IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 1, 2019, accepted October 12, 2019, date of publication October 21, 2019, date of current version November 14, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2948704

A Task Scheduling Algorithm With Improved Makespan Based on Prediction of Tasks Computation Time algorithm for Cloud Computing

BELAL ALI AL-MAYTAMI[1,2], PINGZHI FAN[1], ABIR HUSSAIN[3], THAR BAKER[3], AND PANOS LIATSIS[4]

[1]Institute of Mobile communication, Southwest Jiaotong University, Chengdu 611756, China
[2]Faculty of Science, Ibb University, Ibb 740005, Yemen
[3]Department of Computer Science, Liverpool John Moores University, Liverpool L3 3AF, U.K.
[4]Department of Electrical Engineering and Computer Science, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates

Corresponding author: Abir Hussain (a.hussain@ljmu.ac.uk)

This work was supported by the 111 project under Grant 111-2-14.

ABSTRACT Cloud computing is extensively used in a variety of applications and domains, however task and resource scheduling remains an area that requires improvement. Put simply, in a heterogeneous computing system, task scheduling algorithms, which allow the transfer of incoming tasks to machines, are needed to satisfy high performance data mapping requirements. The appropriate mapping between resources and tasks reduces makespan and maximises resource utilisation. In this contribution, we present a novel scheduling algorithm using Directed Acyclic Graph (DAG) based on the Prediction of Tasks Computation Time algorithm (PTCT) to estimate the preeminent scheduling algorithm for prominent cloud data. In addition, the proposed algorithm provides a significant improvement with respect to the makespan and reduces the computation and complexity via employing Principle Components Analysis (PCA) and reducing the Expected Time to Compute (ETC) matrix. Simulation results confirm the superior performance of the algorithm for heterogeneous systems in terms of efficiency, speedup and schedule length ratio, when compared to the state-of-the-art Min-Min, Max-Min, QoS-Guide and MiM-MaM scheduling algorithms.

INDEX TERMS Scheduling algorithm, task scheduling, resource utilization, cloud computing.

I. INTRODUCTION

A. MOTIVATION AND AIM

Cloud computing has grown to be a major technological enabler in companies and organizations [1]–[3]. It has been shown to increase reliability, deliver cost-cutting solutions, and provide 24/7/365 access to hard/soft resources from anywhere based on pay/use pricing policy [4], [5]. The cloud offers services in the structure of Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS) [3]. Task scheduling is a major challenge in widely distributed heterogeneous systems (e.g., cloud computing), which chooses the preeminent resources for a provided task. Also, in heterogeneous systems, task scheduling is more convoluted in comparison to homogeneous computing (HC) systems because of the various communication and execution rates amid various processors.

The main aim of cloud computing is to provide a highly efficient platform for appropriate exploitation of computational properties embedded in organizations, and to support the enterprise to capitalize on end-user demands [9]. However, the decentralized and heterogeneous nature of cloud networks makes them intricate to deal with. Last but not least, deciding on suitable assets for tasks has become an acute issue due to the swift rise of users and resources.

For heterogeneous clustering systems, task scheduling is a computationally demanding problem, even under abridged conventions, as it is NP-hard [9], [12].

The overarching aim of this research is to improve the performance of task scheduling, while reducing computational costs. A key objective is to predict the ideal algorithm for incoming/available data as and when needed. In order

The associate editor coordinating the review of this manuscript and approving it for publication was Yaser Jararweh.

to achieve this, we perform a systematic analysis of heuristic techniques for resource utilization by means of Principal Components Analysis (PCA) in the cloud environment. Moreover, we analyze the requirements and consequences of utilizing Quality of Service (QoS) with the proposed Prediction of Tasks Computation Time algorithm (PTCT).

B. CONTRIBUTIONS

As described in Section 3, there are many works in the literature in regards to task scheduling in the cloud computing environment. This research proposes the following novel contributions:

1. Typically, task scheduling algorithms focus on improving either computation or communication costs in the cloud data center. The proposed PTCT method decreases both of these costs noticeably, as discussed and explained in Sections 3 and 5.
2. PTCT uses PCA to reduce the size of the required matrices (refer to sections 4 and 5). To the best of the authors' knowledge, this is the first attempt of using PCA in task scheduling in the context of cloud computing.
3. PTCT examines the incoming data (in tasks) to allocate the appropriate/capable processor from the data center. In this way, PTCT guarantees to achieve near-optimal re-/allocation of resources, hence providing superior scheduling performance, compared to benchmark algorithms, as illustrated in Section 5.

C. STRUCTURE

The rest of this research paper is structured as follows. Section 2 discusses heuristic scheduling algorithms and summarizes their essential features. Section 3 provides an overview of related work in task scheduling. Section 4 introduces the definition of the scheduling problem. Section 5 describes the proposed PTCT method and presents strategies for practical algorithms based on PTCT. Section 6 presents simulation results and provides comparisons to the state-of-the-art methods. Section 7 provides the conclusions of this research and identifies opportunities for further work.

II. HEURISTIC SCHEDULING ALGORITHMS

Scheduling is a decision-making process carried out in real time, where processors are allocated to an extensive set of tasks. Due to resource constraints, task scheduling is a challenging problem. Various studies investigated task scheduling algorithms and proposed schemes to improve resource utilization in widely distributed environments, i.e., cloud computing. Solving the associated NP-hard problem leads to the optimal solution.

Various heuristics-based schemes have been shown to provide semi-optimal solutions. Heuristic scheduling algorithms are rule-based, and extensively used in the IaaS cloud computing environment. They are intended to sort out challenging problems faster than meta-heuristic algorithms, where implantation is time consuming. Furthermore, heuristic algorithms are used in finding an optimal solution, when meta-heuristic method fail to do so. The associated solutions are achieved with improved speed, accuracy, optimum transaction and completeness [13]. This section focuses on popular heuristic algorithms for semi-optimal scheduling.

A. MIN-MIN HEURISTIC ALGORITHM

In the Min-Min scheduling algorithm, tasks with shorter execution time are determined and prioritized, and then, resources that generate the minimum accomplished time are assigned to them. This process is executed repeatedly for all scheduled tasks. Hence, the Min–Min scheduling algorithm picks the smaller tasks to be completed first [19], [20].

B. MAX-MIN HEURISTIC ALGORITHM

In contrast to Min-Min, the Max-Min algorithm operates on the concept of completing the largest task first. The time duration of each task is provided in advance and all tasks are mapped to the appropriate processor. This process is repeated until all unmapped tasks are processed. It should be noted that for small-scale distributed systems, Min-Min and Max-Min are suitably utilized [17], [21]. For all short scale tasks transported in the network, Max-Min simultaneously schedules the longer tasks, followed by the shorter ones, while Min-Min performs the opposite, i.e., schedule the shorter ones first, followed by the longer tasks, which implies a larger makespan.

C. QUALITY OF SERVICE (QoS) GUIDED MIN-MIN TASK SCHEDULING HEURISTIC ALGORITHM

The standard Min-Min heuristic scheme ignores Quality of Service (QoS) in its implantation. As a result, the effectiveness of the algorithm in the cloud is questionable. QoS Guided Min-Min allows the QoS constraint to be utilized with standard Min-Min heuristic scheduling [19]. Specifically, some scheduled tasks, particularly those with sensitive data, may necessitate the use of high bandwidth, while others can be accomplished with low bandwidth. This scheduling algorithm allocates tasks with excessive QoS requests first using the Min-Min heuristic. When all tasks entail either high QoS or low QoS, the scheme requires $O(n^2m)$ computational complexity.

D. MIM-MAM ALGORITHM

- This scheduling algorithm builds upon the advantages and limitations of the Max-Min and Min-Min heuristic algorithms. In this case, information about the upcoming deadline for each task, arrival rate, cost of execution of using each available resource, and communication costs is considered [23]. Two types of policies are used to classify task scheduling problems, specifically static and dynamic scheduling. In the former, information about tasks, including communication costs for each task, execution and the relationship with other tasks, is determined in advance. In the latter, decisions are determined in runtime, since the details of the tasks

are not obtainable. In addition, dynamic scheduling signifies runtime scheduling, while static scheduling is a representation of compile-time scheduling. There are two major groups of static scheduling algorithms: (i) Heuristic-based algorithms, and (ii) Guided Random Search based algorithms. The former provides good approximate solutions with polynomial time complexity [14]. Dynamic scheduling is faster in execution than static scheduling, since it's basically not aware of any thread dependencies [7].

1) RELATED WORK

As discussed in Section 2, there exists a variety of heuristic scheduling algorithms, which can operate in both batch and online modes. Some of these schemes are appropriate in heterogeneous scheduling scenarios, however they cannot always attain good makespan, speedup, reduced costs and increased efficiency [1], [6]–[8], [13]. Hence, QoS-based techniques are essential in obtaining the maximum objectives so as to retain QoS characteristics for both tasks and resources.

Wang and Yu [29] propose an enhanced Min-Min algorithm to consider the proficiency of task scheduling in cloud computing. As previously indicated, the Min-Min algorithm first determines the tasks with shorter execution times and then the resources which result in the shortest times. This can lead to delays when examining the use of the algorithm in the cloud environment. Zhang et al. [30] propose QoS constraints in the cloud environment as a criteria for scheduling a task in the Min-Min algorithm, named Mul-QoS-Min-Min. The proposed algorithm finds resources with similar tasks to deliver task scheduling, then requests users to carry out their needs. The simulation results indicate that the performance of the Mul-QoS-Min-Min scheme is improved in terms of execution times, when benchmarked against the traditional Min-Min algorithm.

Both Mao et al. [31] endorse the Max-Min algorithm in order to stabilize the load for the cloud. The algorithm conserves a table that holds details about task position and evaluates the real-time workload for virtual machines (VMs) with the estimated task execution times. The Max-Min algorithm boosts the utilization of resources and decreases task scheduling response time by using VMs instead of traditional resources. Li et al. [32] schedules tasks using improved max-min task scheduling then largest task is too large compared to other tasks in Meta-task in this case overall makespan is increased because too large task is executed by slowest resource.

Henning et al. [34] study task scheduling in the parallel method challenge with a fixed number of processors and the best schedule for high performance outcomes. They indicate that this can be achieved by mapping tasks to machines according to precedence constraints. In [35], the authors propose a task scheduling mechanism for allocating computing processors to a so-called ''task graph templates''. Since the authors do not consider the network connection as a criterion, this is deemed one of the limitations of their study. To overcome this limitation, Sinnen and Sousa [36] use network contention in their task scheduling method, without considering the fee charged to customers for using these resources.

Two factors must be considered in the cloud computing environment, i.e., high performance of data transfer and satisfaction of budget constraints. The authors in [37] and [38] introduce a cost-efficient algorithm to select the most appropriate system in a cloud environment to implement the workflow based on using the deadline and cost saving constraints. Li et al. [39] illustrate a scheduling algorithm, which can be applied in large graph processing, where both cost and schedule length constraints are considered. However, their scheme does not consider failed devices.

Issues of task scheduling have been widely considered in the literature. As expected, a multitude of approaches have been proposed due to its crucial effects on performance [9], [15]. The heuristic algorithm based on list scheduling strategies [9] is one of the conventional scheduling algorithms for cloud environments. This provides low time complexity, however the limitations of minimal universality and poor convergence have.

In [42], the authors study load balancing in the cloud environment to avoid problems, which may occur due to increase in power consumption, node failure, and machine failure. However, the research dealt with a limited number of parameters, e.g., there is no analysis on the effects of dynamic scheduling, increase in the number of tasks and machines, as well the growth of users. In [43], additional parameters are considered. Optimization of task scheduling is addressed by introducing the iterative selection operator. However, this study overlooks the issue of load balancing. Shimada et al., [44] proposed a novel algorithm, which can transfer the task with the shorter path while eliminating redundant tasks. However, the issue of the increase in the number of machines as the number of tasks increases remains an open challenge. In [45], the authors propose a model to increase the overall system utilization, however, load balancing and other performance parameters need to be further improved. Other works, such as [52], [53] explore the cooperation and collaboration among cloud servers using multi-agent approaches to best assign resources to incoming tasks.

III. PROBLEM FORMULATION

In this research, we tackle the problem of static scheduling of a single application in a widely distributed and heterogeneous environment. Let us consider the sets of tasks and processors, T and P, respectively. Let P processors be available for the set of tasks T, which are not shared during task execution. Let ETC is the Expected Time to compute, the matrix which contains in each row the estimated execution time of a given task on each resource, and the estimated execution time of a resource in each column. The aim is to reduce the makespan time of task execution in the data center. To minimize the makespan and increase the utilization of resources, the tasks

of parallel applications have to be efficiently scheduled on the available resources using the ETC matrix model [28].

A. TASK GRAPH MODE

Using Directed Acyclic Graph (DAG), it is not possible to start at one point in the graph and traverse the entire graph, while evaluating computation costs in the correct order. Let us consider a DAG, where G = (V, E). In this case, V represents the set of v nodes. Each node $v_i \in V$ refers to an application task, which is comprised of instructions executed on the same processor. The parameter E represents a set of e connection boundaries between the tasks. Boundary e(i, j) $\in$ E characterizes the task-dependency constraint, where task v_i needs to be completed before task v_j is able to start. The DAG is supplemented by the matrix ETC using $v \times p$ tabulated cost. The parameters v and p are the number of tasks and the number of processors in the system, respectively. Element $w_{i,j}$ of W provides the estimated time of completing task v_i on processor p_j. The mean execution time of task v_i is given by:

$$\overline{w_i} = \sum_{j \in P} w_{i,j}/p \tag{1}$$

Figure 1 represents a simple model commonly used in the scheduling problem [14]–[17], which allows us to provide a competitive likeness with state-of-the-art solutions, since these simplifications correspond to real systems. Table 1 shows the computation costs of the 11 tasks shown in Figure 1, with randomly generated values to provide a sophisticated example on extracting the values of ETC.

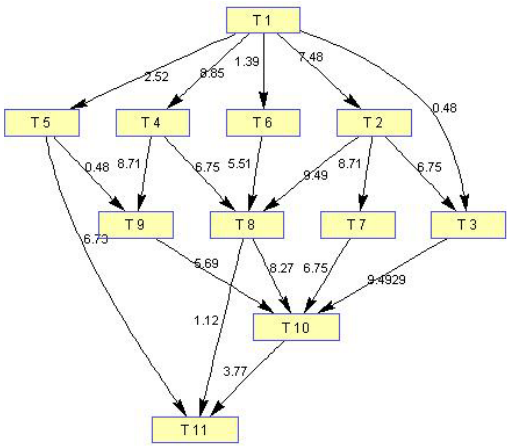


FIGURE 1. A simple task graph with 11 tasks.

IV. PROPOSED METHOD

This section introduces the general framework of the proposed PTCT algorithm, including algorithmic details.

A. OVERVIEW

In heterogeneous computing, effective task scheduling is of the utmost importance to increase the advantages of accomplishing an application. Consequently, the task scheduling problem has been widely studied and many algorithms have been proposed including list scheduling, clustering, and task duplication scheduling based on Genetic Algorithm. In summary, list-scheduling algorithms are ideal in delivering low cost solutions, in comparison to other approaches. Clustering algorithms perform better in the case of homogeneous processors. Finally, task duplication scheduling algorithms are utilized for communication intensive programs. A point to note is that a review of the open literature on task scheduling revealed a number of enhancements for homogeneous processors [8], [10], [16]–[18], however there appears to be less progress in the case of heterogeneous processors [1]–[22]. This provides further motivation for the development of our proposed framework in the context of a heterogeneous environment.

Consider the following two attributes, Earliest Start Time (EST) and Earliest Finish Time (EFT), used to outline the objectives of the task scheduling issue. EST(v_i, p_j) represents the EST for task v_i on processor $_{pj}$, and similarly, EFT(v_i, $_{pj}$) represents EFT for task v_i on processor p_j. EST(v_i) and EFT(v_i) represent the values of these attributes over the set of processors, respectively. For any initial entry task, v_{entry}, EST(v_{entry}) = 0, the values of EST and EFT are calculated from the entry to the exit tasks, traveling the task graph from top to bottom. All immediate predecessor tasks of v_i should be scheduled to allow the calculation of EST. The

TABLE 1. Computation costs of tasks in Fig 1.

Edge(E)	Node(V)
(1,2)	7.48
(1,3)	0.48
(2,3)	6.75
(1,4)	8.85
(1,5)	2.52
(1,6)	1.39
(2,7)	8.71
(2,8)	9.49
(4,8)	6.75
(6,8)	5.51
(4,9)	8.71
(5,9)	0.48
(3,10)	9.493
(7,10)	6.75
(8,10)	8.27
(9,10)	5.69
(10,10)	4.58
(5,11)	6.73
(8,11)	1.12
(10,11)	3.77

task scheduling problem is defined as follows:

$$EST\left(v_i, p_j\right) = \max\left\{p_{avail[v_i,p_j]}, \max\left(EFT\left(v_p, v_i\right) + C\left(v_p, v_i\right)\right)\right\} \quad (2)$$

$$where\ v_p \epsilon pred\left(v_i\right), EFT\left(v_p, p_k\right) = EFT\left(v_p\right)$$

$$C\left(v_p, v_i\right) = 0\ when\ k = j \quad (3)$$

$$EFT\left(v_i, p_j\right) = T\left(v_i, p_j\right) + EST\left(v_i, p_j\right) \quad (4)$$

$P_{avail[vi,pj]}$ is defined as the earliest time for processor pj to execute task v_i, pred(vi) while EST(v_i, p_j) represents the maximum time when processor p_j will be available. k is a counter, meantime this represents the time where last message arrives from any of task v_i predecessors.

$$makespan = EFT(v_{Exit}, p_j), \quad (5)$$

where v_{Exit} is the exit task and ETC(i,j) and PCA are defined as:

$$ETC(i,j) = \begin{bmatrix} ETC_{1,1} & ETC_{1,2} & ETC_{1,3} & \dots & ETC_{1,j} \\ ETC_{2,1} & ETC_{2,2} & ETC_{2,3} & \dots & ETC_{2,j} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ ETC_{i,1} & ETC_{i,2} & ETC_{i,3} & \dots & ETC_{i,j} \end{bmatrix}$$

$$PCA(ETC) = \begin{bmatrix} \lambda(ETC_1) & \dots & \dots & \dots & \dots \\ \vdots & \lambda(ETC_2) & \vdots & \vdots & \vdots \\ \vdots & \vdots & \lambda(ETC_3) & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \dots & \dots & \dots & \dots & \lambda(ETC_i) \end{bmatrix}$$

Let us consider the notation of Table 2.

TABLE 2. Notation.

G	group of data with high and low QoS
H	Hosts
Ψ	Mean
Φ	Variance
C	Covariance
V	Task
P	Machine
F	Algorithms
k	index of algorithm
$\lambda(ETC_I)$	Value of the row after using PCA

B. SYSTEM MODEL OF THE PROPOSED PTCT

The aim of the proposed PTCT scheme is to minimize the scheduling rate (makespan), as well as estimate the optimal algorithm for scheduling as shown in Figure 2.

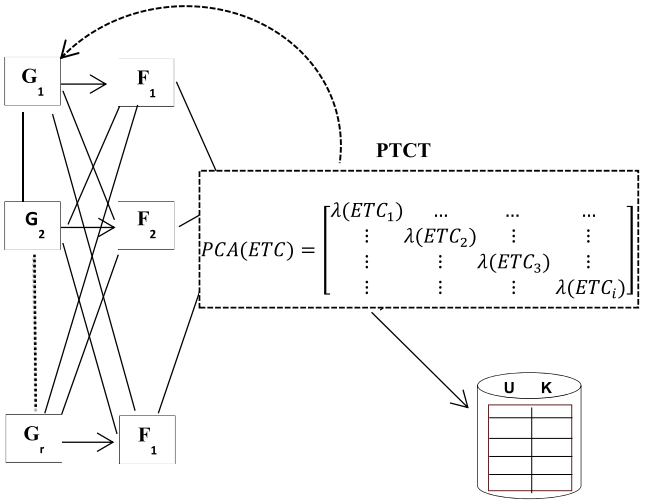


FIGURE 2. The architecture of the proposed PTCT scheme.

The proposed PTCT algorithm is shown in Algorithm 1, while Algorithm 2 illustrates how to calculate the execution time of task v_i on host p_j and Algorithm 3 provides details about calculating the expected execution time of task v_i on host p_j.

V. SIMULATION RESULTS

Simulation experiments were run in MATLAB R2013a on a PC with Intel Core i5 processor, using 2.40 GHz CPU and 8 GB RAM. Windows 7 was utilised as the OS for the platform. The proposed PTCT algorithm was benchmarked with Min-Min [20], Max-Min [21], QoS guided [33] and MiM-MaM [23], developed for heterogeneous system. Simulation experiments were repeated 20 times, using the parameters in Table 3.

TABLE 3. Simulation parameters.

Parameter	Value
Tasks	[10-90]
Machines	8
Algorithms	4
Type of data	High, medium and low QoS

The following assumptions were made:

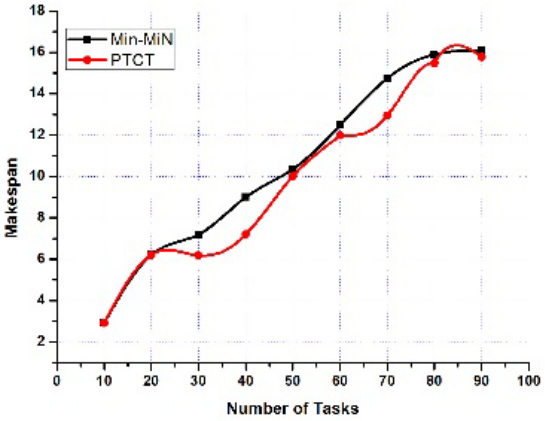
1. Expected Time to Compute (ETC) of size $v \times p$ is used, where v and p represent the number of tasks and resources, respectively.
2. Tasks have no priorities associated with them.
3. Independent tasks are assigned to available resources.
4. The availability of resources and the number of tasks to be executed are known in advance [24].
5. We consider three types of QoS data: high, medium, and low.

A. PERFORMANCE ANALYSIS

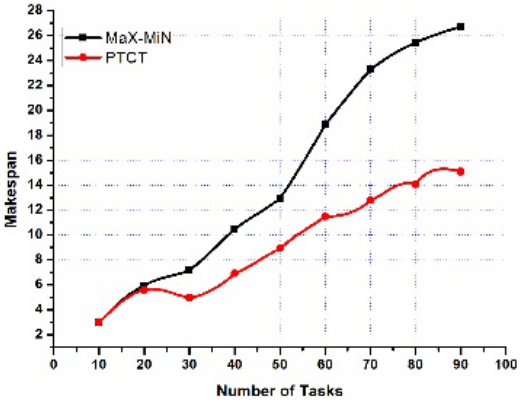
Performance analysis was carried out based on three quality measures, i.e., makespan, speedup and efficiency. These were defined as follows:

TABLE 4. Makespan of PTCT vs four state-of-the-art algorithms as a function of the number of tasks.

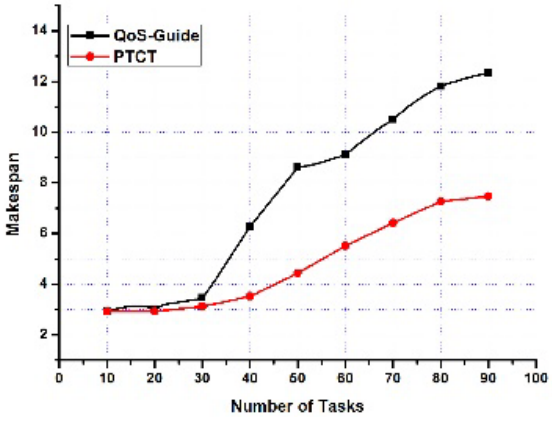
Number of Tasks	Min-Min	PTCT	Max-Min	PTCT	QoS-Guide	PTCT	MiM-MaM	PTCT
10	2.9213	2.9213	3.0069	3.0069	2.9213	2.9213	2.9213	2.421
20	6.2041	6.2041	5.9335	5.5867	3.024	2.9213	4.324	3.029
30	7.1933	6.1852	7.1822	4.9633	3.455	3.1182	5.455	3.513
40	9.0017	7.2059	10.482	6.9191	6.256	3.5182	6.256	3.934
50	10.359	10.018	12.925	8.9163	8.622	4.4341	8.622	4.434
60	12.480	11.983	18.855	11.469	9.12	5.5095	9.12	5.51
70	14.742	12.946	23.256	12.779	10.505	6.4206	10.505	6.421
80	15.9	15.484	25.416	14.073	11.805	7.2664	11.805	7.266
90	16.084	15.781	26.722	15.089	12.335	7.4705	12.335	8.271



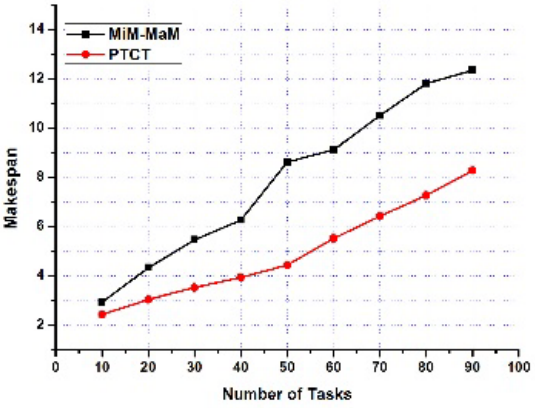
a. Makespan for Min-Min and PTCT algorithms as a function of the number of tasks.



b. Makespan for Max-Min and PTCT algorithms as a function of the number of tasks.



c. Makespan for QoS-guided and PTCT algorithms as a function of the number of tasks.

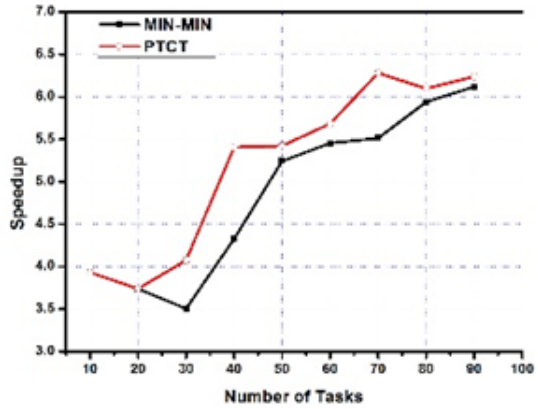


d. Makespan for MiM-MaM and PTCT algorithms as a function of the number of tasks.

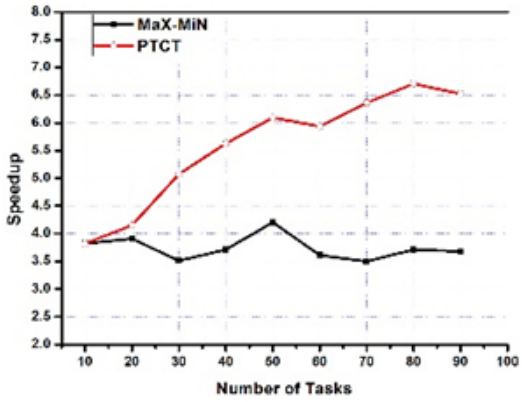
FIGURE 3. Makespan of state-of-the-art algorithms vs the proposed PTCT scheme as a function of the number of tasks.

a) Makespan: this represents the main quality measure as it provides the completion time for all tasks in a graph. Makespan is utilized to find the maximum completion time by estimating the finishing time of the last task [27] and is calculated according to:

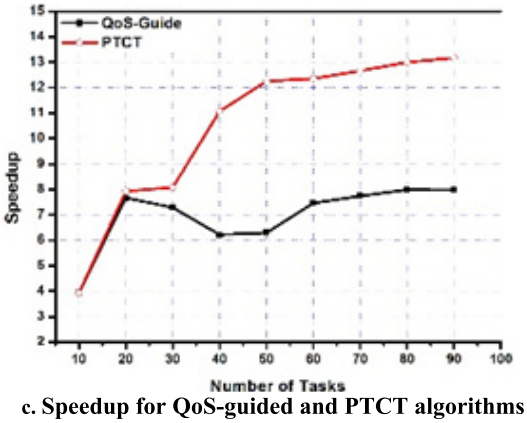
$$Makespan = EFT(v_i, p_j) \quad (6)$$



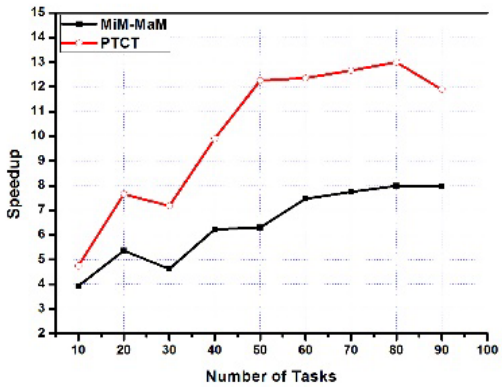
a. Speedup for Min-Min and PTCT algorithms as a function of the number of tasks.



b. Speedup for Max-Min and PTCT algorithms as a function of the number of tasks.



c. Speedup for QoS-guided and PTCT algorithms as a function of the number of tasks.



d. Speedup for MiM-MaM and PTCT algorithms as a function of the number of tasks.

FIGURE 4. Speedup of different algorithm in comparison to our proposed PTCT scheme.

b) *Speedup*: this is defined as the ratio of the sequential schedule length calculated by allocating all tasks to the fastest processor over the execution time of the task schedule (makespan). as shown in Equation (7). The sequential execution time is the cumulative computation cost when assigning all the tasks v_i sequentially to a single computing host p, is the host, H the set of hosts, V is the set of tasks

$$Speedup = \frac{min_{p \in H}(\sum_{v_i \in V} w_{i,j})}{makespan} \tag{7}$$

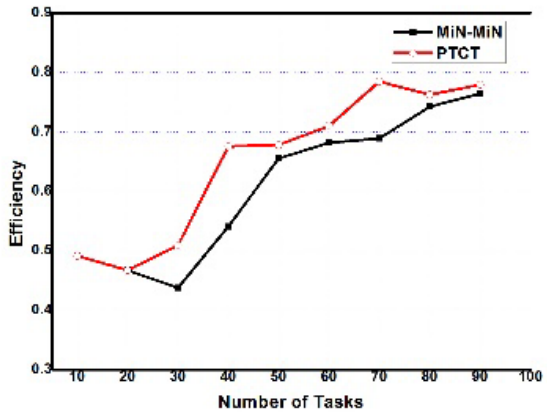
c) *Efficiency*: this is the ratio of speedup to the total number of processors, p, utilized to schedule the entire DAG application:

$$Efficincy = \frac{Speedup}{p} \tag{8}$$

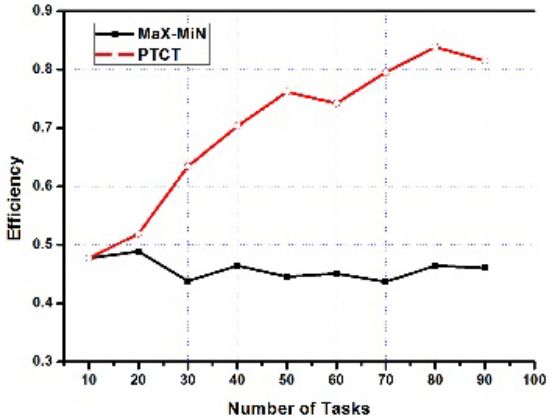
d) *Complexity*: Using the PCA algorithm for all tasks and machines is feasible for low-complexity scheduling algorithms. The information requested from all tasks is utilized to calculate the critical path and the scheduling algorithm is executed in such a way that it stops after a schedule is attained for the task ready to run at its next scheduled time, thus decreasing the time complexity of each schedule. The computational complexity of MiN-MiN, MaX-MiN, QoS-Guide and MiM-MaM is O (n^2m), where n refers to the number of nodes, and m is the number of edges. The computational complexity of PTCT is O(m).because of the matrix of ETC will contains only one dimension instead of two.

Makespan is the maximum finish time of the exit task in the scheduled DAG. From Figure 3, it can be noted that the makespan of PTCT decreases following the application of PCA. Since the PTCT algorithm minimizes the communication overheads, the time required for completing application execution by the PTCT algorithm is lower than the benchmarked algorithms.

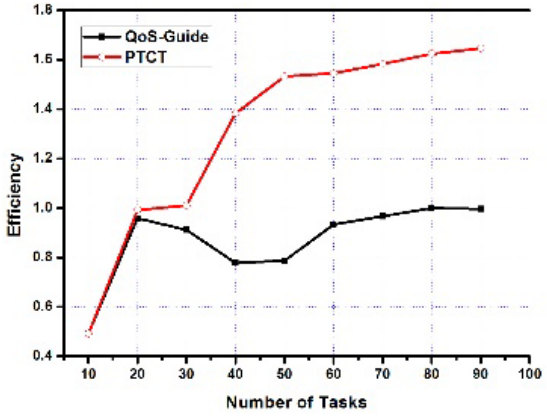
In Equation 7, $w_{i,j}$ represents the weight of task t_i on processor p_j. Speedup is a good quality measure for executing the application program using a parallel system. From Figure 4, it can be seen that the speedup of PTCT algorithm is higher than the other algorithms, since ETC is reduced to one instead of two dimensions. Figure 5 illustrates that the efficiency of the proposed algorithm when benchmarked



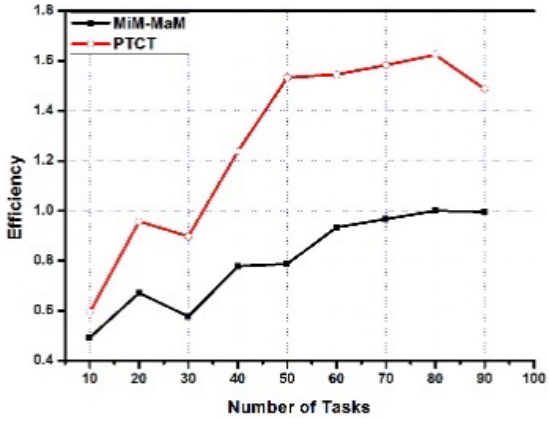
a. Efficiency for Min-Min and PTCT algorithms as a function of the number of tasks.



b. Efficiency for Max-Min and PTCT algorithms as a function of the number of tasks.



c. Efficiency for QoS-guided and PTCT algorithms as a function of the number of tasks.



d. Efficiency for MaX-MaM and PTCT algorithms as a function of the number of tasks.

FIGURE 5. Efficiency of state-of-the-art algorithms vs the proposed PTCT scheme as a function of the number of tasks.

with the state of the art algorithms. It shows improved results, in particular, when the number of tasks is increased to above 40.

B. DISCUSSION

For comparison purposes, the makespan, speedup and efficiency are used to illustrate the performance of makespan.

Figure 3 shows comparative results of four state-of-the-art algorithms with the PTCT scheme. In the experiments, the same data was used to compare the performance of the algorithms. Compare the PCTC algorithm with each algorithm separately. In other words, compare each algorithm before and after using PTCT. The results show an improvement in the performance of each algorithm with PTCT. By repeating the experiment 20 times, the best algorithm will be selected. MiN-MiN shows less effect with using PTCT, and this usually depends on the quality of the used data. But as long as the PTCT algorithm decreases the size of the ETC matrix, that will leads to reduce the time used to complete tasks. We used 8 processors and a range of [10-90] tasks. The results in Table 4 and Figure 3 indicate that the algorithm provided positive results, i.e., reduced times to complete tasks on resources, due to reducing the size of the ETC matrix by using PCA.

Figure 4 also illustrates positive results in terms of speedup, which is one of the essential criteria for measuring the performance of algorithms for the scheduling task. In addition to the efficiency, which refers to the ratio of speedup with the number of processors used, Figure 5 depicts better results for PTCT in comparison with the other algorithms. The simulation results indicate that the efficiency of our proposed technique is significantly improved, when increasing the number of tasks apart from the Min-Min algorithm, which shows similar performance (when the number of tasks is less or equal to 90). However, Min-Min does not take into consideration QoS as is the case with the proposed algorithm. Furthermore, with larger number of tasks (>90), additional simu-

Algorithm 1 Prediction of Tasks Computation Time (PTCT)

Input:
Output:
1: Generate data (*v, p*)
2: Group the data in S groups, where each group can include high and low QoS
3: For each group of data, run six algorithms and output the makespan time for each algorithm
4: $H_i = \{h_1, h_2, \ldots\ldots, h_m\}$, m is number of machines, H (hosts), i=1,..., 4 (for all algorithms)
5: Select the best pair (H_k, F_k), i.e., minimum makespan denoting the best algorithm k.
6: Use PCA
7: $\psi = \frac{1}{m}\sum_{i=1}^{m} h_i$ mean value for every dimension of the matrix ETC.
8: $\phi_i = |h_i - \psi|$
9: Set $A = [\phi_1, \phi_2, \ldots\ldots\ldots\phi_m]$
10: Covariance $C = A\ A^T$
11: Find the eigenvector $W = A\ A^T$
12: Save <U, k> in the database
13: Find $\min||W - U_i||$, i.e., the closest eigenvector in the database and choose its algorithm F_i
14: Predicated algorithm.

Algorithm 2 Calculation of Execution Time of Task v_i on Host p_j

Input:
p = processors;
v = tasks;
Output: $ETC(v_i, p_j)$
While there are more tasks to be scheduled
  For all v_i to schedule
    For all p_j
      Compute $CT_{i,j} = CT(v_i, p_j)$
    End for loop
    Compute $i = F(CT_{i,1}, CT_{i,2}, \ldots\ldots\ldots)$
  End for loop
  Determine the best F match m
  Find minimum $CT_{m,n}$
  task m will be scheduled on n
End while loop.

lation results indicated that the proposed PTCT generated significantly improved results in comparison to MiN-MiN algorithm. This in agreement with existing literature research [40], [41].

One limitation of the proposed algorithm is that minimizing the ETC matrix may lead to reduced accuracy values. Nevertheless, the simulation results demonstrated that this did not affect the quality measures benchmarked against the state-of-the-art techniques.

Algorithm 3 Main Algorithm to Calculate the Execution Time of Task v_i on Host p_j

Input:
p = processors;
v = tasks;
$ETC(v_i, p_j)$
Output: $PTCT_m(ETC(v_i, p_j))$, m= number of algorithms
Group the data (*v, p*)in S groups randomly, where each group may include high and low QoS
While there is a group of QoS
  For all G_n $(ETC(v_i, p_j))$ (generate the matrix ETC
    For all algorithms = m
      Compute makespan for each algorithm
      $F_m = \{f_1, f_2, .., f_m\}$;
      makespan PCA($F_m = \{f_1, f_2, \ldots f_m\}$);
      Select best pair (minimum makespan) Eq.5
      (F_k, G_k), which denotes the best algorithm k
    End for
  End for
End while.

VI. CONCLUSION

In this work, a novel algorithm, Prediction of Tasks Computation Time, was presented. This results in a performance improvement in cloud-based task scheduling by using Principal Component Analysis. This permits the reduction of the size of the Expected Time to Compute (ETC) matrix.

The proposed algorithm was applied to simulated task graphs, and its performance was assessed in terms of speed-up, makespan, schedule length ratio and efficiency. The simulation results showed improved performance, when benchmarked with four state-of-the-art scheduling algorithms, namely Min-Min, Max-Min, QoS-guided and MiM-MaM. In the cloud computing context, the simulation results indicated that the proposed PTCT can reduce the overall makespan and task execution time.

The simulation setup was based on static scheduling, where task arrival at the processors and speed are assumed to be known. Future work will consider dynamic scheduling for real-world application graphs and benchmarking in real-world problems. The focus will be on improving the total energy utilization and consumption of task scheduling using the PTCT algorithm and comparing the findings with relevant state-of-the-art algorithms for cloud energy consumption, such as GreeDi and GreeAODV [47]–[51].

REFERENCES

[1] A. I. Avetisyan, R. Campbell, I. Gupta, M. T. Heath, S. Y. Ko, G. R. Ganger, M. A. Kozuch, D. O'Hallaron, M. Kunze, T. T. Kwan, K. Lai, M. Lyons, D. S. Milojicic, H. Y. Lee, Y. C. Soh, N. K. Ming, J.-Y. Luke, and H. Namgoong, "Open cirrus: A global cloud computing testbed," *Computer*, vol. 43, no. 4, pp. 35–43, Apr. 2010.

[2] S. K. Panda and P. K. Jana, "Efficient task scheduling algorithms for heterogeneous multi-cloud environment," *J. Supercomput.*, vol. 71, no. 4, pp. 1505–1533, Apr. 2015.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.* vol. 25, no. 6, pp. 599–616, Jun. 2009.

[4] K. A. Beaty, V. K. Naik, and C.-S. Perng, "Economics of cloud computing for enterprise IT," *IBM J. Res. Develop.* vol. 55, no. 6, pp. 1–12, Nov./Dec. 2011.

[5] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proc. Grid Comput. Environ. Workshop*, Nov. 2008, pp. 1–10.

[6] A. Masood, "HETS: Heterogeneous edge and task scheduling algorithm for heterogeneous computing systems," in *Proc. IEEE 17th Int. Conf. High-Perform. Comput. Commun.*, Aug. 2015, pp. 1865–1870.

[7] R. Hoffmann, A. Prell, and T. Rauber, "Dynamic task scheduling and load balancing on cell processors," in *Proc. 18th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Feb. 2010, pp. 205–212.

[8] E. U. Munir, J. Li, and S. Shi, "QoS sufferage heuristic for independent task scheduling in grid," *Inf. Technol. J.*, vol. 6, no. 8, pp. 1166–1170, Aug. 2007.

[9] R. K. Bawa and G. Sharma, "Modified min-min heuristic for job scheduling based on QoS in grid environment," in *Proc. 2nd Int. Conf. Inf. Manage. Knowl. Economy (IMKE)*, Dec. 2013, pp. 166–171.

[10] J. Napper and P. Bientinesi, "Can cloud computing reach the top500?" in *Proc. Combined Workshops Unconventional High Perform. Comput. Workshop Plus Memory Access Workshop*, Ischia, Italy, May 2009, pp. 17–20.

[11] W. E. Dong, W. Nan, and L. Xu, "QoS-oriented monitoring model of cloud computing resources availability," in *Proc. Int. Conf. Comput. Inf. Sci.*, Jun. 2013, pp. 1537–1540.

[12] C. Zhang, R. Huang, and J. Zhang, "Distributed adaptive consensus tracking of unknown heterogenous linear systems via output feedback," in *Proc. 35th Chin. Control Conf.*, Chengdu, China, Jul. 2016, pp. 8008–8013.

[13] Z. Beheshti and S. M. Shamsuddin, "A review of population-based meta-heuristic algorithms," *Int. J. Adv. Soft Comput. Appl.*, vol. 5, no. 5, pp. 1–5, 2013.

[14] C. Feng, H. Xu, and B. Li, "An alternating direction method approach to cloud traffic management," Jul. 2014, *arXiv:1407.8309*. [Online]. Available: https://arxiv.org/abs/1407.8309

[15] S. Begum and P. C S R, "Stochastic based load balancing mechanism for non-iterative optimization of traffic in cloud," in *Proc. Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET)*, Mar. 2016, pp. 1249–1254.

[16] A. V. Smirnov, K. A. Borisenko, A. V. Shorov, and E. S. Novikova, "Network traffic processing module for infrastructure attacks detection in cloud computing platforms," in *Proc. 16th IEEE Int. Conf. Soft Comput. Meas. (SCM)*, May 2016, pp. 199–202.

[17] L. Kang and X. Ting, "Application of adaptive load balancing algorithm based on minimum traffic in cloud computing architecture," in *Proc. Int. Conf. Logistics, Inform. Service Sci. (LISS)*, Jul. 2015, pp. 1–5.

[18] R. Sahu and A. K. Chaturvedi, "Many-objective comparison of twelve grid scheduling heuristics," *Int. J. Comput. Appl.*, vol. 13, no. 6, pp. 9–17, Jan. 2011.

[19] T. Amudha and T. T. Dhivyaprabha, "QoS priority based scheduling algorithm and proposed framework for task scheduling in a grid environment," in *Proc. Int. Conf. Recent Trends Inf. Technol. (ICRTIT)*, Tamil Nadu, India, Jun. 2011, pp. 650–655.

[20] G. Patel, R. Mehta, and U. Bhoi, "Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing," *Procedia Comput. Sci.*, vol. 57, pp. 545–553, Jan. 2015.

[21] J. K. Konjaang, J. Y. Maipan-uku, and K. K. Kubuga, "An efficient max-min resource allocator and task scheduling algorithm in cloud computing environment," Nov. 2016, *arXiv:1611.08864*. [Online]. Available: https://arxiv.org/abs/1611.08864

[22] X. He, X. Sun, and G. Von Laszewski, "QoS guided min-min heuristic for grid task scheduling," *J. Comput. Sci. Technol.*, vol. 18, no. 4, pp. 442–451, Jul. 2003.

[23] S. V. Kfatheen and M. N. Banu, "MiM-MaM: A new task scheduling algorithm for grid environment," in *Proc. Int. Conf. Adv. Comput. Eng. Appl.*, Mar. 2015, pp. 695–699.

[24] S. V. Kfatheen and A. Marimuthu, "ETS: An efficient task scheduling algorithm for grid computing," *Adv. Comput. Sci. Technol.* vol. 10, no. 10, pp. 2911–2925, 2017.

[25] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Resource scheduling for infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities," *J. Netw. Comput. Appl.*, vol. 68, pp. 173–200, Jun. 2016.

[26] S. M. Abdulhamid, M. S. A. Latiff, G. Abdul-Salaam, and S. H. H. Madni, "Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm," *PLoS ONE*, vol. 7, Jul. 2016, Art. no. e0158102.

[27] S. H. H. Madni, M. S. A. Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," *PLoS ONE*, vol. 12, no. 5, May 2017.

[28] B. A. Al-Maytami, P. Fan, and A. Hussain, "I-MMST: A new task scheduling algorithm in cloud computing," in *Proc. Int. Conf. Intell. Comput.* Cham, Switzerland: Springer, 2018.

[29] G. Wang and H. C. Yu, "Task scheduling algorithm based on improved min-min algorithm in cloud computing environment," *Appl. Mech. Mater.*, vol. 303, pp. 2429–2432, Feb. 2013.

[30] Y. Zhang and B. Xu, "Task scheduling algorithm based-on QoS constrains in cloud computing," *Int. J. Grid Distrib. Comput.*, vol. 8, no. 6, pp. 269–280, 2015.

[31] Y. Mao, X. Chen, and X. Li, "Max–min task scheduling algorithm for load balance in cloud computing," in *Proc. Int. Conf. Comput. Sci. Inf. Technol.* New Delhi, India: Springer, 2014, pp. 457–465.

[32] X. Li, Y. Mao, X. Xiao, and Y. Zhuang, "An improved max-min task-scheduling algorithm for elastic cloud," in *Proc. Int. Symp. Comput., Consum. Control*, Jun. 2014, pp. 340–343.

[33] H. Han, Q. Deyui, W. Zheng, and F. Bin, "A Qos guided task scheduling model in cloud computing environment," in *Proc. 4th Int. Conf. Emerg. Intell. Data Web Technol.*, Sep. 2013, pp. 72–76.

[34] S. Henning, K. Jansen, M. Rau, and L. Schmarje, "Complexity and inapproximability results for parallel task scheduling and strip packing," in *Proc. Int. Comput. Sci. Symp. Russia*. Cham, Switzerland: Springer, 2018, pp. 169–180

[35] J. Wolf, N. Bansal, K. Hildrum, S. Parekh, D. Rajan, R. Wagle, K.-L. Wu, and L. Fleischer, "SODA: An optimizing scheduler for large-scale stream-based distributed computer systems," in *Proc. 9th ACM/IFIP/USENIX Int. Conf. Middleware*, Leuven, Belgium, Dec. 2008, pp. 306–325.

[36] O. Sinnen and L. A. Sousa, "Communication contention in task scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 6, pp. 503–515, Jun. 2005.

[37] V. Ruben den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, Miami, FL, USA, Jul. 2010, pp. 228–235.

[38] V. Ruben den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds," in *Proc. 3rd IEEE Int. Conf. Cloud Computing Technol. Sci.*, Athens, Greece, Nov./Dec. 2011, pp. 320–327.

[39] J. Li, S. Su, X. Cheng, Q. Huang, and Z. Zhang, "Cost-conscious scheduling for large graph processing in the cloud," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun.*, Banff, AB, Canada, Sep. 2011, pp. 808–813.

[40] S. Anousha, S. Anousha, and M. Ahmadi, "A new heuristic algorithm for improving total completion time in grid computing," in *Multimedia and Ubiquitous Engineering*. Berlin, Germany: Springer, 2014, pp. 17–26.

[41] J. Y. Maipan-uku, A. Muhammed, A. Abdullah, and M. Hussin, "Max-average: An extended max-min scheduling algorithm for grid computing environtment," *J. Telecommun., Electron. Comput. Eng.*, vol. 8, no. 6, pp. 43–47, Sep. 2016.

[42] R. Pratap and T. Zaidi, "Comparative study of task scheduling algorithms through Cloudsim," in *Proc. 7th Int. Conf. Rel., Infocom Technol. Optim. (ICRITO)*, Aug. 2018, pp. 397–400

[43] D. Wu, "Cloud computing task scheduling policy based on improved particle swarm optimization," in *Proc. Int. Conf. Virtual Reality Intell. Syst. (ICVRIS)*, Aug. 2018, pp. 99–101.
[44] K. Shimada, I. Taniguchi, and H. Tomiyama, "Work-in-progress: Communication-aware scheduling of data-parallel tasks," in *Proc. Int. Conf. Compil., Archit. Synth. Embedded Syst. (CASES)*, Sep./Oct. 2018, pp. 1–2.
[45] S. Yang and Q. Deyu, "Study on static task scheduling based on heterogeneous multi-core processor," in *Proc. Int. Conf. Comput. Netw., Electron. Automat. (ICCNEA)*, Sep. 2017, pp. 180–182.
[46] K. Baital and A. Chakrabarti, "Dynamic scheduling of real-time tasks in heterogeneous multicore systems," *IEEE Embedded Syst. Lett.*, vol. 11.1, pp. 29–32, Mar. 2018.
[47] T. Baker, B. Al-Dawsari, H. Tawfik, D. Reid, and Y. Ngoko, "GreeDi: An energy efficient routing algorithm for big data on cloud," *Ad Hoc Netw.*, vol. 35, pp. 83–96, Dec. 2015.
[48] T. Baker, M. Asim, H. Tawfik, B. Aldawsari, and R. Buyya, "An energy-aware service composition algorithm for multiple cloud-based IoT applications," *J. Netw. Comput. Appl.*, vol. 89, pp. 96–108, Jul. 2017.
[49] T. Baker, J. M. García-Campos, D. G. Reina, S. Toral, H. Tawfik, D. Al-Jumeily, and A. Hussain, "GreeAODV: An energy efficient routing protocol for vehicular ad hoc networks," in *Proc. Int. Conf. Intell. Comput.* Cham, Switzerland: Springer, Aug. 2018, pp. 670–681.
[50] I. A. Ridhawi, M. Aloqaily, Y. Kotb, Y. Jararweh, and T. Baker, "A profitable and energy- efficient cooperative fog solution for IoT services," *IEEE Trans. Ind. Informat.*, to be published.
[51] S. Oueida, Y. Kotb, M. Aloqaily, Y. Jararweh, and T. Baker, "An edge computing based smart healthcare framework for resource management," *Sensors*, vol. 18, no. 12, p. 4307, Dec. 2018.
[52] M. Al-Khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, "Improving fog computing performance via fog-2-fog collaboration," *Future Gener. Comput. Syst.*, vol. 100, pp. 266–280, Nov. 2019.
[53] Y. Kotb, I. Al Ridhawi, M. Aloqaily, T. Baker, Y. Jararweh, and H. Tawfik, "Cloud-based multi-agent cooperation for IoT devices using workflow-nets," *J. Grid Comput.*, pp. 1–26, 2019.