Journal of Engineering and Applied Science

RESEARCH Open Access

Black widow optimization algorithm for efficient task assignment in cloud computing

Huimin Wu[1*]

*Correspondence: wuhuiminny@163.com

[1] Henan Polytechnic Institute, Nanyang 473000, Henan, China

Abstract

Cloud computing was developed by blending virtualization and grid computing technologies. Its purpose is to provide Internet-based, on-demand, and consumption-based access to pools of computing resources in a measurable, adaptable, and scalable manner. Task scheduling is essential to cloud computing to ensure the performance of cloud services. However, inefficient scheduling can lead to resource issues such as under-allocation and over-allocation, which wastes resources and degrades service performance. Therefore, metaheuristic algorithms are incorporated into task scheduling systems to efficiently and timely distribute complex and diverse incoming tasks to limited resources. This study aims to analyze task priorities and precisely assign them to virtual machines. This is achieved by utilizing the Black Widow Optimization (BWO) algorithm. The primary objectives are to reduce time and energy consumption, improve task success rates, and optimize turnaround efficiency. Ultimately, these improvements aim to enhance the overall trustworthiness of the system.

Keywords: Cloud computing, Energy consumption, Task scheduling, Trust

Introduction

Today, small and medium-sized businesses (SMBs) are increasingly aware that they can access the best business applications and dramatically increase their infrastructure resources by leveraging the cloud [1]. Cloud computing allows these businesses to access high-performance computing resources at a fraction of the cost of traditional computing solutions [2]. By leveraging the cloud, SMBs can scale their business quickly, and cost-effectively and stay competitive. The term "cloud computing" refers to a style of computing in which computing power is made available over the Internet as a service to external customers [3]. Cloud computing encompasses multiple delivery models, namely Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), which are illustrated in Fig. 1. These services form the layered architecture of cloud computing [4]. At the infrastructure layer, the underlying compute resources, including storage, processing, and network, are standardized and provided across the network. Customers of cloud providers can utilize this infrastructure to deploy and manage operating systems and software. The middle layer, PaaS, offers services and abstractions that facilitate application development,

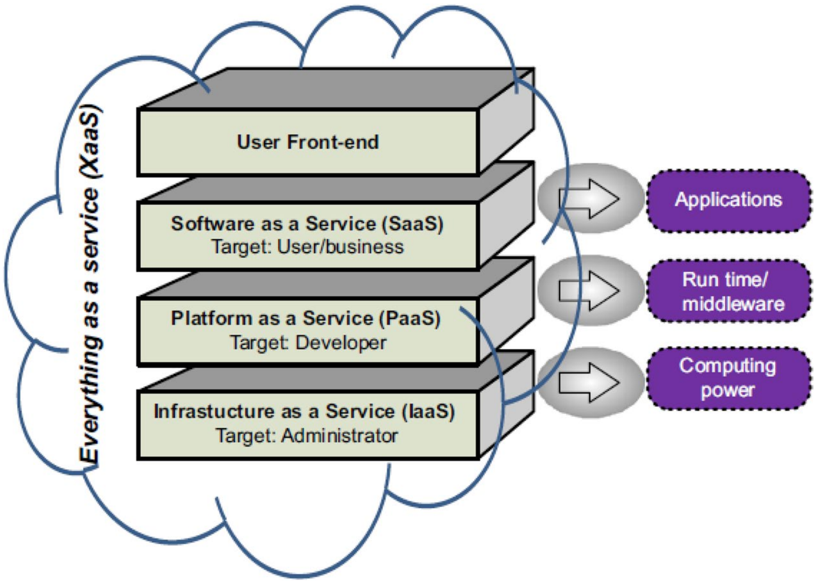
Everything as a service (XaaS)
User Front-end
Software as a Service (SaaS)
Target: User/business
Platform as a Service (PaaS)
Target: Developer
Infrastructure as a Service (IaaS)
Target: Administrator
Applications
Run time/ middleware
Computing power

Fig. 1 Cloud computing paradigm

testing, deployment, hosting, and maintenance. Finally, the application layer provides a comprehensive suite of SaaS applications. All the layers in the XaaS (Everything as a Service) model are seamlessly interconnected through the user interface layer [5].

In recent years, cloud computing has made it possible to arrange geographically dispersed data centers to provide high-quality services quickly. It has become a convenient way to provide computational resources pay-per-use [6]. Cloud computing paradigm bringing conformity and change to the IT industry. As cloud computing continues to grow in application and promotion, it presents many opportunities for advancing traditional information technology while posing a variety of challenges. The advent of cloud computing has revolutionized the way clients access and utilize resources [7]. Cloud providers can quickly provision and release resources without requiring extensive management or interaction by offering a shared pool of configurable assets [8]. This technology brings numerous advantages to the market, including improved time-to-market, cost savings, scalability, and storage efficiency. Applications can be deployed on virtual platforms, with resources efficiently distributed among virtual machines (VMs) [9]. This enables cloud providers to cater to multiple users and applications simultaneously, enhancing efficiency and cost-effectiveness. Furthermore, cloud computing facilitates seamless access to data and applications from any device at any time, enhancing its convenience and accessibility [10].

An effective and dynamic task scheduler is crucial to efficiently manage the simultaneous and diverse service requests originating from heterogeneous resources in a cloud environment [9]. The scheduler needs to adapt to the workload and allocate resources to cater to the needs of various cloud users. Inadequate scheduling mechanisms can significantly impact cloud service quality, leading to a loss of trust in the provider and potential business repercussions [11]. Therefore, in the cloud paradigm, it is imperative to employ a scheduler capable of dynamically scheduling tasks based on the regular influx of workloads to the console. Such a scheduler benefits cloud users and service providers by facilitating optimal resource utilization and maintaining a high level of service quality. This paper investigates the significance of trust-aware scheduling strategies concerning Service Level Agreement (SLA) criteria like response time, success rate, and availability. These indicators positively impact the quality of services and negatively influence power consumption and makespan. The study carefully prioritizes tasks and VMs, followed by precise mapping using the Black Widow Optimization (BWO) algorithm. The research findings reveal an underlying correlation between trust parameters and performance metrics, where improving trust parameters leads to reduced makespan and energy consumption. This research has made the following key contributions.

- This paper presents the development of the Trust-Aware BWO (TABWO) algorithm, which incorporates the trust factor in task scheduling. This algorithm allocates tasks to VMs based on their trustworthiness, improving overall performance and reliability.
- TABWO effectively schedules tasks and VMs by considering the electricity unit costs associated with them.
- The study highlights the significance of SLA parameters in evaluating the performance of cloud services, with a specific focus on the relationship between makespan and trust.
- A robust trust calculation mechanism has been established by incorporating SLA-related metrics such as the success rate of resources, VM availability, and response time, enabling an accurate evaluation of trustworthiness in the cloud environment.
- This study introduced a task assignment approach to meet the deadline constraint that prioritizes pending tasks after the current execution.
- Extensive simulations have been conducted using Cloudsim, a popular cloud computing simulator.
- By incorporating fabricated workload distributions (uniform, normal, left-skewed, and right-skewed) along with real-time workloads from HPC2N and NASA, the study encompasses a wide range of workload scenarios, allowing for a comprehensive evaluation of the proposed approach.

The rest of the paper is structured as follows: the "Methods" section provides an overview of existing research to establish the context for the proposed algorithm. The "Results" section discusses the algorithm's features and its effectiveness in addressing the challenges associated with task scheduling. The "Discussion" section provides an in-depth analysis and discussion of the findings. The "Conclusions" section concludes the study and suggests potential directions for future research and improvement in the field.

Related work

Workflow scheduling in cloud computing is challenged by inherent unreliability and uncertainty. Thus, service-oriented approaches should be considered when scheduling workflows. Tan, et al. [12] developed an algorithm for scheduling workflows based on trust services. A trust metric is used in the scheduling algorithm, combining direct trust and recommendation trust. A balanced policy was also provided that enabled users to balance different requirements, such as time, cost, and trust. The suggested algorithm was illustrated through a case study. The proposed approach has been proven feasible and effective through experiments. Rjoub, et al. [13] introduced a trust-aware scheduling approach involving three phases: computation of the trust level of VMs, determination of task priority levels, and scheduling based on trust metrics. Experiments have demonstrated that the proposed approach outperforms PSO, RR, and SJF approaches regarding makespan and the monetary cost under untrusted VM conditions.

Ali, et al. [14] introduce a multi-level trust improvement strategy to enhance task scheduling in mobile cloud environments. The strategy involves several steps to ensure effective task offloading and scheduling. First, the trustworthiness of tasks suitable for offloading to the mobile cloud is calculated. This step determines which tasks can benefit from being executed in the mobile cloud environment. To further enhance task scheduling, the researchers propose the integration of an efficient and dynamic scheduler. This scheduler incorporates trust computation methods for social and environmental contexts to improve trust-based decision-making. By considering factors such as social interactions and environmental conditions, the scheduler optimizes the allocation of tasks to resources. The method also includes energy and time request analysis, which are evaluated against existing methods. This comparison is conducted to enhance the efficiency of these computations and make them more effective in the context of task scheduling. The proposed strategy utilizes mobile cloud computing to handle the continuous updates of trust values from incoming devices. This allows for constant synchronization of trust values, ensuring up-to-date information is available for decision-making. Moreover, the authors highlight the advantages of employing a centralized data distribution method in the mobile cloud computing environment. This method leverages the benefits of mobile cloud computing to distribute and manage data across the system efficiently.

Govindaraj and Natarajan [15] propose a novel trust-based fruit fly optimization algorithm (TFOA) for cloud task scheduling. The objective of the TFOA algorithm is to enable cloud service providers to receive customer tasks faster and allocate them to the most trustworthy resources available. By incorporating trust as a key factor in the scheduling process, the algorithm aims to enhance task allocation's overall efficiency and effectiveness. To evaluate the performance of TFOA, the authors compare it with existing transitional algorithms such as round robin and particle swarm optimization (PSO). The results demonstrate that TFOA outperforms these algorithms regarding reduced turnaround time and efficient utilization of cloud resources. By considering trust as a crucial factor in resource allocation decisions, TFOA provides improved performance and resource utilization, leading to enhanced overall scheduling efficiency in cloud environments.

Ebadifard, et al. [16] present an effective method for estimating the trustworthiness of VMs that run workflows. Finding the optimal Pareto front becomes more difficult as the number of requests increases, the diversity of VMs expands, and the conflict between objectives grows. Thus, multi-objective evolutionary algorithms face many permutations to identify an optimal trade-off between the objectives. A multi-objective workflow-scheduling algorithm is presented in this paper using a multiverse optimizer algorithm to increase diversity and convergence to take into account the QoS requirements for service providers and customers simultaneously.

Kaur and Auluck [17] present a real-time trust-aware dynamic scheduling framework specifically designed for fog computing environments. This framework considers important factors such as privacy, trust, and real-time performance. The authors propose a trust computation model that enables the calculation of the trustworthiness of fog devices. This model incorporates both direct and recommended trust techniques for each fog device. The aggregated trust values of fog devices are periodically updated based on these trust computations. Tasks submitted by users are categorized into three types: private, semi-private, and public.

Furthermore, fog devices are classified into different trust levels: highly trusted, extremely highly trusted, normal trusted, low trusted, and untrusted. The proposed algorithm aims to improve real-time performance by mapping input jobs to trustworthy fog devices, considering the privacy constraints associated with each task. This mapping process increases the overall success ratio of the scheduling algorithm.

Table 1 summarizes trust-aware task scheduling methods. These methods are designed to optimize task scheduling taking into account trust-aware factors such as trust levels, reputation, and trustworthiness. They can also be used to mitigate the effects of malicious actors such as attackers or untrustworthy peers. Finally, they can be used to improve the overall trustworthiness of a task-scheduling system.

Table 1 Trust-aware task scheduling methods

Reference	Approach	Key features
Tan, et al. [12]	Trust-based workflow scheduling	• Trust metric combining direct and recommendation trust • Balance policy to consider time, cost, and trust
Rjoub, et al. [13]	Trust-aware scheduling	• Computation of trust level for VMs • Task priority determination based on trust metrics • Outperforms PSO, RR, and SJF in makespan and monetary cost under untrusted VM conditions
Ali, et al. [14]	Multi-level trust improvement strategy for mobile cloud	• Trust calculation for offloaded tasks • Dynamic scheduler based on social and environmental trust
Govindaraj and Natarajan [15]	Trust-based fruit fly optimization algorithm for task scheduling	• Trust-based scheduling using a novel TFOA algorithm • Improved turnaround time and resource utilization compared to round-robin and PSO algorithms
Ebadifard, et al. [16]	Multi-objective workflow-scheduling algorithm	• Estimation of VM trustworthiness • Multi-objective evolutionary algorithm with multiverse optimizer
Kaur and Auluck [17]	Real-time trust-aware dynamic scheduling framework	• Trust computation model for fog devices • Categorization of tasks based on privacy • Mapping of tasks to trustworthy fog devices

Methods

Problem definition

In our scenario, this study has a set of tasks denoted as $T=[T_1, T_2, T_3, \ldots, T_n]$, and these tasks need to be assigned to a set of virtual resources represented as $R=[R_1, R_2, R_3, \ldots, R_n]$. These resources are hosted on a set of hosts denoted as $H=[H_1, H_2, H_3, \ldots, H_n]$, which are located in a set of data centers referred to as $D=[D_1, D_2, D_3, \ldots, D_n]$. This research focuses on assigning T_n tasks to R_n VMs, which are then allocated to D_n datacenters. This allocation process considers the priorities of both the tasks and VMs, considering electricity unit costs. The main objective of this problem is to minimize the makespan, which represents the total execution time, while simultaneously enhancing the SLA-based trust parameters. These trust parameters include turnaround efficiency, success rate, and availability. By optimizing the task allocation and scheduling process, This study aims to improve these trust parameters, ensuring that the assigned tasks are executed efficiently and meet the required SLA criteria.

System architecture

As shown in Fig. 2, the architecture effectively illustrates the sequential progression of tasks and the interaction of users within the system. Initially, cloud users from heterogeneous resources submit their requests through the cloud console. To facilitate the process, a cloud broker acts as an intermediary and captures these requests, subsequently forwarding them to the task manager. The task manager then assesses the validity of the requests by comparing them against the SLA agreements. Once validated, the service request dispatcher takes charge of dispatching the approved requests to the scheduler. Task length, run time, and capacity determine task priorities.

Moreover, our system incorporates the calculation of VM priorities based on their electricity costs. By considering this factor, This study determines the relative importance of each VM. Subsequently, the tasks are placed in the execution queue, and the

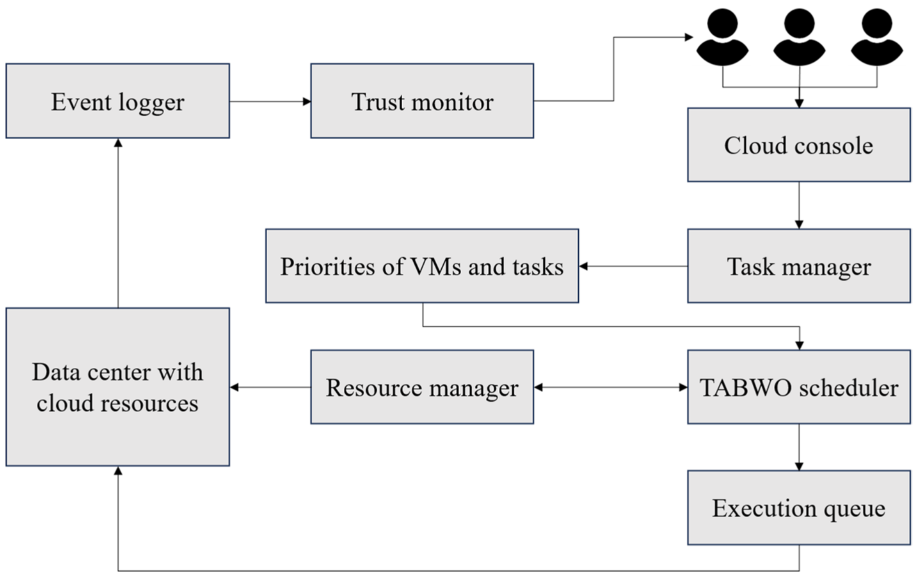

Fig. 2 System architecture

scheduler takes responsibility for assigning them to the most suitable VMs. During the assignment process, the scheduler takes into account the consumption status of virtual resources at the resource manager level. This approach aims to minimize the makespan (the duration of task execution) and enhance trust parameters based on SLA requirements.

To optimize task scheduling, a crucial deadline constraint is enforced. This constraint guarantees that while a task is being executed on a VM, no other tasks are scheduled on the same VM. By adhering to this constraint, the system maintains the integrity of task execution and maximizes resource utilization. Equation 1 is utilized to calculate the workload of all VMs, enabling us to gain insights into the current workload status. This calculation provides a comprehensive understanding of the distribution of tasks across the VMs. Additionally, Eq. 2 is employed to determine the workload on all physical hosts, taking into account that VMs are hosted on these hosts. This assessment allows us to gauge the overall workload of the system and make informed decisions regarding resource allocation and load balancing.

$$\text{load}_{nv} = \sum \text{load}_{v} \tag{1}$$

$$\text{load}_{ho_q} = \frac{\text{load}_{nv}}{ho_q} \tag{2}$$

The Eq. 1 calculates the total workload across all virtual machines (load_{nv}) by summing up the individual workloads of each virtual machine (load_{v}). It provides insights into the current workload status across the entire virtualized environment.

In Eq. 2, this equation determines the workload on a specific physical host (load_{ho_q}), represented by the subscript q. It divides the total workload across all virtual machines (load_{nv}) by the number of hosts (ho_q), providing a measure of the overall workload distribution among physical hosts. The ho_q represents a specific physical host, denoted by the subscript q. This notation is used to distinguish between different physical hosts within the cloud computing environment. Each physical host is assigned a unique identifier or label, which is represented by the subscript q in this context.

Moving forward, the subsequent step entails the mapping of tasks onto VMs. To facilitate this mapping process, task priorities are established, considering both the length of the tasks and the processing capacities of the VMs. Equation 3 is employed to calculate the processing capacity of an individual VM, which plays a crucial role in determining task priorities and optimizing the scheduling process. Furthermore, to obtain a holistic view of the system's processing capacity, Eq. 4 is utilized to compute the combined processing capacity of all VMs. Task sizes are determined by Eq. 5, which is then used to calculate task priorities using Eq. 6. In Eq. 6, the processing capacity of VMs and the size of the corresponding task are considered.

$$pr_{n_v} = pr_{no} * pr_{MIPS} \tag{3}$$

$$ToT^{pr}_{n_v} = pr_{nv} \tag{4}$$

$$m_t^{size} = m_t^{MIPS} * m_t^{pr} \tag{5}$$

$$prio_{m_t} = \frac{m_t^{size}}{pr_{n_v}} \tag{6}$$

Equation 3 calculates the processing capacity of an individual VM (pr_{n_v}) by multiplying the priority of the VM (pr_{no}) with its MIPS (million instructions per second) capacity($\mathrm{pr}_{\mathrm{MIPS}}$). It is used to determine task priorities based on the processing capabilities of VMs. In Eq. 4, it represents the total processing capacity of all virtual machines ($\mathrm{ToT}_{n_v}^{pr}$), which is equal to the processing capacity of an individual VM (pr_{nv}). It provides an overview of the system's processing capacity for task scheduling purposes. In Eq. 5, this equation calculates the size (m_t^{size}) of a task (m_t) by multiplying its MIPS requirement (m tMIPS) with its priority (m_t^{MIPS}). It helps determine task priorities based on their resource requirements. In Eq. 6, it calculates the priority (prio_{m_t}) of a task (m_t) by dividing its size (m_t^{size}) by the processing capacity of the corresponding VM (pr_{n_v}). It is used to prioritize tasks for scheduling based on their resource needs. Equation 7 calculates the electricity cost-based priorities. This equation compares the electricity cost at each data center with the highest electricity cost among all data centers. The priority is determined by taking the rate of the highest electricity cost to the electricity cost at the corresponding data center. By incorporating electricity cost-based priorities for VMs, resource allocation and workload balancing can be optimized. VMs with lower electricity costs are assigned higher priorities, indicating their suitability for task execution and promoting resource utilization and cost-effectiveness in the cloud environment.

$$\mathrm{prio}_{\mathrm{nv}} = \frac{\mathrm{high}_{\mathrm{ele}_{\mathrm{cost}}}^{\mathrm{DC}}}{\mathrm{DC}_{\mathrm{ele}_{\mathrm{cost}}}} \tag{7}$$

The Eq. 7 calculates the electricity cost-based priority ($\mathrm{prio}_{\mathrm{nv}}$) for a virtual machine (nv). It compares the electricity cost at each data center ($\mathrm{DC}_{\mathrm{ele}_{\mathrm{cost}}}$) with the highest electricity cost among all data centers ($\mathrm{high}_{\mathrm{ele}_{\mathrm{cost}}}^{\mathrm{DC}}$). VMs with lower electricity costs are assigned higher priorities, optimizing resource allocation and workload balancing based on cost-effectiveness.

After determining the priorities of tasks and VMs, the task manager forwards these priorities to the scheduler for schedule generation. However, before generating schedules, the scheduler needs to communicate with the resource manager to check the status of virtual resources, specifically whether the VMs are currently busy or idle. In addition to task and VM priorities, the scheduler considers SLA-based trust parameters crucial for maintaining the quality of service provided by the cloud provider. These parameters, including availability, success rate, and turnaround efficiency, are significant in establishing trust between providers and users. Violations of these parameters can erode the trust in the cloud provider. Equation 8 is utilized to calculate a VM's availability. The availability metric is the ratio of the number of tasks accepted by a VM to the total number of tasks submitted by the user. This metric provides insights into the reliability and responsiveness of the VM in handling user tasks.

By monitoring and optimizing VM availability, the cloud provider can ensure a reliable and efficient execution environment for tasks.

$$\mathrm{AV}(n_v) = \frac{\mathrm{AV}_m}{m_t} \tag{8}$$

The Eq. 8 calculates the availability ($\mathrm{AV}(n_v)$) of a virtual machine (n_v) as the ratio of the number of tasks accepted by the VM (AV_m) to the total number of tasks submitted by the user (m_t). It measures the reliability and responsiveness of the VM in handling user tasks, aiding in monitoring and optimizing VM availability for efficient task execution.

The success rate is a significant SLA-based trust parameter that evaluates the performance and reliability of a cloud provider. It measures the proportion of successfully executed requests or tasks on a VM out of the total number of requests submitted within a specific timeframe. The success rate can be calculated using the following equation:

$$\mathrm{SR}(n_v) = \frac{s_{m_t}}{\mathrm{AV}_m} \tag{9}$$

The Eq. 9 calculates the success rate ($SR(n_v)$) of a virtual machine (n_v) as the proportion of successfully executed tasks (s_{m_t}) out of the total number of tasks submitted to the VM (AV_m). It evaluates the performance and reliability of the cloud provider by measuring the VM's ability to execute tasks successfully within a specific timeframe.

The turnaround efficiency is another important SLA-based trust parameter that measures the effectiveness and efficiency of a VM in terms of task scheduling and execution. It quantifies the ratio of the estimated turnaround time (ESTT) provided by the cloud provider to the actual turnaround time (ACTT) experienced during task scheduling and execution. The turnaround efficiency can be calculated using the following equation:

$$\mathrm{TE}(n_v) = \frac{\mathrm{EST}_T}{\mathrm{ACT}_T} \tag{10}$$

The Eq. 10 calculates the turnaround efficiency $\mathrm{TE}(n_v)$ of a virtual machine (n_v) as the ratio of the estimated turnaround time (EST_T) provided by the cloud provider to the actual turnaround time experienced during task scheduling and execution (ACT_T). It assesses the accuracy of the cloud provider's estimated turnaround time and the efficiency of task scheduling and execution on the VM.

The estimated turnaround time is the expected time the cloud provider provides for a task to be scheduled and executed on the VM. It is typically based on the resources allocated to the task and the expected workload. On the other hand, the actual turnaround time represents the real-time duration taken for the task to be executed on the VM, including any delays or unexpected events. By calculating the turnaround efficiency, This study can assess how well the cloud provider's estimated turnaround time aligns with the time taken for task execution. A higher turnaround efficiency indicates that the estimated turnaround time is close to the actual time, reflecting accurate predictions and efficient scheduling. This parameter is crucial for users to evaluate the reliability and performance of a cloud provider in meeting the promised turnaround time for task execution.

Equation 11 represents the trust calculation in a cloud service provider based on the SLA-based trust parameters. The equation involves positive weights assigned to the parameters *X1*, *X2*, and *X3*, which are used to calculate the cloud service provider's overall trust value (*X*). The weights assigned to these parameters determine their relative importance in the trust calculation process. The specific values of the weights can vary depending on the context and requirements of the cloud service. In the given example, *X1* is assigned a weight of 0.5, *X2* is assigned a weight of 0.2, and *X3* is assigned a weight of 0.1. These weights can be adjusted based on the cloud service's and its users' specific needs and priorities. By plugging in the values of *X1*, *X2*, and *X3*, along with the calculated values of the SLA-based trust parameters (availability, success rate, and turnaround efficiency), Eq. 11 allows for the computation of the overall trust value in the cloud service provider. The resulting trust value will fall between 0 and 1, where a higher value indicates a higher level of trust in the cloud service provider.

$$\text{trust}^{\text{csp}} = X1 * AV(n_v) + X2 * SR(n_v) * X3 * \text{TE}(n_v) \tag{11}$$

In Eq. 11, the $\text{trust}^{\text{csp}}$ represents the overall trust value of the cloud service provider. *X1*, *X2*, and *X3* represent positive weights assigned to different SLA-based trust parameters (availability, success rate, turnaround efficiency), determining their relative importance in trust calculation. $AV(n_v)$ depicts the availability of a VM, calculated as the ratio of accepted tasks to total submitted tasks, reflecting the reliability and responsiveness of the VM. The $\text{SR}(n_v)$ represents the success rate of a VM, representing the proportion of successfully executed tasks out of the total submitted tasks, crucial for evaluating the performance and reliability of the cloud provider. The $\text{TE}(n_v)$ represents the turnaround efficiency of a VM, indicating the ratio of estimated turnaround time to actual turnaround time, measuring the accuracy of the cloud provider's predictions and scheduling efficiency.

Proposed trust-aware task scheduling algorithm

The proposed TABWO algorithm introduces a novel approach to task scheduling in cloud computing environments. The methodology of TABWO is built upon the foundation of the BWO algorithm, known for its efficacy in addressing NP-hard problems through efficient exploration of solution spaces. TABWO extends the capabilities of BWO by incorporating trust awareness into the task-scheduling process. This trust awareness encompasses considerations of SLA-based trust parameters, such as availability, success rate, and turnaround efficiency, which are vital for evaluating the reliability and performance of cloud service providers. By integrating trust-related factors into the optimization process, TABWO aims to enhance the overall performance and reliability of task scheduling in cloud computing environments. The algorithm seeks to achieve improved efficiency, resource utilization, and trustworthiness by combining optimization objectives with trust considerations.

The TABWO model comprises several phases aimed at optimizing task scheduling in cloud computing environments. Firstly, the population of spiders is initialized to represent potential solutions to the task scheduling optimization problem. Each spider corresponds to an individual solution in the solution space, with floating-point variables defining their characteristics. These characteristics, including position and fitness, are crucial for evaluating the quality of solutions. Through the mating process, spiders reproduce offspring that inherit certain characteristics from their parents, facilitating the generation of new solutions. Additionally, the algorithm incorporates cannibalism mechanisms inspired by the behavior of black widow spiders, promoting the survival of stronger solutions and the elimination of weaker ones. Furthermore, the mutation component of TABWO employs an adaptive scheme to alter mutation rates dynamically, ensuring effective exploration of the solution space. By leveraging these phases and mechanisms, TABWO can effectively optimize task scheduling in cloud computing environments, considering both optimization objectives and trust-related factors for informed scheduling decisions.

The BWO algorithm, with its inherent optimization capabilities, is a suitable choice for addressing this problem by finding near-optimal solutions [18]. By utilizing the BWO algorithm as the foundation, the proposed TABWO algorithm incorporates trust awareness into the task scheduling process. In addition to optimizing task allocation and resource utilization, the algorithm considers trust-related factors, such as SLA-based trust parameters discussed earlier, to make informed scheduling decisions. The use of TABWO aims to enhance the overall performance and reliability of task scheduling in cloud computing environments by considering both optimization objectives and trust considerations. By combining the capabilities of BWO with trust awareness, the algorithm strives to achieve improved task scheduling results in terms of efficiency, resource utilization, and trustworthiness.

In the BWO algorithm, the population of spiders is initialized to represent potential solutions to the optimization problem. Each spider corresponds to an individual solution in the d-dimensional solution space. The population size is denoted as NP, and the candidate matrix is represented by an *NP * d* array, as defined in Eq. 12.

$$\text{Black window} = [x_1, x_2, \ldots, x_d] \tag{12}$$

The array contains floating-point variables that define the characteristics of each spider in the population. The values of these variables determine the attributes of each solution, such as position, fitness, or other relevant parameters. The initial population is generated randomly to explore the solution space effectively, as indicated in Eq. 13. Initializing the population sets the foundation for the optimization process in BWO. By creating a diverse set of initial solutions, the algorithm can explore different regions of the solution space and increase the chances of finding promising solutions. The subsequent steps of the BWO algorithm, such as mating and selection, build upon this initial population to iteratively improve the quality of the solutions and converge towards optimal or near-optimal solutions.

$$\text{Black window} = xl + \text{rand}(1, d). * (xu - xl) \tag{13}$$

In the BWO algorithm, the mating process occurs independently for each pair of spiders in the population. This process simulates the natural reproduction of black widow spiders in their webs. As each pair of spiders' mates independently of the rest of the web, in the BWO algorithm, each pair mates to produce offspring. A new generation of spiders is created during mating by reproducing an alpha array using arbitrary numbers. The parents are denoted as u_1 and u_2, while the children are represented as v_1 and v_2. The mating process combines the genetic information of the parents to generate offspring that inherit certain characteristics from each parent.

$$\left\{ \begin{array}{l} v_1 = a \times u_1 + (1 - \alpha) \times u_2 \\ v_2 = a \times u_2 + (1 - \alpha) \times u_1 \end{array} \right\} \tag{14}$$

In the context of the BWO algorithm, cannibalism refers to a mechanism inspired by the behavior of black widow spiders. The algorithm incorporates three types of cannibalism: female cannibalism, sibling cannibalism, and baby spider cannibalism. Female black widow spiders eat male black widow spiders after or during mating. In the BWO algorithm, fitness values are used to identify female and male spiders. Female spiders, representing better solutions, have higher fitness values. The algorithm incorporates this concept by allowing female spiders to consume male spiders, thereby eliminating weaker solutions. Sibling cannibalism occurs when stronger spiderlings (baby spiders) eat weaker spiderlings. This behavior is mimicked in the BWO algorithm to promote the survival of stronger solutions. Spiderlings with higher fitness values, indicating better solutions, are more likely to survive and reproduce, while weaker spiderlings may be cannibalized. In some cases, baby spiders may consume their mother. In the BWO algorithm, this behavior is simulated by evaluating the fitness values of spiderlings. Weak or strong spiderlings are assessed based on their fitness values, and this information may influence their survival and reproduction.

The mutation is another important component of the BWO algorithm. It is based on the simulated binary crossover (SBC), which produces new individuals or chromosomes at constant crossover and mutation rates. However, a permanent mutation rate may not lead to the optimal solution for the optimization problem. An adaptive scheme is employed to alter the mutation rate to address this. The proposed algorithm combines three crossover operators (single point crossover, uniform crossover, and SBC) to generate new individuals. The mutation rate is updated using a linear function and is adjusted based on the value of p_s, calculated using Eq. 16.

$$m_r = \frac{p_s}{L} \tag{15}$$

$$p_s = P + (t - 1) \times \frac{1 - P}{t_M - 1} \tag{16}$$

In Eq. 16, t_M represents the maximum generation, a predefined value indicating the total number of generations in the algorithm. The variable t represents the current generation number during the optimization process. The value of P, as given by Eq. 17, is a fixed real number used in calculating the mutation rate.

$$P_0 = \frac{L}{50} \tag{17}$$

Results

In this section, this study provides details about the simulations conducted in this study. The simulations were carried out using the Cloudsim simulator, a Java-based framework for simulating cloud computing environments. This powerful tool allowed them to model and analyze the behavior of various components within the cloud infrastructure. This study employed a combination of fabricated datasets with different distributions and real-time workloads obtained from sources like HPC2N and NASA to generate realistic workloads. Utilizing these datasets ensured that the simulated workload closely resembled real-world scenarios encountered in cloud computing environments. This study generated workload datasets by fabricating datasets with different distributions. These distributions included normal, uniform, right-skewed, and left-skewed distributions. These fabricated datasets were labeled S01, S02, S03, and S04, respectively. The purpose of using different distributions for workload generation is to simulate various scenarios and test the performance of the proposed scheduler under different workload conditions. Each distribution represents a different pattern of task arrival and resource requirements, allowing one to assess the scheduler's ability to handle different workload characteristics.

During the simulations, This study focused on evaluating several important performance metrics. These metrics included SLA-based trust parameters, such as availability, success rate, and turnaround efficiency, which play a significant role in assessing the trustworthiness of a cloud service provider. Additionally, This study measured the makespan, which represents the total time taken to complete all tasks, and energy consumption, which indicates the amount of energy consumed during the execution of tasks. To assess the effectiveness of the proposed approach, This study compared it against baseline approaches, namely the ACO, GA, and PSO algorithms. These algorithms are commonly used in optimization problems, including task scheduling in cloud computing. To determine the advantages and drawbacks of each approach, it needs to subject both the baseline approaches and the proposed approach to the same workload and performance metrics and conduct a comparative analysis.

This study conducted simulations to calculate the makespan of the proposed scheduler. The makespan is an important metric in the design of a scheduler, as it directly affects the effectiveness and performance of the scheduler. Minimizing the makespan leads to improved scheduler performance. This study conducted simulations with varying tasks ranging from 100 to 1000 to calculate the makespan. They ran the simulations for 50 iterations to ensure reliable and consistent results. The simulation settings used were based on the specifications provided in Table 2. This study utilized different workloads, labeled S01, S02, S03, and S04, as input for the TABWO scheduler. These workloads were generated or obtained from external sources. By applying the proposed approach and the baseline algorithms to these workloads, This study could compare the scheduler's performance in terms of the generated makespan. Figure 3 shows that the proposed TABWO scheduler consistently outperforms the baseline approaches regarding generated makespan across the 50 iterations and for the considered range of tasks. This comparison highlights the superiority of the TABWO scheduler in minimizing the makespan and demonstrates its effectiveness in improving scheduler performance compared to the baseline algorithms.

This study calculated the energy consumption of the TABWO scheduler as an important parameter for evaluating its performance. Energy consumption is a crucial aspect to consider from the perspective of the cloud consumer and service providers' perspectives. To calculate energy consumption, the researchers conducted

Table 2 Simulation parameters

Parameter	Value
Number of data centers	5
Hypervisor type	Monolithic
Hypervisor name	Xen
Number of processing elements	1050 MIPS
Bandwidth capacity	100 Mbps
Memory capacity	1 GB
Number of VMs	30
Task length	800,000
Number of tasks	100–1000

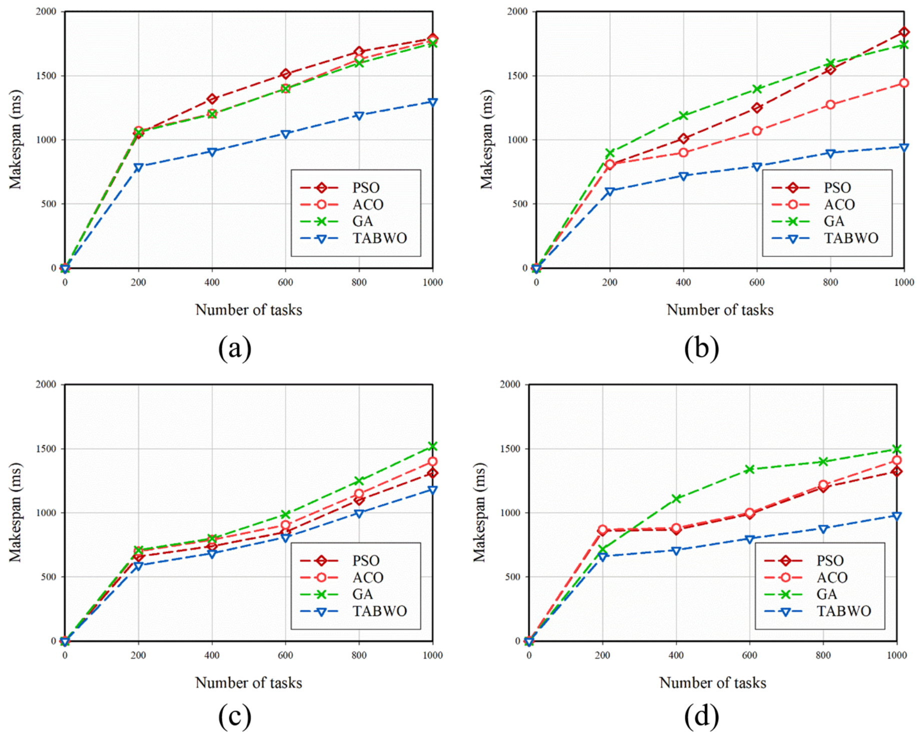

Fig. 3 Makespan comparison **a** normal distribution, **b** uniform distribution, **c** right-skewed distribution, **d** left-skewed distribution

simulations by varying the number of tasks from 100 to 1000 and running 50 iterations. The efficiency of the scheduler in terms of energy usage can be assessed by comparing its energy consumption results with the baseline algorithms. The objective was to minimize energy consumption, which benefits the cloud provider in reducing power bills and the cloud consumer accessing services at a lower cost.

Additionally, minimizing energy consumption also has environmental benefits. The results of the simulations, including the generated energy consumption values for different workloads and the corresponding comparison with baseline approaches, are presented in Fig. 4, respectively. These results show that the proposed TABWO scheduler outperforms the baseline algorithms in terms of energy consumption across the 50 iterations and for the considered tasks.

The success rate is a critical metric in evaluating the efficiency and reliability of a cloud service provider. It represents the percentage of successfully executed requests onto VMs, indicating how well the scheduler completes tasks and meets SLAs. To calculate the success rate, This study conducted extensive simulations by varying the number of tasks from 100 to 1000 and running 50 iterations. The primary objective was to improve the success rate percentage, which has several benefits for the cloud service provider and consumers. For the cloud service provider, a higher success rate indicates a better quality of service, improved reliability, and more efficient resource utilization. This, in turn, enhances consumers' trust in the cloud provider's ability to deliver services as per SLAs. For cloud consumers, a higher success rate means a more reliable and consistent performance of their applications, leading to increased satisfaction with the cloud service. The results of the simulations, including the generated success rate values for different workloads and the comparison with the baseline algorithms, are presented in Fig. 5, respectively. These results show that the proposed TABWO scheduler consistently outperforms the baseline algorithms regarding the success rate across the 50 iterations and for the considered tasks.

On the other hand, one notable aspect to consider in this comparison is the methodology employed by each approach. The TABWO leverages a trust-aware optimization strategy, which incorporates trust parameters such as availability, success rate, and turnaround efficiency. This novel approach not only focuses on optimizing traditional

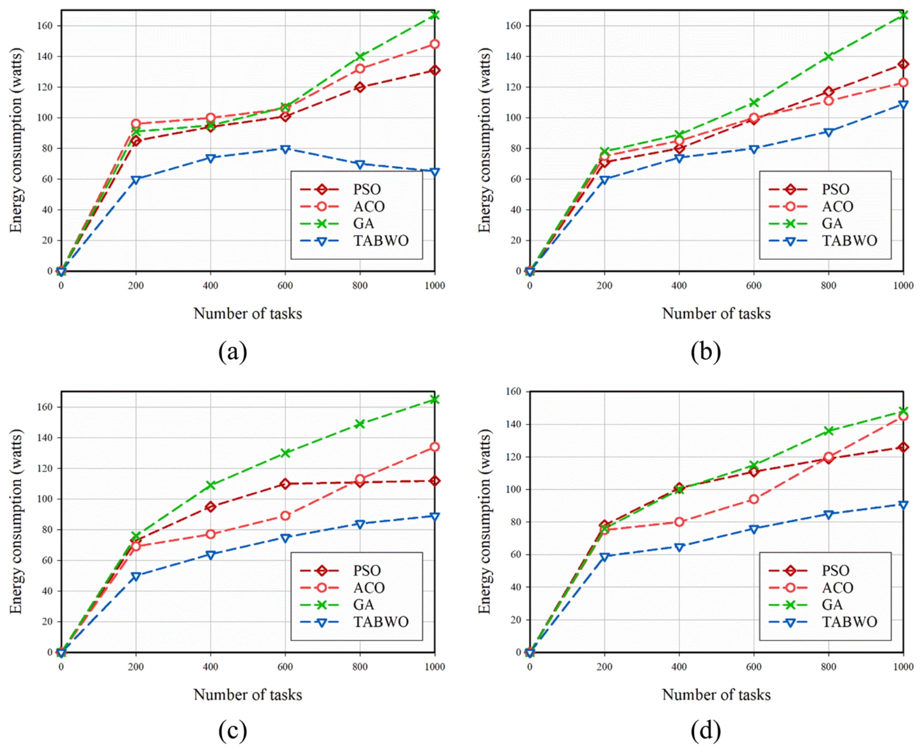

Fig. 4 Energy consumption comparison **a** normal distribution, **b** uniform distribution, **c** right-skewed distribution, **d** left-skewed distribution

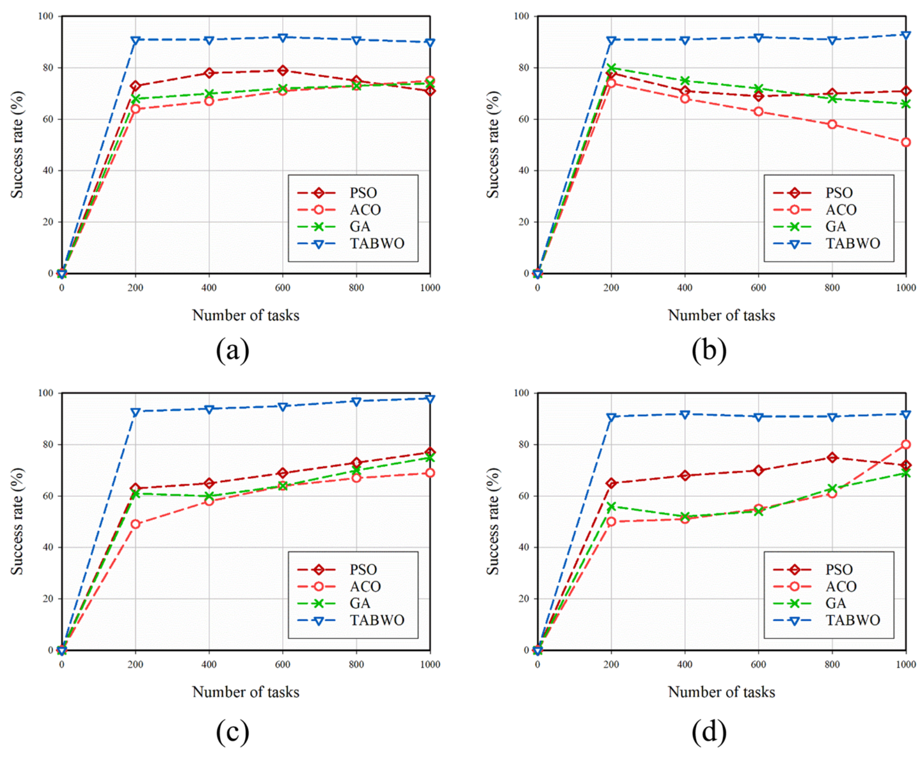

Fig. 5 Success rate comparison **a** normal distribution, **b** uniform distribution, **c** right-skewed distribution, **d** left-skewed distribution

metrics like makespan and energy consumption but also emphasizes the trustworthiness of the cloud service provider. In contrast, ACO, GA, and PSO algorithms typically prioritize optimization without explicitly considering trust aspects. While these traditional algorithms are effective in optimizing certain performance metrics, they may fall short in ensuring the reliability and trustworthiness of cloud services.

Furthermore, the complexity and time efficiency of the scheduling algorithms play a crucial role in practical deployment scenarios. TABWO introduces a unique binary whale optimization technique, which offers a balance between exploration and exploitation phases, resulting in improved convergence speed and solution quality. This characteristic is particularly advantageous in dynamic cloud environments where rapid adaptation to changing workloads is essential. On the other hand, ACO, GA, and PSO algorithms may suffer from scalability issues or longer convergence times, especially when dealing with large-scale task scheduling problems. Additionally, the preprocessing steps involved in each approach contribute to their overall complexity. TABWO's integration of trust parameters into the optimization process adds a layer of complexity compared to traditional algorithms. However, this additional complexity is justified by the enhanced trustworthiness and reliability achieved in task scheduling, which ultimately benefits both cloud providers and consumers.

Another aspect worth considering is the adaptability of the scheduling algorithms to different workload characteristics and distribution patterns. TABWO demonstrates superior adaptability by utilizing fabricated datasets with various distributions, including normal, uniform, right-skewed, and left-skewed distributions. This comprehensive evaluation allows for a thorough assessment of the scheduler's performance under diverse workload conditions, ensuring its robustness and versatility. In contrast, while ACO, GA, and PSO algorithms are widely applicable and effective in many optimization problems, their performance may vary depending on the specific characteristics of the workload. Without explicit consideration of trust parameters and diverse workload distributions, these traditional algorithms may struggle to achieve consistent and reliable performance across various scenarios. Therefore, the qualitative comparison highlights TABWO's ability to outperform existing methods not only in optimizing traditional metrics like makespan and energy consumption but also in addressing the trustworthiness and adaptability requirements of cloud computing environments.

During the simulations, the trustworthiness of VMs was assessed based on SLA-based trust parameters, which include availability, success rate, and turnaround efficiency. These parameters play a crucial role in determining the reliability and performance of a cloud service provider. Here are some clarifications to demonstrate the trustworthiness of VMs during the simulations:

Availability

The availability of VMs was calculated as the ratio of the number of tasks accepted by a VM to the total number of tasks submitted by the user. A higher availability metric indicates that the VM is reliably handling user tasks and responding promptly to requests. By monitoring VM availability, the cloud provider can ensure a dependable execution environment for tasks, thus enhancing trust in its services.

Success rate

The success rate represents the proportion of successfully executed requests or tasks on a VM out of the total number of requests submitted within a specific timeframe. A higher success rate signifies that the scheduler effectively completes tasks and meets SLAs. This metric reflects the reliability and efficiency of VMs in executing tasks, contributing to increased trust in the cloud provider's service quality.

Turnaround efficiency

Turnaround efficiency measures the effectiveness of VMs in terms of task scheduling and execution by quantifying the ratio of the estimated turnaround time to the actual turnaround time. A higher turnaround efficiency indicates that the cloud provider's estimated turnaround time aligns well with the actual execution time, reflecting accurate predictions and efficient scheduling. This parameter is crucial for evaluating the reliability and performance of VMs in meeting promised turnaround times, thus influencing trust in the cloud provider.

By evaluating these SLA-based trust parameters for VMs during the simulations, This study gains insights into the reliability, responsiveness, and efficiency of VMs in handling tasks. Consistently high levels of availability, success rate, and turnaround efficiency demonstrate the trustworthiness of VMs and contribute to building trust between cloud providers and consumers. These metrics provide assurance to consumers regarding the quality of service provided by the cloud provider, leading to enhanced trust and satisfaction with cloud services.

Discussions

The proposed trust-aware task scheduling algorithm, TABWO, introduces a novel approach to addressing the intricate challenges of task scheduling within cloud computing environments. One of its primary advantages lies in its ability to consistently outperform baseline algorithms, as demonstrated through metrics such as makespan, energy consumption, and success rate. By integrating trust awareness into the scheduling process, TABWO makes more informed decisions, leading to enhanced performance and efficiency in task execution. This improvement is particularly significant in complex cloud environments where optimizing resource allocation is paramount for meeting service level agreements (SLAs) and ensuring user satisfaction. TABWO achieves this by dynamically assigning tasks to virtual machines (VMs) based on their processing capacities and workload distribution, thus optimizing resource utilization and minimizing wastage.

Furthermore, the trust-aware approach of TABWO contributes to the enhancement of reliability within cloud computing systems. By considering SLA-based trust parameters such as availability, success rate, and turnaround efficiency, the algorithm ensures a more dependable and consistent performance of VMs. This reliability fosters trust between cloud providers and consumers, consequently elevating the overall quality of service. The algorithm's ability to optimize resource allocation while simultaneously prioritizing trust-related factors ensures that critical tasks are allocated to the most reliable VMs, thereby minimizing the risk of service disruptions and enhancing the user experience.

However, despite its numerous advantages, the implementation of TABWO also presents certain limitations. Chief among these is the inherent complexity associated with trust-aware scheduling algorithms. Implementing TABWO requires a deep understanding of cloud computing architectures, SLA parameters, and optimization techniques, which may pose challenges for users lacking expertise in these domains. Additionally, the computational overhead introduced by the algorithm's trust assessment and decision-making processes may impact its scalability, particularly in large-scale cloud environments with a high volume of tasks and VMs.

Another limitation of TABWO stems from its heavy reliance on SLA parameters for making scheduling decisions. Inaccurate or unreliable SLA parameters, influenced by external factors such as network latency or VM failures, may compromise the effectiveness of the algorithm, leading to suboptimal performance. Moreover, like many optimization algorithms, TABWO requires careful tuning of parameters and heuristics to achieve optimal performance across diverse workload scenarios. Finding the right balance of parameters and ensuring robustness in varied environments can be challenging and time-consuming, potentially hindering the algorithm's widespread adoption.

As a result, while TABWO offers significant advantages in improving task scheduling efficiency and reliability in cloud computing, its implementation and deployment require careful consideration of its inherent complexities and limitations. Through ongoing research and development efforts focused on mitigating these challenges, TABWO has the potential to emerge as a powerful tool for optimizing resource allocation, enhancing reliability, and fostering trust within cloud computing ecosystems.

Conclusions

Task scheduling is critical to cloud computing as it involves managing virtualized resources. With clients utilizing numerous virtual assets per task, manual scheduling is impractical. Task scheduling is primarily intended to enhance performance by minimizing time loss. However, in cloud computing environments, scheduling becomes challenging due to task dependencies, resource heterogeneity, and high computational intensity. One of the most complex challenges is determining the execution time and trustworthiness of VMs used for task execution. Hence, it is essential to simultaneously reduce Makespan time and allocate tasks to VMs with the highest level of trust. This study presented a trust-aware task scheduling approach called TABWO for cloud computing environments by leveraging the BWO algorithm. Our main objective was to maximize the performance and trustworthiness of cloud service providers by considering important factors such as makespan, energy consumption, and success rate. Extensive simulations were conducted using the Cloudsim framework to evaluate the effectiveness of the TABWO scheduler.

Acknowledgements
I would like to take this opportunity to acknowledge that there are no individuals or organizations that require acknowledgment for their contributions to this work.

Research involving human participants and animals
The observational study conducted on medical staff needs no ethical code. Therefore, the above study was not required to acquire an ethical code.

Author's contributions
The author contributed to the study's conception and design. Data collection, simulation, and analysis were performed by Huimin Wu. The author read and approved the final manuscript.

Funding
This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Availability of data and materials
Data can be shared upon request.

Declarations

Ethics approval and consent to participate
This option is not necessary due to that the data were collected from the references.

Competing interests
The author declares no competing interests.

Received: 1 November 2023 Accepted: 11 June 2024
Published online: 20 June 2024

References

1. Bagale GS, Vandadi VR, Singh D, Sharma DK, Garlapati DVK, Bommisetti RK, Gupta RK, Setsiawan R, Subramaniyaswamy V, Sengan S (2023) Small and medium-sized enterprises' contribution in digital technology. Ann Oper Res 326:3–4
2. Hayyolalam V, Pourghebleh B, Chehrehzad MR, Pourhaji Kazem AA (2022) Single-objective service composition methods in cloud manufacturing systems: recent techniques, classification, and future trends. Concurr Comput Pract Exp 34(5):e6698
3. Pourghebleh B, Anvigh AA, Ramtin AR, Mohammadi B (2021) The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments. Cluster Comput 24:2673–2696
4. Amini Motlagh A, Movaghar A, Rahmani AM (2020) Task scheduling mechanisms in cloud computing: A systematic review. Int J Commun Syst 33(6):e4302
5. Deshpande P (2020) Cloud of everything (CLeT): the next-generation computing paradigm, in computing in engineering and technology. Adv Intell Syst Comput 1025; pp 207–214. Springer
6. Hayyolalam V, Pourghebleh B, Kazem AAP, Ghaffari A (2019) Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques. The International Journal of Advanced Manufacturing Technology 105(1–4):471–498

7. Hayyolalam V, Pourghebleh B, Pourhaji Kazem AA (2020) Trust management of services (TMoS): Investigating the current mechanisms. Trans Emerg Telecommun Technol 31(10):e4063
8. Mohamed A, Hamdan M, Khan S, Abdelaziz A, Babiker SF, Imran M, Marsono MN (2021) Software-defined networks for resource allocation in cloud computing: A survey. Comput Netw 195:108151
9. Narwal A, Dhingra S (2023) A novel approach for Credit-Based Resource Aware Load Balancing algorithm (CB-RALB-SA) for scheduling jobs in cloud computing. Data Knowl Eng 145:102138
10. Iftikhar S, Ahmad MMM, Tuli S, Chowdhury D, Xu M, Gill SS, Uhlig S (2023) HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments. Internet Things 21:100667
11. Praveenchandar J, Tamilarasi A (2021) Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing. J Ambient Intell Humaniz Comput 12(3):4147–4159
12. Tan W, Sun Y, Li LX, Lu G, Wang T (2013) A trust service-oriented scheduling model for workflow applications in cloud computing. IEEE Syst J 8(3):868–878
13. Rjoub G, Bentahar J, Wahab OA (2020) BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments. Futur Gener Comput Syst 110:1079–1097
14. Ali A, Iqbal MM, Jamil H, Akbar H, Muthanna A, Ammi M, Althobaiti MM (2021) Multilevel central trust management approach for task scheduling on IoT-based mobile cloud computing. Sensors 22(1):108
15. Govindaraj P, Natarajan J (2020) Trust-based fruit fly optimisation algorithm for task scheduling in a cloud environment. Int J Internet Manuf Serv 7(1–2):97–114
16. Ebadifard F, Babamir SM, Labafiyan F (2020) A Multi Objective & Trust-Based Workflow Scheduling Method in Cloud Computing based on the MVO Algorithm. In 2020 11th International Conference on Information and Knowledge Technology (IKT), IEEE, pp. 26–30
17. Kaur A, Auluck N (2023) Real-time trust aware scheduling in fog-cloud systems. Concurr Comput Pract Exp 35(10):e7680
18. Hayyolalam V, Kazem AAP (2020) Black widow optimization algorithm: a novel meta-heuristic approach for solving engineering optimization problems. Eng Appl Artif Intell 87:103249