Cluster Computing (2025) 28:848
https://doi.org/10.1007/s10586-025-05586-5

A hybrid DECSO algorithm for efficient multi objective task scheduling in cloud computing environments

Ali Gunduz[1] · **Sibel Senan**[1] · **Zeynep Gurkas-Aydin**[1]

Received: 17 December 2024 / Revised: 15 March 2025 / Accepted: 7 June 2025 / Published online: 19 September 2025

Abstract
One of the biggest problems in the rapidly developing cloud computing field in recent years is efficient task scheduling. Task scheduling in cloud computing is recognized as an NP-complete problem, presenting significant challenges due to the large task sizes and the complexity of efficiently managing diverse computational resources. Task scheduling in cloud computing aims to ensure that tasks are assigned to virtual machines to minimize completion time and maximize resource utilization. To address these challenges, this study introduces a novel hybrid optimization algorithm named Differential Evolution Cat Swarm Optimization (DECSO). Unlike traditional hybrid approaches, DECSO dynamically balances exploration and exploitation, ensuring a more adaptive and efficient task scheduling strategy. DECSO synergizes the global exploration ability and adaptive capabilities of Differential Evolution (DE) with the local search efficiency and explorative and exploitative strengths of Cat Swarm Optimization (CSO). The proposed DECSO algorithm is compared with PSO (Particle Swarm Optimization) and CSO via the CloudSim simulation experiment platform. DECSO's performance is evaluated using makespan, resource utilization, and migration time, which are critical metrics for efficient cloud task scheduling. The experimental results demonstrate that DECSO achieves up to 22.6% reduction in MakeSpan compared to CSO and 9.6% compared to PSO, 11.9% improvement in resource utilization compared to CSO and 14.7% compared to PSO, and 20.6% reduction in migration time compared to CSO and 11.2% compared to PSO. The results obtained from the simulation studies carried out show that the presented optimization model provides significant improvements in terms of MakeSpan, resource utilization, and migration time.

Keywords Cloud computing · Multitask scheduling · Cat swarm optimization · Differential evolution · Evolutionary approach

1 Introduction

Cloud computing has experienced a rapid rise in recent years, transforming the way organizations deploy, manage, and scale their IT services and infrastructure. Unlike traditional distributed computing approaches, cloud computing offers unparalleled elasticity and scalability in service delivery, enabling consumers to access computing resources on demand and in a pay-as-you-go manner [1]. The basis of cloud computing is the task dispatching mechanism, which involves allocating computing tasks to a resource pool composed of numerous heterogeneous virtualized servers, commonly referred to as virtual machines (VMs). This process is essential for optimizing the utilization of underlying resources and ensuring efficient service delivery [2]. To facilitate access to cloud resources and the provision of cloud services efficiently, a sophisticated scheduling mechanism is essential within the cloud computing paradigm. This mechanism plays a crucial role in dynamically allocating resources while optimizing system performance and resource utilization. In the cloud computing model, where numerous users submit requests simultaneously, each with potentially varying processing requirements, the complexity

✉ Sibel Senan
ssenan@iuc.edu.tr

Ali Gunduz
ali.gunduz@iuc.edu.tr

Zeynep Gurkas-Aydin
zeynepg@iuc.edu.tr

[1] Faculty of Engineering, Department of Computer Engineering, Istanbul University - Cerrahpasa, Avcilar, Istanbul 34320, Turkey

of task allocation escalates. Addressing this challenge necessitates an advanced scheduling mechanism capable of efficiently mapping user tasks to the appropriate virtual machines, ensuring that each task is running on a resource that matches its specific needs and processing capacity. Within the cloud computing framework, the simultaneous submission of requests by a multitude of users necessitates a robust intermediary, often a broker, responsible for managing these requests. The broker forwards the incoming tasks to a Task Manager, which verifies each request before passing it to the scheduler. The scheduler's role is important, as it orchestrates the mapping of a vast number of requests onto the available virtual machines (VMs) located within physical servers. This intricate process involves matching each request with a VM that not only has the requisite processing capacity but also optimizes key parameters such as MakeSpan.

Task scheduling is a critical component of cloud computing that directly affects the efficiency and performance of cloud services [3]. Optimizing various performance metrics such as Resource utilization, MakeSpan, and Migration time when assigning computing tasks to virtualized resources such as virtual machines (VMs) is one of the main goals of task scheduling in cloud computing environments. Traditional task scheduling typically focuses on a single objective like minimizing completion time or optimizing resource utilization while multi-objective scheduling addresses a combination of goals. Therefore, multi-objective task scheduling is an advanced approach that aims to simultaneously optimize various performance metrics, such as increasing resource utilization while reducing MakeSpan and migration time.

Multi-objective task scheduling in cloud computing is inherently challenging due to the conflicting nature of its objectives. Minimizing MakeSpan often requires maximizing resource utilization, leading to a trade-off between execution efficiency and computational load. Additionally, optimizing migration time to reduce task relocation can negatively impact system-wide load balancing. The complexity of this problem is further exacerbated by the high-dimensional search space, where the number of possible task-to-VM assignments grows exponentially with the number of tasks and available resources. Moreover, cloud environments are highly dynamic, with fluctuating workloads and continuously changing resource availability, making static scheduling approaches ineffective. Traditional scheduling methods often focus on a single objective, neglecting the trade-offs necessary for holistic performance optimization. Thus, an adaptive and hybrid approach is required to effectively balance multiple objectives in real-time cloud computing environments.

The complexity of task scheduling arises from the need to manage numerous, often heterogeneous, computing resources and to handle a wide range of task requirements and priorities. Given its complexity, task scheduling is known as an NP-complete problem, posing significant challenges in finding optimal solutions within reasonable computational times. Given the NP-complete characteristics of the task scheduling problem in cloud computing, numerous researchers have turned to nature-inspired algorithms, drawing on their proven efficacy in navigating complex optimization landscapes to devise efficient scheduling solutions. Each new approach that improves the optimization of key performance metrics, such as MakeSpan, resource utilization, and migration time, plays a crucial role in advancing the state-of-the-art in cloud task scheduling. Numerous existing studies have explored the problem of task scheduling in cloud computing by employing a variety of nature-inspired algorithms. These algorithms are known for their effectiveness in finding near-optimal solutions within polynomial time, as opposed to the exponential time often required by conventional methods. Despite the extensive research on cloud task scheduling using metaheuristic approaches, existing methods often fail to simultaneously optimize multiple conflicting objectives. Many studies focus on reducing MakeSpan while neglecting resource utilization efficiency or migration overhead. Moreover, several hybrid algorithms, such as DE-PSO and WOA-based approaches, have been proposed to improve task scheduling, but they often struggle with premature convergence and imbalanced exploration-exploitation trade-offs.

To address these limitations, DECSO integrates Differential Evolution (DE) and Cat Swarm Optimization (CSO) to dynamically balance global exploration and local exploitation. Unlike conventional hybrid models, DECSO enhances adaptability in dynamic workloads and ensures robust task allocation across heterogeneous cloud resources. By combining DE's superior search and adaptive capabilities with CSO's efficient local refinement, our proposed method achieves a better trade-off among MakeSpan, resource utilization, and migration time, effectively filling the research gap in multi-objective cloud task scheduling.

Differential Evolution (DE), provides efficient solutions to complex optimization problems. As a powerful evolutionary algorithm, DE has the advantage of a small number of control variables but also exhibits effective convergence performance [4]. Cat Swarm Optimization Algorithm (CSO) is an algorithm inspired by how cats behave in their natural lives. CSO is an effective metaheuristic method that can provide a balance between local search and global search with its seeking and tracing mode [5].

In this study, we introduce a hybrid method called Differential Evolution Cat Swarm Optimization (DECSO) to

address the problem of multi-objective task scheduling in cloud environments. This innovative method combines the strengths of CSO and DE by combining these two powerful algorithms, DECSO improves overall performance by maximizing resource utilization, reducing MakeSpan and migration time. This study presents a detailed implementation of DECSO by performing extensive simulations using the CloudSim toolkit [6] and demonstrates its efficiency in managing up to 1,000 tasks in a cloud environment. MakeSpan, migration time, and resource utilization are used as performance metrics to evaluate our method. The results indicate that our approach goes beyond traditional, thereby offering a more sustainable and cost-effective solution to address multi-objective task scheduling problems in cloud computing environments.

The main highlights of the paper are mentioned below.

- In this research, we developed a hybrid algorithm called DECSO for multi-objective task scheduling by leveraging the principles of Cat Swarm Optimization and advanced optimization strategies of Differential Evolution methods.
- Evaluations were provided for the proposed algorithm by comparing it with DE and CSO on CloudSim. Simulations managing up to 1,000 tasks in the cloud environment were executed using CloudSim.

The results show that the proposed DECSO hybrid algorithm has good convergence speed and search ability, and it performs better in resource utilization, migration time and MakeSpan ability.

2 Related work

Task scheduling remains a challenging problem in cloud computing because of the inherent complexity and dynamic characteristics of cloud environments. The need to manage heterogeneous resources and diverse user requirements adds to the complexity, making efficient task scheduling crucial for optimizing cloud performance.

Over the years, numerous mathematical models have been developed to handle the task scheduling problem. Such approaches often rely on linear programming, integer programming, and other optimization techniques to formulate and solve scheduling problems. For instance [7], represented the relationship between the due date and fetched on crossover clouds as a blended numbers nonlinear programming issue with usage in a numerical programming language (AMPL).

To overcome the limitations of mathematical methods, heuristic approaches have been extensively used in recent years. Heuristic algorithms, such as greedy algorithms, priority-based scheduling, and list scheduling, offer faster solutions by simplifying the decision-making process. In [8], a cost-effective task-scheduling algorithm by implying two heuristic techniques according to the thought of Pareto dominance was proposed. While these methods are computationally efficient, they often result in suboptimal solutions as they do not explore the entire solution space. A multi-objective task scheduling algorithm introduced by [9], mapping tasks to virtual machines to improve data center throughput and minimize costs without compromising SLA requirements. The algorithm takes into account multiple criteria including execution time, cost, and user bandwidth to provide optimal scheduling in cloud environments. [10] used heuristic approach for optimizing resource allocation in task scheduling in cloud computing. In [11], a novel task scheduling algorithm derived from the RAO algorithm, leverages heuristic-based methods to reduce search space and time to find optimal solutions efficiently.

Since the task planning problem is an NP-complete complexity problem, metaheuristic algorithms that can achieve high accuracy rates in short times have become widely used in this field. As Approaches like Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Genetic Algorithms (GA)have been effectively applied to task scheduling. These approaches drawing inspiration from natural phenomena and employ iterative methods to explore the solution space effectively.

Genetic algorithms, have proven effective in optimizing task allocations, with notable applications demonstrated by studies such as those by [12] and [13], which explore genetic approaches to task scheduling optimization in cloud settings. Furthermore [14], presented a genetic algorithm that enhances task scheduling efficiency by minimizing completion time and costs, and optimizing resource utilization in cloud computing. [15] introduced a genetic algorithm that significantly improves task scheduling in cloud services by optimizing resource allocation to minimize both completion time and cost. [16] introduced a modified genetic algorithm integrated with a greedy strategy (MGGS), which improves task scheduling by reducing total completion time and enhancing QoS parameters. [17] presented the Balancer Genetic Algorithm, a novel approach that provides an optimized scheduling solution focusing on balancing workload and reducing energy usage in cloud systems. Cloud computing has also seen an adaptation of Particle Swarm Optimization by [18], who enhanced its efficacy for dynamic task scheduling. [19] illustrated the ongoing evolution and adaptation of scheduling algorithms that address the specific needs of cloud computing environments. [20] developed a PSO-based task scheduling model that significantly minimizes processing costs by optimizing transferring and

processing times in cloud computing environments. [21] introduced an LBMPSO algorithm that optimizes task scheduling by considering reliability and availability, and outperforms standard PSO in execution time and load balancing. [22] proposed an improved PSO with iterative methods to enhance time efficiency and prevent local optimization in task scheduling. [23] proposed a modified PSO algorithm that dynamically adjusts the inertia weight coefficient aimed at enhancing convergence speed and reduce energy consumption in cloud computing environments. [24] introduced a hybrid artificial bee colony algorithm for multi-objective task scheduling problem that effectively navigates the complexities of cloud workflow scheduling by simultaneously reducing completion time, device workload, and energy consumption. [25] developed an evolutionary algorithm-based model that addresses multi-objective optimization by focusing on reliability, cost, MakeSpan, and energy consumption, offering a comprehensive solution for complex cloud environments. In [26], Ant colony optimization has been utilized to efficiently schedule diverse tasks with innovative strategies, focusing on optimizing execution time and cost simultaneously. Furthermore, [27] introduced an ACO-based task scheduling policy designed to reduce MakeSpan, thereby improving the processing capacity and efficiency of cloud computing systems. [28] combines genetic algorithms with ant colony optimization to overcome limitations and enhance efficiency in cloud computing environments. [29] proposed an MQoS-GAAC algorithm that integrates ant colony optimization and genetic algorithm with multi-QoS constraints to improve the performance of task scheduling in cloud computing. [30] proposed a SACO algorithm that leverages slave ants for more accurate pheromone trail management, resulting in improved task scheduling and resource utilization in cloud environments. [31] proposed a Modified Ant Colony Optimization (MACO) algorithm that focuses on multi-objective task scheduling in cloud computing. The algorithm focuses on reducing MakeSpan and optimize resource utilization by assigning pheromone amounts based on virtual machine efficiency. [32] presented an improved ACO-based task scheduling model that addresses issues like load balance and fast convergence, showing significant improvements in scheduling efficiency and resource utilization. [33] introduced Cat Swarm Optimization based Simulated Annealing algorithm to solve multi objective task scheduling problem in cloud computing. This method enhances local search procedures with improved simulated annealing to adapt dynamically changing tasks and resources, minimizing execution time and cost. [34] proposed a fuzzy self-defense approach for multi-objective task scheduling problem in cloud computing. This approach aims to improve task completion time, resource load balance, and minimize costs. [35] proposed a task scheduling algorithm derived from the coronavirus herd immunity optimizer (CHIO) to optimize scheduling in cloud computing. This method using meta-heuristic algorithms to discover optimal solutions. [36] introduced an Enhanced Marine Predator Algorithm (EMPA) for task scheduling in cloud computing environments. Their approach improves the standard Marine Predator Algorithm (MPA) by integrating Whale Optimization Algorithm (WOA) operators, a nonlinear inertia weight coefficient, and a golden sine strategy to enhance convergence speed and global search capabilities.

An advanced method for task scheduling in cloud computing was proposed by [1], utilizing the Whale Optimization Algorithm (WOA) with a multi-objective optimization approach. Their strategy called Improved WOA (IWC) for Cloud Task Scheduling aims to boost the WOA's ability to search for the task scheduling solutions. The research aims to enhance the efficiency of cloud systems using computing resources. It takes into account factors, like task completion time, system workload and cost effectiveness. In CloudSim experiments using simulations as a basis, for testing the IWC showed results in terms of how it reached optimal outcomes and the level of accuracy achieved when compared to conventional metaheuristic algorithms. It also proved to be more efficient, in utilizing resources and cost effective in handling tasks of sizes.

[37], to solve the complicated task of scientific workflow scheduling in cloud computing, have put forward an Improved Many-Objective Particle Swarm Optimization algorithm (MOPSO). Their study focuses on optimizing workflow scheduling, which is a acknowledged NP-complete problem, by focusing on four conflicting objectives: increasing reliability and reducing cost, energy consumption, and MakeSpan. They introduced four significant improvements to boost the traditional MOPSO algorithm's ability to converge to optimal solutions while maintaining a balance between exploration and exploitation in the scheduling process. Their test results demonstrated that the suggested MOPSO algorithm significantly outperforms existing algorithms like LEAF, MOPSO, and EMS-C in terms of the HyperVolume criterion. [38] introduced a hybrid and lightweight metaheuristic approach for multi-objective resource scheduling and application placement in fog environments. Their method, Enhanced Multi-Objective Particle Swarm Optimization with Clustering (EMOPSOC), integrates evolutionary computation with machine learning-based clustering techniques to optimize scheduling decisions dynamically. The proposed algorithm minimizes network latency, resource wastage, and load imbalance, improving overall efficiency.

[39] have tackled the complex issue of multiobjective cloud workflow scheduling. Their study models addressing cloud workflow scheduling as a multiobjective optimization

challenge, aiming to optimize execution time and cost at the same time. To this end, they introduced an advanced multi-objective ant colony system that leverages a mutual evolutionary framework with diverse populations, each dedicated to a specific objective. This approach incorporates innovative elements such as an updated pheromone update mechanism guided by a group of Pareto-optimal solutions and a complementary heuristic strategy to balance the optimization of both objectives. Their comprehensive evaluation, utilizing real-world scientific workflows and reflecting the characteristics of the Amazon EC2 cloud platform,

[40] proposed an innovative approach to Cloud computing environments and multi-objective task scheduling by introducing a hybrid antlion optimization method enhanced with elite-based differential evolution, termed MALO. This novel method addresses the challenges of scheduling in cloud computing by optimizing for both resource utilization and MakeSpan simultaneously. The antlion optimization algorithm's search efficiency is significantly improved through the incorporation of elite-based differential evolution, enhancing its ability to bypass local optima and ensuring a more comprehensive exploration of the solution space. The efficacy of MALO was rigorously tested against traditional and contemporary algorithms utilizing the CloudSim, across both simulated and real-world trace datasets.

[41] explored the challenge of task scheduling in cloud computing by leveraging the Grey Wolf Optimizer (GWO), inspired by the hunting patterns of grey wolves. Their research underscores the NP-complete characteristics of task scheduling, where the primary objectives include identifying suitable resources for timely task execution, enhancing optimization of resource use, and minimizing the overall time to complete tasks. The proposed GWO-based scheduling algorithm was tested for its robustness and efficiency. [42] introduced a multi-objective task scheduling algorithm in cloud computing based on GWO. Their approach makes scheduling decisions dynamically at runtime by considering the current status of cloud resources and incoming workload demands. The method also takes into account user budget constraints and task priorities to allocate resources more effectively. [43] introduced a new hybrid multi-objective optimization algorithm for task scheduling in heterogeneous cloud environments. The proposed method combines the pollination behavior of flowers with the search exploration capability of the Grey Wolf Optimizer (GWO) and further incorporates evolutionary algorithm crossover operators to improve the balance between exploration and exploitation. [44] introduced a hybrid task scheduling algorithm that combines Genetic Algorithm (GA) with Grey Wolf Optimization (GWO) to enhance both exploration and exploitation in cloud computing environments. The proposed approach incorporates GA's crossover and mutation operations into GWO to mitigate premature convergence and improve solution diversity.

Metaheuristic algorithms are often utilized in cloud computing due to their flexibility, adaptability, and robustness. They offer a practical means to tackle the complexity of task scheduling by balancing exploration and exploitation of the solution space, making them suitable for dynamic and large-scale environments. However, these methods present several trade-offs. PSO, while effective in swarm-based search, suffers from premature convergence and lacks an efficient local refinement mechanism, making it less effective in balancing exploration and exploitation. ACO, inspired by pheromone-based path optimization, excels in finding optimal routes but is computationally expensive in large-scale dynamic environments. GA, known for its powerful global search capabilities through crossover and mutation, often struggles with slow convergence and requires extensive parameter tuning.

In contrast, Differential Evolution (DE) and Cat Swarm Optimization (CSO) offer a complementary hybrid approach that overcomes these limitations. DE provides a strong global search mechanism through adaptive mutation and crossover, allowing it to efficiently explore diverse solutions while avoiding premature convergence. CSO, on the other hand, introduces a structured local search mechanism through its seeking mode, which refines solutions with high precision. This combination allows DECSO to effectively balance exploration and exploitation, making it an ideal choice for multi-objective task scheduling in cloud computing environments.

Recent advancements in hybrid optimization algorithms have highlighted the potential of integrating DE with other heuristic methods to enhance task scheduling outcomes. For instance, [45] introduced a hybrid model combining the WOA with DE to enhance virtual machine scheduling processes in cloud environments. Their work demonstrates improved efficiency in handling multiple objectives, including balancing load and reducing energy consumption, which are crucial for sustainable cloud operations.

Similarly, [46] explored the synergy between genetic algorithms and DE, applying these techniques applied to multi-objective task scheduling within cloud computing. Their study showcased how the collective advantages of these algorithms could achieve superior optimization regarding cost, time, and resource utilization.

Furthermore, [47] developed a novel approach by integrating the moth search algorithm with DE to tackle task scheduling challenges in cloud systems. Their research, published in Knowledge-Based Systems, not only enhanced the scheduling efficiency but also contributed to the robustness and scalability of the solutions, demonstrating the versatility of DE in hybrid optimization settings.

While these methods leverage DE's global search capabilities, they often suffer from premature convergence, where the algorithm settles into suboptimal solutions too early. Additionally, DE-PSO hybrids tend to lack a robust local search mechanism, making them inefficient in fine-tuning solutions. WOA-based hybrids [45], on the other hand, struggle with balancing exploration and exploitation, leading to fluctuating performance in dynamic environments.

DECSO overcomes these limitations by combining DE's adaptive global exploration with CSO's structured local refinement strategy. Unlike DE-PSO, which mainly relies on particle-based movements, DECSO's seeking mode in CSO provides a more controlled local search, preventing premature convergence. Additionally, unlike DE-WOA, which lacks adaptive search intensity, DECSO dynamically adjusts exploration-exploitation trade-offs, making it more suitable for dynamic and heterogeneous cloud environments.

These studies have predominantly focused on optimizing key performance metrics, with MakeSpan, the time required to complete all tasks, being a primary consideration due to its critical impact on overall system efficiency. Beyond MakeSpan, several researchers have utilized additional metrics including, total execution cost, execution time, and energy consumption, to measure the performance and sustainability of cloud computing services.

With the growing demand for cloud computing continues to expand, the development of effective and efficient task-scheduling algorithms remains critical. These strategies not only enhance the immediate performance of cloud services While ensuring both long-term sustainability and adaptability, underlining the importance of continued research and innovation in this field.

In this study, we propose an innovative approach called Differential Evolution Cat Swarm Optimization (DECSO) to achieve more efficient task scheduling. By integrating advanced optimization techniques from both CSO and DE, DECSO focuses on improving accuracy and convergence speed, addressing the multiobjective nature of cloud task scheduling problems. Through extensive simulations and performance evaluations, we demonstrate DECSO's effectiveness in improving MakeSpan, resource utilization, and migration time.

An overview of the related works, including the algorithms used, evaluation parameters, and limitations, is presented in Table 1.

3 Materials and methods

This section introduces the basic concepts of Differential Evolution (DE) and conventional Cat Swarm Optimization (CSO) methods.

3.1 Differential evolution

DE is an optimization algorithm introduced by [48]. DE is known for its robust and straightforward nature. It proves to be highly efficient for optimizing complex, multi-modal functions and is widely applied in different fields like machine learning, engineering, and scientific research. DE iteratively enhances a candidate solution in relation to a specified quality metric known as the fitness function.

The algorithm keeps a population of candidate solutions, each represented as a vector in the solution space. Unlike traditional optimization methods, DE does not require gradient information and instead relies on the differences between randomly chosen pairs of solutions to drive the search process. DE operates through three main processes: mutation, crossover and selection.

Mutation: At each generation , for every individual $X_i^t = \{x_{i,1}^t, x_{i,2}^t, \ldots, x_{i,n}^t\}$ $(i \in \{1,2,\ldots,NP\})$, a mutant vector $Y_i^t = \{y_{i,1}^t, y_{i,2}^t, \ldots, y_{i,n}^t\}$ is generated by applying one of the mutation strategies. These strategies increase the diversity of the population and improve the efficiency of the search space. The most commonly used mutation strategies are:

- Rand/1: $y_{i,j}^t = x_{r[1],j}^t + F \cdot \left(x_{r[2],j}^t - x_{r[3],j}^t\right)$
- Best/1: $y_{i,j}^t = x_{\text{best},j}^t + F \cdot \left(x_{r[1],j}^t - x_{r[2],j}^t\right)$
- Current to best/1: $y_{i,j}^t = x_{i,j}^t + F \cdot \left(x_{\text{best},j}^t - x_{i,j}^t\right) + F \cdot \left(x_{r[1],j}^t - x_{r[2],j}^t\right)$
- Best/2: $y_{i,j}^t = x_{\text{best},j}^t + F \cdot \left(x_{r[1],j}^t - x_{r[2],j}^t\right) + F \cdot \left(x_{r[3],j}^t - x_{r[4],j}^t\right)$
- Rand/2: $y_{i,j}^t = x_{r[1],j}^t + F \cdot \left(x_{r[2],j}^t - x_{r[3],j}^t\right) + F \cdot \left(x_{r[4],j}^t - x_{r[5],j}^t\right)$

In these formulas, r[k] (k ∈ {1, 2,…, 5}) are distinct random indices uniformly sampled from the range [1, NP], with i ∉ r[k]. F is a scaling factor that regulates the amplification of the differential variation, usually selected from the range [0,2].

Following mutation, a crossover operation is performed to generate a trial vector $Z_i^t = \{z_{i,1}^t, z_{i,2}^t, \ldots, z_{i,n}^t\}$. This operation combines the genes of the mutant vector Y_i^t and the target vector X_i^t based on a predefined crossover probability (CR). This approach encourages diversity while still leveraging certain aspects of the search space.

The crossover formula is given as:

Table 1 An overview of the related works

Algorithm	References	Parameters	Limitations	Year
PSO	[18]	Makespan, Resource Utilization	Premature convergence, weak local search	2023
MOPSO	[37]	Makespan, Energy Consumption, Cost	Computational overhead, requires fine-tuning	2016
ACO-based	[26]	Execution Time, Workflow Efficiency	High computational cost, slow convergence	2018
GWO	[41]	Load Balancing, Resource Utilization	Weak diversification, stagnation in large-scale problems	2019
Hybrid WOA-DE	[45]	Makespan, Energy Efficiency, Load Balancing	Parameter sensitivity, risk of performance instability	2022
GA-GWO Hybrid	[44]	Makespan, Execution Cost, SLA Compliance	Complex tuning process, high dependency on initialization	2024
Enhanced Moth Search Algorithm with DE	[47]	Convergence Speed, Search Efficiency, Makespan	Hybrid model complexity, requires additional optimization	2021
Hybrid PSO-DE	[46]	Makespan, Resource Utilization	High dependency on parameter initialization	2020
Fuzzy Self-Defense Approach	[34]	Task Completion Time, Cost Optimization	Requires expert knowledge for fuzzy rule definitions	2021
Antlion Optimization	[40]	Makespan, Execution Time, Cost	Slow convergence, prone to local optima	2021
Whale Optimization Algorithm	[1]	Makespan, Load Balancing, Energy Savings	Requires extensive parameter tuning, unstable performance in dynamic workloads	2020
Modified ACO	[31]	Workflow Execution Time, Cost Efficiency	Computationally expensive, parameter sensitivity	2018
Genetic Algorithm	[12]	Load Balancing, Task Allocation	Slow convergence, requires evolutionary tuning	2018
Deep Reinforcement Learning	[50]	Adaptive Scheduling, Performance Optimization	High computational cost, requires deep learning infrastructure	2023
Hybrid Marine Predator Algorithm	[36]	Makespan, Resource Allocation, Efficiency	Parameter dependency, needs large-scale testing	2024

$$z_{i,j}^{t} = \begin{cases} y_{i,j}^{t} & \text{if rand} \leq \text{CR or } j = j_{\text{rand}} \\ x_{i,j}^{t} & \text{otherwise} \end{cases}. \quad (1)$$

In this context, *rand* represents a random number drawn from a uniform distribution within the range [0, 1], while j_{rand} is a randomly selected index to guarantee that at least one gene is derived from the mutant vector Y_i^t.

After the trial vector $Z_i^t = \left\{z_{i,1}^t, z_{i,2}^t, \ldots, z_{i,n}^t\right\}$ is created through the crossover process, a selection operation takes place. The offspring individual Z_i^t competes with the parent individual X_i^t based on a greedy criterion, and the survivor advances to the $t + 1$ generation. The selection process is outlined as follows:

$$X_i^{t+1} = \begin{cases} Z_i^t & \text{if} f\left(Z_i^t\right) \leq f\left(X_i^t\right), \\ X_i^t & \text{otherwise.} \end{cases} \quad (2)$$

Here $f(\cdot)$ represents the fitness function, which evaluates the quality of solution. The process guarantees the evolution of the population by selecting the superior solution between Z_i^t and X_i^t. Various techniques are integrated into DE to solve constraint optimization problems (COPs). Constraint adaptation by differential evolution (CADE) [49] is one of these techniques. This technique merges the concepts of constraint adaptation and DE, employing a region of acceptability (ROA) as its selection mechanism. If Z_i^t falls within the ROA, it is accepted as $X_i^{t+1} = Z_i^t$. If not, the DE procedure is repeated multiple times. If the generated offspring still be outside the ROA, the solution remains as $X_i^{t+1} = X_i^t$.

The DE algorithm iterates through these steps, the execution continues until a stopping criterion, such as a maximum number of generations or convergence to a satisfactory solution, is met. Its simplicity, ease of implementation, and robustness make DE a popular choice to resolve a variety of optimization problems. The pseudo-code of the Differential Evolution Algorithm is given in the following:

Algorithm 1 : The procedure of the Differential Evolution Algorithm

Initialize parameters: Population size (P_s), Maximum Generations (G_{max}), Crossover Rate (CR), Scaling Factor (F)
Initialize population with random task-to-VM assignments
Evaluate fitness of each individual in the population
FOR each iteration from 1 to G_{max} DO
  FOR each individual X_i in the population DO
    Select three random individuals X_{r1}, X_{r2}, X_{r3} ($r_1 \neq r_2 \neq r_3 \neq i$)
    Perform mutation: $V_i = X_{r1} + F * (X_{r2} - X_{r3})$
    Perform crossover to generate trial vector U_i
    IF rand() < CR THEN
      $U_{i[j]} = V_{i[j]}$
    ELSE
      $U_{i[j]} = X_{i[j]}$
    END IF
    Evaluate fitness of U_i
    Selection: If fitness(U_i) is better than fitness(X_i), replace X_i with U_i
  END FOR
END FOR
Return the best individual as the optimized task schedule

3.2 Cat swarm optimization

CSO is an algorithm based on the natural actions of cats, developed by [50]. This algorithm mimics the way cats behave, alternating between periods of activity and rest. In CSO, cats are considered as candidate solutions, aiming to reach an optimal target by cycling through various cats and optimizing the solution over time. The algorithm operates with cats in two primary modes: seeking mode and tracing mode. Cats are at rest in seeking mode but remain alert, exploring their surroundings passively. In tracing mode, cats actively chase targets, moving towards potential solutions. This operation carries on iteratively until a predefined number of iterations is completed. Initially, all cats are randomly distributed across the solution space, which is defined by the problem's dimensions. These cats are then divided into seeking and tracing modes based on the Seeking Memory Ratio (SMR), which determines the proportion of cats in each mode. In each cycle, solutions are evaluated, and the fitness of all cats is measured until the best result is achieved.

During the seeking mode, the cats go into resting mode and explore potential solutions in their vicinity. When cats switch to tracing mode, they actively pursue the best solutions found so far.

3.2.1 Seeking mode

In the seeking phase, cats are considered to be in a passive state but are constantly alert, exploring potential solutions around their current positions. This phase is designed to mimic the exploratory and curious nature of cats when they are at rest but attentive to their environment. Several candidate solutions, also known as "kittens," are generated around the current position of each cat by adding a small perturbation to the cat's position. Seeking Range of Dimensions (SRD) determines the range of this perturbation, allowing for a dynamic adjustment of the search area. Each candidate position's fitness is then evaluated based on a predefined fitness function, assessing how close the candidate is to the optimal solution. The cat's position is updated to the best candidate position based on the fitness evaluation, ensuring that the cat moves towards a better solution while still in the seeking mode.

3.2.2 Tracing mode

In the tracing phase, cats actively chase their targets, representing an active search for optimal solutions. This mode simulates the hunting behavior of cats when they are actively pursuing prey. Each cat moves towards the best result obtained up to this point, known as the global best position (g). The position update formula in tracing mode is given in (1):

$$x_i(t+1) = x_i(t) + c.(g - x_i(t)) \tag{3}$$

where $x_i(t)$ represents current position of the i_{th} cat at iteration t, c is a constant and g represents the global best position, representing the speed of movement towards the

target. The new positions of the fitness function is used to evaluate the cats, and if a cat's new position yields a better fitness value, it becomes the new best solution. In tracing mode, the system is repeated for each cat in the population, with cats continuously updating their positions towards the global best position until the termination conditions, such as an acceptable fitness level or the maximum iterations are met.

The CSO algorithm alternates between the seeking and tracing modes to balance exploitation and exploration. This combined approach ensures a Comprehensive exploration of the solution space, utilizing the strengths of both modes. By iterating through these modes, CSO is capable of efficiently navigating complex optimization landscapes and discovering high-quality solutions. The key advantages of CSO include its simplicity, ease of implementation, and capability to discover top-tier solutions for difficult optimization problems. It has been effectively used in various fields, such as scheduling, engineering and machine learning. The pseudo-code of the Cat Swarm Optimization Algorithm is given in the following:

Algorithm 2 : The procedure of the Cat Swarm Optimization Algorithm

```
Initialize parameters: number of cats (N), seeking mode ratio (SMR), seeking memory pool (SRD),
   maximum iterations (MaxIter), etc.
Initialize the positions of all cats in the solution space
Measure the fitness of each cat and locate the global optimal position (g)
FOR each iteration from 1 to MaxIter DO
   FOR each cat i in the population DO
      Randomly assign cat i to seeking mode or tracing mode based on SMR
      IF cat i is currently in seeking mode THEN
         Generate K candidate solutions around the current position of cat i
         Measure the fitness of each candidate position
         Select the best candidate position as the new position of cat i
      ELSE IF cat i is in tracing mode THEN
         Update the position of cat i towards the global best position (g)
      END IF
   END FOR
   Measure the fitness of all cats
   Update the global best position (g) if a better solution is found
END FOR
Return the global best position (g) as the optimal solution
```

4 DECSO model

This study presents a novel optimization methodology named the Differential Evolution Cat Swarm Optimization (DECSO) model which integrates the exploitative capabilities of CSO with the adaptive strategies of DE. Simulation studies were carried out to verify the performance of the theoretical framework. This hybrid model is designed to efficiently manage the multi-objective nature of task scheduling, focusing on MakeSpan, resource utilization, and migration time.

The process begins with DE's explorative capabilities to initialize the population of cats in CSO. Instead of randomly placing the cats, DE conducts a thorough search in the solution space, generating high-quality initial solutions. These solutions, produced by DE, serve as a starting point for the CSO algorithm. Once the initial population is established, CSO takes over, applying its unique mechanisms to further refine the solutions. In each iteration, the best cat identified by the CSO is forced into the seeking mode, ensuring a detailed local search around the current best solution. This strategic integration combines DE's global search effectiveness with CSO's local search refinement to achieve multi-objective task scheduling that minimizes completion time, maximizes resource utilization, and reduces migration time. The combination of these methods allows for a comprehensive and adaptive approach to solving the complex, problem of multi-objective task scheduling in cloud computing environments.

The algorithm for the proposed DECSO model is presented in Algorithm 3.

Algorithm 3 : DECSO

Input: Population Size ($\mathbf{P_s}$), Max Generations ($\mathbf{G_{max}}$), Objective Function (**f**)
CSO Parameters: Mixture Ratio (**MR**), Seeking Memory Pool (**SMP**), Seeking Range of the Dimensions (**SRD**), Counts of Dimension to Change (**CDC**),
DE Parameters: Crossover Rate ($\mathbf{C_r}$), Scaling Factor (**F**), Problem Dimension (**d**)

Output: The individual who has the best fitness value. ($\mathbf{X_{best}}$)

Initialize DE Parameters: Set $\mathbf{C_r}$, **F**, **d** for Differential Evolution.

Generate an initial random population X using DE with size $\mathbf{P_s}$:

For each X **do**
Mutation: Generate a mutant vector $\mathbf{V_i}$ using Eq. (4).
Crossover: Generate a trial vector $\mathbf{Z_i}$ controlled by $\mathbf{C_r}$ using Eq. (1).
Selection: Calculate the fitness for $\mathbf{Z_i}$ and compare it with $\mathbf{X_i}$.
IF $\mathbf{f(Z_i) > f(X_i)}$ **THEN** $\mathbf{X_i = Z_i}$

Evaluate the fitness value for each individual $\mathbf{X_i}$ in **X** and identify the best cat ($\mathbf{X_{best}}$)

i = 1; do while ($\mathbf{i \geq G_{max}}$)**:**
Generate a random number `**r**` in the range [0,1]
IF r < **MR** or $\mathbf{X_i = X_{best}}$
Generate **SMP** candidate solutions around the current position of $\mathbf{X_i}$ by changing **CDC** dimensions according to **SRD**.
Evaluate the fitness value of each candidate
Select the best candidate as the new position of $\mathbf{X_i}$
ELSE
Evaluate new velocity $\mathbf{V_i}$ for $\mathbf{X_i}$ by Eq. (3).
Change the position of $\mathbf{X_i}$ by $\mathbf{V_i}$ and evaluate fitness value for $\mathbf{X_i}$
IF $\mathbf{f(X_i) > f(X_{best})}$
$\mathbf{X_{best} = X_i}$
Return $\mathbf{X_{best}}$

In our proposed methodology, Differential Evolution (DE) is employed as the initial step in the hierarchical process for scheduling tasks in cloud computing. DE is a population-based optimization algorithm regarded for its ease of use and efficiency in handling continuous optimization problems. The DE algorithm begins with the initialization phase, where a population of potential solutions is randomly created within the solution space. Each solution represents a possible assignment of tasks to VMs.

In the context of task scheduling, each individual in the DE population represents a potential assignment of tasks to VMs. The goal is to find an optimal assignment that maximizes resource utilization, minimizes MakeSpan, and reduces migration time. During initialization, each x_i is encoded as a vector representing the assignment of tasks T to VMs . During the mutation and crossover steps, new task assignments are generated by exploring the solution space, helping to find diverse solutions and avoid local optima. The mutation formula for creating a mutant vector is given in (4):

$$v_i^{t+1} = x_{r1}^t + F.\left(x_{r2}^t + x_{r3}^t\right) \tag{4}$$

Where v_i^{t+1} is the mutant vector, x_{r1}^t, x_{r2}^t, x_{r3}^t are individuals randomly selected from the population and F represents a scaling factor.

Using DE in the initial step provides several advantages for task scheduling. DE's capability to effectively explore the global solution space helps in finding high-quality solutions that balance multiple objectives. Its simplicity and computational efficiency make it suitable for large-scale cloud environments. Additionally, DE can adapt to various optimization problems by tuning parameters F and C_r, offering flexibility in handling different task scheduling scenarios. By starting with a robust exploration of the solution space, DE ensures better overall performance in managing cloud resources, laying a strong foundation for the subsequent steps in our hierarchical approach. In DECSO, DE

uses a crossover rate of 0.9 to ensure a high probability of exchanging genes between the mutant and target vectors, promoting diversity and preventing premature convergence. The mutation factor is set to 0.8, allowing a sufficient degree of variation in the population while maintaining stability. These values were selected through experimental tuning to achieve a balance between exploration and exploitation, ensuring effective task scheduling in cloud environments.

Following the Differential Evolution (DE) phase, we employ Cat Swarm Optimization (CSO) to further refine the task scheduling solutions. CSO is a metaheuristic algorithm inspired by the natural actions of cats, combining seeking and tracing modes to balance exploitation and exploration in the search space. In CSO, every cat denotes a possible solution, with its position in the search space corresponding to a specific assignment of tasks to VMs. The algorithm operates with cats in two primary modes: tracing mode and seeking mode. These modes enable the CSO algorithm that effectively explore the search space, refining the solutions generated by the initial DE phase.

Seeking mode *In this mode, cats explore their local neighborhoods to find better positions. Each cat generates multiple candidate solutions* $\left(X_{i,1}^t, X_{i,2}^t, \ldots, X_{i,k}^t\right)$ *by slightly perturbing its current position* X_i^t*. It can be expressed as in (5).*

$$X_{ij} = x_i + r \cdot \text{SRD} \cdot x_i \tag{5}$$

X_{ij} represents the j-th copy of the i-th cat. SRD is a ratio that defines the boundaries within which the cat can move. r is a random number within the range [− 1, 1]. Each candidate solution undergoes a fitness evaluation $f\left(X_{i,j}^t\right)$, and

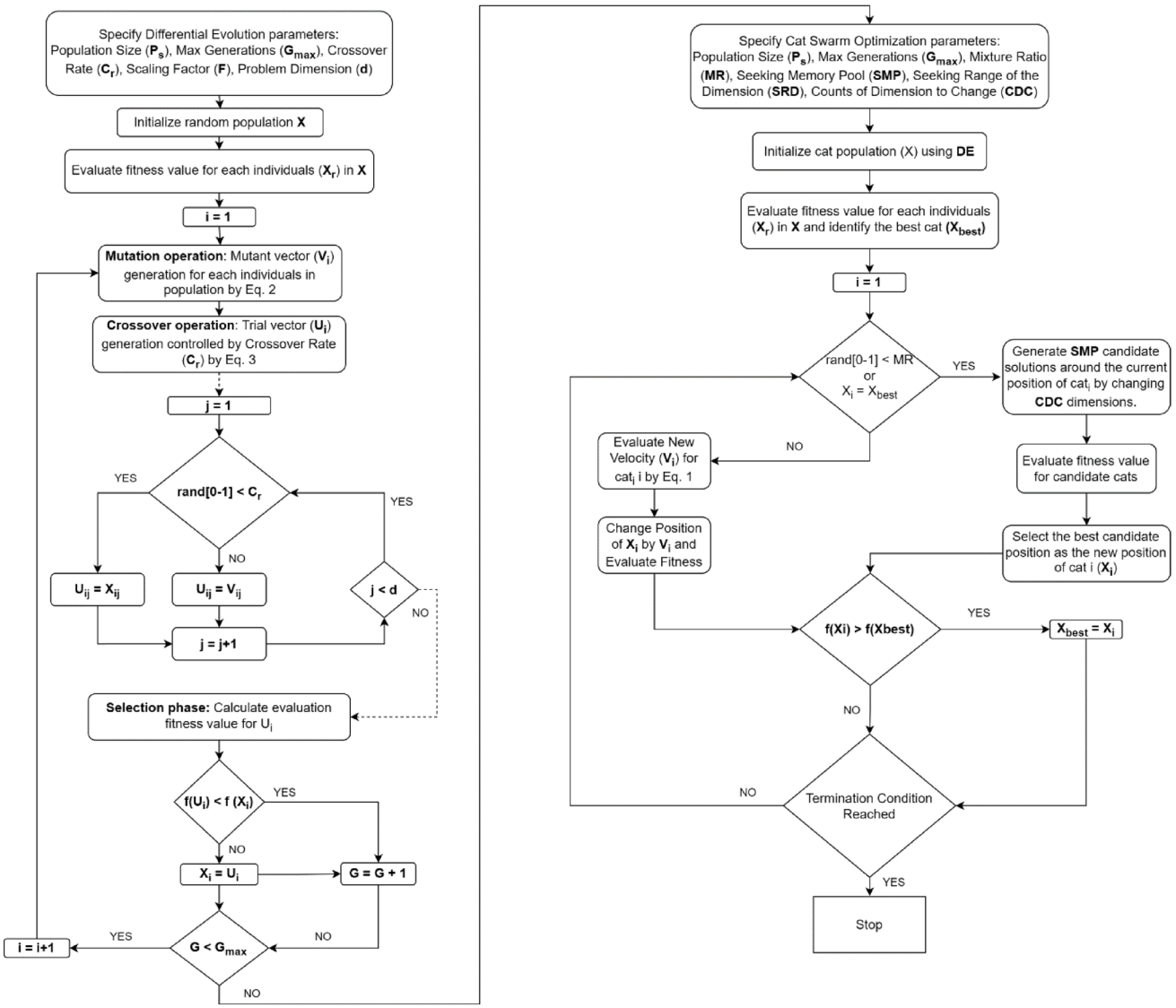

Fig. 1 Flowchart of the DECSO model

the cat moves to the position of the best candidate found. Defined as:

$$X_i^{t+1} = \arg \min_{j \in \{1,2,\ldots,k\}} f\left(X_{i,j}^t\right), \tag{6}$$

Where k is the number of candidate solutions generated. This local search mechanism is crucial for fine-tuning solutions and enhancing the precision of task assignments. In our implementation, we ensure that the highest-quality solution found so far (X_{best}) is always forced into the seeking mode in every iteration. This strategy guarantees that the local search is intensively conducted around the current best solution, improving the chances of finding an even better solution.

Tracing mode *In tracing mode, cats move towards that the highest-quality solution (X_{best}) found so far so far by following their velocities. Each cat's position is adjusted according to its velocity, influenced by the global best position found by the swarm. Velocity equation is given as follows.*

$$\begin{aligned} V_i^{t+1} &= w \cdot V_i^t + c_1 \cdot r_1 \cdot \left(X_{\text{best}} - X_i^t\right), \\ X_i^{t+1} &= X_i^t + V_i^{t+1}, \end{aligned} \tag{7}$$

Where w is the inertia weight, c_1 is the cognitive acceleration coefficient, r_1 is a random number uniformly distributed in [0,1], V_i^t is the velocity of cat i at iteration t. This mode exploits the search space by converging towards high-quality solutions, ensuring that the algorithm capitalizes on the top solutions discovered during the search process.

In DECSO, the Cat Swarm Optimization (CSO) algorithm is configured with a Seeking Memory Pool (SMP) of 50, ensuring a diverse set of candidate solutions during the seeking phase. The Counts of Dimension to Change (CDC) is set to 0.8, meaning that 80% of the dimensions are modified during each iteration, promoting sufficient diversity in the search process.

Table 2 Parameter settings for DECSO

Parameter	Value	Description
SMP	50	Number of candidate solutions stored in the seeking phase
CDC	0.8	Percentage of dimensions modified in each iteration
MR	0.1 -> 0.7	Starting from 0.1 and increasing up to 0.7 to balance exploration and exploitation
SRD	0.1 -> 0.4	Starts at 0.1 and increases up to 0.4, widening search range in early stages and refining in later stages
P_s	20	Number of individuals in the population
CR	0.9	Probability of crossover between individuals
F	0.8	Mutation factor, controlling the differential variation

Unlike static approaches, DECSO dynamically adjusts Mixture Ratio (MR) and Seeking Range of Dimensions (SRD) based on the evaluation progress to improve adaptability and convergence stability. MR starts at 0.1 and gradually increases up to 0.7, allowing more cats to transition into the tracing mode as the search progresses, which helps intensify exploitation in later iterations. SRD starts at 0.1 and increases up to 0.4, enabling a wider search range in earlier stages to enhance exploration and progressively narrowing it down for more precise local refinements as the algorithm converges. This adaptive strategy prevents premature convergence, maintains diversity, and ensures a smooth transition between exploration and exploitation, which is essential for effective task scheduling in cloud environments.

Integration with DE *After the DE phase, the refined solutions are transferred to the CSO algorithm. Each solution obtained from DE serves as an initial position for a cat in the CSO swarm. This integration leverages DE's global exploration capabilities to generate a diverse and high-quality set of starting solutions, which are then further optimized by CSO's local search mechanisms.*

During the CSO phase, the algorithm iteratively applies seeking and tracing modes to refine the solutions. In each iteration, we ensure that the best cat (i.e., the highest fitness value cat) is placed in seeking mode. This approach intensifies The local search aimed at finding the best solution, increasing the likelihood of discovering superior solutions. The remaining cats are probabilistically assigned to either seeking or tracing mode based on a predefined mixture ratio. The continuous evaluation and updating of fitness values ensure that only the best solutions are retained. The iterative process of seeking and tracing modes allows CSO to effectively balance exploration and exploitation, refining the task scheduling solutions obtained from DE. Figure 1 represents the architecture flowchart of the proposed DECSO model.

The proposed hybrid model merges the benefits of DE and CSO. By leveraging DE's ability to search for the wider solution space and CSO's capability to fine-tune solutions locally, the DECSO approach enhances the overall search efficiency and enhances the effectiveness of task scheduling solutions. In conclusion, the DECSO methodology provides an effective and efficient solution for the multi-objective task scheduling problem in cloud computing. By integrating DE and CSO, the model optimizes primary performance metrics such as resource utilization, MakeSpan, and migration time, leading to greater overall performance and efficiency in cloud computing systems. The key parameter settings for DECSO, including the configurations for

Differential Evolution (DE) and Cat Swarm Optimization (CSO), are presented in Table 2.

4.1 Algorithm complexity analysis

Accurately determining the asymptotic complexity of meta-heuristic algorithms can be challenging due to their adaptive nature and dynamic parameter adjustments. However, a theoretical worst-case time complexity can be established for DECSO by analyzing its two main components: DE and CSO.

The time complexity of DE is primarily influenced by the number of generations (G), the population size (NP), and the dimensionality of the search space (D). Since each generation applies mutation, crossover, and selection operators across the entire population, its computational complexity can be expressed as $O(G \times NP \times D)$.

Similarly, CSO involves both a seeking mode and a tracing mode, operating on a swarm of N individuals over I iterations. The computational complexity of CSO is therefore $O(I \times N \times D)$.

In the proposed DECSO algorithm, DE is responsible for generating an initial population, which is then refined through CSO. Since CSO directly enhances the solutions produced by DE, these two phases are executed sequentially rather than in parallel. As a result, the total time complexity of DECSO is $O(G \times NP \times D) + O(I \times N \times D)$. Given that both DE and CSO are evolutionary algorithms designed to explore and exploit the search space efficiently, the overall complexity of DECSO remains in line with other hybrid meta-heuristic algorithms. This ensures that DECSO maintains computational feasibility while effectively optimizing task scheduling in cloud computing environments.

5 Results

To evaluate the effectiveness of the proposed DECSO algorithm, three key performance metrics are used: MakeSpan, resource utilization, and migration time. MakeSpan represents the total time required to complete all scheduled tasks, where a lower value indicates a more efficient task allocation strategy. Resource utilization refers to the percentage of computational resources actively used during task execution, with higher values signifying better workload distribution. Migration time measures the total time spent migrating tasks between virtual machines, and reducing it helps minimize overhead and improve scheduling efficiency.

Table 3 Simulation settings

S. No	Name of the entity	Quantity
1	Number of tasks	100–1000
2	Length of tasks	1500–10,000
3	Number of datacenters	2
4	Capacity of VM	9192 MB
5	Name of hypervisor	xen
6	Number of VMs	50
7	VM Processing capacity	100–1000
8	VM Ram	1024

5.1 Experimental setup

This section details the simulation framework applied to evaluate the performance of the proposed DECSO hybrid model for task scheduling in cloud computing. To evaluate the performance of the proposed DECSO hybrid model for task scheduling, we utilized CloudSim, a widely used cloud computing simulation toolkit. CloudSim provides a comprehensive framework for simulating, modeling and conducting experiments related to cloud computing. It offers a flexible and extensible environment for simulating a variety of cloud elements, including VMs, data centers, and tasks (cloudlets). The toolkit enables researchers to create detailed simulations of cloud environments, allowing for the analysis and optimization of different cloud computing strategies without the need for physical hardware.

To ensure the robustness and reliability of our simulation results, we used real-world workload traces from the High-Performance Computing Center North (HPC2N) and NASA. These workload traces provide detailed information about the resource requirements and execution times of various tasks, reflecting the characteristics of real cloud computing environments. The HPC2N dataset, originating from high-performance computing (HPC) environments, contains heterogeneous workloads that simulate cloud-based scientific computations. These workloads reflect dynamic job arrival patterns and varying resource demands, making them suitable for evaluating the adaptability of DECSO in complex scheduling scenarios. The NASA dataset, on the other hand, consists of scientific workflow traces from real-world computational tasks, providing insights into task dependencies, execution times, and resource allocation complexities. By incorporating these real-world workload traces into our simulation, we can ensure that our evaluation accurately reflects the complexities and challenges of scheduling tasks in cloud computing. HPC2N and NASA workloads allow us to validate the effectiveness of the DECSO hybrid model in handling diverse and dynamic workloads, ensuring that it can efficiently allocate resources under varying conditions. This validation ultimately demonstrates its potential for improving the efficiency and performance of cloud computing systems, making DECSO applicable beyond synthetic simulation environments. Table 3 presents the cloud simulation parameters.

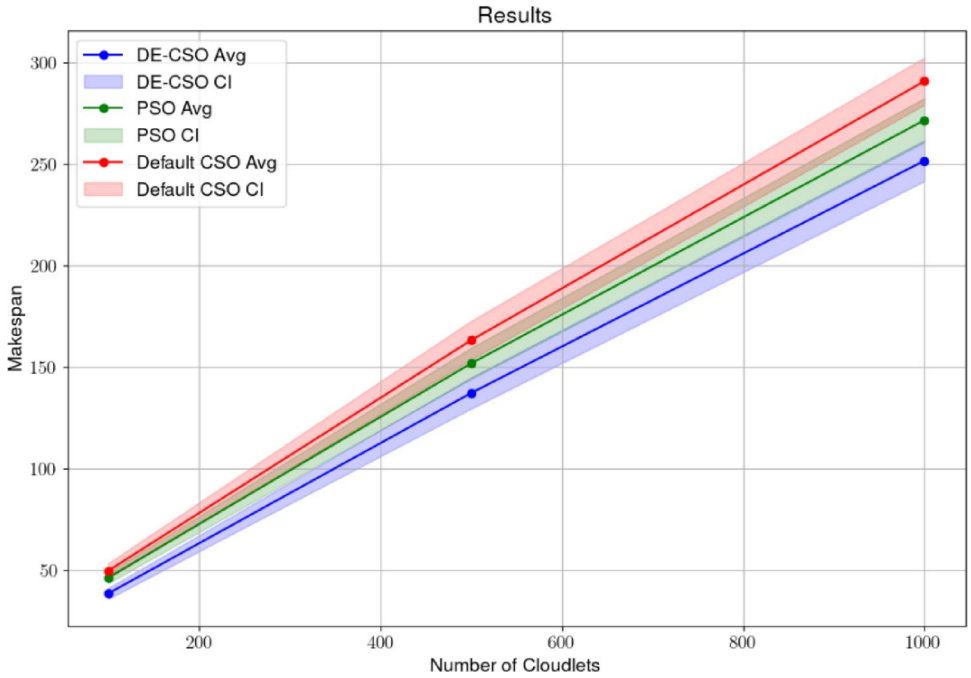

Fig. 2 Evaluation of MakeSpan performance utilizing the HPC2N workload

5.2 Objective functions

Assigning tasks in a way that optimizes variety of performance metrics is one of the most popular and critical features in cloud computing. Efficient task scheduling aims to allocate computational tasks to VMs hosted in data centers, balancing the load across available resources and ensuring the timely task execution.

Tasks (T) *Let* $T = \{T_1, T_2, T_3, \ldots, T_m\}$ *illustrate the group of tasks to be scheduled. Each task* T_i *specifies its own resource requirements and execution durations.*

Virtual machines (VMs) *Let* $V = \{V_1, V_2, V_3, \ldots, V_n\}$ *denote the set of VMs available in the data center. Each VM* V_j *has certain capacities in terms of CPU, memory, and storage.*

Data centers (DC) *Let* $DC = \{D_{C1}, D_{C2}, D_{C3}, \ldots, D_{Ck}\}$ *represent the set of data centers hosting the VMs. Each data center has multiple hosts, and each host can run multiple VMs.*

Hosts (H) *Let* $H = \{H_1, H_2, H_3, \ldots, H_p\}$ *denote the set of physical hosts in the data center. Each host* H_q *has finite computational resources that can be allocated to VMs.*

The task scheduling problem can be expressed as an optimization problem where the focus is on maximize resource utilization, minimize the MakeSpan, and reduce migration time. The decision variables involve the assignment of tasks T_i to VMs V_j.

Table 4 Evaluatıon of makespan performance utilizing the HPC2N workload

Cloudlets	CSO	PSO	DECSO
100	49.62	46.18	**38.42**
500	163.03	151.64	**137.04**
1000	290.64	271.44	**251.47**

MakeSpan *(M)*: The total execution time for all tasks, defined as the time when the last task finishes execution. It can be expressed as in (8):

$$m = \max_{i \in \{1,\ldots,m\}} C_i \tag{8}$$

where C_i is the execution time of task T_i.

Resource utilization (RU) *The effective use of available resources, expressed as the ratio of the utilized resources to the overall resource capacity over a given period. It can be formulated as in (9).*

$$RU = \frac{\sum_{j=1}^{n}\sum_{i=1}^{m} R_{ij}.x_{ij}}{\sum_{j=1}^{n} R_j} \tag{9}$$

where R_{ij} is the resource requirement of task T_i on VM V_j, x_{ij} represents a binary variable indicates whether if task T_i has been assigned to VM V_j, and R_j is the total resource capacity of VM V_j.

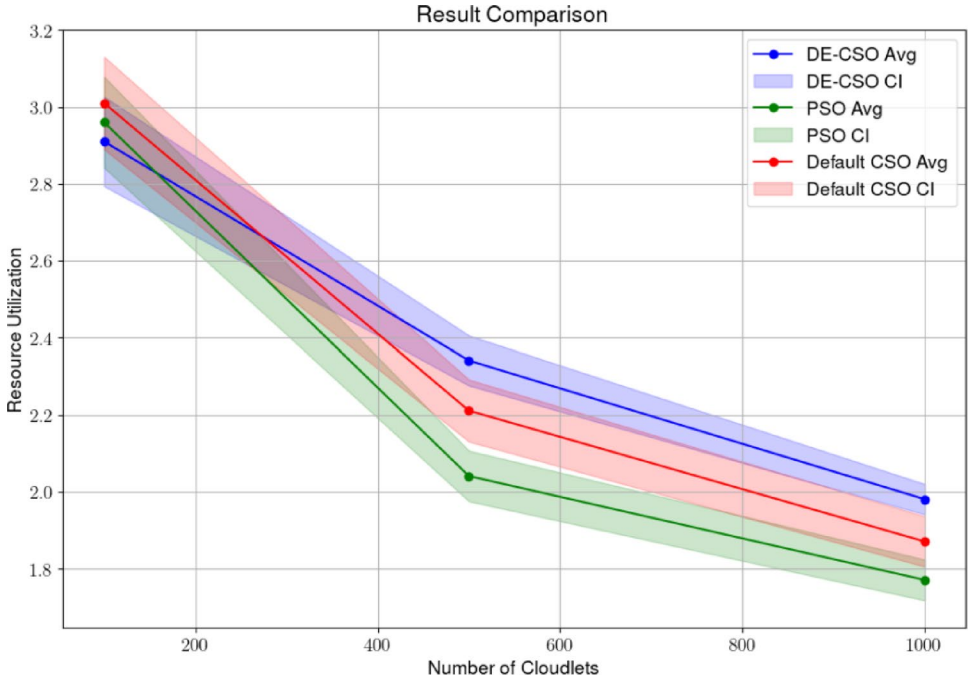

Fig. 3 Evaluation of resource utilization using HPC2N workload

Table 5 Evaluation of resource utilization using HPC2N workload

Cloudlets	CSO	PSO	DECSO
100	**3.01**	2.96	2.91
500	2.21	2.04	**2.34**
1000	1.87	1.77	**1.98**

Migration time (MT) *The total time spent on migrating tasks between VMs, which should be minimized to ensure efficiency. It can be expressed as in (10).*

$$MT = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} M_{ij} \tag{10}$$

where M_{ij} represents the time taken for task T_i to migrate to VM V_j.

The DECSO model's fitness function is created to evaluate the performance of task scheduling by combining the above metrics. The objective is to increase resource utilization, reduce the MakeSpan, and minimize migration time. The formulation of the fitness function F is provided in (11):

$$F = w_1.\left(\frac{1}{M}\right) + w_2.RU - w_3.MT \tag{11}$$

where w_1, w_2, and w_3 are the weights assigned to the respective metrics, indicating their relative importance. This formulation provides a comprehensive framework for evaluating and optimizing task scheduling processes in cloud computing environments, making use of the strengths of both DE and CSO algorithms.

The proposed mathematical formulations for task scheduling are designed to be applicable in real-world cloud computing environments by considering key performance metrics such as MakeSpan, Resource Utilization, and Migration Time. The model assumes a heterogeneous cloud infrastructure where tasks arrive dynamically and need to be assigned to virtual machines with varying capacities. These assumptions align with practical cloud computing scenarios, including large-scale data centers and edge computing environments. Furthermore, the optimization objectives ensure that the scheduling strategy remains adaptable to different workload conditions, making the model generalizable across various cloud-based platforms.

5.3 Experimental results

This section details the experimental results of our proposed model, DECSO (Differential Evolution Cat Swarm Optimization), and compare its performance with two well-known optimization algorithms: CSO and PSO. The experiments were conducted to evaluate the efficiency of these algorithms in addressing three distinct objectives: MakeSpan, Resource Utilization, and Migration Time. The evaluation involved varying the number of tasks between 100 and 1000, utilizing workloads from HPC2N and NASA datasets. This range was selected to assess the algorithm's performance under different workload sizes. The results demonstrate that DECSO maintains its optimization efficiency across different task scales, effectively adapting to increasing workload complexities and resource variations. The results highlight DECSO's effectiveness in improving task scheduling for

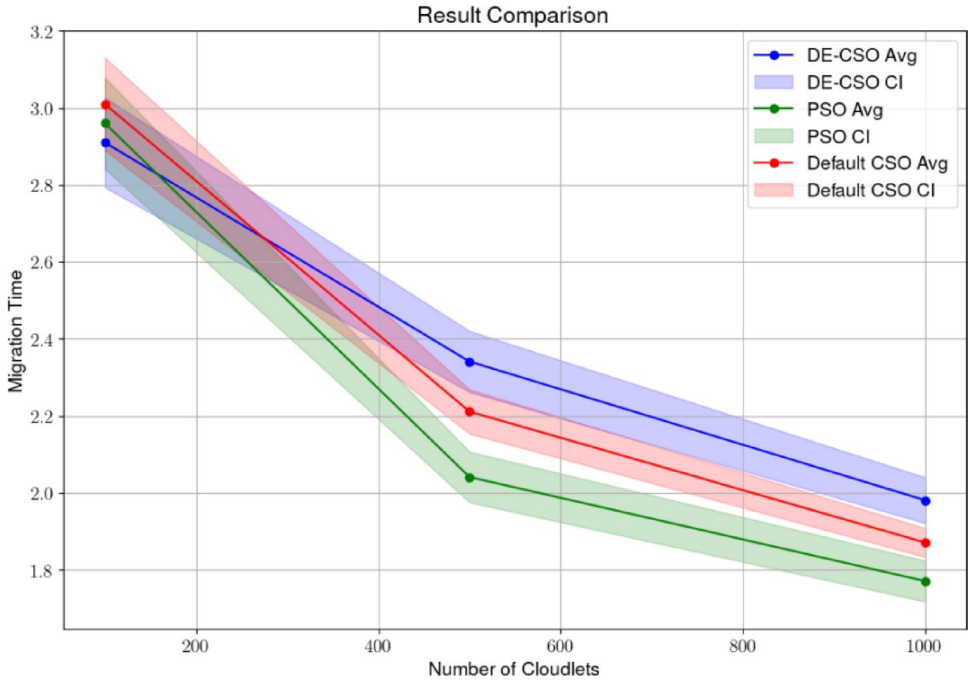

Fig. 4 Migration time evaluation with the HPC2N workload

Table 6 Migration time evaluation with the HPC2N workload

Cloudlets	CSO	PSO	DECSO
100	41.22	38.46	**31.88**
500	141.91	126.90	**112.65**
1000	277.04	239.73	**235.87**

cloud computing environments, providing comprehensive insights into its performance across different scenarios. Each objective's results are discussed in detail, highlighting the advantages of DECSO over PSO and CSO in achieving optimal scheduling outcomes.

In this study, Particle Swarm Optimization (PSO) and Cat Swarm Optimization (CSO) were selected as baseline algorithms due to their extensive use in cloud task scheduling and metaheuristic optimization. PSO is a leading optimization algorithm, frequently employed for its efficient global search capability, but it often struggles with premature convergence, limiting its adaptability in dynamic cloud environments. CSO, on the other hand, provides a more adaptive local search mechanism, making it effective in fine-tuning scheduling decisions. However, both methods face challenges in balancing exploration and exploitation, which are critical for task scheduling performance.

DECSO integrates Differential Evolution (DE) with CSO to overcome these limitations, ensuring a more adaptive and balanced optimization process. By comparing DECSO against standard CSO, this study evaluates the impact of hybridization, highlighting how incorporating DE enhances CSO's global search capability while retaining its efficient local refinement. By benchmarking against PSO and CSO, the study effectively assesses how DECSO improves upon conventional swarm-based approaches while maintaining computational efficiency.

To ensure statistical reliability, the confidence intervals in all performance comparison figures have been set to 95%.

The evaluation of MakeSpan for the proposed DECSO model, alongside PSO and CSO, is illustrated in Fig. 2 and detailed in Table 4. The experiments utilized the HPC2N workload with cloudlet numbers varying from 100 to 1000.

Figure 2 illustrates the average MakeSpan values for each algorithm. With an increase in the number of cloudlets, the MakeSpan values for all algorithms also increase, which is expected due to the increased computational load. However, DECSO consistently demonstrates lower MakeSpan values compared to PSO and CSO across all task counts, indicating its superior efficiency in task scheduling. Table 4 provides a detailed comparison of the MakeSpan values for 100, 500, and 1000 cloudlets. For 100 cloudlets, DECSO achieves a MakeSpan of 38.42, which is 16.8% lower than PSO's 46.18 and 22.6% lower than CSO's 49.62. As the task count increases to 500 cloudlets, DECSO maintains its performance advantage with a MakeSpan of 137.04, which is 9.6% lower than PSO's 151.64 and 15.9% lower than CSO's 163.03. This trend continues with 1000 cloudlets, where DECSO records a MakeSpan of 251.47, which is 7.3% lower than PSO's 271.44 and 13.5% lower than CSO's 290.64. These results clearly demonstrate that DECSO outperforms both PSO and CSO in terms of MakeSpan, effectively shortening the time needed for task completion. This highlights the robustness and efficiency of DECSO in handling diverse cloud computing workloads, making it a promising approach for multi-objective task scheduling.

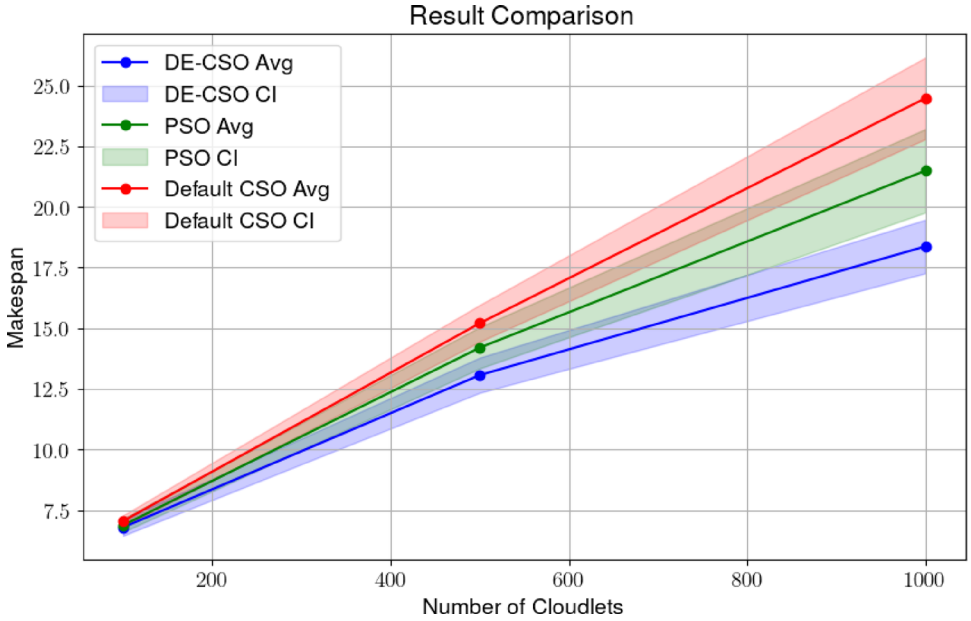

Fig. 5 Evaluation of MakeSpan performance utilizing the NASA workload

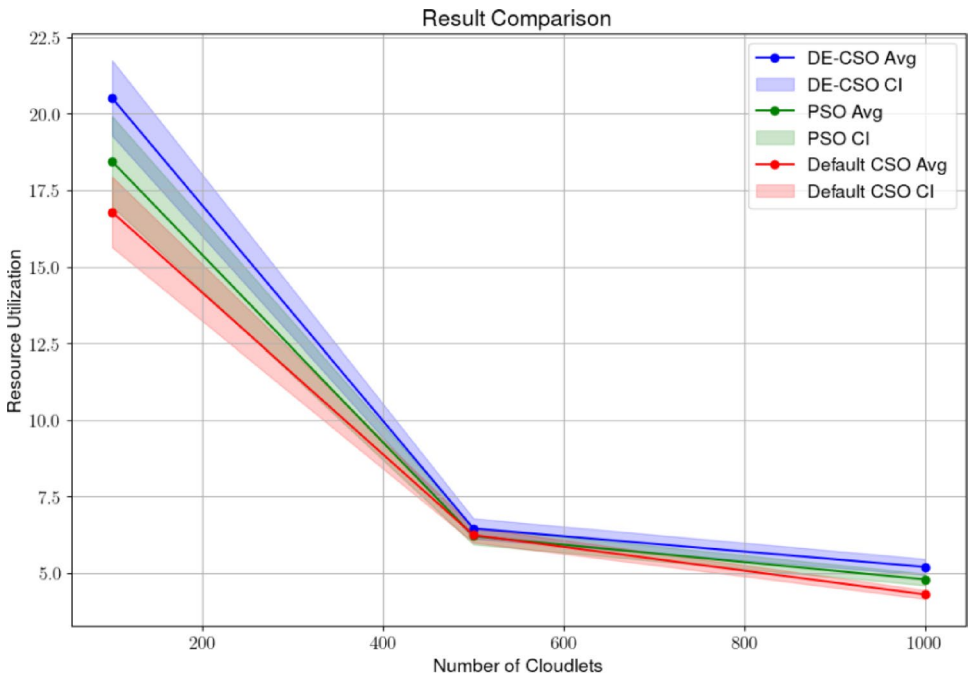

Fig. 6 Evaluation of resource utilization using NASA workload

The evaluation of Resource Utilization for the proposed DECSO model, alongside PSO and CSO, is illustrated in Fig. 3 and detailed in Table 5. The experiments utilized the HPC2N workload with cloudlet numbers varying from 100 to 1000.

Figure 3 illustrates the average Resource Utilization values for each algorithm. With an increase in the number of cloudlets, the Resource Utilization values for all algorithms decrease, which is expected due to the increased distribution of tasks across available resources. DECSO consistently demonstrates better Resource Utilization values compared to PSO and CSO across most task counts, indicating its superior efficiency in resource management. However, it is worth noting that for 100 cloudlets, DECSO performs slightly worse compared to both PSO and CSO. While DECSO generally outperforms PSO and CSO, its lower resource utilization at smaller task counts (100 cloudlets) can be attributed to its exploration-heavy search strategy in the early optimization stages. At low task counts, the problem space is relatively simple, making rapid convergence possible, whereas DECSO's adaptive mechanism continues to explore potential solutions rather than directly exploiting the best-known assignments. Future enhancements could

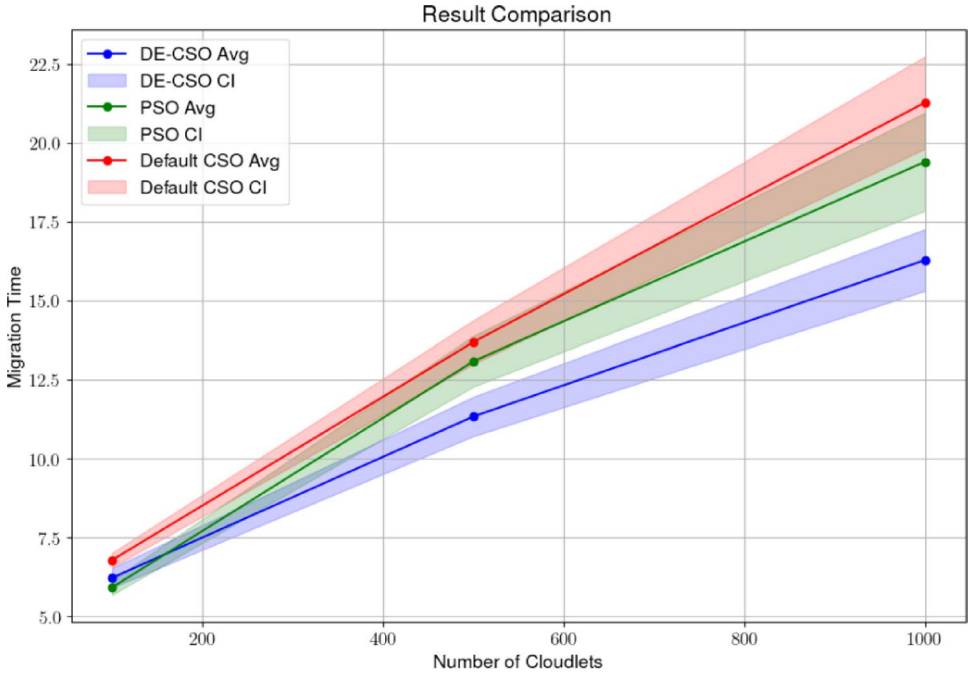

Fig. 7 Migration time evaluation with the NASA workload

Table 7 Evaluation of make span performance utilizing the NASA workload

Cloudlets	CSO	PSO	DECSO
100	7.06	6.88	**6.80**
500	15.21	14.20	**13.06**
1000	24.48	21.49	**18.37**

Table 8 Evaluation of resource utilization using NASA workload

Cloudlets	CSO	PSO	DECSO
100	16.79	18.44	**20.52**
500	6.24	6.21	**6.45**
1000	4.30	4.79	**5.20**

Table 9 Migration time evaluation with the NASA workload

Cloudlets	CSO	PSO	DECSO
100	6.79	**5.92**	6.23
500	13.69	13.08	**11.34**
1000	21.27	19.40	**16.28**

focus on dynamic exploration-exploitation tuning to optimize performance across different workload scales.

Table 5 provides a detailed comparison of the Resource Utilization values for 100, 500, and 1000 cloudlets. For 100 cloudlets, DECSO achieves a Resource Utilization of 2.91, which is 1.7% lower than PSO's 2.96 and 3.3% lower than CSO's 3.01. Despite this, as the task count increases, DECSO shows significant improvements. At 500 cloudlets, DECSO achieves a Resource Utilization of 2.34, which is 14.7% higher than PSO's 2.04 and 5.9% higher than CSO's 2.21. This trend continues with 1000 cloudlets, where DECSO records a Resource Utilization of 1.98, which is 11.9% higher than PSO's 1.77 and 5.9% higher than CSO's 1.87. These results clearly demonstrate that DECSO outperforms both PSO and CSO in the context of Resource Utilization, effectively maximizing the use of available resources, particularly with an increase in the number of cloudlets. This highlights the robustness and efficiency of DECSO in managing resource allocation in diverse cloud computing workloads, making it a promising approach for multi-objective task scheduling. However, the slight underperformance in the lower task count scenario suggests potential areas for further refinement and optimization.

The evaluation of Migration Time for the DECSO model, in comparison to PSO and CSO, is depicted in Fig. 4 and detailed in Table 6. The experiments utilized the HPC2N workload and were conducted with varying numbers of cloudlets, ranging from 100 to 1000.

Figure 4 demonstrates the average Migration Time values for each algorithm with increasing cloudlet numbers. As expected, the Migration Time values for all algorithms rise with the increasing computational load. Nevertheless, DECSO consistently shows lower Migration Time values than both PSO and CSO, indicating a more efficient task scheduling process.

Table 6 offers a comprehensive comparison of Migration Time values for 100, 500, and 1000 cloudlets. For 100 cloudlets, DECSO achieves a Migration Time of 31.88, which is 17.1% less than PSO's 38.46 and 22.6% less than CSO's 41.22. With 500 cloudlets, DECSO maintains its advantage, recording a Migration Time of 112.65 11.2% lower than PSO's 126.90 and 20.6% lower than CSO's 141.91. For 1000 cloudlets, DECSO continues to outperform, with

a Migration Time of 235.87, which is 1.6% less than PSO's 239.73 and 15% less than CSO's 277.04.

To assess the statistical significance of DECSO's performance improvements over PSO and CSO, we conducted independent t-tests on the MakeSpan, Resource Utilization, and Migration Time metrics across different task sizes (100, 500, and 1000 tasks). The results indicate that DECSO significantly outperforms both PSO and CSO in MakeSpan and Migration Time across all tested scenarios ($p < 0.05$). Specifically, for 1000 tasks, DECSO achieved a statistically significant reduction in MakeSpan ($p < 0.001$) and Migration Time ($p < 0.001$) compared to both baseline algorithms. In Resource Utilization, the statistical significance varied depending on the task size. While the improvement was not statistically significant for 100 tasks ($p = 0.145$ vs. PSO, $p = 0.089$ vs. CSO), it became significant for 500 and 1000 tasks ($p < 0.05$), demonstrating DECSO's ability to efficiently allocate resources as the workload scales. These results confirm that DECSO's hybrid approach provides a statistically meaningful advantage over CSO and PSO, particularly in large-scale cloud environments.

These findings underscore DECSO's superior performance in reducing Migration Time compared to PSO and CSO, highlighting its effectiveness in minimizing task migration duration. This demonstrates DECSO's potential as an efficient solution for scheduling tasks in cloud computing.

In addition to the HPC2N workload, the performance of DECSO, PSO, and CSO was also evaluated using the NASA workload, as illustrated in Figs. 5, 6 and 7, and detailed in Tables 7, 8 and 9. The MakeSpan, Resource Utilization and Migration Time metrics were analyzed for varying numbers of cloudlets from 100 to 1000.

These results underscore DECSO's robustness and efficiency in optimizing task scheduling and resource utilization, not only with the HPC2N workload but also with the NASA workload, strengthening its capability as an optimization algorithm for task scheduling.

6 Conclusion and future work

Task scheduling in cloud computing poses considerable challenges because of the dynamic and diverse nature of cloud environments. Efficiently allocating resources while minimizing MakeSpan, maximizing Resource Utilization, and reducing Migration Time are critical objectives that require robust and adaptive algorithms. Conventional optimization techniques frequently struggle to tackle these multi-objective complexities. In this study, we proposed DECSO model, an innovative hybrid algorithm, designed to tackle these challenges. Our experimental results demonstrated that DECSO consistently outperformed established algorithms such as PSO and CSO in optimizing MakeSpan, Resource Utilization, and Migration Time across different workloads and varying task counts. DECSO's superior performance highlights its robustness and adaptability in efficiently managing cloud resources and scheduling tasks. While DECSO demonstrates strong performance in task scheduling, its efficiency is sensitive to parameter tuning, requiring careful adjustment to achieve optimal results. Additionally, its initial exploration phase can lead to slower convergence in scenarios where rapid scheduling decisions are needed. Future improvements could focus on adaptive mechanisms to refine parameter selection and accelerate convergence in time-sensitive applications. Beyond cloud computing, DECSO's adaptive scheduling capabilities make it a strong candidate for other resource-intensive domains. In edge computing, where computational resources are distributed across network edges, DECSO's ability to dynamically allocate tasks can minimize latency and optimize resource usage. Future work will focus on refining DECSO to further improve its scalability and efficiency. We aim to integrate additional real-world constraints and workloads to verify the algorithm's effectiveness in more diverse scenarios. Moreover, exploring hybrid approaches that combine DECSO with other optimization techniques could yield even better results. Future enhancements to DECSO could be integrating it with reinforcement learning-based adaptive mechanisms, allowing the algorithm to dynamically adjust its exploration-exploitation balance based on real-time scheduling conditions.

Author contributions A.G: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing– original draft SS: Conceptualization, Supervision, Validation, Investigation, Writing– review & editing Z.G-A: Conceptualization, Supervision, Validation, Writing– review & editing.

Funding Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK). Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK). The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

1. Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y., Murphy, J.: A WOA-based optimization approach for task scheduling in cloud computing systems. IEEE Syst. J. **14**(3), 3117–3128 (2020). https://doi.org/10.1109/JSYST.2019.2960088
2. Kaur, P., Mehta, S.: Resource provisioning and workflow scheduling in clouds using augmented shuffled frog leaping algorithm. J. Parallel Distrib. Comput. **101**, 41–50 (2017). https://doi.org/10.1016/j.jpdc.2016.11.003
3. Arunarani, A.R., Manjula, D., Sugumaran, V.: Task scheduling techniques in cloud computing: A literature survey. Future Generation Comput. Syst. **91**, 407–415 (2019). https://doi.org/10.1016/j.future.2018.09.014
4. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997)
5. Chu, S.C., Tsai, P.W., Pan, J.S.: Cat swarm optimization. In PRICAI 2006: Trends in Artificial Intelligence: 9th Pacific Rim International Conference on Artificial Intelligence Guilin, China, August 7–11, 2006 Proceedings 9 (pp. 854–858). Springer Berlin Heidelberg. (2006). https://doi.org/10.1007/978-3-540-36668-3_94
6. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**, 23–50 (2011). https://doi.org/10.1002/spe.995
7. Malawski, M., Figiela, K., Nabrzyski, J.: Cost minimization for computational applications on hybrid cloud infrastructures. Future Generation Comput. Syst. **29**(7), 1786–1794 (2013). https://doi.org/10.1016/j.future.2013.01.004
8. Su, S., Li, J., Huang, Q., Huang, X., Shuang, K., Wang, J.: Cost-efficient task scheduling for executing large programs in the cloud. Parallel Comput. **39**(4–5), 177–188 (2013). https://doi.org/10.1016/j.parco.2013.03.002
9. Lakra, A.V., Yadav, D.K.: Multi-objective tasks scheduling algorithm for cloud computing throughput optimization. Procedia Comput. Sci. **48**, 107–113 (2015). https://doi.org/10.1016/j.procs.2015.04.158
10. Gawali, M.B., Shinde, S.K.: Task scheduling and resource allocation in cloud computing using a heuristic approach. J. Cloud Comput. **7**, 1–16 (2018). https://doi.org/10.1186/s13677-018-0105-8
11. Younes, A., Elnahary, M.K., Alkinani, M.H., El-Sayed, H.H.: Task scheduling optimization in cloud computing by Rao algorithm. Computers Mater. Continua. **72**(3) (2022). https://doi.org/10.32604/cmc.2022.022824
12. Hamed, A.Y., Alkinani, M.H.: Task scheduling optimization in cloud computing based on genetic algorithms. Comput. Mater. Contin. **69**(03), 3289–3301 (2021). https://doi.org/10.32604/cmc.2021.018658
13. Gobalakrishnan, N., Arun, C.: A new multi-objective optimal programming model for task scheduling using genetic Gray Wolf optimization in cloud computing. Comput. J. **61**(10), 1523–1536 (2018). https://doi.org/10.1093/comjnl/bxy009
14. Hamad, S.A., Omara, F.A.: Genetic-based task scheduling algorithm in cloud computing environment. Int. J. Adv. Comput. Sci. Appl. **7**(4), 550–556 (2016). https://doi.org/10.14569/IJACSA.2016.070471
15. Gawanmeh, A., Parvin, S., Alwadi, A.: A genetic algorithmic method for scheduling optimization in cloud computing services. Arab. J. Sci. Eng. **43**(12), 6709–6718 (2018). https://doi.org/10.1007/s13369-017-2812-8
16. Zhou, Z., Li, F., Zhu, H., Xie, H., Abawajy, J.H., Chowdhury, M.U.: An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. Neural Comput. Appl. **32**, 1531–1541 (2020). https://doi.org/10.1007/s00521-019-04119-7
17. Gulbaz, R., Siddiqui, A.B., Anjum, N., Alotaibi, A.A., Althobaiti, T., Ramzan, N.: Balancer genetic algorithm—A novel task scheduling optimization approach in cloud computing. Appl. Sci. **11**(14), 6244 (2021). https://doi.org/10.3390/app11146244
18. Pirozmand, P., Jalalinejad, H., Hosseinabadi, A.A.R., Mirkamali, S., Li, Y.: An improved particle swarm optimization algorithm for task scheduling in cloud computing. J. Ambient Intell. Humaniz. Comput. **14**(4), 4313–4327 (2023). https://doi.org/10.1007/s12652-023-04541-9
19. Alla, H.B., Alla, S.B., Ezzati, A.: A novel architecture for task scheduling based on dynamic queues and particle swarm optimization in cloud computing. In 2016 2nd international conference on cloud computing technologies and applications (CloudTech) (pp. 108–114). IEEE. (2016)., May https://doi.org/10.1109/CloudTech.2016.7847686
20. Guo, L., Zhao, S., Shen, S., Jiang, C.: Task scheduling optimization in cloud computing based on heuristic algorithm. J. Networks. **7**(3), 547 (2012). https://doi.org/10.4304/jnw.7.3.547-553
21. Awad, A.I., El-Hefnawy, N.A., Abdel-kader, H.M.: Enhanced particle swarm optimization for task scheduling in cloud computing environments. Procedia Comput. Sci. **65**, 920–929 (2015). https://doi.org/10.1016/j.procs.2015.09.064
22. Wu, D.: Cloud computing task scheduling policy based on improved particle swarm optimization. In *2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)* (pp. 99–101). IEEE. (2018)., August https://doi.org/10.1109/ICVRIS.2018.00032
23. Zhou, Z., Chang, J., Hu, Z., Yu, J., Li, F.: A modified PSO algorithm for task scheduling optimization in cloud computing. Concurrency Computation: Pract. Experience. **30**(24), e4970 (2018). https://doi.org/10.1002/cpe.4970
24. Li, J.Q., Han, Y.Q.: A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system. Cluster Comput. **23**(4), 2483–2499 (2020). https://doi.org/10.1007/s10586-019-03022-z
25. Ramezani, F., Lu, J., Taheri, J., Hussain, F.K.: Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments. World Wide Web. **18**, 1737–1757 (2015). https://doi.org/10.1007/s11280-015-0335-3
26. Song, X., Gao, L., Wang, J.: Job scheduling based on ant colony optimization in cloud computing. In 2011 International Conference on Computer Science and Service System (CSSS) (pp. 3309–3312). IEEE. (2011)., June https://doi.org/10.1109/CSSS.2011.5972226
27. Wang, L., Ai, L.: Task scheduling policy based on ant colony optimization in cloud computing environment. In *LISS 2012: Proceedings of 2nd International Conference on Logistics, Informatics and Service Science* (pp. 953–957). Springer Berlin Heidelberg. (2013). https://doi.org/10.1007/978-3-642-32054-5_133
28. Liu, C.Y., Zou, C.M., Wu, P.: A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing. In *2014 13th International Symposium on Distributed computing and applications to business, engineering and science*

(pp. 68–72). IEEE. (2014)., November https://doi.org/10.1109/DCABES.2014.18
29. Dai, Y., Lou, Y., Lu, X.: A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-QoS constraints in cloud computing. In *2015 7th international conference on intelligent human-machine systems and cybernetics* (Vol. 2, pp. 428–431). IEEE. (2015)., August https://doi.org/10.1109/DCABES.2014.18
30. Moon, Y., Yu, H., Gil, J.M., Lim, J.: A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. Human-centric Comput. Inform. Sci. **7**, 1–10 (2017). https://doi.org/10.1186/s13673-017-0109-2
31. Reddy, G.R.N., Phanikumar, S.: Multi objective task scheduling using modified ant colony optimization in cloud computing. Int. J. Intell. Eng. Syst., **11**(3), 242–250 (2018). https://doi.org/10.22266/ijies2018.0630.26
32. Wei, X.: Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing. J. Ambient Intell. Humaniz. Comput. 1–12 (2020). https://doi.org/10.1007/s12652-020-02614-7
33. Gabi, D., Zainal, A., Ismail, A.S., Zakaria, Z.: Scalability-Aware scheduling optimization algorithm for multi-objective cloud task scheduling problem. In *2017 6th ICT International Student Project Conference (ICT-ISPC)* (pp. 1–6). IEEE. (2017)., May https://doi.org/10.1109/ICT-ISPC.2017.8075304
34. Guo, X.: Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. Alexandria Eng. J. **60**(6), 5603–5609 (2021). https://doi.org/10.1016/j.aej.2021.04.051
35. Hamed, A.Y., Elnahary, M.K., El-Sayed, H.H.: Task scheduling optimization in cloud computing by coronavirus herd immunity optimizer algorithm. Int. J. Adv. Netw. Appl. **14**(6), 5686–5695 (2023)
36. Gong, R., Li, D., Hong, L., Xie, N.: Task scheduling in cloud computing environment based on enhanced marine predator algorithm. Cluster Comput. **27**(1), 1109–1123 (2024). https://doi.org/10.1007/s10586-023-04054-2
37. Alkayal, E.S., Jennings, N.R., Abulkhair, M.F.: Efficient task scheduling multi-objective particle swarm optimization in cloud computing. In 2016 IEEE 41st conference on local computer networks workshops (LCN workshops) (pp. 17–24). IEEE. (2016)., November https://doi.org/10.1109/LCN.2016.024
38. Sabireen, H., Venkataraman, N.: A hybrid and light weight metaheuristic approach with clustering for multi-objective resource scheduling and application placement in fog environment. Expert Syst. Appl. **223**, 119895 (2023). https://doi.org/10.1016/j.eswa.2023.119895
39. Chen, Z. G., Zhan, Z. H., Lin, Y., Gong, Y. J., Gu, T. L., Zhao, F.,… Zhang, J.(2018). Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach. IEEE transactions on cybernetics, 49(8), 2912–2926. https://doi.org/10.1109/TCYB.2018.2832640
40. Abualigah, L., Diabat, A.: A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Cluster Comput. **24**(1), 205–223 (2021). https://doi.org/10.1007/s10586-020-03075-5
41. Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I., Tuba, M., Zivkovic, M.: Task scheduling in cloud computing environment by grey wolf optimizer. In 2019 27th Telecommunications Forum (TELFOR) (pp. 1–4). IEEE. (2019)., November https://doi.org/10.1109/TELFOR48224.2019.8971223
42. Mangalampalli, S., Karri, G.R., Kumar, M.: Multi objective task scheduling algorithm in cloud computing using grey Wolf optimization. Cluster Comput. **26**(6), 3803–3822 (2023). https://doi.org/10.1007/s10586-022-03786-x
43. Malti, A.N., Hakem, M., Benmammar, B.: A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems. Cluster Comput. **27**(3), 2525–2548 (2024). https://doi.org/10.1007/s10586-023-04099-3
44. Behera, I., Sobhanayak, S.: Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach. J. Parallel Distrib. Comput. **183**, 104766 (2024). https://doi.org/10.1016/j.jpdc.2023.104766
45. Rana, N., Abd Latiff, M.S., Abdulhamid, S.I.M., Misra, S.: A hybrid Whale optimization algorithm with differential evolution optimization for multi-objective virtual machine scheduling in cloud computing. Eng. Optim. **54**(12), 1999–2016 (2022). https://doi.org/10.1080/0305215X.2021.1969560
46. Li, Y., Wang, S., Hong, X., Li, Y.: Multi-objective task scheduling optimization in cloud computing based on genetic algorithm and differential evolution algorithm. In 2018 37th Chinese Control Conference (CCC) (pp. 4489–4494). IEEE. (2018)., July https://doi.org/10.23919/ChiCC.2018.8483505
47. Abd Elaziz, M., Xiong, S., Jayasena, K.P.N., Li, L.: Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowl. Based Syst. **169**, 39–52 (2019). https://doi.org/10.1016/j.knosys.2019.01.023
48. Storn, R., Price, K.: Differential Evolution Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces. International Computer Science Institute (1995)
49. Storn, R.: System design by constraint adaptation and differential evolution. IEEE Trans. Evol. Comput. **3**(1), 22–34 (1999). https://doi.org/10.1109/4235.752918
50. Chu, S.C., Tsai, P.W., Pan, J.S.: Cat swarm optimization. In PRICAI 2006: Trends in Artificial Intelligence: 9th Pacific Rim International Conference on Artificial Intelligence Guilin, China, August 7–11, 2006 Proceedings 9 (pp. 854–858). Springer Berlin Heidelberg (2006). https://doi.org/10.1007/978-3-540-36668-3_94

ALİ GÜNDÜZ Ali Gündüz received his B.Sc. degree in Computer Engineering from Istanbul University in 2019 and his M.Sc. degree in Computer Engineering from Istanbul University-Cerrahpaşa in 2023. He is currently pursuing a Ph.D. in Computer Engineering and serving as a Research Assistant at Istanbul University-Cerrahpaşa. His research interests include meta-learning, few-shot learning and cross-domain generalisation, aiming to build models that remain data-efficient yet highly adaptable.

SİBEL SENAN Sibel Senan received her B.Sc. degree from Dept. of Computer Science Engineering, Istanbul University in 2002, and, M.Sc., and Ph.D. degrees from Dept. of Computer Engineering, Istanbul University, 2005, and 2009, respectively. She completed her postdoctoral research in the Dept. of Electrical and Computer Engineering at the University of Illinois at Chicago, USA, in 2011. She is currently an Associate Professor at the Dept. of Computer Engineering, Istanbul University-Cerrahpasa. Her research interests include artificial intelligence, neural networks, machine learning, image processing, and medical decision support systems.

Zeynep Gürkaş-AYDIN Dr. Zeynep Gürkaş-AYDIN completed her undergraduate studies in Computer Engineering at Istanbul University Engineering Faculty in 2003. She completed her Master's degree in Computer Engineering at Istanbul University Institute of Science in 2005, her first PhD in Computer Engineering at Istanbul University Institute of Science in 2011, and her second PhD in the field of Informatique as part of the Ecole Doctorale program at Université Pierre-et-Marie-Curie: Paris VI in France in 2014. Currently, she serves as a faculty member in the Department of Cybersecurity at Istanbul University-Cerrahpaşa Engineering Faculty. Her research covers a wide range of topics in Computer Science and Engineering.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com