Arabian Journal for Science and Engineering (2019) 44:3943–3952
https://doi.org/10.1007/s13369-018-03698-2

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Parental Prioritization-Based Task Scheduling in Heterogeneous Systems

Muhammad Shahzad Arif[1] · Zeshan Iqbal[1] · Rehan Tariq[2] · Farhan Aadil[2] · Muhammad Awais[1]

Received: 21 April 2018 / Accepted: 19 December 2018 / Published online: 17 January 2019

Abstract
Efficient task scheduling is important for achieving high performance in heterogeneous distributed computing systems. The main focus of this research is to build a task scheduling algorithm for a heterogeneous environment. We proposed an algorithm named parental prioritization earliest finish time. It has two phases, tasks prioritization phase and processor assigning phase. In the tasks prioritization phase, tasks will schedule in parental priority queue (PPQ) on the basis of downward rank and parental priority. Task prioritization is based on the directed acyclic graph. It can schedule the task of successor row before the current row if it has less communication cost. In the processor assigning phase, the processor will allocate to the scheduled tasks obtained from PPQ keeping the computation cost to a minimum. This proposed algorithm is compared with HEFT and CPOP algorithms through graphs generated from random task graph generator and a set of tasks. The experimental results show that our proposed scheduling algorithm performs significantly better than other algorithms in terms of both cost and makespan of schedules.

Keywords Heterogeneous systems · Task scheduling · Cloud computing · Directed acyclic graph · Parental priority queue

1 Introduction

A cloud computing is an on-demand provision of processing capability and storage of data and provides a different type of resources through the Internet. Through Cloud, the customer can gain access to computing power, storage and number of software services by using different applications. Microsoft Azure, Amazon Web Services and IBM are the cloud service providers for customers. The cloud computing can be linked to a large number of resources. Resources size can be dynamically adjusted according to the customer demand, by which customer can make full use of different resources in the cloud computing. The cloud service provider gives different services to the customers, such as infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS).

With increasing tendency of usage of heterogeneous computing systems, scheduling of tasks is very important to achieve high reliability, computing powers, scalability and accessibility in a number of research fields. A group of connected computers has the same specifications (processing capabilities, RAM, etc.) called a homogeneous system, while the group of connected computers has different specifications called a heterogeneous system. Homogeneous distributed systems are easy to build and administrate because all processing units and their capabilities are same, whereas heterogeneous distributed systems are complex to build and administrate because every node has different processing capabilities and different internal bus architectures [1]. Heterogeneous distributed systems are capable of solving different types of problems with the same magnitude. Homo-

✉ Muhammad Shahzad Arif
mshehzad77@gmail.com

Zeshan Iqbal
zeshan.iqbal@uettaxila.edu.pk

Rehan Tariq
rehan@ciit-attock.edu.pk

Farhan Aadil
farhan.aadil@ciit-attock.edu.pk

Muhammad Awais
awaisntu@gmail.com

[1] Department of Computer Science, University of Engineering and Technology, Taxila, Pakistan

[2] Department of Computer Science, COMSATS Institute of Science and Technology, Attock, Pakistan

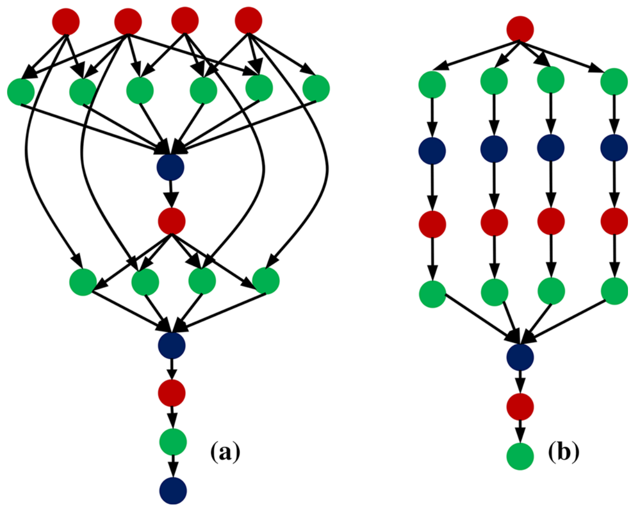

Fig. 1 Scientific workflows. **a** Epigenomics. **b** Montage

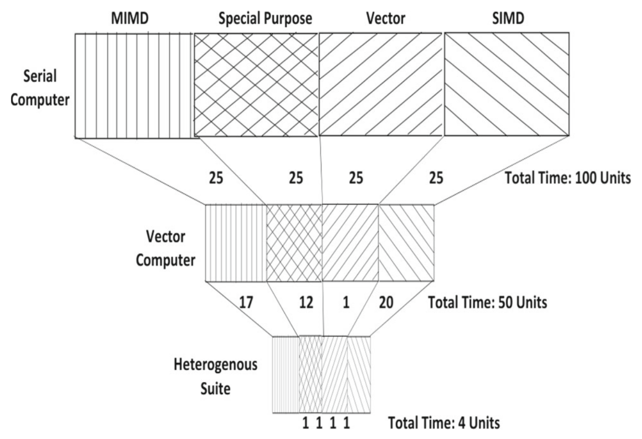

Fig. 2 Hypothetical homogeneous and heterogeneous systems

geneous distributed systems can only work efficiently for the same type of tasks.

The availability of a library of scientific workflows is very helpful in the development and comparison of task scheduling heterogeneous systems. Such workflows allow the scheduling and data management for algorithms within workflow systems. Today some scientific workflows have been available for open use, including the access to their data and code. The Montage [2] workflow is one of the main workflows which is used for the evaluation of task scheduling algorithms. It is an open-source toolkit created by NASA Infrared Science Archive. The Epigenomics workflow [3] is basically a data processing pipeline that uses the Pegasus workflow management system to automate the implementation of the numerous genome sequencing processes. The Epigenomics workflow is the map of epigenetic state of human cells. Figure 1 shows the approximate structure of both the workflows used in our experiments.

In Fig. 2, we have discussed homogeneous and heterogeneous distributed systems. We have four different computers, vector, multiple instruction stream multiple data streams (MIMD), single instruction stream multiple data stream (SIMD) and special purpose. It is assumed that when homogeneous systems work independently each computer takes 25 units of time to solve a problem. When these are connected in serial, total computation time is 100 units of time. When we execute the same problem on vector system, vector application solved in 1 unit of time, MIMD application in 17 units of time, SIMD application in 20 units of time and special purpose application in 12 units of time. Total execution time on vector system is 50. Now we execute the same problem on heterogeneous distributed system, all different applications having 1 unit of time for execution. Total 4 units of time execute all the problems in the heterogeneous suite, which is better as compared to serial homogeneous systems (100 units of time) and vector systems (50 units of time). This simply tells us that high performance can only be achieved by using heterogeneous distributed systems.

Heterogeneous distributed computing systems (HeDCSs) are interconnected resources having different processing capabilities, RAM and speed of computation [4]. When an application is executed in HeDCSs environment, the main concern to be observed is task scheduling. The better task scheduling can only be possible to achieve better computing performance as shown in Fig. 1. Any negligence or improper schedule implicitly degrades the performance of the whole computing system. Scheduling of tasks focuses on plotting the tasks on the existing set of resources without violating the priority constraints and aiming to get minimum schedule time [5]. Task scheduling algorithms are classified into two main classes, dynamic and static task scheduling. In dynamic task scheduling, decision can be made at runtime of tasks, while in static task scheduling, decision can be made at compile time of tasks, data dependencies, priorities, etc. [6].

In this paper, we are focusing on the scheduling of dependent tasks in heterogeneous computing environment. We proposed a new task scheduling algorithm, namely parental prioritization earliest finish time (PPEFT). The PPEFT algorithm is proposed to schedule dependent tasks of directed acyclic graph (DAG) tasks. For experimentation setup, we generated DAG from random task graph generator (RTGG). Also we used standard real-world workflow applications (Epigenomics and Montage) for experimentation.

Task scheduling is done with processing units of the same capabilities having negligible communication cost. In this situation, assigning a task with high communication cost to the same processor results in low makespan. The inspiration behind this work is to develop a new algorithm for task scheduling to achieve high performance in terms of cost and time. We are trying to assign tasks to the same machine and if not possible then trying to pick another machine having minimum computation time of that task. DAG properties must be observed, i.e., children cannot get execution time until their parents are executed [7]. Then, we pick the processing machine having minimum computation time of that picked task.

2 Related Work

Scheduling and distribution of resources in cloud computing affect the overall performance of the system. For scheduling and scaling of resources, numbers of researchers have developed different algorithms which perform scheduling in a well-organized way. Existing algorithms are implemented for comparative analysis. The following are the few algorithms related to task scheduling.

2.1 Heterogeneous Earliest Finish Time (HEFT) Algorithm

HEFT is an algorithm which is used to schedule dependent tasks on a heterogeneous network communication. HEFT algorithm execution is divided into two phases. First one is tasks prioritization and second is a selection of processes. In the first phase, ranks of all tasks are calculated. In the next phase, tasks are assigned to the processor on the basis of the earliest possible time. The task can become ready to execute only when the previous tasks in the graph complete their execution. HEFT algorithm [8] is appropriate for processors which have a bounded number. The time complexity of the HEFT algorithm is $O(n^2)$.

2.2 Critical Path on a Processor Algorithm (CPOP)

CPOP algorithm works same as HEFT algorithm, but it is an extended version which uses diverse strategy. This algorithm has two phases: First is prioritizing different tasks and second is a selection of tasks. The first phase is managed by priority queue through keys value, downward rank and upward rank. Binary heaps are used to perform that queues. In the next phase, if the path that is selected is a critical path, this will be scheduled on the critical path processor (CCP), and if it is not a critical path, then the tasks will be assigned to the processors which have minimum earliest start time (EFT). Critical path on a processor algorithm [9] is applied to the heterogeneous computing system. The time complexity of CPOP algorithm is $O(n^2)$.

2.3 Granularity-Based Workflow Scheduling Algorithm (GSS)

GSS algorithm is given by Kumar [10] for workflow applications in cloud computing environment. This algorithm has three phases, namely B-level calculation, score adjustment and task ranking and scheduling. The objective of GSS algorithm is to minimize the overall makespan and also maximize the utilization of virtual machines.

2.4 Normalization-Based Task Scheduling Algorithms for Heterogeneous Multi-cloud Environment (CZSN, CDSN, CDN and CNRSN)

Panda [11] have presented normalization-based task scheduling algorithms for heterogeneous multi-cloud environment. These algorithms consist of two-phase scheduling. The CZSN and CDSN algorithms are based on traditional normalization techniques, while the CDN and CNRSN algorithms are based on distributed scaling and nearest radix scaling normalization, respectively.

2.5 Heterogeneous Dynamic List Task Scheduling Heuristics (HDLTS) Algorithm

HDLTS algorithm is given by Qasim et al. [12] for scheduling of tasks in heterogeneous computing environment. This algorithm has three main phases: first, selection of entry task which reduces the execution time; second, mapping of tasks; third, selection of the task which takes the maximum execution time and assigning the task to the resource which takes minimum execution time to execute the task.

2.6 Synthesized Heuristic Task Scheduling (HCPPEFT) Algorithm

This algorithm works on the basis of duplications base technique as well as lists base technique in a heterogeneous system. HCPPEFT algorithm [13] consists of two steps. The first step is prioritizing the tasks followed by selecting the resources. In this step, the newest technique is applied in which there are three levels. The second step is optimizing the replication of different tasks performed.

2.7 Sorted Nodes in Leveled DAG Division (SNLDD) Algorithm

SNLDD algorithm is used for task scheduling in a heterogeneous distributed computing systems [14]. This algorithm is also called a high-performance scheduling of tasks algorithm. SNLDD algorithm has better performance for a less number of processors. In SNLDD algorithm, DAG is distributed and used to divide into the levels in view of dependent prioritized condition between different tasks, in which every task level is arranged in a descending order with respect to computation sizes.

2.8 Allocation-Aware Min–Min Max–Min Batch (AMinMaxB) Algorithm

Panda et al. [15] have proposed allocation-aware Min–Min Max–Min batch (AMinMaxB) algorithm for heterogeneous multi-cloud systems, which consists of three phases, namely matching, allocating and scheduling to fit them in multi-cloud environment. The allocating phase is used to fill the gaps between matching and scheduling phases.

2.9 Constrained Earliest Finish Time (CEFT) Algorithm

Jiang [16] has proposed improved CEFT algorithm for heterogeneous system scheduling by the use of concepts of constraining critical path. Constraining critical path is established for the relevant DAG, while the task scheduling is performed on the basis of EFT algorithm technique. This algorithm technique is used for scheduling tasks of limited time span.

3 Problem Definition

In HeDCSs, the scheduling of static tasks is represented by DAG for a parallel application. It contains n tasks of set T and number of edges e, and both are defined as a tuple (T, E). In a parallel application, every task $t_i \in T$, while the communication path between t_i and t_j is represented by edge (t_i, t_j). If the $(t_i, t_j) \in E$, then $t_i \in T$ must finish its execution before the start of $t_j \in T$ execution. In an edge (t_i, t_j), the task t_j is a child task and t_i is a parent task. If task t_i has no parent, then it is called an initial task and if t_j has no child, then task is called a final task of DAG. On every edge (t_i, t_j), a value cc_{ij} linked with it shows the communication cost from tasks t_i to t_j.

The HeDCSs contain m processors of set P where each processor has various computation capabilities. The matrix C contains the computation cost $m \times n$ where m is the number of processors and n is a set of tasks. Every element of matrix $w_{ij} \in C$ shows the computation cost of task t_i on processor pj, and here it assumes each processor is completely linked. For the execution of concurrent tasks on different processors, the autonomous communication unit is used that helps in communication among processors. If two tasks are scheduled on different processors, the total data transmitted on edge (t_i, t_j) are equal to the communication cost cc_{ij} from task t_i to task t_j. If task t_i and task t_j are executed on the same processor, then its communication cost cc_{ij} will be taken as zero. Execution of child task can only start on the processor when parent task is finished its execution, and all data are available to that processor. Each task must be assigned to the processor in a schedule that total makespan (runtime) should be minimum for parallel execution of tasks. Figure 2 shows an example of a DAG for the dependent task with a computation time matrix.

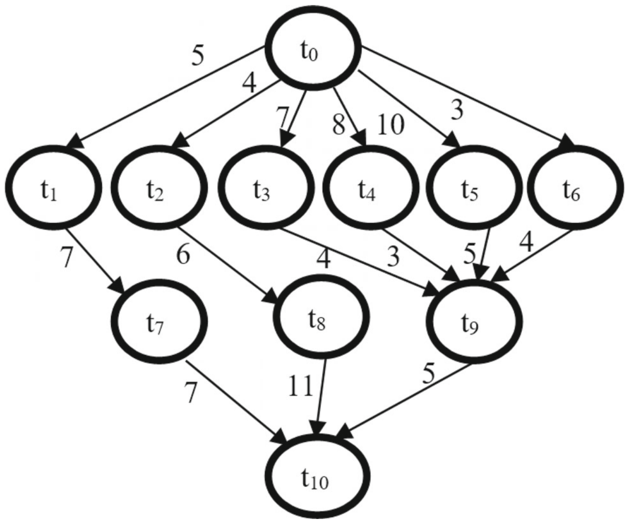

Fig. 3 DAG example for dependent tasks

Figure 3 shows the example of DAG consisting of 11 tasks with t_0 as an initial task and t_{10} as a final task. Tasks t_1, t_2, t_3, t_4, t_5 and t_6 are dependent on t_0, all these tasks can only be executed once t_0 finished its execution, and all data are available. An edge (t_0, t_1) has a communication cost of 5, while an edge (t_1, t_7) has a communication cost of 7. Task t_{10} is dependent on tasks t_7, t_8 and t_9.

4 The Proposed Algorithm Parental Prioritization Earliest Finish Time

PPEFT algorithm is proposed to schedule dependent tasks in a heterogeneous environment. PPEFT algorithm has two phases, prioritization of task phase and processor assigning phase for execution of tasks. Tasks prioritization phase ranks all tasks calculated on the basis of parental prioritization. Processor selection phase tasks are assigning to the processor on the basis of earliest possible finish time.

4.1 Tasks Prioritization Phase

Tasks prioritization phase (TTP) is an important phase of task scheduling. This phase decides the priority of each task on the basis of ranks calculated by parental prioritization. A tasks list is generated by sorting the ranks of all tasks. Ranks are calculated top-down of a DAG starting from the initial task. Then, it calculates the rank of tasks which is dependent on the initial task. The rank values of each task are calculated by combining communication costs (CCs) and mean computation time. The parental prioritization give the advantage to schedule the task which is on next layer before the tasks of the current layer in DAG, if it has less rank value as well as not dependent on any other tasks of which rank value is not calculated. Rank of node n_i can be calculated using mean computation time (w_i) of all processors of the current task, maximum communication cost of parent task and its rank value. All the calculated ranks are sorted in a descending order and line up in a queue to assign to the processor. To calculate the rank of task n_i, it is defined as

$$\text{rank}\,(n_i) = w_i + \max_{n_j \in \text{par}(n_i)} \left(\text{cc}_{ij} + \text{rank}\left(n_j\right)\right) \tag{1}$$

where n_i is the current node, w_i is the mean computation time of task n_i, par(n_i) is parent tasks of n_i and cc_{ij} is a communication cost of edge (t_i, t_j). As ranks are calculated by traversing from top to down on the tasks graph, it is started from the initial task. Rank of initial task n_{ini} is equal to

$$\text{rank}\,(n_{\text{ini}}) = w_{\text{ini}} \tag{2}$$

where n_{ini} is a first node and w_{ini} is an average computation time of the initial task.

4.2 Processor Assigning Phase

In this phase, the processor assigns to each task on the basis of task scheduled in tasks prioritization phase. Most algorithms for task scheduling consider the earliest possible available time, at the time processor finished its execution. But in PPEFT algorithm, it considers the insertion policy in which tasks assign to the processor on the basis of earliest start time (EST) of available processors and earliest finish time (EFT). EST is the earliest possible time to start the execution of the task. The EST of node n_i can be calculated by

$$\text{EST}\left(n_i, p_j\right) = \max\left[\text{avail}\left(p_j\right), \left(\text{AFT} + \text{cc}_{ij}\right)\right] \tag{3}$$

where n_i is the current node, $p_j$is the total computation time on the processor, avail(p_j) is the availability of processor p_j and cc_{ij} is the communication cost. For initial task, EST for each processor considers as zero because computation time starts from zero on each processor. EFT can be calculated by

$$\text{EFT}\,(n_i) = w_i + \text{EST}\left(n_i, p_j\right) \tag{4}$$

where w_i is the computation time of each processor. EFT is the earliest finish time of tasks on each processor, considering EST and computation of each processor for task n_i. The processor which has least EFT value considers as idle time slot of task n_i; it will be scheduled on that processor. Figure 4 shows the flowchart of our proposed algorithm. It starts with the input of computation time and communication cost of tasks. Two different types of input are taken from real-world applications workflows (Montage and Epigenomics) and RTGG. At the end of flowchart, our proposed algorithm outputs the scheduled tasks.

5 Experimental Results and Discussion

In this section, we present the performance comparison of our proposed algorithm PPEFT with well-known task scheduling algorithms such as HEFT and CPOP algorithm. To test the performance of task scheduling algorithms, simulation environment is run on CloudSim 3.0.3 and WorkflowSim 1.0 with graphs generated through python library. Different sets of dependent tasks graphs are created with random and manually application DAGs. We also performed experiments using standard scientific workflows. Each task scheduling algorithm is run on tasks DAG to generate resulted schedules of given tasks. Finally, a set of performance metrics is applied to each algorithm for performance comparison.

5.1 Performance Comparison Metrics

The performance of scheduling algorithms is compared on the basis of the following metrics:

- Schedule length ratio (SLR): As a different set of tasks graphs (DAGs) are used, it is important to calculate the makespan (scheduling length) of the output of above-mentioned algorithms. It is compulsory to normalize the makespan to lower bound called SLR.

$$\text{SLR} = \frac{\text{makespan}}{\sum n_i \in \text{CP}_{\min} \min p_j \in Q \left\{w_{i,j}\right\}} \tag{5}$$

where makespan is the scheduling length and $\text{CP}_{\min}$ is the minimum computation cost of critical path tasks in SLR. Since SLR is lower bound, it cannot be less than one for every scheduling algorithm. The best algorithm on performance is the one which has the lowest SLR on the graph.

- Speedup: The speedup value for a DAG is the ratio of sequential execution time (SET) to the parallel execution time (PET). The SET is calculated by assigning each task to one processor, while PET is calculated by assigning tasks to more than one processor.

$$\text{Speedup} = \frac{\min p_j \in Q \left\{\sum n_i \in V W_{i,j}\right\}}{\text{makespan}} \tag{6}$$

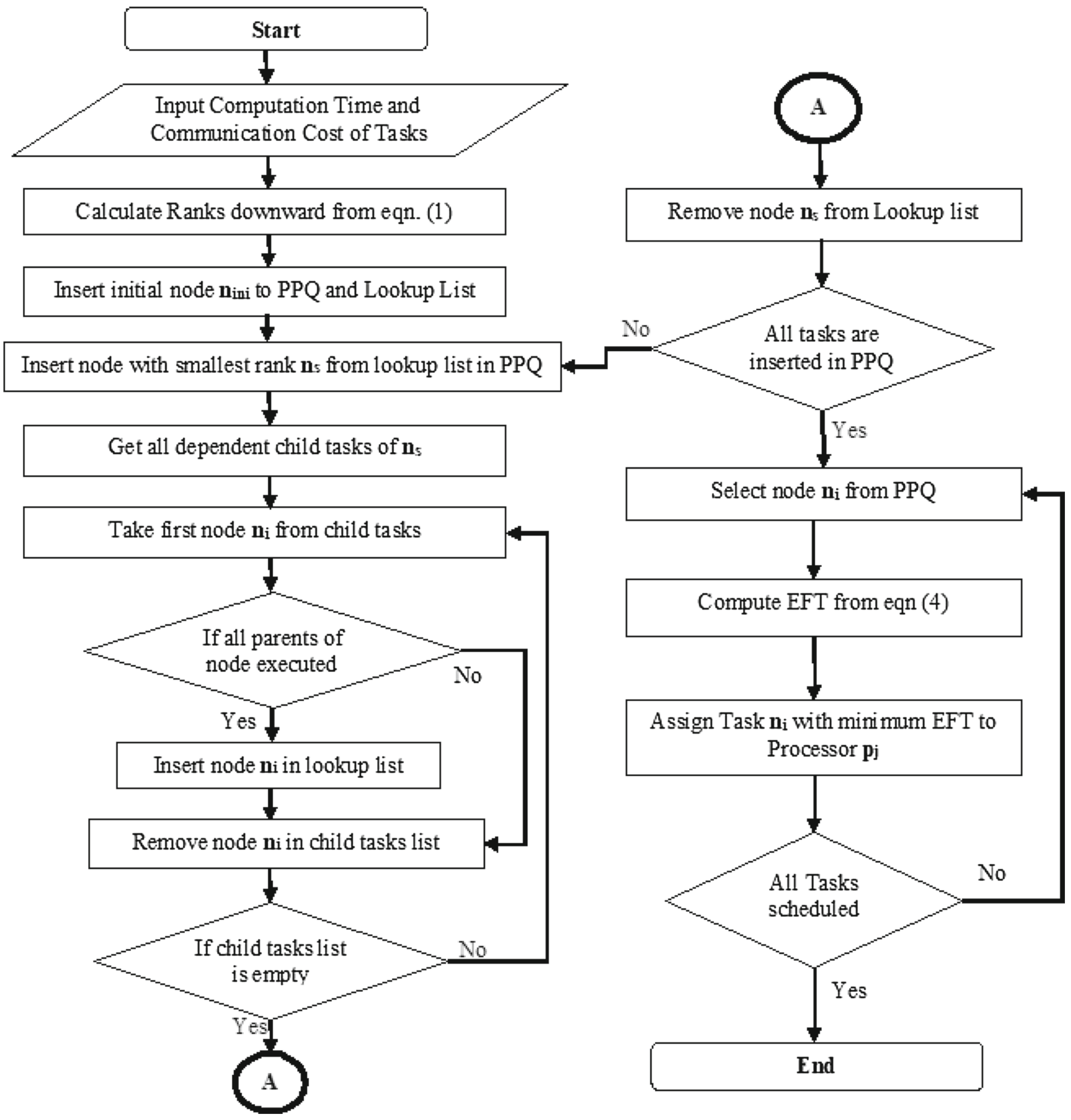
Start
Input Computation Time and Communication Cost of Tasks
Calculate Ranks downward from eqn. (1)
Insert initial node n_{ini} to PPQ and Lookup List
Insert node with smallest rank n_s from lookup list in PPQ
Get all dependent child tasks of n_s
Take first node n_i from child tasks
If all parents of node executed
No
Yes
Insert node n_i in lookup list
Remove node n_i in child tasks list
If child tasks list is empty
No
Yes
A
A
Remove node n_s from Lookup list
All tasks are inserted in PPQ
No
Yes
Select node n_i from PPQ
Compute EFT from eqn (4)
Assign Task n_i with minimum EFT to Processor p_j
All Tasks scheduled
No
Yes
End

Fig. 4 PPEFT algorithm (flowchart)

- Efficiency: It is obtained by the ratio of speedup value to total resources used for task scheduling graph, which is defined as

$$\text{Efficiency} = \frac{\text{speedup}}{\text{total resources}} \tag{7}$$

5.2 Randomly Generated Task Graphs and Performance Comparison

To assess the performance of our proposed algorithm PPEFT, we considered task graphs which are randomly generated. RTGG was implemented to get weighted tasks graphs with different properties which rely on various input parameters. Our simulation structure permits us to give different values to the parameters used in RTGG. The simulation framework works in a manner. It first executes RTGG program to generate random tasks DAGs, then it executes the different task scheduling algorithms to get the result in scheduling outputs, and finally, it computes the performance metrics of the output schedules.

5.3 Real-World Scientific Workflow Performance Comparison

In addition to the RTGG, we evaluate the performance of our proposed algorithm and we also perform simulations with two real-world scientific workflows such as Montage [2] and

Table 1 Details of workflows used in experiments

Workflows	Total number of tasks in different workflow sizes			
	Small	Medium	Large	XL
Montage	25	50	100	1000
Epigenomics	24	46	100	997

Table 2 Schedule produced by the PPEFT algorithm in each iteration

Step	Ready queue	Task selected	EFT			Processor assigned
			p_1	p_2	p_3	
1	t_0	t_0	12	23	19	p_1
2	$t_1 t_2,t_3,t_4,t_5,t_6$	t_1	25	34	27	p_1
3	t_2,t_3,t_4,t_5,t_6,t_7	t_2	42	45	41	p_3
4	t_3,t_4,t_5,t_6,t_7,t_8	t_5	69	89	73	p_1
5	t_3,t_4,t_6,t_7,t_8	t_6	75	91	81	p_1
6	t_3,t_4,t_7,t_8	t_8	102	97	103	p_2
7	t_3,t_4,t_7	t_4	104	114	109	p_1
8	t_3,t_7	t_3	124	117	121	p_2
9	t_7,t_9	t_9	126	125	129	p_2
10	t_7	t_7	145	172	152	p_1
11	t_{10}	t_{10}	189	213	194	p_1

Epigenomics [3]. For each workflow, we have used four different sizes in our experiments (small, medium, large and extra large) given in Table 1. Small-size workflow of Montage has 25 tasks and Epigenomics has 24 tasks, medium-size workflow of Montage has 50 tasks and Epigenomics has 46 tasks, large-size workflow of Montage and Epigenomics has 100 tasks, and extra-large-size workflow of Montage has 1000 and Epigenomics has 997 tasks.

The performance of our proposed algorithm PPEFT is compared with HEFT and CPOP algorithms on different graphs generated from RTGG on the basis of SLR, speedup and efficiency. First, we compare the performance of scheduling algorithms by considering a number of tasks to a fixed value of 10 ($t0, t1, t2 \ldots t9$) and a fixed number of processors to 3 ($p0$, $p1$and$p2$). Each of the results is calculated from the average of 100 random task graphs generated from RTGG. The weight of edges is given randomly from 5 to 25, while the weight of tasks varies from 10 to 30. Table 2 shows the schedule produced by the PPEFT algorithm in each iteration. This table consists of ready queue, task selected (tasks which are ready to be assigned to the processor), EFT of tasks on each processor and the possessor assigning to the task.

Figures 5 and 6 show that results after simulation of our proposed algorithm PPEFT outperform other algorithms on the basis of SLR and speedup. By comparing the average SLR value of each scheduling algorithm, the PPEFT algorithm gives improved results than the HEFT algorithm by 13.00% and the CPOP algorithm by 15.00%. Figures 7 and 8 show that average efficiency of PPEFT algorithm is far better than of HEFT algorithm and CPOP algorithm. With increasing the number of processors, PPEFT algorithm gives more improved results.

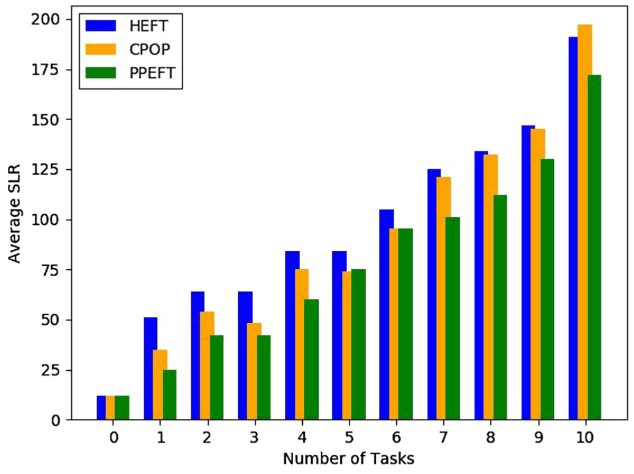

Fig. 5 Average SLR with ten tasks

Figure 9 shows the comparison of the overall schedule length of PPEFT algorithm with HEFT and CPOP algorithms. This comparison is made by taking three processors and ten tasks. Random DAG was generated from RTGG. Our proposed PPEFT algorithm gives improved results as compared to other algorithms.

Figure 10 shows the average SLR of PPEFT, HEFT and CPOP algorithms by increasing the number of tasks to 100. We can see that with the increase in a number of tasks, PPEFT algorithm gives improved results. PPEFT algorithm clearly outperforms HEFT and CPOP algorithms.

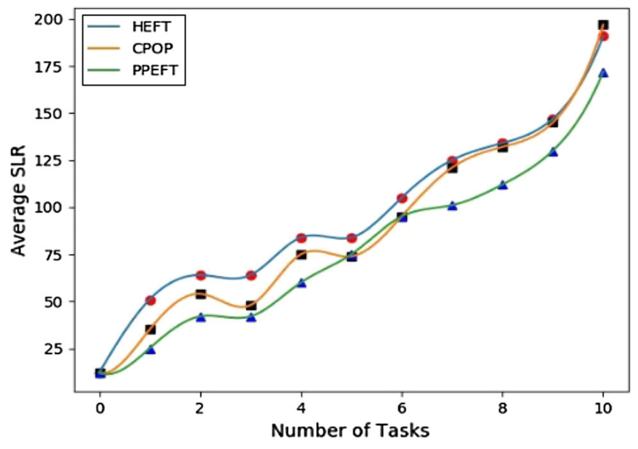

Fig. 6 Average SLR of ten tasks

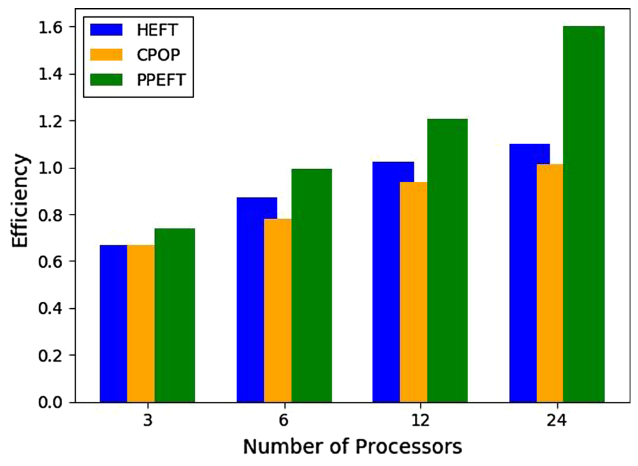

Fig. 7 Efficiency of PPEFT, HEFT and CPOP algorithms on 24 processors

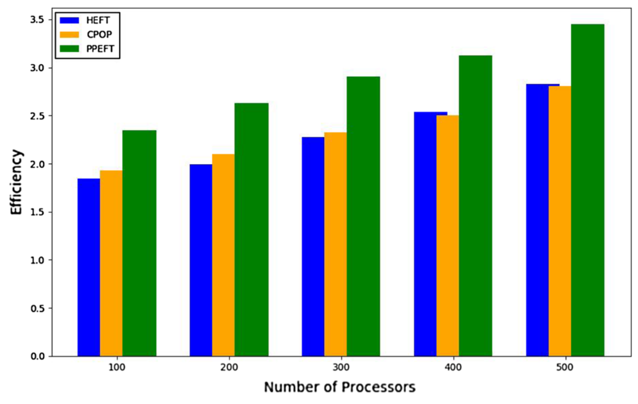

Fig. 8 Efficiency of PPEFT, HEFT and CPOP algorithms on 500 processors

Figures 11 and 12 show the increase in the number of tasks to 500. The average SLR of PPEFT, HEFT and CPOP algorithms is calculated. The average schedule length ratio of CPOP is highest, followed by HEFT algorithm and our proposed algorithm having the smallest SLR value. So increasing the number of tasks and processor effects in performance improvement in PPEFT algorithm.

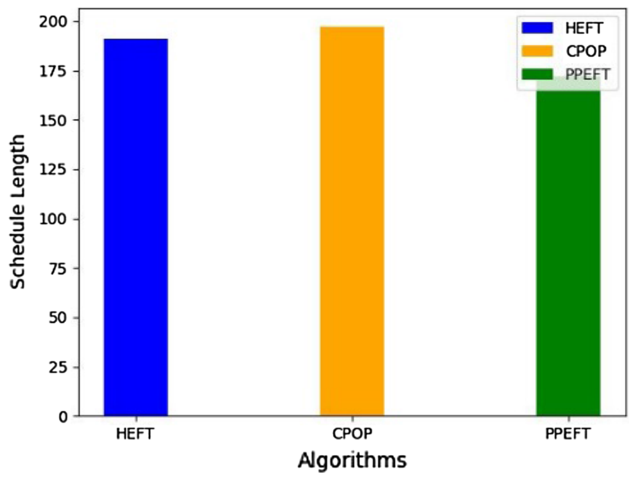

Fig. 9 Execution time of PPEFT, HEFT and CPOP algorithms

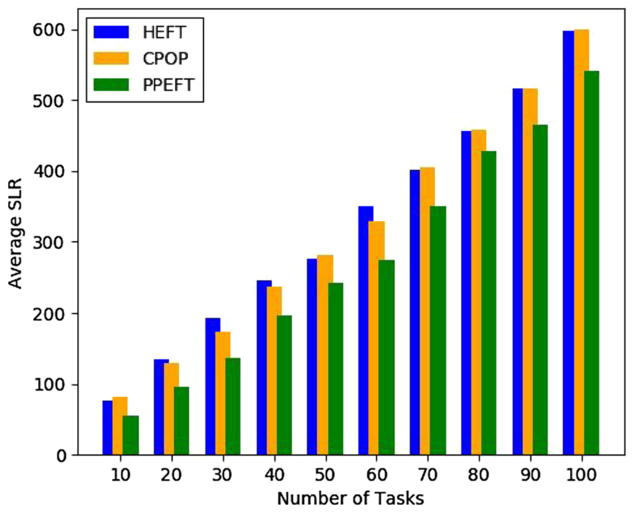

Fig. 10 Average SLR of 100 tasks

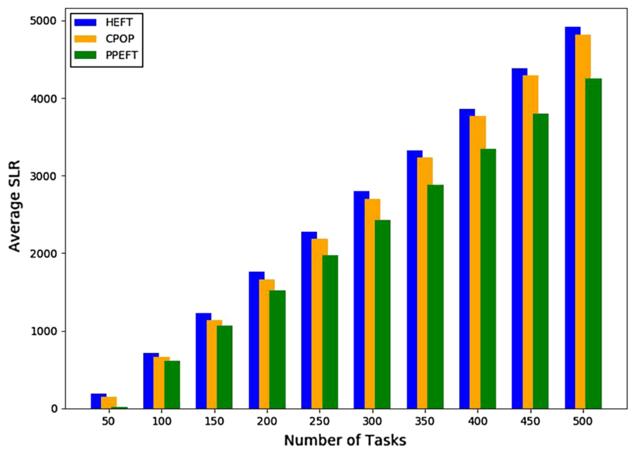

Fig. 11 Average SLR of 500 tasks

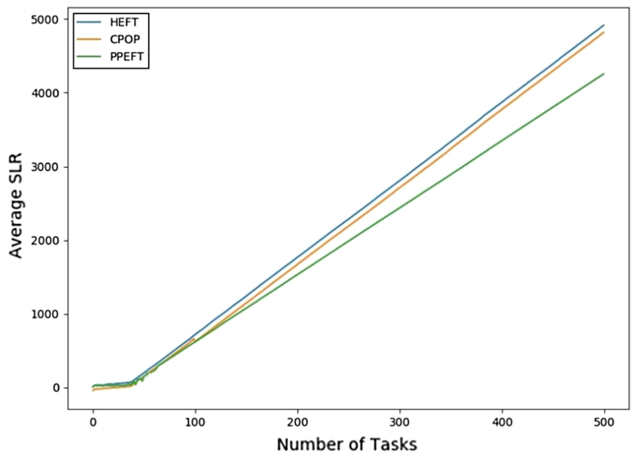

Fig. 12 Average SLR of 500 tasks

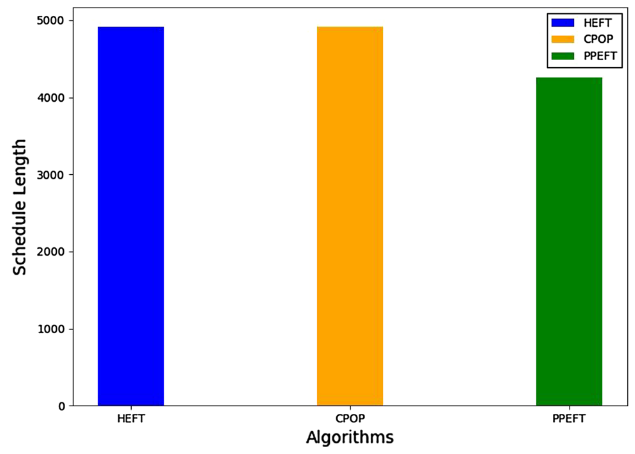

Fig. 13 Schedule length of HEFT, CPOP and PPEFT on 500 tasks

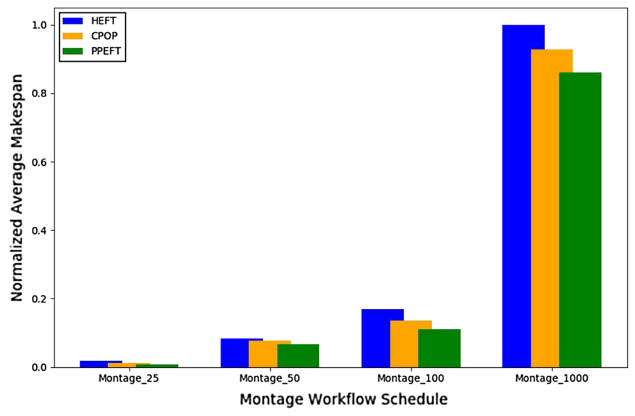

Fig. 14 Average makespan of Montage workflow

Figure 13 shows the comparison of schedule length of HEFT, CPOP and PPEFT, respectively. This experimental results show that schedule length of PPEFT algorithm is less as compared to other algorithms. Increasing the total number of tasks and processor results in better task scheduling.

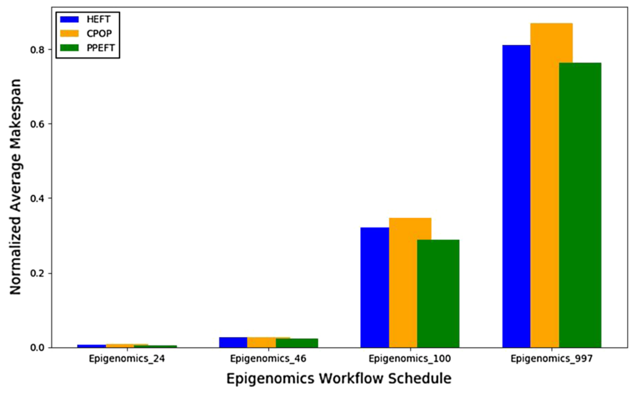

Fig. 15 Average makespan of Epigenomics workflow

Figure 14 shows the average makespan of Montage workflow. In this experiment, four different sizes of Montage workflow are used. Montage_25 is the smallest size containing 25 tasks, Montage_50 is the medium size containing 50 tasks, Montage_100 is the large size containing 100 tasks, and Montage_1000 is the extra large size containing 1000 tasks. Experimental results show that PPEFT algorithm works better on Montage workflow as compared to HEFT and CPOP algorithms. Figure 15 shows the Epigenomics workflow used for experimentation. Four different sizes are used (small, medium, large and extra large) containing 24, 46, 100 and 997 tasks, respectively. Experimental results show that PPEFT algorithm outperforms HEFT and CPOP algorithm on Epigenomics workflow.

6 Conclusion and Future Work

In this paper, we have proposed the scheduling of a task algorithm for heterogeneous computing systems called PPEFT. This new algorithm has two phases, tasks prioritization phase and processor assigning phase. In tasks prioritization phase, tasks are scheduled on the basis of downward ranks and parental priority. PPEFT uses parental prioritization queue to schedule tasks. PPQ uses to schedule dependent tasks of next row of task graph (DAG) before the current row, which have less communication cost. In the processor assigning phase, the processor is assigned to tasks list obtained from PPQ, which give the minimum computation time. Therefore, PPEFT gives more efficient results for scheduling of tasks than other task scheduling algorithms. The performance of our new proposed algorithm PPEFT is compared with well-known algorithms HEFT and CPOP. The comparison of the algorithm is made on the basis of the RTGG and various set of tasks graphs (DAGs). The PPEFT algorithm gives improved results in terms of average SLR, efficiency and speedup. Our future research plan is to extend the PPEFT algorithm for the scheduling of tasks that will respond to the changes in virtual machines and network load.

References

1. Convolbo, M.W.; Chou, J.: Cost-aware DAG scheduling algorithms for minimizing execution cost on cloud resources. J. Supercomput. **72**(3), 985–1012 (2016)
2. Montage: An astronomical image engine. http://montage.ipac.caltech.edu. Accessed 19 Aug 2018
3. Epigenomics: USC Epigenome Center. http://epigenome.usc.edu. Accessed 19 Aug 2018
4. Xie, Y.; Wu, J.; Liu, F.: A hierarchic hybrid scheduling algorithm for static task with precedence constraints. In: 2016 IEEE Trustcom/BigDataSE/ISPA, 2079-2085 (2016)
5. Zhang, L.; Li, K.; Xu, Y.; Mei, J.; Zhang, F.; Li, K.: Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster. Inf. Sci. **319**, 113–131 (2015)
6. Zhang, Q.; Zhani, M.F.; Boutaba, R.; Hellerstein, J.L.: Dynamic heterogeneity-aware resource provisioning in the cloud. IEEE Trans. Cloud Comput. **2**(1), 14–28 (2014)
7. Wu, W.; Bouteiller, A.; Bosilca, G.; Faverge, M.; Dongarra, J.: Hierarchical DAG scheduling for hybrid distributed systems. In: 2015 IEEE International Conference on Parallel and Distributed Processing Symposium, pp. 156–165 (2015)
8. Keshanchi, B.; Souri, A.; Navimipour, N.J.: An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. J. Syst. Softw. **124**, 1–21 (2017)
9. Dai, Y.; Zhang, X.: A synthesized heuristic task scheduling algorithm. Sci. World J. **2014**, 1–9 (2014)
10. Kumar, M.S.; Gupta, I.; Panda, S.K.; Jana, P.K.: Granularity-based workflow scheduling algorithm for cloud computing. J. Supercomput. **73**(12), 5440–5464 (2017)
11. Panda, S.K.; Jana, P.K.: Normalization-based task scheduling algorithms for heterogeneous multi-cloud environment. Inf. Syst. Front. **20**(2), 373–399 (2018)
12. Qasim, M.; Iqbal, T.; Munir, E. U.; Tziritas, N.; Khan, S. U.; Yang, L. T.: Dynamic mapping of application workflows in heterogeneous computing environments. In: IEEE international parallel and distributed processing symposium workshops, pp. 462–471 (2017)
13. Nasr, A.A.; Bahnasawy, N.A.E.; Sayed, A.E.: A new duplication task scheduling algorithm in heterogeneous distributed computing systems. Bull. Electr. Eng. Inform. **5**(3), 373–382 (2016)
14. Eswari, R.; Nickolas, S.; Arock, M.: A path priority-based task scheduling algorithm for heterogeneous distributed systems. Int. J. Commun. Netw. Distrib. Syst. **12**(2), 183–201 (2014)
15. Panda, S.K.; Gupta, I.; Jana, P.K.: Task scheduling algorithms for multi-cloud systems: allocation-aware approach. Inf. Syst. Front. **19**, 1–19 (2017)
16. Jiang, Y.; Shao, Z.; Guo, Y.: A DAG scheduling scheme on heterogeneous computing systems using tuple-based chemical reaction optimization. Sci. World J. **2014**, 1–23 (2014)