International Journal of Mathematical, Engineering and Management Sciences
Vol. 9, No. 5, 1089-1105, 2024
https://doi.org/10.33889/IJMEMS.2024.9.5.057

Ram Arti Publishers

Security-Aware Deadline Constraint Task Scheduling using Hybrid Optimization of Modified Flying Squirrel Genetic Chameleon Swarm Algorithm

G. Kiruthiga
Department of Computer Applications,
Guru Nanak College, Chennai, 600042, Tamil Nadu, India.
Corresponding author: g_kiruthiga@yahoo.co.in

S. Mary Vennila
Department of Computer Science,
Presidency College, Chennai, 600005, Tamil Nadu, India.
E-mail: maryvennila13@gmail.com

(Received on November 9, 2023; Revised on May 28, 2024; Accepted on June 29, 2024)

Abstract
Cloud computing enables cost-effective resource sharing in hybrid clouds to tackle the problem of insufficient resources by elastically scaling the service capability based on the users' needs. However, task scheduling (TS) in cloud environments is challenging due to deadline-based performance and security constraints. To remove the existing drawbacks based on deadline and security constraints, a Security-Aware Deadline Constraint TS (SADCTS) approach is presented using a hybrid optimization algorithm of the Modified Flying Squirrel Genetic Chameleon Swarm Algorithm (MFSGCSA). The proposed MFSGCSA is developed by integrating the genetic operators into CSA and combining it with the modified Flying Squirrel Optimization (FSO) algorithm in which the position update and global search equations are enhanced by adaptive probability factor to reduce the local optimum problem. In this SADCTS approach, the task assignment process is modeled into an NP-hard problem concerning a multi-level security model using user authentication, integrity, and confidentiality. This maximizes tasks' completion rate and decreases the resource costs to process tasks with different deadline limitations. The optimal schedule sequence is obtained by MFSGCSA, where tasks are scheduled concurrently based on security constraints, demand, and deadlines to improve the prominence of cost, energy, and makespan. Experiments are simulated using the CloudSim toolkit, and the comparative outcomes show that the suggested SADCTS approach reduced the makespan, cost, and energy by 5-20% more than the traditional methods. Thus, SADCTS provides less task violation of 0.0001%, high energy efficiency of 700GHz/W, high resource utilization of 92%, less cost efficiency of 72GHz/$, and less makespan of 480minutes to satisfy the necessary security and deadline requirements for TS in shared resource hybrid clouds.

Keywords- Cloud computing, Hybrid clouds, Task scheduling, Security, Deadline constraint, Genetic chameleon swarm algorithm, Flying squirrel optimization.

1. Introduction

Cloud computing provides vast benefits, including pay-per-use, elastic resource provisioning, high dependability, easy availability, cost-effective scalability, dynamic customization, and high security (Khalil et al., 2014). It provides various resources, such as software, hardware, or virtual machines (VM). These resources offer high availability and flexibility to users preferring high-performance services. Although these provisional resources are considered unlimited to the users, they are limited in the real world. The ideal cloud system aims to adaptively share these resources based on the peak demands of the users with the utmost satisfaction of their requests (Aithal and Pai, 2017). However, the public and private clouds cannot meet such peak demands without resources. In such scenarios, most cloud systems perform one of the three solutions, namely 1) reject a few of the non-vital requests, mostly the reasonable requests, 2) include different architecture to increase the resource pool, or 3) utilize cloud bursting to form a hybrid

of private and public clouds such that the resources can be elastically increased or decreased based on the requirements (Wang et al., 2020). In the first two options, the service providers are bound to lose their customers with non-vital requests or incur additional costs for infrastructure, which are not ideal for small organizations. Therefore, the third option of hybrid clouds seems to be the perfect solution for the ideal resource-sharing scenario. This solution increases the diversity of cloud applications and provides both open-source and commercial tools with adequate cloud-bursting properties (Wang et al., 2021). However, even hybrid clouds have their own set of limitations. The most important limitations are the inability to handle deadline-sensitive applications and high-security constraints. Therefore, the task execution must be optimized through optimal TS regarding makespan, cost, SLA violations, security, complexity, and energy consumption (Lavanya et al., 2020).

TS algorithms must be developed considering various factors, including the user and service provider's requirements. Security is regarded as the top priority for the applications, which are mostly owned and operated by enterprises (Zhu et al., 2021). Most existing TS methods utilized two-level security models. The reserved tasks with high-security requirements are performed in the private cloud, and open tasks with minimized requirements are accomplished in both cloud models. However, these models become ineffective in the real world as the security problem is not a two-level problem but a multi-dimensional problem (Meng et al., 2021). The security-required tasks are sometimes outsourced to the public cloud to optimize task execution. But this process might lead to resource overheads. Therefore, the security model must be multi-level and effective in all applications. In terms of performance, the other major constraint is handling the deadlines for tasks and services to ensure higher satisfaction for users. The deadline constraints are based on the tasks and sub-tasks in a workflow, cloud resource accessibility, and the processing capability of the system processors (Alworafi and Mallappa, 2018). Resource heterogeneity must be considered when designing the TS on shared VMs in hybrid clouds. In addition to the security requirements, these requirements make the TS process a multi-objective problem that must ensure minimized makespan, cost, SLA violation, etc. Since there are multiple objectives, the optimization of the task execution is often performed using metaheuristic models (Hosseinzadeh et al., 2020). Utilizing advanced metaheuristic algorithms to maximize task completion, minimize costs, and satisfy the deadlines and protection layers in hybrid clouds (Sofia and GaneshKumar, 2018).

The main objective of this paper is to develop an efficient task scheduling model that incorporates security and reliability for allocating tasks to VMs with suitable resource sharing. The formation of a multi-objective hybrid optimization algorithm achieves this objective. This paper formulates the task assignment process in hybrid clouds as a multi-objective, non-linear encoding problem. Since this problem is NP-hard, an advanced scheduling scheme of SADCTS is proposed, which utilizes the hybrid optimization algorithm of MFSGCSA to minimize the multi-objective problem in polynomial time. This paper offers the following solutions: A) The Task assignment process on the hybrid clouds is formulated into a multi-objective non-linear encoding problem considering a multi-level protection scheme to maximize the task completion rate and cost minimization with effective handling of deadline and security constraints. B) To solve the multi-objective non-linear encoding problem using the SADCTS approach developed using hybrid MFSGCSA to optimize the task accomplishment in hybrid clouds. This proposed MFSGCSA is developed by incorporating the genetic operators into CSA and combining them with the global search-modified FSO. The optimal schedule sequence is obtained by MFSGCSA, where tasks are assigned based on security constraints, demand, and deadlines. C) Broad simulation tests are conducted with real-world trace data to examine the efficiency of the suggested SADCTS approach and equate the performance against the traditional schemes. The remainder of this paper is structured as follows: Section 2 explains the recent related works. Section 3 illustrates the development of the proposed SADCTS approach for task scheduling in the cloud. Section 4 simulates the proposed model and discusses the

obtained results. Section 5 provides a conclusion of this paper and also suggests possible directions for future work.

2. Related Works

Different variants of popular optimization algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Firefly Optimization, Artificial Bee Colony (ABC), Cuckoo Search Algorithm (CSA), Chaotic Quantum Whale Optimization Algorithm (CWOA) (Kiruthiga and Mary Vennila, 2019), Chaotic Quantum-Behaved Chicken Swarm Optimization (CQCSO) (Kiruthiga and Mary Vennila, 2021), Chaotic Darwinian Chicken Swarm Optimization (CDCSO) (Kiruthiga and Mary Vennila, 2020), etc. have been employed for cloud task assignment process. Most recent studies have experimented with different and advanced bio-inspired optimization algorithms for TS.

Manikandan et al. (2022) presented a Modified Bee Algorithm (MBA) based Random Double Adaptive WOA (RDWOA) model for the task assignment process. In this model, the execution time and cost of the TS are reduced by introducing the hybrid MBA-RDWOA in which the mutation operator of MBA improves the WOA output. With faster convergence, this approach reduced the completion time, computation cost, and execution time for large scheduling problems. Nabi et al. (2022) recommended an Adaptive PSO (AdPSO)-based TS approach in which the standard PSO is fine-tuned by introducing Linearly Descending and Adaptive Inertia Weight (LDAIW). This AdPSO algorithm reduced the TS time by 10% and increased the throughput and Average Resource Utilization Ratio (ARUR) by 12% and 60%, respectively. Yet, this model has a high response time, causing poor performance for delay-sensitive applications. Panda et al. (2022) offered a pair-based TS algorithm that utilized the Hungarian algorithm (HA) for fast convergence and reduced response and execution times. However, this model did not consider network usage while scheduling and has scalability issues. Kandan et al. (2022) proposed a Quasi-Oppositional Aquila Optimizer-based TS (QOAO-TS) approach. It integrates quasi-oppositional learning with the Aquila optimization algorithm to reduce the makespan and increase the throughput and exploitation ratio of the tasks. Abualigah and Alkhrabsheh (2022) developed a Multi-Verse Optimizer with GA to form the MVO-GA algorithm to improve the cloud's speed, capacity, task size, and throughput properties TS. This model reduced the task transfer time by 15% and minimized execution time through fast convergence. Ajitha and Indra (2022) offered Fisher linear discriminant and discrete global swarm-based TS (FLD-DGSTS) methods. This hybrid optimization function considers the CPU cycles, RAM, energy, and bandwidth to optimize the TS problem and reduce the makespan, energy, and memory constraints.

Bezdan et al. (2022) developed Multi-objective TS using hybridized Bat Algorithm-ABC exploitation quasi-reflection learning (BAAEQRL). This model utilized the QRL to improve the exploitation based on the modified search solutions by the ABC algorithm. This model reduced the cost by 3-5% and makespan by 1-5%. However, this model does not suit the deadline constraint applications. Zhang et al. (2022) proposed an interval many-objective evolutionary algorithm (InMaOEA) with an interval reliability policy, hyper-volume, and overlap degree to improve convergence and population diversity. This algorithm reduced uncertainty problems, makespan, and scheduling costs while improving the load balance and task accomplishment ratio by 12%. Abdullahi et al. (2023) suggested constrained multi-objective adaptive benefit factors based on symbiotic organisms search (CMABFSOS) for constrained TS. The SOS control parameters are adaptively tuned, and an adaptive constrained handling strategy reduces premature convergence. This CMABFSOS model reduced workloads by 17-48% and performance improvement by 20-50%. Attiya et al. (2022) developed a DJSD TS method using a Dynamic Jellyfish Search Algorithm (DJSA) based on Simulated Annealing (SA) and Disruption factors for better convergence. In this approach, the SA was used to enhance the exploration phase, and the

disruption operator was used to boost the diversity of solutions in the exploitation phase to avoid the local optima problem. This approach reduced the makespan and energy consumption by 3-6%. Zubair et al. (2022) introduced Geometric Mean based improved Symbiotic Organisms Search Algorithm (G-SOS) for heterogeneous TS to minimize the makespan by 1-25%, cost by 10-31%, and response time by 8-42%. However, this model has a trade-off in deadline-constraint applications such that the cost increases when the makespan is smaller. Mangalampalli et al. (2022) proposed Cat Swarm Optimization Algorithm-based TS with makespan, energy, total power cost, and migration time as parameters. Implemented using NASA and HPC2N workloads, this model reduced makespan by 10-27%, energy consumption by 12-31%, total power cost by 31-39%, and migration time by 14-34%.

Fu et al. (2023) developed a TS model using a hybrid PSO and genetic algorithm to form PSO_PGA. This hybrid model enhanced the PSO population by using the phagocytosis mechanism and crossover mutation of genetic algorithm to improve the search capabilities and apply the PSO update functions to adaptively allocate the tasks to the VMs. Saravanan et al. (2023) developed Improved wild horse optimization with a levy flight (IWHOLF) algorithm for TS to address the problems of lengthy scheduling time, higher computation costs, and higher loads. This model reduced the makespan by 2%, energy by 8%, and cost by 12%. Behera and Sobhanayak (2024) proposed a hybrid GA-Grey Wolf Optimization Algorithm (GWO) approach for TS in which the crossover and mutation operators of GA are used to improve the convergence of GWO. Compared to standard GWO, GA, and PSO algorithms, this model reduced the makespan by 19%, 21%, and 15% and energy by 17%, 19%, and 23%. However, this model increased the computation complexity for deadline-sensitive tasks. From the studies in the literature, it is clear that the metaheuristic optimization algorithms-based schedulers have greater advantages, which is evident from the numerous studies that utilize them compared to the traditional TS algorithms. The major finding from the literature is that the task scheduling methods have prioritized energy and other resource utilization while neglecting the deadline constraints, which are very critical in deadline-sensitive applications. This ignorance has raised reliability issues. Therefore, this paper considers these two problems and develops an efficient SADCTS approach using the novel hybrid model of MFSGCSA for reducing the makespan, cost, and energy consumption with increased security and reliability. The MFSGCSA combines the strengths of the Flying Squirrel Search Algorithm, Genetic Algorithms, and Chameleon Swarm Algorithm to enhance optimization performance. Compared to other standard optimization algorithms such as GA, PSO, ACO, WOA, CS, ABC, etc., this proposed MFSGCSA is robust and flexible, which effectively balances exploration and exploitation throughout the search process and improves the scalability and convergence speed with efficient prevention of premature convergence and ensuring better global search capabilities.

3. Methods

The proposed SADCTS approach has been developed by utilizing a hybrid optimization algorithm of MFSGCSA. The model is implemented on a hybrid cloud framework with independent tasks. The tasks include workflows, parallel rendering, and software testing tasks, as they are prevalent in large numbers in parallel and distributed systems. **Figure 1** shows the overall structure of the proposed SADCTS approach.

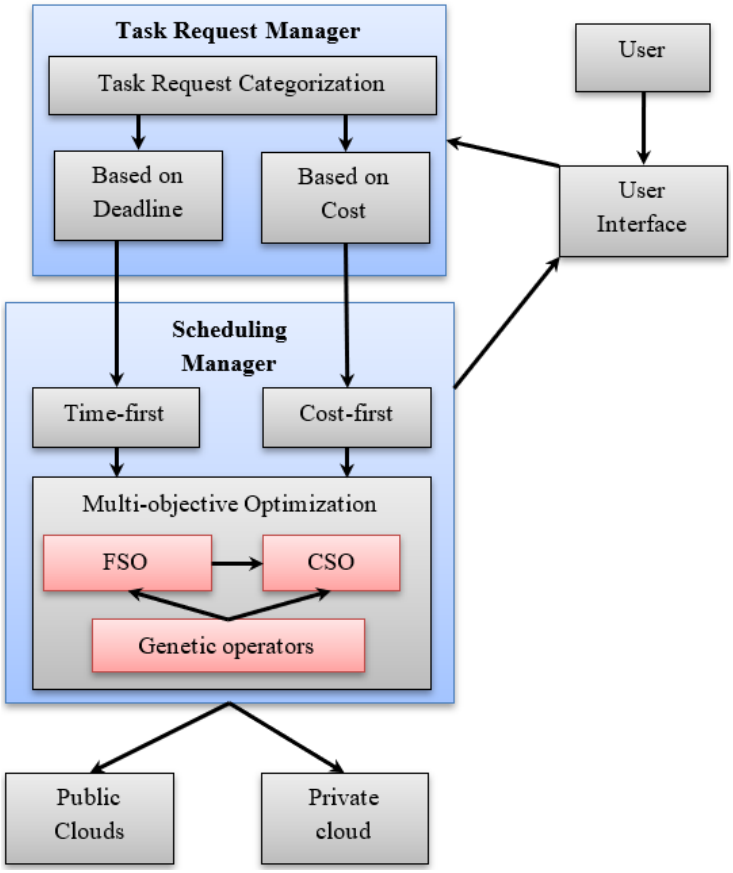


Figure 1. Proposed methodology.

3.1 Hybrid Cloud and Multi-Level Security Model

The hybrid cloud model comprises a private and a public cloud, consisting of multiple public clouds combined into a big public cloud for executing independent tasks with effective resource provisioning. **Figure 2** shows the implementation environment of the hybrid cloud model for performing the proposed TS method.

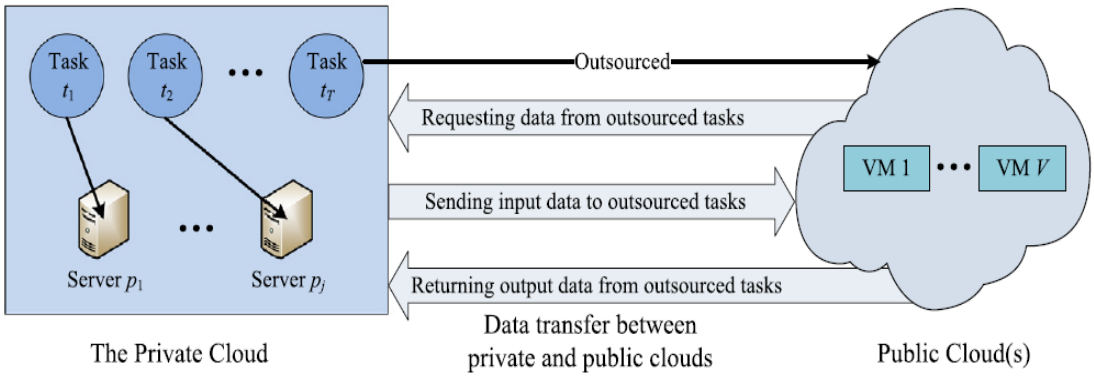


Figure 2. Hybrid cloud model.

There are P servers in the private cloud to execute the input tasks, where each $p_j (j \in [1, P])$ server contains c_j cores with a great computing capacity g_j (in terms of MIPS). Apart from the private cloud, the public cloud offers V VM types, each with c_{vm_k} cores with a computing capacity g_{vm_k}. The cost per unit time of the VM requests with vm_k type is U_k. The private cloud selects suitable VM instances for rent whenever there is a shortage of resources to match the user task requirements (Ajitha and Indra, 2022).

The task execution model considers all T tasks ($t_i,\ i \in [1, T]$) with size s_i based on the server units and consumes s_i/g time to complete the task allocated to the core with computing capability g. If a task t_i is assigned, the data size is determined as in_i and out_i, while the deadline is represented as d_i. It is assumed that the deadline is without loss of generality, i.e. $d_1 \leq d_2 \leq \cdots \leq d_T$. The input and output data are assumed to be stored in a distributed file system (DFS) on the private cloud. The p_j server bandwidth for reading and writing data is represented as r_j and w_j. When a task t_i is assigned to p_j, the input and output data consumed in_i/r_j and out_i/w_j time, the completion time of the task t_i is given as,

$$\tau_{i,j} = \frac{in_i}{r_j} + \frac{s_i}{g_j} + \frac{out_i}{w_j} \quad (1)$$

The information handover bandwidth among the public and private clouds is bw, then the input and output data without security requirements will consume in_i/bw and out_i/bw time. The time to complete t_i devoid of security necessities on a VM will be,

$$\tau_{i,vm_k} = \frac{in_i + out_i}{bw} + \frac{s_i}{g_{vm_k}} \quad (2)$$

For a task t_i, the protection layer values [0, 1] to describe the minimum-security necessities in three traits: user authentication, integrity, and confidentiality. Allocating a task to the private cloud, there is no additional security process since they already provide the maximum level of security. When a task is farmed out to the public cloud, the necessary security technologies are employed to increase the low-level security of the public cloud.

For tasks t_i having protection layer requirements $sl_{i,\alpha}$ for validation, a user authentication method is applied to ensure high security, i.e.

$$sl_{i,\alpha} \leq sl_{a_i}^{\alpha} \quad (3)$$

Here, $sl_{a_i}^{\alpha}$ denotes the confidence delivered by the a_i validation system ($a_i \in \{0,1, \ldots\ldots, A\},\ A = 3$). It is anticipated that without generalization cost, the overhead $o_{a_i-1}^{\alpha} < o_{a_i}^{\alpha}, \forall a_i \in \{0,1, \ldots\ldots, A\}$, where $o_{a_i}^{\alpha}$ denotes the overhead of a_i. Thus, a_i satisfies the protection layers with reduced overhead for tasks t_i subject to the conditions,

$$sl_{a_i-1}^{\alpha} < sl_{i,\alpha} \leq sl_{a_i}^{\alpha} \quad (4)$$

This system has a static overhead when it is performed for an outsourced t_i, and it increases the execution time,

$$\tau_{i,vm_k}(a_i, .,.) = o_{a_i}^{\alpha} + \tau_{i,vm_k} \quad (5)$$

Here, $\tau_{i,vm_k}(a_i, .,.)$ denotes the time to complete the outsourced t_i assigned to the VM in the public cloud by using the a_i authentication method that ensures protection layer requirements only in user verification.

Once the authentication is done, the input data demanded for the job is transmitted from the private to the public cloud. After completion, the results are forwarded to the storage unit. In this stage, a simple but effective cryptographic algorithm must be utilized to prevent malicious attacks so that the data and resources are not shared with unauthorized users or applications to improve the protection layer necessities for information privacy $sl_{i,\beta}$ with minimum overhead.

$$sl_{b_i-1}^{\beta} < sl_{i,\beta} \leq sl_{b_i}^{\beta} \quad (6)$$

Here, $sl_{b_i}^{\beta}$ denotes the protection layer delivered by b_i cryptographic procedure ($b_i \in \{0,1,\ldots..,B\}$). Therefore, the time to complete the outsourced t_i will be,

$$\tau_{i,vm_k}(.,b_i,.) = o_{b_i}^{\beta}.(in_i + out_i) + \tau_{i,vm_k} \quad (7)$$

Here, $\tau_{i,vm_k}(.,b_i,.)$ denotes the time to complete the outsourced t_i using b_i, a cryptographic algorithm that ensures protection layer requirements only in information privacy and $o_{b_i}^{\beta}$ denotes the interval overhead of b_i cryptographic procedure per information. Once the private cloud obtains the output data from the public cloud, the information reliability is tested to avoid attacks, which can be accomplished by using hash functions. The tasks' protection layer required for the data integrity is given as $sl_{i,\varepsilon}$ and is satisfied by the e_i, a hash function that affords a protection layer of $sl_{e_i}^{\varepsilon}$ ($e_i \in \{0,1,\ldots..,\Gamma\}$ subject to

$$sl_{e_i-1}^{\varepsilon} < sl_{i,\varepsilon} \leq sl_{e_i}^{\varepsilon} \quad (8)$$

The time to complete the task t_i devising only the data reliability protection layer by the e_i hash function is given as,

$$\tau_{i,vm_k}(.\ ,.,e_i) = o_{e_i}^{\varepsilon}.out_i + \tau_{i,vm_k} \quad (9)$$

Here, $\tau_{i,vm_k}(.\ ,.,e_i)$ denotes the time to complete the t_i using the e_i hash function that ensures protection layer necessities only in data reliability and $o_{e_i}^{\varepsilon}$ denotes the time overhead of e_i hash function per data. Considering such characteristics, the time taken to complete the t_i task execution with protection layer necessities in consumer validation, information privacy, and data reliability is given by,

$$\tau_{i,vm_k}(a_i\ , b_i, e_i) = o_{a_i}^{\alpha} + o_{b_i}^{\beta}.(in_i + out_i) + o_{e_i}^{\varepsilon}.out_i + \tau_{i,vm_k} \quad (10)$$

Here, $\tau_{i,vm_k}(a_i\ , b_i, e_i)$ denotes the time to complete the t_i having protection layer requirements by a_i an authentication method, b_i cryptographic algorithm and e_i hash function.

3.2 Problem Formulation and MFSGCSA Modeling for SADCTS

The scheduling problem is based on two major constraints: full task completion and minimum cost for filling the target and safety requirements. The multi-objective fitness utility is formulated as,

$$\text{Maximizing } \sum_{i=1}^{T}\left(\sum_{j=1}^{p}\sum_{l=1}^{c_j} x_{i,j,l} + \sum_{k=1}^{V}\sum_{m=1}^{n_k}\sum_{l=1}^{c_{vm_k}} y_{i,k,m,l}\right),$$

$$\text{Minimizing } \sum_{k=1}^{V}\sum_{m=1}^{n_k} Cost_{k,m} + Cost_{bw} \quad (11)$$

Here, $x_{i,j,l}$ and $y_{i,k,m,l}$ denotes whether the task execution of t_i is carried out in the core l of the server p_j and core l of m instance of the VM type vm_k, respectively. $Cost_{k,m}$ and $Cost_{bw}$ denotes the cost for leasing data m of the VM category vm_k and the cost of system resources for completing outsourced tasks, respectively. Makespan is the maximum of all task completion time, and energy is the power depletion for a task on a VM based on the completion time. $Makespan = max(CT_{ij})$ and $E_{ij} = P(U_{ij}) \times CT_{ij}$. Here

CT_{ij} is the completion time, U_1 denotes the minimum range of CPU utilization U, and U_2 denotes the maximum range of CPU utilization U.

The MFSGCSA utilizes Equation (11) to formulate the solution for the TS challenge. MFSGCSA is formed by merging the Modified Flying Squirrel Optimization (MFSO) and Genetic Chameleon Swarm Algorithm (GCSA) algorithms. The FSO algorithm is based on two motions of the flying squirrels in their ultimate goal of food search and predator avoidance (Azizyan et al., 2019). The two motions of random walking on the grounds and gliding based on Levy flights for aerial movements are considered. In this algorithm, the global search process is enhanced by adding an adaptive probability of the predator to reduce the local optima convergence. First, the squirrel population is initialized in MFSO, and the random walk process is applied, consisting of many sequential, random steps. Mathematically, S_N forms a random walk in N dimensions with S_N denoting the sum of each sequential, random step X_i. The random walk is given as,

$$S_N = \sum_{i=1}^{N} X_i = X_1 + X_2 + \cdots + X_N \quad (12)$$

Here X_i denote the random steps obtained from an arbitrary distribution, and this association is rewritten as a recursive equation,

$$S_N = \sum_{i=1}^{N-1} X_i + X_N = S_{N-1} + X_N \quad (13)$$

Here, S_N becomes the next state entirely dependent on the current state S_{N-1} while X_N denoting the motion from the present state to the subsequent state. The step size can be either fixed or varied in this random walk. This study determines this step size by Normal distribution and converts the random walk into the Brownian motion.

A semi-sigmoid function called Sigma Reduction Factor (SRF) has been used in the FSO to increase the accuracy of the random phase search operations. SRF is deliberated as a factor of standard deviations of the motion steps, which induces a decrease in the standard deviation of the flying squirrels' motion at each iteration. In this equation, the global search process is enhanced by adding an adaptive probability of the predator that can dynamically vary based on the iterations, which is denoted as,

$$P_{adap} = (P_{dmax} - P_{dmin}) \times \left(1 - \frac{t}{T}\right)^2 + P_{dmin} \quad (14)$$

Here, P_{dmax} and P_{dmin} are the maximum and minimum probability of predators in the trajectory, t denotes the iterations, and T denotes the maximum iterations. The inverted sigmoid function can be used for SRF computation.

$$SRF_t = \sqrt{\left(-\ln\left(1 - \frac{1}{\sqrt{t}}\right)\right)^2} + P_{adap} \quad (15)$$

Similar to performing the random walk for ground motion, the second movement of the Levy flight is used for the flight motion of the squirrels. The flying squirrels do not fly like birds but use their bodies for paragliding and float in the air to reach the next tree. The angle of the glide and the size of the squirrels play a significant role in achieving optimal glide.

When implementing this algorithm, a symmetric Levy stable distribution must be used to obtain the bi-directional steps, which can be positive or negative. In this case, a random variable U and its probability distribution are stable only when a linear combination of the variables possesses the same distribution. The step size of such distribution can be estimated as,

$$s = \frac{u}{|v|^{1/\beta}} \tag{16}$$

Here, u and v are the variables obtained from the normal distribution and are expressed as,

$$u \sim N(0, \sigma_u^2), v \sim N(0, \sigma_v^2) \tag{17}$$

Here, σ_u^2 and σ_v^2 are the variance of u and v. After applying the MFSO to select the initial best solutions for scheduling the tasks, the GCSA algorithm is applied to validate the selected solution. GCSA is developed by integrating the mutation and crossover operations of the genetic algorithm into the Chameleon swarms (Braik, 2021). Initially, the swarm population is determined along with the initial positions and the bounds of the d-dimensional exploration region with all chameleons signifying a candidate solution denoted as a two-dimensional matrix y of size $n \times d$. Then, two individuals are selected randomly as parents, and the two-point crossover operation is applied. The resulting offspring are improvised using a 20% mutation process. These two processes increase the quality of the solutions, and these new solutions are added to the archive for the optimization problem. A trajectory can represent the position of the i-th chameleon at t iterations.

$$y_t^i = \left[y_{t,1}^i, y_{t,2}^i, \ldots, y_{t,d}^i\right]; \quad i = 1,2, \ldots, n \tag{18}$$

Here, t denotes the current iteration, d denotes the dimension, and $y_{t,d}^i$ denotes the location of the *i*-th chameleon at the *d*-th aspect. The preliminary population is computed as,

$$y^i = l_j + r \times (u_j - l_j) \tag{19}$$

Here, y^i denotes the primary trajectory of the *i*-th chameleon, l_j and u_j represents the minor and major bounds of the exploration region in the *j*-th dimension, and *r* is a homogeneously formed arbitrary integer $r \in [0,1]$.

While the positions of chameleons and the velocity of their tongues are interrelated, the random positions are defined with rotating coordinates to match their eye orientations. Therefore, the positions are updated as per the orientation.

$$y_{t+1}^i = yr_t^i + \bar{y}_t^i \tag{20}$$

Here, y_{t+1}^i indicates the recent location of the chameleon after the orientation, $\bar{y}_t^i$ defines the focal point of the present location of the chameleon before the orientation and yr_t^i symbolizes the orientation concentrated coordinates in the exploration region given by,

$$yr_t^i = rm \times yc_t^i \tag{21}$$

Here, $rm = R(\theta, \vec{V}\overrightarrow{z_1}, \overrightarrow{z_2})$ denotes the orientation matrix with $\overrightarrow{z_1}$ and $\overrightarrow{z_2}$ representing the two orthonormal trajectories of size $d \times 1$, R denotes the orientation matrices in the corresponding axes, $\vec{V}$ denotes the position gyrating trajectories, $r \times sgn(rand - 0.5) \times 180°$ represents the orientation angle of an r arbitrary chameleon and $sgn(rand - 0.5)$ is the direction of orientation, which is ± 1, and yc_t^i denotes the center coordinates at iteration *t*.

$$yc_t^i = y_t^i - \bar{y}_t^i \tag{22}$$

The behavior of the chameleon can be modeled using the position updating process as given below.

$$y_{t+1}^{i,j} = \begin{cases} y_t^{i,j} + p_1\left(P_t^{i,j} - G_t^j\right)r_2 + p_2\left(G_t^j - y_t^{i,j}\right)r_1 & r_i \geq P_p \\ y_t^{i,j} + \chi\left(\left(u_j - l_j\right)r_3 + l_b^j\right)sgn(rand - 0.5) & r_i < P_p \end{cases} \tag{23}$$

Here, $y_{t+1}^{i,j}$ denotes the recent location of the *i*-th chameleon in the (t + 1) iteration step, $y_t^{i,j}$ denotes the present location at the t iteration step, $P_t^{i,j}$ denotes the best position obtained till now by *i* in the *j*th facet at iteration loop *t*, G_t^j denotes the global best position obtained till now by any chameleon in the *t*-th iteration, p_1 and p_2 are two positive integers for controlling the exploration ability, $r_1, r_2, r_3 \in r_i$ are uniform random numbers between $[0, 1]$, P_p denotes the likelihood of the chameleon identifying target, computed as 0.1 in this work and χ refers to the utility of iterations that decreased with increasing iterations and is employed to minimize the convergence speed and improvise the search characteristics. It is shown in the following equation.

$$\chi = \phi e^{\left(-\frac{\mathrm{At}}{\mathrm{T}}\right)^{\mathrm{B}}} \tag{24}$$

Here, $\phi, \mathrm{A}, \mathrm{B}$ are the three constant integers used to regulate the search process, and (t, T) denotes the current and maximum iterations. Based on the position updating equation, the speed of the chameleon's tongue can be modeled as,

$$v_{t+1}^{i,j} = \omega v_t^{i,j} + c_1\left(G_t^j - y_t^{i,j}\right)r_2 + c_2\left(P_t^{i,j} - y_t^{i,j}\right)r_1 \tag{25}$$

Here, $v_{t+1}^{i,j}$ represents the updated velocity of the *i*-th chameleon in the (t + 1) iteration step, $v_t^{i,j}$ denotes the current velocity of an i-th chameleon at the t iteration step, c_1 and c_2 are two positive integers for controlling the influence of $P_t^{i,j}$ and G_t^j on falling the chameleon's tongue, and ω refers to the inertia weight, which decreases linearly and is given as,

$$\omega = (1 - t/T)^{\rho\sqrt{\left(\frac{t}{T}\right)}} \tag{26}$$

Here, ρ denotes a positive integer to control search volume and is mostly equal to 1. This inertia weight ω improves the convergence of GCSA, and the point of the chameleon's tongue, when moved en route to the target, becomes the chameleon's location.

$$y_{t+1}^{i,j} = y_t^{i,j} + \frac{\left(\left(v_t^{i,j}\right)^2 - \left(v_{t-1}^{i,j}\right)^2\right)}{2a} \tag{27}$$

Here, $v_{t-1}^{i,j}$ represents the previous speed of the *i*-th chameleon in the (t + 1) iteration step, and a denotes the acceleration ratio of the chameleon's tongue projection given as,

$$a = 2590 \times (1 - e^{-\log(t)}) \tag{28}$$

Here, the value of a enhances steadily towards the utmost value of 2,590m/s. Equations (23), (25) and (27) are the main equations to simulate the GCSA. Once GCSA returns the solutions, they are validated and returned as the final solutions by the MFSGCSA algorithm. The complexity of this model is given as $O(2D.N.T)$ where N – population size, D - dimension number, and T - maximum iteration. The overall process of the proposed SADCTS approach using MFSGCSA is presented in Algorithm 1.

***Algorithm* 1: MFSGCSA-based SADCTS Approach**

Begin
Initialize the servers, VMs, input tasks, and other cloud parameters
Initialize flying squirrels and chameleons
Apply protection layer based on three aspects
Set initial candidates' solution set
Set iteration t=0

Compute multiple objectives
Formulate the objective function using Equation (11)
Calculating the core with high objective values (private cloud)
Renting the VMs with the maximal security and earliest deadline satisfaction (public cloud)
For each squirrel
Initiate the random walk of MFSO using Equations (12), (13)
Compute SRF using Equation (15) with an adaptive probability of the predator
Update the location of each squirrel toward the local best
Reassign the squirrels
Perform Levy flight motion
Change the step size
Follow the normal distribution of the solutions
Update the minimum seasonal movement parameter
Update the location of each squirrel toward the global best
End for
Return global best solution
Assign the global best solutions to the GCSA
Initialize the position, search space, and dimensions
Select two solutions randomly
Apply two-point crossover and 20% mutation to enhance the solutions
Set the initial position, velocity, and orientation coordinates of the tongue using Equations (18) and (19)
For each chameleon in the new set
    Compute the fitness using Equation (11)
    Update the position using Equation (23)
Define χ, ω, and a using Equations (24), (26), and (28)
    Appraise the speed of the chameleon's tongue by Equation (25)
    Appraise the location with χ, ω, and a using Equation (27)
End for
Return global best G_t^j
Assign the tasks based on G_t^j
End

4. Results

The proposed SADCTS-based TS approach is assessed through extensive simulation trials using the CloudSim 3.0.3 simulator. The experiments are conducted over the independent tasks obtained from the University of Luxemburg Gaia log in Parallel Workloads Archive (Meng et al., 2021). The workload traces contain simple tasks with information that can be executed on 2GHz cores. The tasks with the negatively used memory are removed; hence, the final dataset contains about 49000 tasks. The trials are conducted on an Intel ® core i7 processor with a 2GHz frequency CPU, 8GB RAM, and Windows 10 operating system using Eclipse IDE with JDK 1.8. The read/write IO bandwidths are assigned as 7 and 6 GB/s. Parameter settings are given in **Table 1**.

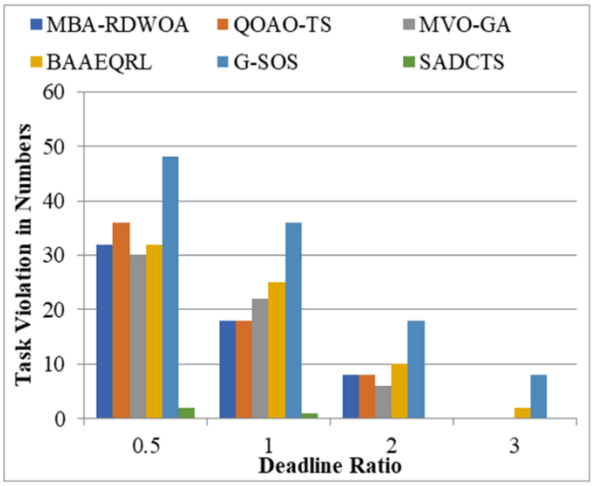
The existing methods, namely MBA-RDWOA (Manikandan et al., 2022), QOAO-TS (Kandan et al., 2022), MVO-GA (Abualigah and Alkhrabsheh, 2022), BAAEQRL (Bezdan et al., 2022), and G-SOS (Zubair et al., 2022) implemented in the same environment, and the outcomes are equated with the performance of the suggested SADCTS approach. The performance is compared in terms of task violation, resource utilization, energy efficiency, cost, and makespan. **Figure 3** displays the task violation

of the suggested SADCTS approach against the existing methods, with a) denoting the violation in numbers and b) denoting the violation in size.

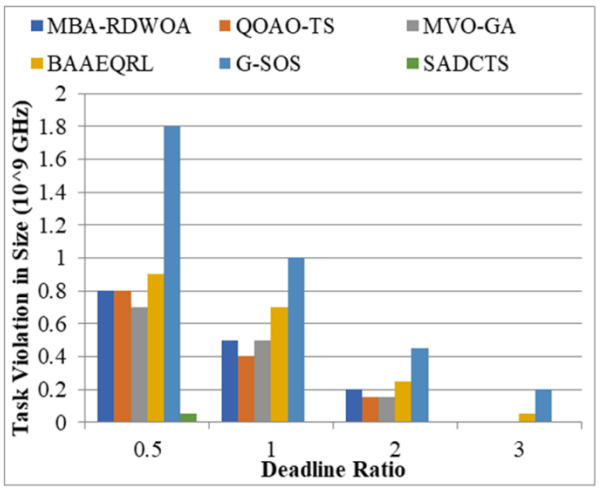
Table 1. CloudSim parameter settings.

Entity type	Parameter	Value
Data centers	Count	2
	Type	Heterogeneous
	Link delay (milliseconds)	10-100
	Bandwidth (Gbps)	2-10
Host	Hosts count	300
	Cores	2-8
	Host RAM (MB)	8192
	Host Storage (MB)	2000000
	Host bandwidth (bps)	20000
VM	VMs count	50-500
	CPU (MIPS)	5000-50000
	RAM (MB)	1024
	Bandwidth (bps)	2000
	Cores per VM	2
Task	Tasks count	49000
	Length (MI)	250-2500
	Size	250-800
	Iterations	20-100

Figure 3 shows that both in terms of the number and size, the task violation of the SADCTS approach is much less when the deadline ratio is 0.5 and 1.0, while there is no violation when the deadline ratio is 2.0 and 3.0. Compared to the proposed SADCTS approach, the existing methods failed to complete all the tasks within the deadline ratio. This is mainly because of the unique design of the SADCTS approach to evade task desecration by using deadline-aware TS and also instantly outsourcing the uncompleted tasks in private clouds to the public clouds.



a) Violation in numbers.



b) Violation in size.

Figure 3. Task violation in numbers.

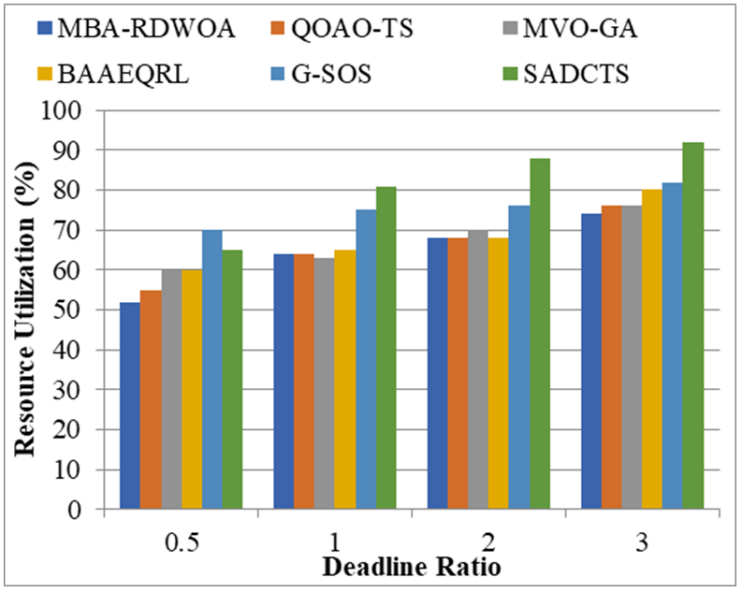


Figure 4. Resource utilization rate.

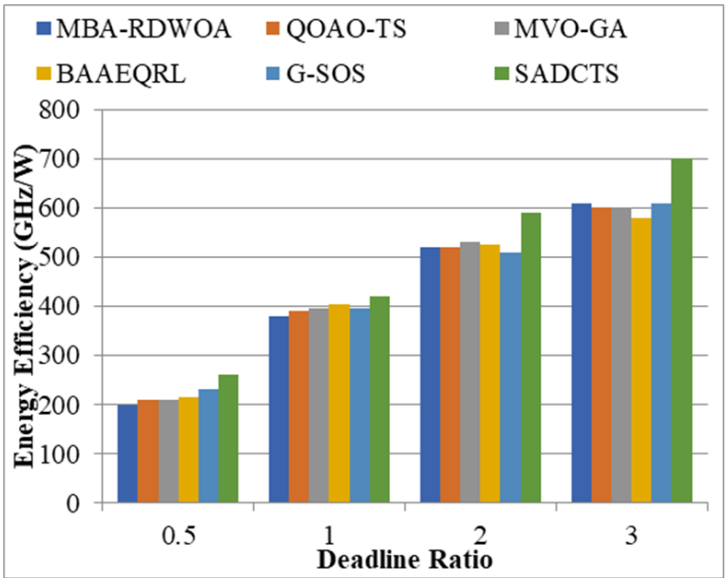


Figure 5. Energy efficiency.

Figure 4 displays the resource utilization of the proposed SADCTS approach against the existing methods.

Figure 4 displays the overall resource utilization of the SADCTS approach is 17%, 20%, and 18% higher than MBA-RDWOA, 17%, 20%, and 16% higher than QOAO-TS, 18%, 18%, and 18% higher than MVO-GA, 16%, 20%, and 12% higher than BAAEQRL, and 6%, 12%, and 10% higher than G-SOS

methods when the deadline ratio is 1.0, 2.0 and 3.0. These demonstrate that the suggested approach employs the hybrid cloud resource more effectively and completes most tasks with effective load balancing among the cores and VMs.

Figure 5 shows the energy efficiency comparison. SADCTS approach has a better energy efficiency of about 30%, 10%, 14% and 15% higher than MBA-RDWOA, 24%, 8%, 14% and 16.66% higher than QOAO-TS, 24%, 6%, 11% and 16.66% higher than MVO-GA, 21%, 4%, 12% and 20.69% higher than BAAEQRL, and 13%, 6%, 16% and 14.75% higher than G-SOS methods, when the deadline ratio is 0.5, 1.0, 2.0 and 3.0, respectively.

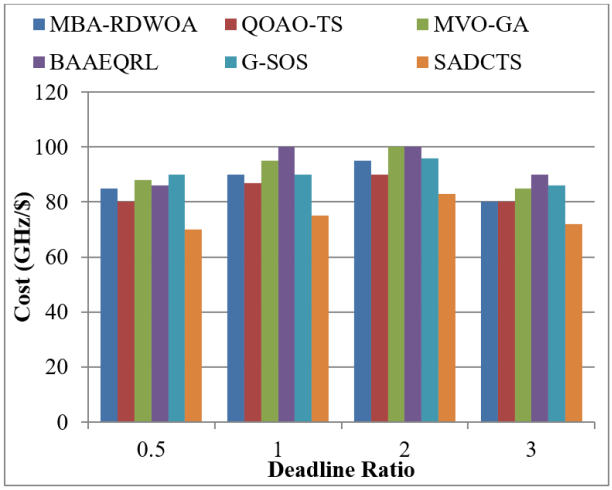


Figure 6. Cost efficiency.

Figure 6 shows the cost comparison. SADCTS approach has a better cost of about 15%, 15%, 12% and 13% lesser than MBA-RDWOA, 10%, 12%, 7% and 7% lesser than QOAO-TS, 18%, 20%, 17% and 12% lesser than MVO-GA, 16%, 25%, 17% and 17% lesser than BAAEQRL, and 20%, 15%, 13% and 13% lesser than G-SOS methods, when the deadline ratio is 0.5, 1.0, 2.0 and 3.0, respectively.

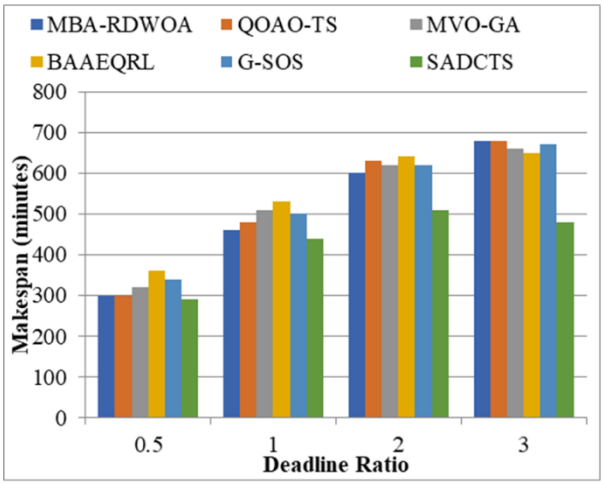


Figure 7. Makespan.

Figure 7 shows the makespan comparison. SADCTS approach has a better makespan of about 3.3%, 4.3%, 15% and 29% lesser than MBA-RDWOA, 3.3%, 8.33%, 19% and 29% lesser than QOAO-TS, 9.4%, 13.73%, 17.75% and 27.27% lesser than MVO-GA, 19.44%, 17%, 20% and 26% lesser than BAAEQRL, and 14.7%, 12%, 17.74% and 28.36% lesser than G-SOS methods, when the deadline ratio is 0.5, 1.0, 2.0 and 3.0, respectively. This improvement is predominantly beneficial for the task reassignment for effectively equating the overloaded VMs. It further confirms that greater resource conservation is significant in the suggested SADCTS method.

5. Conclusion

This paper addresses the problem of optimal task accomplishment in fusion clouds, privacy, and target necessities. The TS problem is modeled as a multi-objective non-linear encoding problem with a multi-level security model to ensure optimal performance. To solve this problem, SADCTS is proposed, which utilizes the hybrid optimization algorithm of MFSGCSA to solve the multi-objective problem to maximize the task completion rate and cost minimization with effective handling of deadline and security constraints. This proposed approach provides less task violation, high energy efficiency, high resource utilization, less cost efficiency, and less makespan to satisfy the necessary security and deadline requirements for TS in shared resource hybrid clouds. Although efficient, the proposed model is suitable for independent tasks, while the workflow-based tasks need to be investigated to adapt this model for real-time applications. In the future, the inter-dependent relationship among tasks will be investigated to identify the complexity-enhancing aspects of hybrid clouds. Additionally, the public cloud model will be exploited to improve the cost diversity for the reserve VMs.

Conflict of Interest

The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for profit sectors. The authors would like to thank the editor and anonymous reviewers for their comments that helped improve the quality of this work.

References

Abdullahi, M., Ngadi, M.A., Dishing, S.I., & Abdulhamid, S.I.M. (2023). An adaptive symbiotic organisms search for constrained task scheduling in cloud computing. *Journal of Ambient Intelligence and Humanized Computing, 14*(7), 8839-8850. https://doi.org/10.1007/s12652-021-03632-9.

Abualigah, L., & Alkhrabsheh, M. (2022). Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. *The Journal of Supercomputing, 78*(1), 740-765. https://doi.org/10.1007/s11227-021-03915-0.

Aithal, P.S., & Pai T.V. (2017). Opportunity for realizing ideal computing system using cloud computing model. *International Journal of Case Studies in Business, IT and Education, 1*(2), 60-71. https://ssrn.com/abstract=3083538.

Ajitha, K.M., & Indra, N.C. (2022). Fisher linear discriminant and discrete global swarm-based task scheduling in cloud environment. *Cluster Computing, 25*(5), 3145-3160. https://doi.org/10.1007/s10586-021-03509-8.

Alworafi, M.A., & Mallappa, S. (2018). An enhanced task scheduling in cloud computing based on a deadline-aware model. *International Journal of Grid and High-Performance Computing, 10*(1), 31-53. https://doi.org/10.4018/ijghpc.2018010103.

Attiya, I., Abualigah, L., Alshathri, S., Elsadek, D., & Abd Elaziz, M. (2022). Dynamic jellyfish search algorithm based on simulated annealing and disruption operators for global optimization with applications to cloud task scheduling. *Mathematics, 10*(11), 1894. https://doi.org/10.3390/math10111894.

Azizyan, G., Miarnaeimi, F., Rashki, M., & Shabakhty, N. (2019). Flying squirrel optimizer (FSO): A novel SI-based optimization algorithm for engineering problems. *Iranian Journal of Optimization, 11*(2), 177-205. https://dorl.net/dor/20.1001.1.25885723.2019.11.2.10.1.

Behera, I., & Sobhanayak, S. (2024). Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach. *Journal of Parallel and Distributed Computing, 183*, 104766. https://doi.org/10.1016/j.jpdc.2023.104766.

Bezdan, T., Zivkovic, M., Bacanin, N., Strumberger, I., Tuba, E., & Tuba, M. (2022). Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm. *Journal of Intelligent & Fuzzy Systems, 42*(1), 411-423. https://doi.org/10.3233/jifs-219200.

Braik, M.S. (2021). Chameleon swarm algorithm: A bio-inspired optimizer for solving engineering design problems. *Expert Systems with Applications, 174*, 114685. https://doi.org/10.1016/j.eswa.2021.114685.

Fu, X., Sun, Y., Wang, H., & Li, H. (2023). Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm. *Cluster Computing, 26*(5), 2479-2488. https://doi.org/10.1007/s10586-020-03221-z.

Hosseinzadeh, M., Ghafour, M.Y., Hama, H.K., Vo, B., & Khoshnevis, A. (2020). Multi-objective task and workflow scheduling approaches in cloud computing: A comprehensive review. *Journal of Grid Computing, 18*(3), 327-356. https://link.springer.com/article/10.1007/s10723-020-09533-z.

Kandan, M., Krishnamurthy, A., Selvi, S.A.M., Sikkandar, M.Y., Aboamer, M.A., & Tamilvizhi, T. (2022). Quasi-oppositional Aquila optimizer-based task scheduling approach in an IoT-enabled cloud environment. *The Journal of Supercomputing, 78*(7), 10176-10190. https://doi.org/10.1007/s11227-022-04311-y.

Khalil, I.M., Khreishah, A., & Azeem, M. (2014). Cloud computing security: A survey. *Computers, 3*(1), 1-35. http://dx.doi.org/10.3390/computers3010001.

Kiruthiga, G., & Mary Vennila, S. (2019). An enriched chaotic quantum whale optimization algorithm based job scheduling in cloud computing environment. *International Journal of Advanced Trends in Computer Science and Engineering, 8*(4), 1753-1760. https://doi.org/10.30534/ijatcse/2019/105842019.

Kiruthiga, G., & Vennila, S.M. (2020). Energy efficient load balancing aware task scheduling in cloud computing using multi-objective chaotic Darwinian chicken swarm optimization. *International Journal of Computer Networks and Applications, 7*(3), 82-92. https://doi.org/10.22247/ijcna/2020/196040.

Kiruthiga, G., & Vennila, S.M. (2021). Multi-objective task scheduling using chaotic quantum-behaved chicken swarm optimization (CQCSO) in cloud computing environment. In: Suma, V., Bouhmala, N., Wang, H. (eds) *Evolutionary Computing and Mobile Sustainable Networks* (pp. 803-814). Springer, Singapore, https://doi.org/10.1007/978-981-15-5258-8_74.

Lavanya, M., Shanthi, B., & Saravanan, S. (2020). Multi-objective task scheduling algorithm based on SLA and processing time suitable for cloud environment. *Computer Communications, 151*, 183-195. https://doi.org/10.1016/j.comcom.2019.12.050.

Mangalampalli, S., Swain, S.K., & Mangalampalli, V.K. (2022). Multi-objective task scheduling in cloud computing using cat swarm optimization algorithm. *Arabian Journal for Science and Engineering, 47*(2), 1821-1830. https://doi.org/10.1007/s13369-021-06076-7.

Manikandan, N., Gobalakrishnan, N., & Pradeep, K. (2022). Bee optimization-based random double adaptive whale optimization model for task scheduling in cloud computing environment. *Computer Communications, 187*, 35-44. https://doi.org/10.1016/j.comcom.2022.01.016.

Meng, S., Huang, W., Yin, X., Khosravi, M.R., Li, Q., Wan, S., & Qi, L. (2021). Security-aware dynamic scheduling for real-time optimization in cloud-based industrial applications. *IEEE Transactions on Industrial Informatics*, *17*(6), 4219-4228. https://doi.org/10.1109/tii.2020.2995348.

Nabi, S., Ahmad, M., Ibrahim, M., & Hamam, H. (2022). AdPSO: Adaptive PSO-based task scheduling approach for cloud computing. *Sensors*, *22*(3), 920. https://doi.org/10.3390/s22030920.

Panda, S.K., Nanda, S.S., & Bhoi, S.K. (2022). A pair-based task scheduling algorithm for cloud computing environment. *Journal of King Saud University-Computer and Information Sciences*, *34*(1), 1434-1445. https://doi.org/10.1016/j.jksuci.2018.10.001.

Saravanan, G., Neelakandan, S., Ezhumalai, P., & Maurya, S. (2023). Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing. *Journal of Cloud Computing*, *12*(1), 24. https://doi.org/10.1186/s13677-023-00401-1.

Sofia, A.S, & GaneshKumar, P. (2018). Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II. *Journal of Network and Systems Management*, *26*(2), 463-485. https://doi.org/10.1007/s10922-017-9425-0.

Wang, B., Wang, C., Huang, W., Song, Y., & Qin, X. (2021). Security-aware task scheduling with deadline constraints on heterogeneous hybrid clouds. *Journal of Parallel and Distributed Computing*, *153*, 15-28. https://doi.org/10.1016/j.jpdc.2021.03.003.

Wang, B., Wang, C., Song, Y., Cao, J., Cui, X., & Zhang, L. (2020). A survey and taxonomy on workload scheduling and resource provisioning in hybrid clouds. *Cluster Computing*, *23*(4), 2809-2834. https://doi.org/10.1007/s10586-020-03048-8.

Zhang, Z., Zhao, M., Wang, H., Cui, Z., & Zhang, W. (2022). An efficient interval many-objective evolutionary algorithm for cloud task scheduling problem under uncertainty. *Information Sciences*, *583*, 56-72. https://doi.org/10.1016/j.ins.2021.11.027.

Zhu, Q.H., Tang, H., Huang, J.J., & Hou, Y. (2021). Task scheduling for multi-cloud computing subject to security and reliability constraints. *IEEE/CAA Journal of Automatica Sinica*, *8*(4), 848-865. https://doi.org/10.1109/jas.2021.1003934.

Zubair, A.A., Razak, S.A., Ngadi, M.A., Al-Dhaqm, A., Yafooz, W.M.S., Emara, A.H.M., Saad, A., & Al-Aqrabi, H. (2022). A cloud computing-based modified symbiotic organisms search algorithm (AI) for optimal task scheduling. *Sensors*, *22*(4), 1674. https://doi.org/10.3390/s22041674.