Journal of Engg. Research Vol. 8 No. (3) September 2020 pp. 71-90

خوارزمية اليعسوب ذاتية الانحدار لجدولة المهام متعددة الأهداف (ADO-MTS) في الحوسبة السحابية المتنقلة

***ماتيش جارج و**راجندر نات**
*معهد بانيبات للهندسة والتكنولوجيا، سمالخا (هاريانا)
**قسم علوم الحاسوب وتطبيقاته، جامعة كوروكشترا، كوروكشترا

الخلاصة

الحوسبة السحابية المتنقلة هي منصة حوسبة معترف بها وتُعتبر نموذجاً تجارياً نظراً لقدرتها على النمو في تقديم الخدمات المطلوبة للمستخدمين. ومع ذلك، فإنها تطرح عدداً من التحديات، من بينها استهلاك الطاقة في مراكز البيانات حيث أنها القضية الرئيسية. ولذلك، تم تصميم تقنية لجدولة المهام المدركة للطاقة، وتسمى جدولة المهام متعددة الأهداف المرتكزة على خوارزمية اليعسوب ذاتية الانحدار، والتي تقوم بجدولة المهام إلى الموارد السحابية المناسبة. تكون جدولة المهام إما في السحابة العامة أو في السحابة المتنقلة (MC) بحيث يتم تقليل استخدام الطاقة. وفقاً لذلك، تم تطوير خوارزمية التحسين، وخوارزمية اليعسوب ذاتية الانحدار، والتي تجمع بين القيمة الذاتية الشرطية المعرضة للخطر (CAViaR) وخوارزمية اليعسوب (DA). علاوة على ذلك، تم تصميم نموذج متعدد الأهداف يتعلق باستهلاك الطاقة، والتصنيع، واستخدام الموارد، لتحقيق التخصيص الأمثل للموارد من أجل المهام. تم استخدام ثلاثة مقاييس، مثل الموارد، التصنيع، والطاقة، لتقييم أداء تقنية ADO-MTS المُقترحة. أظهرت النتائج أن تقنية ADO-MTS قدمت أداءً عالياً من خلال الحصول على الحد الأقصى من استخدام الموارد 0.5795، والحد الأدنى لاستهلاك الطاقة 6.22 و 0.092، على التوالي.

Autoregressive Dragonfly Optimization for Multiobjective Task Scheduling (ADO-MTS) in Mobile Cloud Computing

Matish Garg* and Rajender Nath**

Department of Computer Science and Applications,
Kurukshetra University, Kurukshetra-136119, Haryana, India.
**Corresponding Author: matish1981@gmail.com*

Submitted: 23/10/2018
Revised: 26/11/2019
Accepted: 01/12/2019

ABSTRACT

Mobile Cloud computing is a recognized computing platform and is being considered as a business model due to its potential growth in offering required services to the users. However, it poses a number of challenges, among which the consumption of energy in the data centers is the key issue. Hence, an energy aware task scheduling technique, called Autoregressive Dragonfly Optimization-based Multiobjective Task Scheduling (ADO-MTS), is designed that schedules the tasks to the suitable cloud resources. The scheduling of the tasks is either in the public cloud or Mobile Cloud (MC) such that the utilization of energy is reduced. Accordingly, an optimization algorithm, Autoregressive Dragonfly Optimization (ADO), is developed combining Conditional Autoregressive Value at Risk (CAViaR) with Dragonfly Algorithm (DA). Moreover, a multiobjective model concerning energy consumption, Makespan, and resource utilization is designed for the optimal allocation of resources to the tasks. Three measures, that is, resource utilization, Makespan, and energy, are used to evaluate the performance of the proposed ADO-MTS technique. The results show that the ADO-MTS technique has provided high performance by obtaining the maximum resource utilization of 0.5795, minimum Makespanof 6.22, and minimum energy consumption of 0.092, respectively.

Keywords: Cloud computing; Task scheduling; MC; CAViaR; DA; Makespan.

INTRODUCTION

Mobile Cloud Computing (MCC) is a new business approach that extends cloud computing by providing cloud resources to the mobile users and helps them in saving device end resources (Lin et al. 2014). Mobile cloud applications transmit both the data and data processing away from mobile devices and into the cloud, conserving local storage, battery life, and processing power. Cloud computing provides infrastructure, applications, and platforms as web services via three service delivery models, namely, Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS), and Platform-as-a-Service (PaaS) (Wang et al. 2016 ; Hongyou et al. 2013). In MCC, there involves two kinds of architectures, namely, remote cloud and local mobile cloud. The remote cloud may be a public cloud, private cloud, or hybrid cloud, whereas the local mobile cloud is created by nearby mobile devices in the vicinity.

The compactness and the portability of mobile devices like smart-phones, tablets, and laptops make them a widely used computing platform. Due to the increased usage and energy density of batteries, the durability of the battery for the mobile devices becomes shorter and results in a power crisis. MCC integrates mobile computing and cloud computing in order to increase the capabilities of mobile devices using offloading techniques. MCC intends to improve

the performance and save energy of the mobile devices by (i) selective offloading of processes of an application, such as object recognition, natural language processing, and video/image editing, into the cloud and (ii) careful scheduling of tasks to be executed in the mobile device as well as in the cloud,keeping the task-precedence requirements (Chen, et al. 2017).

The main feature of cloud computing as recognized by the researchers is the provisioning of a shared pool of resources, such as storage, data processing, network, and applications at minimal cost and in Pay-As-You-Go manner. If the jobs are not scheduled correctly, performance is reduced because the cloud processes a large amount of data. Thus, the scheduling mechanism plays an important role in cloud computing. A scheduling algorithm is utilized to plan the task with greatest evaluated gain or benefit and execute the task.To develop an effective scheduling algorithm, it is necessary to understand the various problems associated with different scheduling methodologies and the limitations to overcome.

An important constraint in scheduling the dependent tasks is the energy consumption. Conventional techniques of task scheduling for shared platforms like clusters, grids, and cloudsare concerned with minimizing the time of execution irrespective of the energy consumption. The execution time and the energy consumption may cause larger variations during the runtime, due to physical faults, process variability, and the changes in the voltage or the frequency (Li *et al.,* 2017; Juarez *et al.,* 2016). Therefore, the scheduling process in cloud computing is considered as a challenge, since it requires a useful considerate of characteristics of resources and various energy factors. The scheduling techniques (Ali *et al.,* 2017; Liu *et al.,* 2017) available concentrate more on allocating the resources fairly, but in reality, various resource-based applications operating on the Virtual Machines (VMs) also pose considerable effect on the performance of the system and energy consumption. Hence, it is important to consider these aspects during task scheduling. To schedule the resources optimized and effectively, a scheduling scheme must consider multiple parameters like CPU, bandwidth, memory, etc., of a data center for making the scheduling decisions (Duan *et al.,* 2017; Ding *et al.,* 2015).

Accordingly, this paper designs an Autoregressive Dragonfly Optimization technique for the task scheduling in either a public cloud or MC with energy awareness. The algorithm considers three objectives, energy, Makespan, and resource utilization, which are optimized. The energy estimation in the cloud environment is formulated based on application and dynamic executions, while that in MC is calculated based on the frequency of operation of the cores during task scheduling. Makespan indicates the time of execution of each task in the cloud. The ADO-MTS technique is designed to allocate the tasks in each resource in the cloud, while the ADO algorithm is designed by combining CAViaR approach with DA for the optimal scheduling. The performance of the proposed method is analyzed using three measures, such as resource utilization, Makespan, and energy. From the results, it is exposed that the proposed ADO-MTS technique has provided the high performance by obtaining the maximum resource utilization of 0.5795, minimum Makespanof 6.22, and minimum energy consumption of 0.092, respectively.

The contributions of the proposed ADO-MTS technique for the task scheduling in the cloud are as follows:

- Introduction of ADO algorithm by integrating the concept of CAViaR into DA, for the optimal scheduling of tasks to the particular resources in the cloud such that the energy consumption and the execution time are at minimum, while the resource utilization is maximized.
- Designing a multiobjective model based on three objectives, such as energy, Makespan, and resource utilization, for the effective selection of cloud resources, to assign the tasks.

The rest of the paper is organized as follows: Section 2 presents various existing methods employed for task scheduling in the cloud together with problem formulation. In section 3, the cloud setup and the energy consumption model are presented. The proposed ADO-MTS technique developed using ADO algorithm is described in section 4 with diagrammatic representations. The results of the ADO-MTS technique are demonstrated using a comparative analysis with the performance discussion in section 5, and section 6 concludes the paper.

RELATED WORK

This section presents the review of existing task scheduling works with their contribution, merits, and demerits. Xue Lin *et al.* (2015) investigated the task scheduling problems in MC and proposed an algorithm based on minimal-delay, which reduced the energy consumption by migrating the tasks from the local cores to the cloud. The algorithm utilized Dynamic Voltage and Frequency Scaling (DVFS) technique to reduce the further utilization of energy. For the migration of tasks, a linear-time rescheduling algorithm was developed. However, the algorithm has a large computational complexity of order O (N^3K).

On the basis of Dynamic Voltage Scaling (DVS), Yibin Li *et al.* (2017) had developed a novel Energy-aware Dynamic Task Scheduling (EDTS) algorithm to reduce the utility of total energy for smart-phones such that the constraints, time, and probability were satisfied. The consumption of energy for Cyber Physical System (CPS) could be reduced using EDTS than in critical path scheduling and parallelism-based scheduling methods. The major drawback of EDTS is that it is not applicable for Android devices.

Ting Shi *et al.* (2016) had designed the local mobile clouds using mobile devices in proximity to develop their mathematical models, after which the scheduling problem was formulated. Then, the resource discovery process and a scheduling algorithm, called adaptive probabilistic scheduler, were designed to schedule the tasks to the nearby mobile devices. Even though the approach is scalable and flexible, task rescheduling is impossible.

An Energy-aware Resource Allocation method, called EnReal, was developed by Xiaolong Xu *et al.* (2016), to address the energy constraint issue. The authors had presented an energy consumption model in the cloud environment, designing a resource allocation algorithm with energy awareness. It reduces the consumption of energy in the cloud, but is not appropriate for real world applications, as it fails to consider network bandwidth and storage.

For energy conservation, Xiaomin Zhu *et al.* (2014) had presented a rolling-horizon scheduling technique with a scheduling technique based on energy awareness, known as EARH, for real-time applications in virtualized clouds. The technique utilized an energy consumption model depending on the task. Moreover, two methods regarding scaling up and scaling down of resources were developed to improve the trade-off between schedulability of the task and energy conservation. EARH also suffers from the same limitation most of the existing systems face; i.e., it is not suitable for real-time cloud platforms.

Tarandeep Kaur and Inderveer Chana (2016) analyzed the significance of multicore processors, consolidation, and virtualization methods to attain energy efficiency in the cloud by designing a model, named Green Cloud Scheduling Model (GCSM). GCSM utilized the heterogeneity of tasks and resources through a scheduler unit that could reserve the constrained tasks based on the nodal energy. By the dynamic decisions made, GCSM ensures the performance, reaching the Quality-of-Service (QoS) required. However, GCSM fails to consider the deadline constraints and the energy expenditure issues.

A scheduling scheme, called PreAntPolicy, was designed by Hancong Duan *et al.* (2017) using a prediction model depending on fractal mathematics and Ant Colony Algorithm (ACA)-based scheduler. Scheduling was done based on the load trend prediction model so that the scheduler could allocate the resources with minimized energy consumption satisfying the QoS. The reduced scheduling efficacy due to a large number of VMs is the drawback of the PreAntPolicy.

Seyedmehdi Hosseinimotlagh *et al.* (2015) presented a VM scheduling algorithm based on the utilization level for the optimal energy utility such that the QoS was met. By regulating the scheduled resources of VMs on a host, it could reach the optimal energy level. Only if the CPU utilization of a host is greater than the optimal threshold, the performance of the algorithm has slighter enhancement.

Yonghua Xiong *et al.* (Xiong et al., 2019) analyzed the task scheduling problem for cloud data-centers (CDCs) and established a mathematical model of the scheduling of two-stage tasks. The Johnson's rule was combined with the genetic algorithm to create a Johnson's-rule-based genetic algorithm (JRGA), which takes into account the characteristics of multiprocessor scheduling in CDCs. A comparison of the JRGA with two other scheduling

algorithms shows that the JRGA is better than the conventional LS algorithm in all cases and that it is better than the ILS algorithm when the number of machines is small. However, more efforts are still needed to further this study, and they include considering the scheduling with multiple heterogeneous machines and developing scheduling algorithm with lower complexity.

Pei Yun Zhang and Meng Chu Zhou (Zhang and Zhou, 2018) proposed a method based on a two-stage strategy to maximize task scheduling performance and minimize unreasonable task allocation in clouds. At the first stage, a job classifier motivated by a Bayes classifier's design principle is utilized to classify tasks based on historical scheduling data. A certain number of virtual machines (VMs) of different types are accordingly created. This can save time of creating VMs during task scheduling. At the second stage, tasks are matched with concrete VMs dynamically. Dynamic task scheduling algorithms are accordingly proposed. They minimize the waiting time of VMs to schedule tasks, thus minimizing the cost to be paid by users who utilize VMs. The readily deployable algorithms are designed and illustrated to improve cloud task scheduling and execution results in comparison with those using traditional methods. However, the energy consumption of this method is high.

Summary

From the related work discussed above, the following problems of task scheduling are identified. These problems are considered as the challenges and are addressed by the proposed method.

- Overloading of the cloud servers may lead to large consumption of power, and execution inefficiency. On the contrary, they must not be underloaded, and in this case also energy consumption increases. The problem is critical with the evolution of big data processing in the cloud due to the necessity of using the cloud infrastructure in an energy-efficient way (Ismaila & Fardoun, 2016).
- Since the scheduling algorithms in the cloud technology need a useful study regarding the characteristics of the resources, considering several cost factors of energy, designing such algorithms is a challenge (Duan *et al.,* 2017).
- While processing the tasks scheduled, large variations in the execution time and the consumption of energy may occur, because of inconsistency in the process, physical faults, voltage/frequency variations, and so on (Li *et al.,* 2017).
- For the effective reservation of computing resources, a scheduling technique has to consider the load in the data center, which most of the existing systems ignore, to make the decisions concerning scheduling (Duan *et al.,* 2017).
- The major problems with the energy aware scheduling methods (Lin *et al.,* 2015; Li *et al.,* 2017; Shi *et al.,* 2016) are that they have considered only the energy as the important parameter, but it is necessary to include the Makespan for the efficient scheduling.
- In Xu *et al.* (2016), a task scheduling method was developed with well-defined energy model by considering the dynamic and application oriented parameters. The algorithm did not consider the optimization model, which is better for efficient energy aware scheduling even if multiple objectives are included.

MOBILE CLOUD SYSTEM MODEL

The cloud system model designed for the task scheduling process is depicted in Figure 1. On the submission of a user request, the task is accepted, and the service is provided to the user following a scheduling algorithm. Let $T = \{T_1, T_2, \ldots, T_n\}$ represent the tasks that may be executed in parallel. The allocation of tasks can be in either the public cloud or mobile cloud. The public cloud is comprised of a number of Physical Machines (PMs), represented as $P^M = \{P_1^M, P_2^M, \ldots, P_m^M\}$ Each physical machine is comprised of 'p' VMs, $V^M = \{V_1^M, V_2^M, \ldots, V_p^M\}$, such that the value of p is different for different PMs. PMs consist of four parameters, baseline energy consumption rate, denoted as α, internal communication bandwidth (β), internal communication energy (φ), and energy consumption at sleep

mode (δ). The value of these parameters is fixed in the limit [0,1]. Similarly, the task scheduling in VM is based on active energy consumption (μ^a), baseline energy consumption (μ^b), speed in Million Instructions Per Second (MIPS), processor (P), and memory (M). The sending rate S^R, which is the rate at which the task is sent to the cloud through the channel, is assigned a value 0.5. Network bandwidth, represented as β_{ext}, and energy consumption rate (λ) are the other parameters considered during the communication between a PM and a core, for the energy awareness during the task scheduling process. Meanwhile, a mobile device in MC consists of a number of cores as $C = \{C_1, C_2, \ldots, C_q\}$. Every core can be operated at different frequency scaling factors, denoted as *s*, whose value ranges from 0 to 1 and a constant γ that ranges between 2 and 3. Hence, the total number of resources in the cloud, where the tasks are scheduled, is $z = mp + q$. The process of scheduling follows a dependency graph, and the scheduling process is executed at various levels.

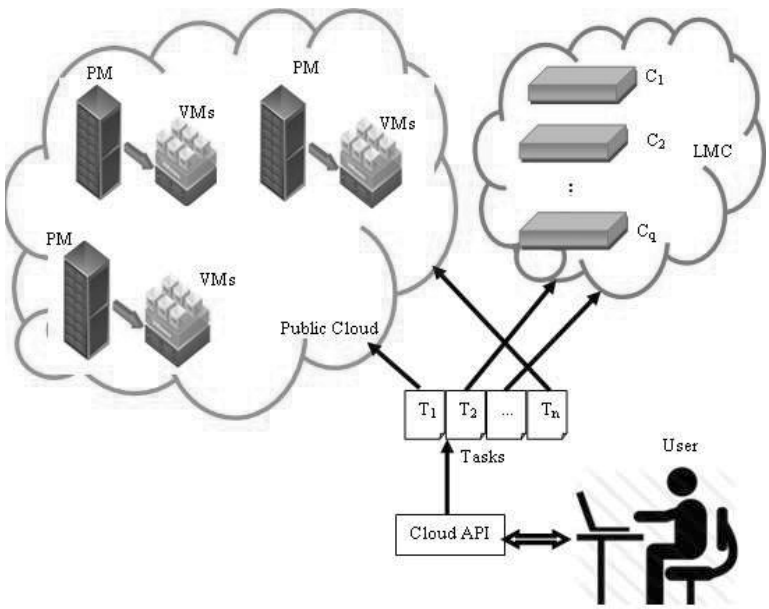

Figure 1. MobileCloud System model for task scheduling.

Energy Consumption Model

The proposed ADO-MTS technique follows two energy models, one during the task scheduling in the public cloud and the other for that in MC. The consumption of energy during the execution of a task in the cloud resource is to be at minimum for the energy aware task scheduling process. Thus, the energy consumed during the scheduling process is given as

$$E^T = \frac{1}{2} * \left[E^{Public} + E^{mobile} \right] \quad (1)$$

where E^{Public} and E^{mobile} represent the energy consumption in the public cloud and the MC, respectively.

Energy model in the public cloud

The energy consumption in the public cloud is modelled (Xu *et al.,* 2016) regarding two classifications, application and dynamic execution. The requirements of resources for application execution depend on the VMs in the physical servers. The energy consumption for dynamic execution is computed based on the PMs in idle and switch operation modes. Thus, the energy consumption in the cloud is given by

$$E^{Public} = \frac{1}{2} * \left[\frac{E^{P(App)}}{N^{P(App)}} + \frac{E^{P(Dyn)}}{N^{P(Dyn)}} \right] \quad (2)$$

where $E^{P(App)}$ is the energy consumed for application execution, $E^{P(Dyn)}$ is the energy consumed for dynamic executions, $N^{P(App)}$ is the normalized factor associated with application execution, and $N^{P(Dyn)}$ is the normalized factor associated with dynamic execution. The estimation of energy for the application execution is based on the following parameters: baseline energy consumption by PMs, energy consumption by VMs for execution, energy consumed by idle VMs, internal communication energy, and external communication energy. Hence, the energy (Xu *et al.,* 2016) is computed as

$$E^{P(App)} = E^{P}_{base} + E^{V}_{exe} + E^{V}_{idle} + E_{\text{int}} + E_{ext} \quad (3)$$

where E^{P}_{base} is the baseline energy consumption in PM, E^{V}_{exe} is the energy consumed by VMs for execution, E^{V}_{idle} is the energy consumed by idle VMs, E_{int} is the internal communication energy, and E_{ext} is the external communication energy. E^{P}_{base} is computed based on the active time, denoted as t^{act}, of each PM and α, while E^{V}_{exe} depends on the execution time of each VM in the particular PM and μ^{a}. E^{V}_{idle} is computed based on μ^{b} and the idle time, t^{idle}, which is calculated as follows:

$$t^{idle} = S - t^{act} \quad (4)$$

where S is the Makespan, and t^{act} is the active time, which will be discussed in section 4.1.1. The estimation of the internal communication energy depends on φ and the ratio of the data transmitted between two PMs to the bandwidth β. Similarly, the external communication energy, E_{ext}, is computed based on those parameters, i.e., λ and β_{ext}, associated with the communication between the PM and the core. The formulation of these parameters is similar to that presented in Xu *et al.*(2016).

The computation of energy for the dynamic execution follows the energy model in Xu *et al.*(2016), which is based on the two modes, idle and switch, as represented by the following equation:

$$E^{P(Dyn)} = E^{idle} + E^{switch} \quad (5)$$

where E^{idle} is the energy consumed when the PM is idle by multiplying the idle time t^{idle} of each VM with the corresponding rate of energy consumption, and E^{switch} is the energy utilized during switch operation, which is computed based on the energy consumption rate at sleep mode, represented as δ, baseline energy consumption in the PM (α), and active energy consumption in the VM (μ^{a}).

Energy model in the MC

This subsection explains the energy utility model (Lin *et al.,* 2015) in the MC environment during the task scheduling process. For the execution of a task in the MC, it does not utilize the energy of the mobile device. Hence, the energy consumed during the execution of a task T_i in the j^{th} core, C_j , is given by

$$E^{mobile} = \frac{\sum_{i=1}^{n} E_i}{N^{mobile}} = \frac{\sum_{i=1}^{n} \left(p_j, e^{M}_{i,j} \right)}{N^{mobile}}; 1 < j \leq q \quad (6)$$

where p_j is the average power, n is the number of tasks, $e^{M}_{i,j}$ is the execution time of i^{th} task in C_j, and N^{mobile} is the normalized factor associated with MC.

The average power consumed by the core can be measured taking the operating frequency as

$$p_j = s_j \left(F_j\right)^{r_j} \tag{7}$$

where r_j is a constant whose value is in the interval [2,3] and F_j is the j^{th} core frequency, given by

$$F_j = s_j F_{\max} \tag{8}$$

where $F_{\max}$ is the maximum operating frequency.

PROPOSED METHODOLOGY

This section explains the proposed task scheduling technique using ADO algorithm in the cloud computing environment, either public cloud or MC. To schedule the tasks in the cloud optimally, ADO algorithm is designed by combining CAViaR approach (Engle & Manganelli, 1999) and DA. Multiobjectives, such as energy utilization, Makespan, and resource utilization, are considered in designing the fitness function, for the proper selection of resources in the cloud to schedule the tasks. Hence, a multiobjective model is designed with the objective of reducing the energy, and the Makespan, but with increasing the resource utilization. The energy utility is computed in both cloud environments based on the resource where the task is scheduled.

ADO-MTS Technique

In this section, the ADO-MTS technique proposed for task scheduling in the cloud is illustrated. The tasks requested by the users to the cloud through the internet are handled by the task buffer and the task manager. The cloud includes a number of resources that consume sufficient energy. The information regarding the energy, Makespan, and resource utilization is collected by the task manager. For the feasible scheduling of tasks to the cloud resources, an efficient task scheduling algorithm is required. Hence, the scheduling algorithm, ADO, is developed utilizing the multiobjective model in the fitness function. Thereby, the tasks are assigned to the appropriate resources in the cloud based on the requirement according to the energy consumption, Makespan, and resource utilization. As energy is the major objective considered in the proposed ADO-MTS technique, the consumption of energy in both cloud environments is combined to formulate an energy model, discussed in mobile cloud system model. Based on the scheduling done, the tasks are executed using the VMs in the public cloud and the heterogeneous cores in the MC in parallel. The block diagram of the proposed ADO-MTS technique is pictured out in Figure 2.

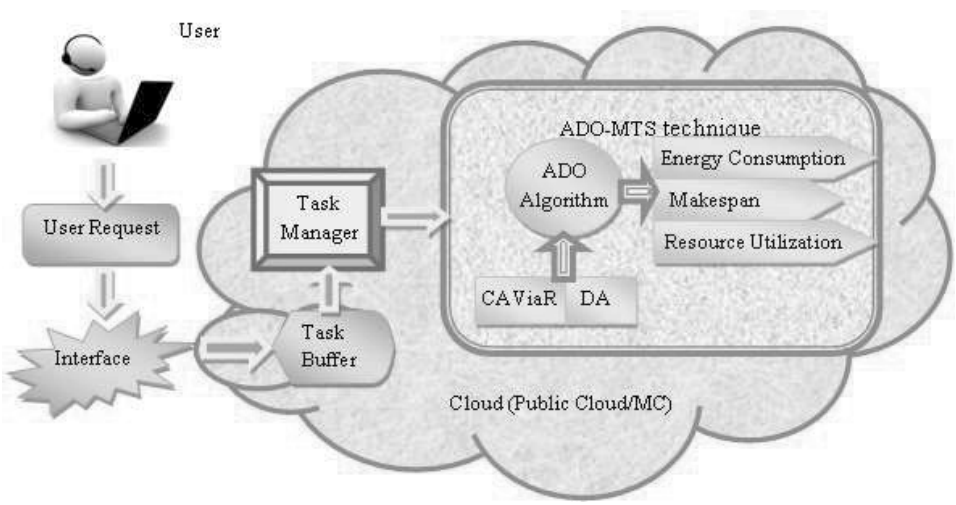

Figure 2. Block diagram of the proposed ADO-MTS technique.

Multiobjective Fitness Model

One of the challenges in considering multiobjective model is the complexity in satisfying all the objectives considered. The proposed technique developed for task scheduling considers three major objectives, that is, energy, Makespan, and resource utilization. The requirement of any resource in the cloud is i) minimizing the consumption of energy, ii) minimizing the Makespan, and iii) enhancing the resource utilization. Based on the three considered objectives, the fitness function can be formulated as follows:

$$f = \frac{S}{N^S} + [1 - R] + E^T \tag{9}$$

where S is the Makespan, N^S is the normalized factor associated with Makespan, R is the resource utilization, and E^T is the total energy consumed. The computation of the energy is based on the energy consumption model.

Makespan: The second objective, Makespan, is the time consumed by the resources for the completion of all the tasks. Scheduling the tasks to different cloud resources is usually to reduce the Makespan. Hence, the Makespan is computed based on the execution time as

$$S = \sum_{i=1}^{n} \max_{k=1}^{N} e_i^k \tag{10}$$

where n is the number of tasks, N is the number of parallel executions possible, and e_i^k is the execution time. Since the execution time of a task differs for the public cloud and the MC, it is represented as

$$e_i = \begin{cases} e_i^P \;; if\ i \in z\ \&\ z \in Public\ cloud \\ e_{i,j}^M \;; if\ i \in z\ \&\ z \in MC \end{cases} \tag{11}$$

where e_i^P is the execution time in the public cloud during the execution of the task T_i, $e_{i,j}^M$ is the execution time in the MC, and z is the number of resources available. The estimation of execution time requires Task Assignment Matrix (TAM), which is a matrix constructed according to the assignment of the task in a resource. Hence, the execution time in the public cloud is computed in accordance with TAM as

$$e_i^P = \frac{D_i^L}{\frac{1}{3}\left[\frac{(TAM_{i,z} * I_z)}{c_1} + \frac{(TAM_{i,z} * P_z)}{c_2} + \frac{(TAM_{i,z} * M_z)}{c_3}\right]} \tag{12}$$

where D_i^L is the length of the data in the i^{th} task, I represents MIPS, P is the processor, M is the memory, $TAM_{i,z}$ represents the TAM of i^{th} task that takes a value 1 for the task assigned in z, and C_1, C_2, and C_3, are the constants taking the maximum of I, P, and M, respectively.

$$\left.\begin{aligned} c_1 &= \max_{z=1}^{p} I_z \\ c_2 &= \max_{z=1}^{p} P_z \\ c_3 &= \max_{z=1}^{p} M_z \end{aligned}\right\} \tag{13}$$

where p is the number of VMs. Based on the execution time of each task in the scheduled resources, Task Time Matrix (TTM) is built.

$$TTM_{i,z} = TAM_{i,z} * e_i \;; 1 < z < p + q \tag{14}$$

where $TAM_{i,z}$ represents the TAM of i^{th} task that takes a value 1 for the task assigned in z, and e_i is the execution time in the clouds. Accordingly, the active time, which is the time at which the resource is involved actively in the execution of a task, is computed based on the TTM of i^{th} task in the resources as

$$t^{act} = \sum_{i=1}^{n} TTM_{i,z} \tag{15}$$

where n is the number of tasks. When the VM is idle, it consumes only the baseline energy.

Meanwhile, the execution time of a task in the MC is computed as the ratio of a decreasing function to the frequency scaling as

$$e_{i,j}^{M} = \frac{e_{i,j}^{M,\min}}{s_j} * D_i^L \tag{16}$$

where $e_{i,j}^{M,\min}$ is the decreasing function of the maximum frequency, s_j is the scaling factor, and D_i^L is the data length.

Resource Utilization: The utilization of resources in the cloud is measured according to the MIPS, processors, and memory used. Also, it depends on the TTM that computes the execution time of every task in the allocated resources.

$$R = \frac{1}{n}\left\{\left[\sum_{z=1}^{p}\sum_{i=1}^{n} TTM_{i,z} * \left(\frac{I_z}{c_1} + \frac{P_z}{c_2} + \frac{M_z}{c_3}\right)\right] + \left[\sum_{z=m+1}^{q}\sum_{i=1}^{n} TTM_{i,z} * s\right]\right\} \tag{17}$$

where q is the number of cores, s is the frequency scaling, I_z is the MIPS, P_z is the processor, M_z is the memory, and c_1, c_2 and c_3 take the value as expressed in equation (6). The fitness is preferred to be at minimum for the solution to be the best, as the solution must have minimized energy consumption, minimized Makespan, and maximum utilization of resources.

Autoregressive Dragonfly Optimization for Optimal Task Scheduling

The proposed ADO algorithm developed for the optimal selection of resources to schedule the tasks is described in this section. By the incorporation of CAViaR approach in DA (Mirjalili, 2016), ADO algorithm is proposed, where the optimal resource to assign a particular task is determined based on a multiobjective fitness function. DA is an optimization technique developed according to the swarming behavior of dragonflies. The reason behind selecting DA in the proposed technique is due to its ability in solving multiobjective problems. Even though DA has better convergence to the global optimum, a further improvement in the algorithm can be attained using the CAViaR model. The following sections explain the proposed ADO algorithm in detail.

Solution Representation

The solution representation provides the simplest view of the technique representing its solution. In the proposed technique, the solution is determined using the proposed ADO algorithm. Since the objective of ADO-MTS is task scheduling, the solution is the resource in the cloud to be selected optimally to schedule a particular task. The order of the task to be executed in the cloud resource is represented by the index. Hence, the solution is a vector having dimensions equal to the number of tasks and is represented as $1 \times n$. Initially, the solutions are generated in random, and the suitable one from the population n is selected based on the fitness function that considers multiple objectives for the optimal selection of the cloud resource. Consider the dependency graph, based on which the task scheduling is performed. Let the number of tasks to be scheduled be four, denoted as T_1, T_2, T_3 , and T_4 . Depending on the priority of the task, at level 1 the task T_1 is executed in a cloud resource, while the tasks T_2 and T_4 are performed at level 2. Finally, at the third level, T_3 is performed. Hence, the order of task execution is as follows: T_1 , T_2 and T_4, T_3. The scheduling

of tasks to the resources is given in the solution representation depicted in figure 3. Let the number of VMs available in the public cloud be three, V_1^M, V_2^M, and V_3^M, and the cores in the MC be two, C_1 and C_2. Hence, the total number of resources is five, i.e., $z = 5$. Each element in the solution vector corresponds to a cloud resource. Hence, as shown in Figure 3, T_1 is scheduled in the resource numbere 2, i.e., V_2^M, T_2 in C_1 (4), T_4 in V_1^M (1), and T_3 in C_2 (5).

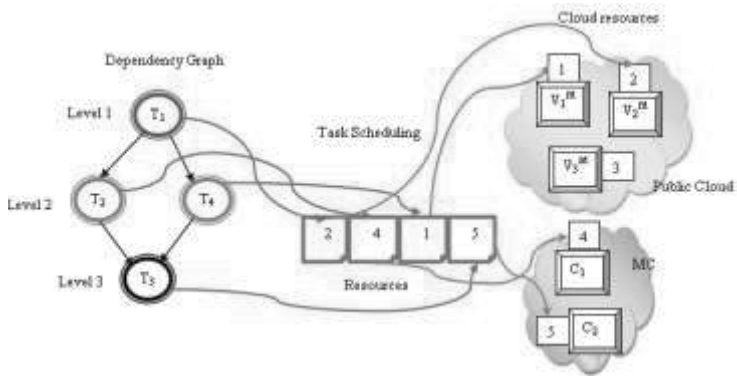

Figure 3. Solution representation of ADO-MTS technique.

Autoregressive Dragonfly Optimization Algorithm

The proposed ADO algorithm designed by incorporating CAViaR model into DA is demonstrated in this section. Determining how the dragonflies interact in navigation, food search, and enemy avoidance, two phases, namely, exploitation and exploration, are designed in DA (Mirjalili, 2016). The position update process of DA is modified using CAViaR model to improve the performance of the algorithm by selecting the optimal solution. The algorithmic steps are as follows.

I) Population Initialization

The foremost step is to initialize the population of dragonflies in random as

$$Z = \{Z_1, Z_2, \ldots, Z_l, \ldots, Z_K\}; 1 < l \leq K \tag{18}$$

where Z_l is the l^{th} position of the solution in the population and K is the dimension of the population. The size of the population is assumed 10, i.e., $K = 10$. In addition, the step vector, $\Delta Z_l; 1 < l \leq K$, is also initialized.

II. Fitness Evaluation

After the initialization of the population, the fitness of each solution is computed using the fitness formulated in equation (9). Every solution updates its position to form the best solution in each consecutive iteration based on its fitness.

III. Defining swarm behavior based on three principles

The algorithm follows three principles, that is, separation, alignment, and cohesion, to define the swarm behavior as follows.

i) Separation: It is defined as "the static collision avoidance of the individuals from other individuals in the neighborhood".

$$A_l = -\sum_{r=1}^{Q} Z - Z_r \tag{19}$$

where Z is the position of the individual at the current iteration, and Z_r is the position of the r^{th} individual in the neighborhood Q.

ii) Alignment: The alignment is defined as "velocity matching of individuals to that of other individuals in neighborhood" and is computed as

$$G_l = \sum_{r=1}^{Q} L_r \tag{20}$$

where L_r is the velocity of r^{th} individual in the neighborhood.

iii) Cohesion: Cohesion is defined as "the tendency of individuals towards the centre of mass of the neighborhood",

$$H_l = \frac{\sum_{r=1}^{Q} Z_r}{Q} - Z \tag{21}$$

where Z and Z_r are the positions of the current and the r^{th} individuals and Q is the number of neighborhoods.

IV. Autoregressive based position update

The position update process depends on two more factors, other than the three principles. The two factors are attraction to a food source, represented as B_l, and distraction from an enemy, D_l, defined as follows:

$$B_l = Z^* - Z \tag{22}$$

$$D_l = Z^- + Z \tag{23}$$

where Z^* represents the best position of the individual and Z^- represents the worst position. For the position update, step vector that indicates the direction at which the dragonflies move is computed.

$$\Delta Z(t+1) = (aA_l + gG_l + hH_l + bB_l + dD_l + W\Delta Z(t)) \tag{24}$$

where t represents the current iteration, a is the weight related to separation, A_l is the separation of l^{th} individual, g is the weight related to alignment, G_l is the alignment of l^{th} individual, h is the weight related to cohesion, H_l is the cohesion of l^{th} individual, b is the food factor, B_l is the food source of l^{th} individual, d is the enemy factor, D_l represents the distraction, and W is the weight of inertia. From equation (24), the position of the individuals can be updated as

$$Z(t) = Z(t+1) - \Delta Z(t+1) \tag{25}$$

where $\Delta Z(t+1)$ is the step vector at iteration $t+1$ and $Z(t)$ is the position of the individual at the current iteration. At this instance, CAViaR approach that indicates the evolution of the quantile utilizing an autoregressive method is adopted. In CAViaR, the unknown parameters are computed by reducing the regression quantiles. The ability in adapting to any environment makes CAViaR model applicable in the optimization algorithm to enhance the performance of the algorithm. The model has a general representation as follows:

$$Z(t+1) = \varepsilon_0 + \varepsilon_1 Z(t) + \varepsilon_2 Z(t-1) + \varepsilon_1 f(Z(t)) + \varepsilon_2 f(Z(t-1)) \tag{26}$$

where ε is a constant, representing a vector of unknown parameters, $f(Z(t))$ is the fitness of the solution at current iteration, and $f(Z(t-1))$ is the fitness of the solution at iteration $t-1$. Subtracting $Z(t)$ at both sides of equation (26),

$$Z(t+1)-Z(t)=\left[\varepsilon_0+\varepsilon_1 Z(t)+\varepsilon_2 Z(t-1)+\varepsilon_1 f(Z(t))+\varepsilon_2 f(Z(t-1))\right]-Z(t) \quad (27)$$

Substituting equation (25), in the above equation, the position update of ADO algorithm can be derived as follows:

$$Z(t+1)-Z(t)=\left[\varepsilon_0+\varepsilon_1 Z(t)+\varepsilon_2 Z(t-1)+\varepsilon_1 f(Z(t))+\varepsilon_2 f(Z(t-1))\right]-Z(t+1)+\Delta Z(t+1) \quad (28)$$

$$2Z(t+1)=Z(t)+\varepsilon_0+\varepsilon_1 Z(t)+\varepsilon_2 Z(t-1)+\varepsilon_1 f(Z(t))+\varepsilon_2 f(Z(t-1))+\Delta Z(t+1) \quad (29)$$

$$Z(t+1)=\frac{1}{2}\left[Z(t)+\varepsilon_0+\varepsilon_1 Z(t)+\varepsilon_2 Z(t-1)+\varepsilon_1 f(Z(t))+\varepsilon_2 f(Z(t-1))+\Delta Z(t+1)\right] \quad (30)$$

where $Z(t)$ is the position of the individual at current iteration, $Z(t-1)$ is the position of the individual at $(t-1)^{th}$ iteration, $Z(t+1)$ is the position of the individual at iteration $t+1$, and $\Delta Z(t+1)$ is the step vector. For the improvement of randomness, exploration, and so on, the algorithm utilizes Levy flight, as given in equation (31), to expand the search space. In this case, the position is updated as

$$Z(t+1)=Z(t)+Levy(u)\times Z(t) \quad (31)$$

where u represents the size of the position vectors, and the levy flight is computed as

$$Levy(u)=0.01\times\frac{v_1\times\sigma}{|v_2|^{\frac{1}{\rho}}} \quad (32)$$

where v_1, v_2 are two numbers that take a value between 0 and 1, $\rho = 1.5$, and σ is given by

$$\sigma=\left(\frac{\Gamma(1+\rho)\times\sin\left(\frac{\pi\rho}{2}\right)}{\Gamma\left(\frac{1+\rho}{2}\right)\times\rho\times 2^{\left(\frac{\rho-1}{2}\right)}}\right) \quad (33)$$

where $\Gamma(u) = (u-1)$.

V. Finding the best solution

Once the positions of the individuals are updated, the fitness of the individuals is computed. The solution that has the minimum fitness is determined as the best solution.

VI. Termination

The steps from II to V are repeated until the algorithm reaches the termination criterion. Here, the criterion is to terminate the procedure when the iteration t exceeds the maximum iteration, t_{max}, before which the best solution is determined.

The algorithmic procedure is described using the pseudocode given in Table 1.

Table 1. Pseudocode of ADO Algorithm.

	Proposed ADO Algorithm
1	Input: Random population $Z = \{Z_1, Z_2, \ldots, Z_l, \ldots, Z_K\}; 1 < l \leq K$
2	Output: Best position Z^*
3	Parameters: Separation A, Alignment G, Cohesion H, Food factor B, and Position of enemy D
4	Begin
5	Initialize the population and the step vectors
6	for ($t < t_{max}$)
7	for each solution in the population
8	Compute the fitness using equation (9)
9	Update A, G, H, B, and D using equations (19), (20), (21), (22), and (23).
10	if a dragonfly has atleast one neighbor
11	Update the position using equation (30)
12	else
13	Update the position using equation (31)
14	end if
15	$t = t + 1$
16	end for
17	Return Z^*
18	end for
19	Terminate

RESULTS AND DISCUSSION

The results of the proposed ADO-MTS technique developed for task scheduling in both the cloud computing environments (public cloud and MCC) are demonstrated in this section. The performance of the proposed technique is compared with existing techniques for the evaluation.

For the performance comparison, three existing techniques, that is, i) EnReal (Xu *et al.,* 2016), ii) Xue Lin *et al.* (Lin *et al.,* 2015), and iii) DA-TS (Applied DA (Mirjalili, 2016) in the proposed technique instead of the proposed

ADO algorithm in the ADO-MTS technique), are employed. The technique presented in Lin *et al.*(2015) is a task scheduling method in MC, where DVFS technique is utilized for the minimization of the energy utility. Another method employed for the comparison is EnReal, which is also designed to solve the problems related to energy constraints. The performance of these three techniques is compared with that of the proposed technique using the evaluation metrics.

The measures used for the evaluation of the performance of the comparative techniques are resource utilization, Makespan, and energy. Makespan and resource utilization are defined and formulated in the fitness model. The conservation of energy during the execution of tasks in the cloud is given in equation (1).

Experimental Setup

The ADO-MTS technique is experimented in a system operated using Windows 10 with 64-bit OS and 2 GB memory. The software used for the implementation of the task scheduling technique is JAVA with CloudSimtool. For the experimentation, four setups are formed based on different numbers of PMs, VMs, and cores, as given below.

Setup 1: The first setup assumes two PMs, and ten VMs in the public cloud and two cores in the MC, for the experimentation.

Setup 2: Setup 2 is designed with the public cloud having three PMs and ten VMs, while the number of cores in the MC is three.

Setup 3: For the third setup, the number of PMs and VMs in the public cloud is 4 and 20; and the MC is comprised of four cores.

Setup 4: The number of PMs and VMs in the setup 4 is five and 20 and the cores assigned in the MC are five.

Comparative Analysis

This subsection presents the comparative analysis, where the performance of existing EnReal, Xue Lin *et al.*, and DA-TSis compared with that of ADO-MTS and evaluated using the three performance evaluation measures by varying the task sizes, as given below.

Comparison based on Resource Utilization

The analysis based on resource utilization in the comparative techniques is illustrated using Figure 4 for the four cloud setups. The analysis is made by fixing the number of tasks as 20, 30, 40, and 50. Maximum utilization of resources indicates that the technique has performed the task scheduling process effectively. Figure 4.a presents the analysis based on resource utilization for the first cloud setup. For the first setup, when the task size is 20, Xue Lin *et al.* have the resource utilization of 0.5516, while that in EnReal and DA-TS is 0.5519. Meanwhile, the proposed ADO-MTS technique has a value of 0.552. Increasing the number of tasks to 30, Xue Lin *et al.,*EnReal, and DA-TS have attained the resource utilization of 0.5111, 0.5114, and 0.5117, respectively, for setup 1. Figure 4.b presents the analysis based on resource utilization for the second cloud setup. The utilization of resources has increased to a value of 0.5494 in Xue Lin *et al.,* for the same number of tasks, for setup 2. However, at that instant, ADO-MTS can attain a value of 0.5502. Figure 4.d presents the analysis based on resource utilization for the second cloud setup. For setup 4, when 0.5355 is the resource utilization obtained using Xue Lin *et al.* with the task size fixed to 20, a little improvement is observed in EnReal method that has the resource utilization of 0.5359. In the same setup and task size, the utilization of resource measured using ADO-MTS is 0.5372. When the number of tasks is kept atits maximum, i.e., 50, the resource utilization attained using all the comparative techniques seems to have a similar result without significant deviation. However, the proposed technique has obtained the maximum resource utilization of 0.5438 for setup 4, while that in the existing DA-TS is 0.5433.

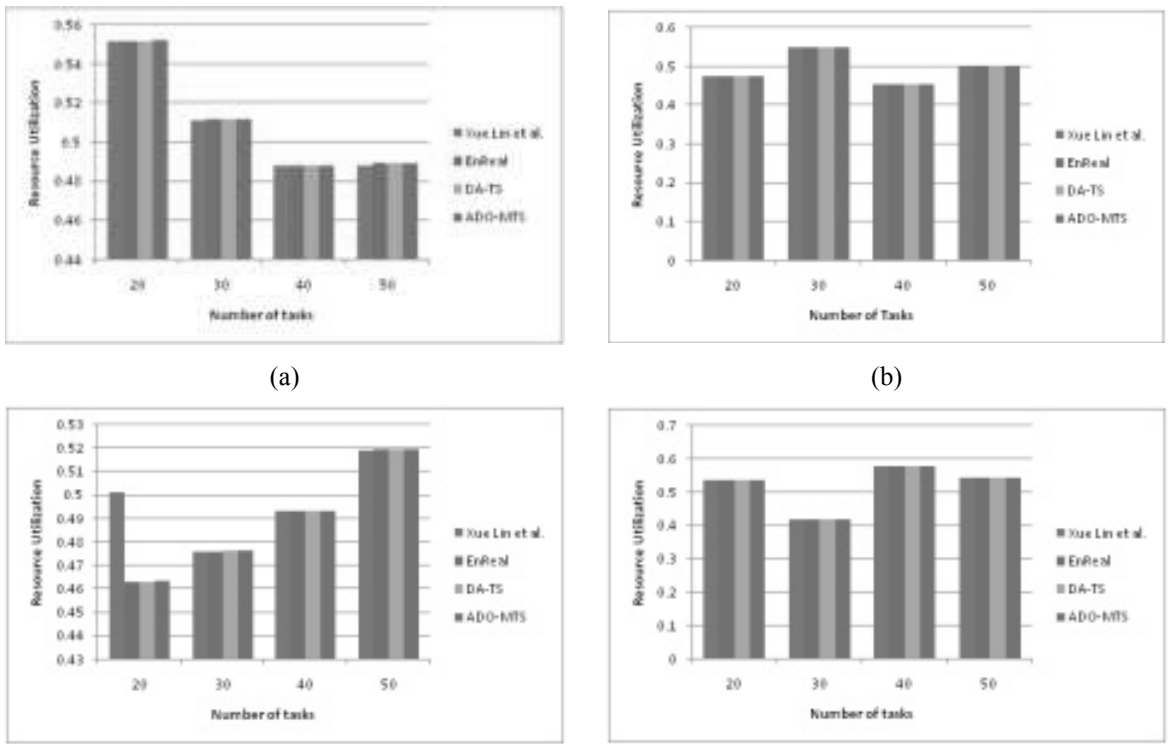

Figure 4. Comparative Analysis based on Resource Utilization in (a) Setup 1, (b) Setup 2, (c) Setup 3, and (d) Setup 4.

Comparison based on Makespan

This subsection illustrates the results of the comparative analysis performed based on Makespan in all the techniques considered for various task sizes in the four setups using Figure 5. Figure 5.a presents the analysis based on Makespan for the first cloud setup. A task scheduling technique is said to be effective if its Makespan is reduced. Xue Lin *et al*, EnReal, and DA-TS have Makespan of 23.75, 16.81, and 10.12, whereas the proposed ADO-MTS has just 6.85. Fixing the task size as 50, the Makespan seems to be increasing in the existing EnReal and DA-TS with values 22.85 and 13.91. Meanwhile, the ADO-MTS technique has a Makespan of only 6.22 and thus, this proves its effectiveness. In figure 5.b, the comparative analysis based on Makespan in setup 2 is depicted. As shown in the plot, the existing Xue Lin *et al* has taken the maximum Makespan than the other techniques compared for all the task sizes considered. As the number of tasks is 50, the Xue Lin *et al.* Makespanis 25.45. For the same task size, the Makespan of the proposed technique is just 8.94. The Makespan analysis for the third setup is pictured out in figure 5.c, where the minimum Makespan achieved by EnReal and DA-TS is 29.55 and 9.47, when the number of tasks is the least, i.e., 20. Meanwhile, ADO-MTS has reduced its Makespan to just 6.95, indicating that it has the minimum Makespan among the existing techniques. The analysis based on Makespan for the cloud setup 4 is presented in figure 5.d. Initially, when the number of tasks is kept atminimum, Xue Lin *et al.* havea Makespan of 39.82, while ADO-MTS has just 6.75. Even though the task size is increased to 50, the proposed technique has the Makespan of only 16.41.

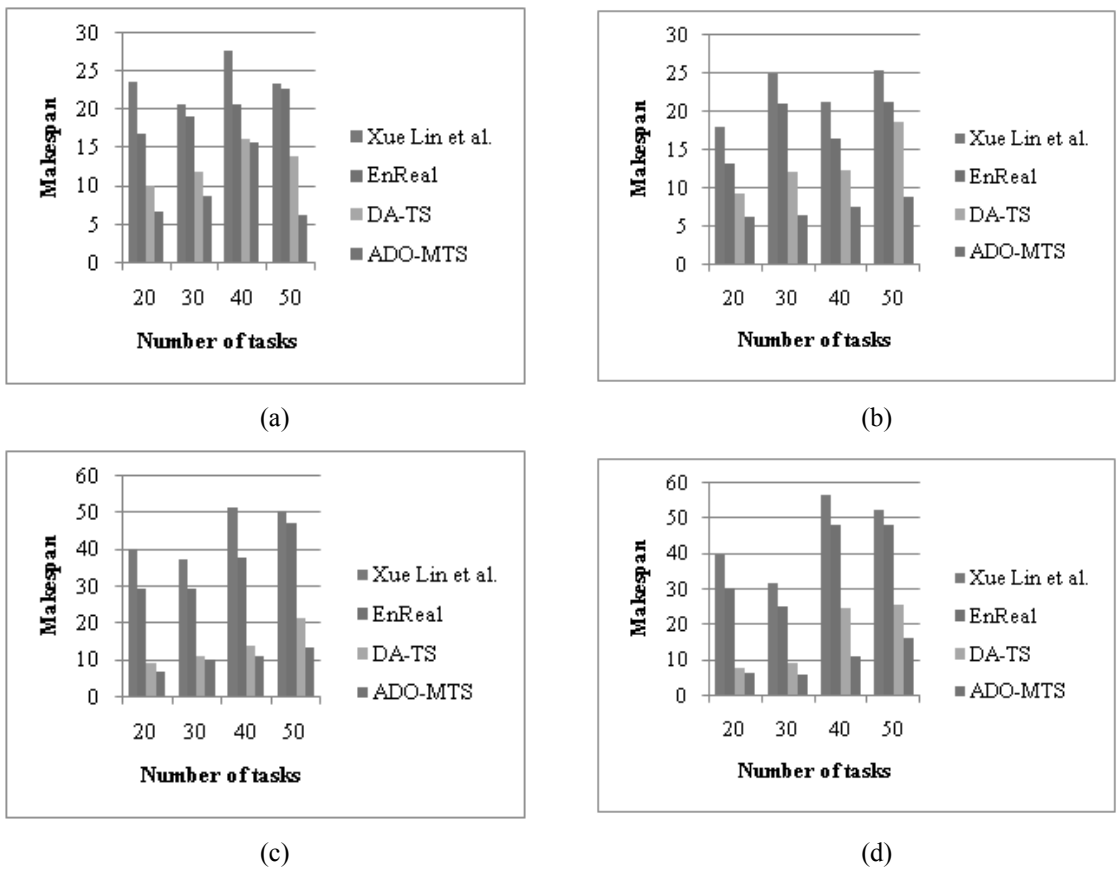

Figure 5. Comparative Analysis based on Makespan in (a) Setup 1, (b) Setup 2, (c) Setup 3, and (d) Setup 4.

Comparison based on Energy

A similar comparison is made regarding the energy consumed in each technique using Figure 6. The objective of any technique employed for task scheduling is to reduce the energy consumption so as to improve its performance. The comparative analysis based on energy for the first setup is given in Figure 6.a by varying the task size as 20, 30, 40, and 50. When 0.4934 is the energy attained by Xue Lin *et al.* for 20 tasks, the proposed ADO-MTS technique has utilized an energy value of just 0.2299. As the number of tasks is increased from 20 to 50, the energy tends to be increasing with a value of 0.6319 in Xue Lin *et al.*, whereas ADO-MTS has just 0.3084. Figure 6.b sketches out the energy analysis for setup 2 in all the comparative techniques. At the maximum number of tasks, the energy consumed by the existing Xue Lin *et al.*, EnReal, and DA-TS is 0.4239, 0.3819, and 0.2437, while that in the proposed ADO-MTS is 0.1847. In figure 6.c, the plot showing the energy analysis for the third setup is pictured out. Here, initially, the energy consumed by the proposed technique is 0.2196, which increases to 0.2512 at the maximum number of tasks. Figure 6.d presents the comparative analysis based on energy for setup 4, where the minimum energy attained among the existing techniques is 0.2255 by DA-TS. However, it is possible to obtain a minimum energy of 0.1756 using the proposed technique. Thus, it is clear that the proposed ADO-MTS has the minimum energy consumption than the existing Xue Lin *et al.*, EnReal, and DA-TS techniques.

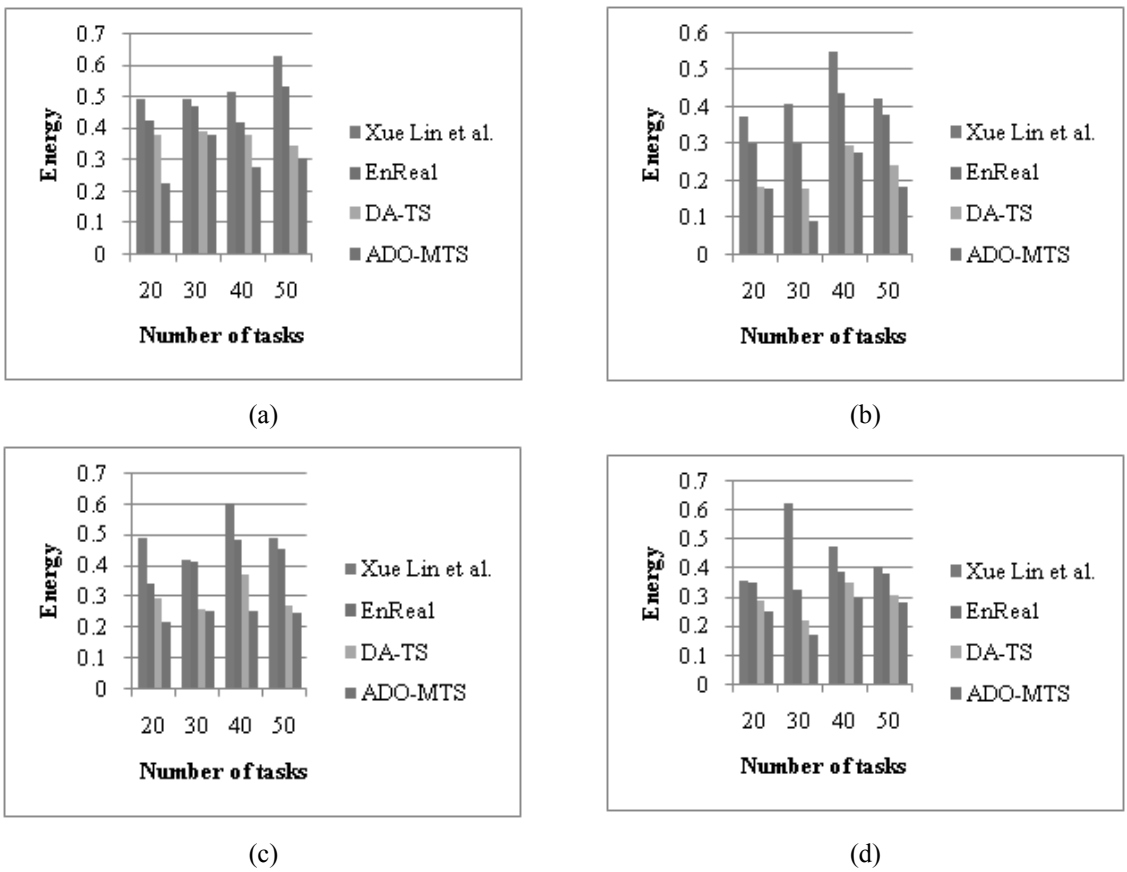

Figure 6. Comparative Analysis based on Energy in (a) Setup 1, (b) Setup 2, (c) Setup 3, and (d) Setup 4.

DISCUSSION

The comparative discussion based on the overall analysis is illustrated in this section using Table 2. It lists the maximum performance attained by Xue Lin *et al.*, EnReal, DA-TS, and ADO-MTS, irrespective of the size of the tasks, for the four cloud setups.

Table 2. Performance Comparison.

Comparative Techniques	*Performance Measures*	*Setup 1*	*Setup 2*	*Setup 3*	*Setup 4*
Xue Lin *et al.*	***Resource Utilization***	0.5516	0.5494	0.5192	0.579
	Makespan	20.74	18.01	37.3	31.82
	Energy	0.4925	0.4107	0.4195	0.3597
EnReal	***Resource Utilization***	0.5519	0.5497	0.5196	0.579
	Makespan	16.81	13.29	29.55	25.3
	Energy	0.4238	0.3012	0.3442	0.3291

DA-TS	***Resource Utilization***	0.5519	0.5501	0.5196	0.5791
	Makespan	10.12	9.27	9.47	8.03
	Energy	0.3471	0.1813	0.2613	0.2255
ADO-MTS	***Resource Utilization***	0.552	0.5502	0.5197	0.5795
	Makespan	6.22	6.26	6.95	6.22
	Energy	0.2299	0.092	0.2196	0.1756

As shown in Table 2, the maximum performances observed among the comparative techniques in terms of resource utilization, Makespan, and energy are represented in bold. When the maximum utility of resources obtained in Xue Lin *et al.* is 0.5494 for setup 2, the proposed ADO-MTS has aresource utilization of 0.5502. The minimum Makespan observed in ADO-MTS is 6.22 for two cloud setups, 1 and 4, while the existing DA-TS has a value 8.03 as its minimum Makespan. Analyzing the energy consumption, it is seen that the proposed technique has the least energy than the comparative techniques with a value of 0.092 for setup 2. Meanwhile, the minimum possible energy in the existing DA-TS is 0.1813, which increases considerably in other techniques. Thus, from the discussion, it is obvious that the proposed ADO-MTS technique has attained the maximum performance for all the cloud setups considered.

CONCLUSION

This paper presents an energy aware task scheduling technique, known as ADO-MTS, considering energy consumption as the major objective. An energy model is designed for the allocation of resources in the public cloud and MC to compute the energy consumption during task scheduling. For the optimal scheduling of tasks to the suitable resources, ADO algorithm is proposed by the integration of CAViaR model in DA, which utilizes the multiobjective model in the fitness function. The fitness function formulated using multiple objectives, that is, energy consumption, Makespan, and resource utilization, helps in determining the best resource for the task based on the priorities of the tasks. Thus, the proposed energy aware ADO-MTS technique schedules the task effectively in both the cloud environments. The performance of the proposed ADO-MTS technique is compared with three existing techniques, namely, Xue Lin *et al.*, EnReal, and DA-TS, and is evaluated using the performance evaluation measures like resource utilization, Makespan, and energy. The results show that the ADO-MTS technique has provided the high performance by obtaining a maximum resource utilization of 0.5795, minimum Makespan of 6.22, and minimum energy consumption of 0.092, respectively. From the comparative analysis, it can be concluded that the ADO-MTS technique outperforms the existing techniques compared, performing the task scheduling process effectively. However, the computational time of the proposed method is slightly high. In future, we will develop the task scheduling algorithm with the aim of minimizing the computational complexity.

REFERENCES

Ali, H.G.E.D.H., Saroit, I.A. & Kotb, A.M. (2017). Grouped tasks scheduling algorithm based on QoS in cloud computing network, *Egyptian Informatics Journal,* **18**(1): 11-19.

Chen, W., Xie, G., Li, R., Bai, Y., Fan, C. & Li, K. (2017). Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems, *Future Generation Computer Systems,* **74**, 1-11.

Ding, Y., Qin, X., Liu, L & Wang, T. (2015). Energy efficient scheduling of virtual machines in cloud with deadline constraint, *Future Generation Computer Systems,* **50**, 62-74.

Duan, H., Chen, C., Min, G & Wu, Y. (2017). Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems, *Future Generation Computer Systems,* **74**, 142-150.

Engle, R.F. & Manganelli, S. (1999). CAViaR: Conditional Value at Risk by Quantile Regression, Journal of Business & Economic

Statistics, *American Statistical Association,* **22**, 367-381.

Hongyou, L., Jiangyong, W., Jian, P., Junfeng, W & Tang, L. (2013). Energy-Aware Scheduling Scheme Using Workload-Aware Consolidation Technique in Cloud Data Centres, *China Communications,* **10**(12): 114-124.

Hosseinimotlagh, S., Khunjush, F & Samadzadeh, R. (2015). SEATS: smart energy-aware task scheduling in real-time cloud computing, *The Journal of Supercomputing,* **71**(1): 45-66.

Ismaila, L & Fardoun, A. (2016). EATS: Energy-Aware Tasks Scheduling in Cloud Computing Systems, *Proceedings Computer Science,* **83**, 870 – 877.

Jiang, Q., Ma, J & Wei, F. (2016). On the Security of a Privacy-Aware Authentication Scheme for Distributed Mobile Cloud Computing Services, *IEEE Systems Journal,* 1-4.

Juarez, F., Ejarque, J & Badia, R.M. (2016). Dynamic energy-aware scheduling for parallel task-based application in cloud computing, *Future Generation Computer Systems.*

Kaur, T & Chana, I. (2016). Energy aware scheduling of deadline-constrained tasks in cloud computing, *Cluster Computing,* **19**(2): 679-698.

Li, K. (2017). Scheduling parallel tasks with energy and time constraints on multiple many core processors in a cloud computing environment, *Future Generation Computer Systems.*

Li, Y., Chen, M., Dai, W & Qiu, M. (2017). Energy Optimization With Dynamic Task Scheduling Mobile Cloud Computing, *IEEE Systems Journal,* **11**(1): 96-105.

Lin, W., Xu, S., He, L. & Li, J. (2017). Multi-resource scheduling and power simulation for cloud computing, *Information Sciences,* 397-398, 168-186.

Lin, X., Wang, Y., Xie, Q. & Pedram, M. (2014). Energy and performance-aware task scheduling in a mobile cloud computing environment, *in the proceedings of IEEE International Conf. Cloud Computing* (Cloud '14), 192-199.

Lin, X., Wang, Y., Xie, Q. & Pedram, M. (2015). Task Scheduling with Dynamic Voltage and Frequency Scaling for Energy Minimization in the Mobile Cloud Computing Environment, *IEEE Transactions on Services Computing,* **8**(2): 175-186.

Liu, Y., Xu, X., Zhang, L., Wang, L. & Zhong, R.Y. (2017). Workload-based multi-task scheduling in cloud manufacturing, *Robotics and Computer-Integrated Manufacturing,* **45**, 3-20.

Luo, C., Yang, L.T., Li, P., Xie, X. & Chao, H. (2015). A holistic energy optimization framework for cloud-assisted mobile computing, *IEEE Wireless Communications,* **22**(3): 118-123.

Mirjalili, S. (2016). Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, *discrete, and multi-objective problems, Neural Computing and Applications,* **27**(4): 1053-1073.

Shi, T., Yang, M., Li, X., Lei, Q. & Jiang, Y. (2016). An Energy-Efficient Scheduling Scheme for Time-constrained Tasks in Local Mobile Clouds, *Pervasive and Mobile Computing,* **27**, 90-105.

Tsai, J & Lo, N. (2015). A Privacy-Aware Authentication Scheme for Distributed Mobile Cloud Computing Services, *IEEE Systems Journal,* **9**(3): 805-815.

Wang, X., Wang, Y. & Cui, Y. (2016). An energy-aware bi-level optimization model for multi-job scheduling problems under cloud computing, *Soft Computing,* **20**(1): 303-317.

Xiong, Y., Huang, S., Wu, M., She, J. & Jiang, K. (2019). A Johnson's-Rule-Based Genetic Algorithm for Two-Stage-Task Scheduling Problem in Data-Centers of Cloud Computing, *IEEE Transactions on Cloud Computing*, **7**(3): 597-610.

Xu, X., Dou, W., Zhang, X. & Chen, J. (2016). EnReal: An Energy-Aware Resource Allocation Method for Scientific Workflow Executions in Cloud Environment, *IEEE Transactions on Cloud Computing,* **4**(2): 166-179.

Zhang, P.Y & Zhou, M.C. (2018). Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy, *IEEE Transactions on Automation Science and Engineering,* **15**(2): 772-783.

Zhao, Q., Xiong C., Yub, C., Zhang, C. & Zhao, X. (2016). A new energy-aware task scheduling method for data-intensive applications in the cloud, *Journal of Network and Computer Applications, 59*, Pages 14-27.

Zhu, X., Yang, L.T., Chen, H., Wang, J., Yin, S. & Liu, X. (2014). Real-Time Tasks Oriented Energy-Aware Scheduling in Virtualized Clouds, *IEEE Transactions on Cloud Computing,* **2**(2): 168-180.