in nature which affects the decision of the rest system, and in cooperative processor scheduling the goal of each processor is to carry out its own portion of task, whereas all processors are working toward the common system. The same discussion has to be presented in the static case and applies for the dynamic case also. When the taxonomy tree reaches the bottom, we have to consider an optimal [3], suboptimal, and heuristic [3] solutions for the multiprocessor.

3 Literature Survey

There are various research has been done in this field such as C. Jianer and L. Chung Yee proposed a model for processor scheduling in which several alternatives are used a process job and each alternative several machine process the assigned job to them. They invented pseudo-polynomial algorithm [3] to solve optimally two-machine processor problem and also provide a heuristic scheduling algorithm to solve the three-machine processor problem to minimize the completion time for all jobs. X. Yuming and L. Kenli address the task scheduling problems and suggested multiple priorities queuing genetic scheduling [5] for distributed system as well as parallel heterogeneous computing system. They used HEFT approach to search an optimal solution for mapping task of the processor. L. Shih-Tang et al. presented a modified ACO approach DDACS [6] to minimize precedence and resource constraint in multiprocessor scheduling problem. Here, to represent the scheduling problem, a matrix graph is adopted. This matrix graph is used to minimize makespan schedule. A. Hadi Lotfi et al. proposed new coarse-grain genetic algorithm [7] to schedule the tasks and reduce the solution search space and to prevent the speedy convergence between the subpopulation. The initial population is divided into multiple subpopulations and the experimental results of this paper show that the proposed technique will reduce the makespan and it also achieves a better scheduling method in comparison with the other existing methods such as MCP and genetic algorithm. K. Yan and Z. Zhenchao proposed an activity-based genetic scheduling algorithm [8] in which the scheduled tasks run on the heterogeneous grid system network that is signified by directed acyclic graphs (DAGs). First, this approach list all the nodes according to the scheduling algorithm to generate the initial population of GA and it also represents the possible operation sequences so that it reduces the coding space when compared to permutation representation. This approach assigns tasks as activity on the processor to improve quality of the random probability solution and the activity is added to crossover and mutation operator. In Table 1, represent the various techniques proposed by different researchers for solving the problem of processor scheduling.

There are various methods like DSC, MCP, LC, PGA, etc. used for scheduling algorithm. The linear clustering (LC) algorithm uses the recursive grouping of all the nodes in the critical path while zeroing all the edges on the path in one step. In the second step, it schedules the same processor for all partitions that do not execute concurrently. Dominant sequence clustering (DSC) uses the two major ideas: (i) to directly reduce the dominant sequence of the graph and (ii) to create an algorithm with low computational complexity. DSC keeps track of the dominant sequence to reduce parallel time. The complexity order of the algorithm is also reduced. We need to specify certain constraints and definitions to describe DSC such as (i) scheduled: if a processor has been assigned to a node, it should be scheduled; (ii) free: if the processor is unscheduled and all its predecessors are arranged, then the node is free; and (iii) partial free: if it is unscheduled and at least one of its predecessors is unscheduled, then the node is partially free. In this algorithm, the complexity is reduced by confining the range of edges to be zero. Priority-based genetic algorithm (PGA) is based on genetic algorithm, whereas the gene location is used to signify the task node and construct the scheduler among candidate, and the priority of the task node value is used. The proposed PGA method [9] first generates the initial random chromosome, whereas each chromosome is called a gene and each gene is used in the priority node in the DAG structure. According to this technique, it easily validates any encoding permutation to the corresponding scheduler, so that the most recent operator is used for the encoding. Modified critical path algorithm (MCP) explains that the precedence list is prepared on the basis of the "highest bot-level first" ordering. The reason behind is that if the same task has same priority, then to overcome with the critical path the algorithm breaks the task and ties by using the highest priority to the successor tasks and the second highest priority of its successors, and so on. The major objective of this algorithm is to minimize the implementation time so that it becomes more cost-effective. Calculate the cost function of the algorithm in the parallel time environment which is equal to

$$\mathrm{PT} = \max(\mathrm{ST}(nj) + Tj) \leq \max\mathrm{ST}(n_j) + \max Ti.$$

This above equation shows that if we minimize the initial time of the last task then we can get the result in the decline of the overall parallel time.

Table 1 Summary of research development in the field of processor scheduling

Research	Methodology	Technique used	Advantages	Disadvantages
Hadi lotfi et al. [7]	(i) Divided the population into multi-sub-population (ii) Reduce the searching speed (iii) Prevent the convergence and migration between the subpopulations	Coarse-grained genetic algorithm	Minimizing makespan Minimizing communication cost Maximizing CPU utilization	For further solution multi-population combinations of genetic algorithm required
Yan and Zhenchao [8]	(i) Here, GA represents the feasible operation and sequences (ii) Reduce the coding space	Activity-based genetic task scheduling algorithm	The quality of the solution will enhance	The iterative algorithm does not perform well
Ravreet and Gurvinder [11]	Leads to the suboptimal solution	Genetic algorithm (GA) is proposed for static	Minimizing the job completion time, completion time	Not feasible for the multiprocessor systems
Rami and Khalid [14]	(1) One way to prepare the chromosome (2) It uses a new methodical process for the crossover operator	Generic algorithm to resolve the task scheduling problem	(i) Minimize the task completion time (ii) Make the most of the throughput of the system (iii) Decrease the length of schedule	Scheduling tasks in homogeneous parallel multiprocessor systems.
Ricardo et al. [15]	Improved genetic algorithm with the introduction of some knowledge about the scheduling problem	Genetic algorithm-based scheduling problem heuristic in the crossover and mutation genetic	(i) Program's execution time is minimized (ii) Produce good quality solutions in shorter time	Works on homogeneous multiprocessor system
Rashid and Deniz [16]	Genetic algorithm used for optimization of multiprocessor in	Multi-population-based parallel genetic algorithm	(i) Reduced the implementation time	(ii) Improved the quality of scheduling solutions

(continued)

Table 1 (continued)

Research	Methodology	Technique used	Advantages	Disadvantages
	the presence of communication costs.			
Jianer and Chung Yee [3]	(1) Schedule jobs to minimize the completion time of all jobs (2) Machines process the job assigned	Pseudo-polynomial algorithm	Combine fully polynomial scheme and a heuristic to solve the three-machine problem	Integer programming cannot be solved
Yuming and Kenli [5]	Advantages of both evolutionary and heuristic-based algorithms	Multiple priorities queuing genetic algorithm (MPQGA)	(i) Minimizing makespan	Not good for larger task graphs and more processors
Ghafarian et al. [17]	Combined the use of ant colony and evolutionary metaheuristics to search	Cellular automata (CA)	Execution time is minimized.	Get an optimal solution before generation 20
Kumar et al. [18]	Comparative study of the performance of two algorithms GFTS and AFTS	Primary backup (PB)-based fault-tolerant scheduling (PBFTS) technique	Minimize the makespan	Getting the immediate optimal possible results for the non-preemptive scheduling on task sets
Shih et al. [10]	(1) Two-dimensional matrix is used for assigning the jobs on the processors (2) The dynamic rule applied to adjust the earliest starting time of jobs	Modified ant colony optimization (ACO)	(1) Solve the multiprocessor system scheduling problems with resource constraints (2) Minimize makespan	Works for homogeneous processors
Savas [19]	Applying a GA to adapt non-identical parallel machine scheduling problem	Genetic algorithm	Solve hard combinatorial optimization problems	(1) Get better result for small-scale problems (2) Dispatching the rule does not guarantee good result in various problems

4 Problem Statement

From the available literature, it is observed that performance of the different techniques on a particular scenario exhibits as shown in Fig. 2. In multiprocessor scheduling, the critical problem is how to provide the task precedence relation between tasks and processor so that program's execution time is minimized. In real time, the most scheduling problems are NP-hard in nature and it is a very complex problem. For that, performance from the start of the algorithm is very poor. This problem is extremely hard to solve. Most scheduling problems are NP-hard [10] in nature. Scheduling algorithms for processors in hard real time are also complex problem. For such large-scale scheduling problem, the performance of state-of-the-art algorithms is very poor. It is observed that evolutionary and swarm-based algorithms exhibit better performance for large-scale combinatorial problems. Our objective is to analyze, study, apply, and implement possible improvements in the said application using evolutionary and swarm-based

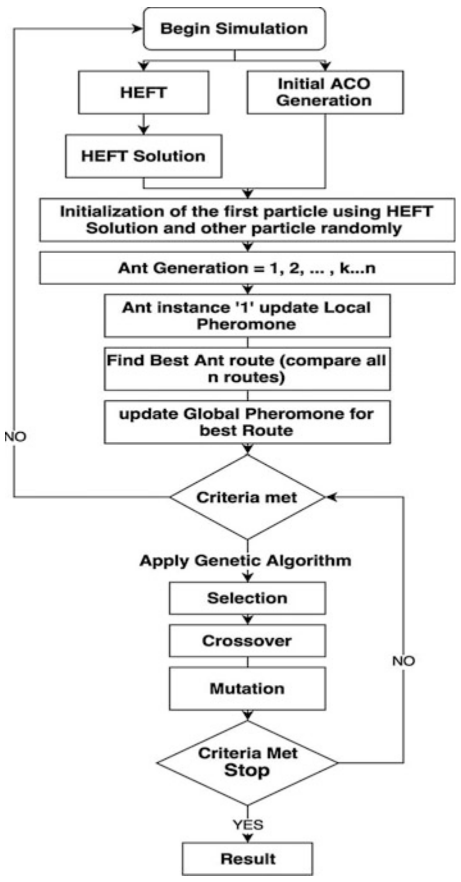

Begin Simulation
HEFT
Initial ACO Generation
HEFT Solution
Initialization of the first particle using HEFT Solution and other particle randomly
Ant Generation = 1, 2, ... , k...n
Ant instance '1' update Local Pheromone
Find Best Ant route (compare all n routes)
update Global Pheromone for best Route
NO
Criteria met
Apply Genetic Algorithm
Selection
Crossover
Mutation
NO
Criteria Met Stop
YES
Result

Fig. 2 Flowchart of hybrid system based on ACO and GA with HEFT for task scheduling on heterogeneous nodes and processors

algorithms. After an exhaustive study of the different research papers and case studies, we like to propose a hybrid system based on ACO and GA with HEFT for task scheduling on heterogeneous nodes and processors. The framework consists of two parts: in the first part, ACO and HEFT which are combined and used for finding the local pheromone and global pheromone, whereas in the second part the result of the ACO and HEFT is taken as input for the GA and applied to the selection, crossover, and mutation process.

5 Discussion

Evolutionary techniques effectively solve the scheduling problems. These methods apply the skills of evolutionary and it is observed that performance of the different techniques on particular scenario exhibits superiority of GA over traditional heuristic approaches as shown in Fig. 2. Generally, genetic algorithm is more efficient to solve NP-complete problem of multiprocessor scheduling problems. Here, in this paper, we have discussed some of the processor scheduling algorithms inspired by evolutionary techniques. A comparative study was carried out, and merit and demerit areas are highlighted. Previously, it is observed that increasing the number of task increases the value of makespan as compared to GA [11] and LA [12] methods. After that, coarse-grain genetic algorithm [13] shows that it has the lower makespan value of proposed algorithm than other algorithms. To make this comparison, various algorithms and different parameters have been taken for the execution.

Figure 3 shows that the makespan of MCP is higher than the other five algorithms, whereas DSC, DCP, and MD have the similar makespan, but the number of processors is higher for DSC to achieve the results of MD and DCP algorithm.

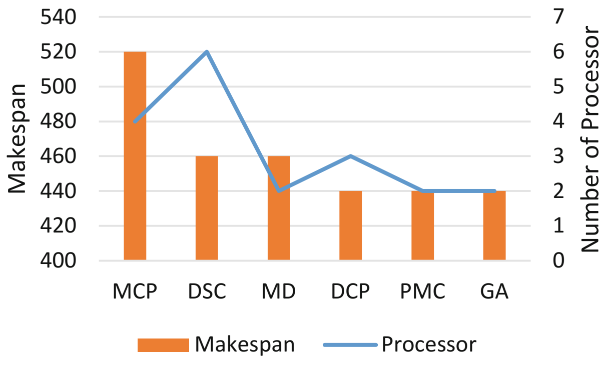


Fig. 3 Comparison between different algorithms

The GA and PMC have used the same number of processor to achieve the same makespan. The comparison of this technique gave the overall scenario for the processor scheduling environment.

6 Conclusion

In this paper, a survey is drawn between different traditional scheduling algorithms used in the parallel multiprocessor system. And it is proven that by using the evolutionary technique such as GA we get the better results. Here, in this paper, we proposed a framework for the multiprocessor task scheduling model using ACO and HEFT. The hybrid structure will generate the best possible result in conventional CPU time. And it also forms the suboptimal solution for allocating the tasks to the homogeneous parallel multiprocessor system. Performance of the hybrid ACO and HEFT scheduling algorithm will produce the better result than GA and other traditional approaches for job scheduling in a multiprocessor environment.

References

1. AI Na'mneh RA, Darabkh KA (2013) A new genetic-based algorithm for scheduling static tasks in homogeneous parallel systems. In: International conference on robotics, biomimetics, intelligent computational systems (ROBIONETICS) Yogyakarta, Indonesia, Nov 2013
2. Wu, Yu H, Jin S, Lin K-C, Schiavone G (2004) An incremental genetic algorithm approach to multiprocessors scheduling. IEEE Trans Parallel Distrib Syst 15(9):824–834
3. Jianer C, Chung-Yee L (1999) General multiprocessor task scheduling. Wiley, Hoboken
4. Apostolos G, Tao Y (1992) A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors. J Parallel Distrib Comput
5. Yuming X, Kenli L, Tung Truong K, Meikang Q (2012) A multiple priority queueing genetic algorithm for task scheduling on heterogeneous computing systems. In: IEEE 14th international conference on high-performance computing and communications 2012
6. Shih-Tang L, Ruey-Maw C, Yueh-Min H, Chung-Lun W (2007) Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system. Elsevier Ltd
7. Hadi lotfii A, Broumandnia A, Shahriar A (2010) Task graph scheduling in multiprocessor systems using a coarse grained genetic algorithm. In: IEEE 2nd international conference on computer technology and development (ICCTD 2010)
8. Yan K, Zhenchao Z, Pengwu C (2011) An activity-based genetic algorithm approach to multiprocessor scheduling. In: IEEE seventh international conference on natural computation
9. ReaKook H, Mitsuo G, Hiroshi K (2006) A performance evaluation of multiprocessor scheduling with genetic algorithm. ReaKook Hwang et al./Asia Pac Manag Rev 11(2):67–72
10. Shih T, Ruey MC, Yueh-Min H, Chung-Lun W (2007) Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system. Elsevier Ltd
11. Ravreet K, Gurvinder S (2012) Genetic algorithm solution for scheduling jobs in multiprocessor environment. IEEE

12. Jahanshahi M, Meybodi MR, Dehghan M (2009) A new approach for task scheduling in distributed systems using learning automata. In: Proceedings of the IEEE international conference on automation and logistics Shenyang, China, Aug 2009
13. Hadi L, Ali B, Shahriar L (2010) Task graph scheduling in multiprocessor systems using a coarse grained genetic algorithm. In: IEEE 2nd international conference on computer technology and development (ICCTD 2010)
14. Rami A, Khalid A (2013) A new genetic-based algorithm for scheduling static tasks in homogeneous parallel systems. In: IEEE international conference on robotics, biomimetics, intelligent computational systems (ROBIONETICS) Yogyakarta, Indonesia, Nov 25–27, 2013
15. Ricardo C, Afonso F, Pascal R (1999) Scheduling multiprocessor tasks with genetic algorithms. IEEE Trans Parallel Distrib Syst 10(8) (Aug 1999)
16. Rashid M, Deniz D (2016) A multi-population based parallel genetic algorithm for multiprocessor task scheduling with communication costs. In: IEEE symposium on computers and communication (ISCC)
17. Ghafarian T, Deldari H, Mohammad R (2009) Multiprocessor scheduling with evolving cellular automata based on ant colony optimization. In: IEEE proceedings of the 14th international CSI computer conference (CSICC’09), 2009
18. Kumar A et al (2014) Aco and Ga based fault-tolerant scheduling of real-time tasks on multiprocessor systems—a comparative study. IEEE
19. Savas_ Balin (2010) Non-identical parallel machine scheduling using genetic algorithm. Elsevier Ltd
20. Kwok Y, Ahmad I (1999) Static scheduling algorithm for allocating directed task graph to multiprocessors. ACM Comput Surv 31(4) (Dec 1999)