Efficient Cloud Workflow Scheduling with Inverted Ant Colony Optimization Algorithm

Hongwei DING, Ying ZHANG*
Hebei Software Institute, Hebei, Bao ding, 071000, China

***Abstract*—Cloud computing has risen as a prominent paradigm, offering users on-demand access to computing resources and services via the Internet. In cloud environments, workflow scheduling plays a vital role in optimizing resource utilization, reducing execution time, and minimizing overall costs. As workflows comprise interdependent tasks that need to be assigned to Virtual Machines (VMs), the complexity of the scheduling problem increases in proportion to workflow size and VM availability. Due to its NP-hard nature, finding an optimal scheduling solution for workflows remains a challenging task. To address this problem, researchers have turned to metaheuristic approaches, which have shown promise in finding near-optimal solutions for complex combinatorial optimization problems. This paper proposes a novel metaheuristic algorithm called Inverted Ant Colony Optimization (IACO) for workflow scheduling in cloud environments. IACO is a variation of the traditional ACO algorithm, where the updated pheromone has an inverted influence on the path chosen by the ants. By leveraging the complementary nature of these two algorithms, our proposed algorithm aims to achieve superior workflow scheduling performance regarding total execution time and cost, surpassing existing approaches.**

***Keywords*—Cloud computing; workflow scheduling; virtualization; task allocation; swarm intelligence; optimization**

I. Introduction

Cloud computing is a technological advancement that harnesses the capabilities of the Internet and distant centralized servers to supply users with flexible services. These services are delivered using a diverse range of distributed resources, catering to various quality of service (QoS) requirements [1]. Prominent cloud computing platforms include Aneka, Microsoft Azure, Google App Engine, and Amazon EC2. Clouds are generally classified into several types: public, private, community, hybrid, and cloud federation [2]. Public clouds are accessible to the general public and are owned and managed by external entities known as independent cloud service providers. Computing resources, like applications, storage, and servers, are available to an array of businesses or individuals [3].

In contrast, private clouds are owned by an individual organization and are either hosted internally or handled exclusively by an external provider for that organization's use [4]. Community clouds are shared among multiple organizations with similar interests or requirements. These clouds are designed to cater to the specific needs of a particular community, such as government agencies, educational institutions, or healthcare providers [5]. Hybrid clouds integrate elements from both public and private clouds. In this model, organizations can distribute applications and data across multiple cloud deployment models, interconnected to function as a cohesive infrastructure. Cloud federation involves the interconnection and collaboration of multiple cloud infrastructures to work as a single unified cloud environment. It enables seamless movement of workloads and data among different cloud providers, enhancing flexibility and scalability in cloud computing [6].

Cloud computing is categorized into three primary service models: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS) [7]. In SaaS, software applications are delivered to users online through subscriptions. Users can use these services remotely without installation or local device maintenance. The responsibility for hosting, maintenance, and updates lies with the SaaS provider. PaaS offers a platform for developers to build, deploy, and manage applications with no need for infrastructure management. PaaS provides developers with access to a set of development tools, programming languages, and runtime environments, facilitating the creation and execution of applications. In IaaS, users subscribe to virtual machines, servers, networking components, and other infrastructure resources from a cloud provider. IaaS gives customers enhanced control over infrastructure without requiring them to invest in physical hardware and its maintenance [8].

Virtualization is a critical technology in cloud computing that enables the coexistence of multiple Virtual Machines (VMs) on a single physical machine. A VM is a simulated computer system that executes tasks assigned by users. This capacity for VM instantiation empowers users to run their applications across resources that encompass diverse functionality and cost attributes. Orchestrating this arrangement within each physical machine or server is a software layer, colloquially referred to as the hypervisor or VM monitor. The hypervisor serves as a facilitator for VM creation and ensures their isolated execution, allowing multiple VMs to operate independently and securely on the same physical hardware. The hypervisor is responsible for efficiently managing the allocation of resources and providing a seamless and robust virtualization environment for cloud computing [9].

Workflow scheduling poses a substantial challenge within the context of cloud computing, entailing the intricate assignment of workflow tasks to VMs based on a multitude of operational and technical requisites [10]. Workflows are constructed from an array of interdependent tasks, interlinked by either data or functional dependencies, necessitating meticulous consideration during the scheduling endeavor. Nonetheless, the task of workflow scheduling in cloud environments is classified as an NP-hard optimization problem, rendering the attainment of an optimal schedule a formidable undertaking [11]. The cloud environment typically encompasses a multitude of VMs, engendering intricacies in orchestrating an array of user tasks while accounting for diverse scheduling objectives and elements. For instance, a scheduling scheme may prioritize supporting Service Level Agreements (SLAs), predetermined timeframes, and cost limitations. Additionally, scheduling strategies may take into account parameters like the availability of cloud resources and services, load balancing, and resource utilization to make informed scheduling plans.

The integration of cutting-edge technologies such as the Internet of Things (IoT), machine learning, deep learning, and neural networks has revolutionized workflow scheduling, particularly in cloud environments. The IoT facilitates the interconnectedness of devices and sensors, offering real-time data collection and sharing [12, 13]. Machine learning techniques, including supervised and unsupervised algorithms, analyze this data to predict and optimize workflow patterns [14, 15]. Deep learning, a subset of machine learning, uses intricate neural network architectures to process complex data representations, making it adept at recognizing patterns and optimizing scheduling decisions [16]. Neural networks, inspired by the human brain's structure, excel in learning from data and making informed decisions based on this acquired knowledge [17]. Their application in workflow scheduling involves predictive analysis, resource allocation, and task optimization [18]. Collectively, these technologies enable intelligent decision-making, predictive scheduling, and adaptive allocation of tasks within cloud environments. By harnessing IoT data with machine learning, deep learning, and neural networks, workflow scheduling becomes more agile, responsive, and adept at handling the dynamic and complex demands of cloud-based applications, ultimately improving efficiency, resource utilization, and overall performance. Their integration not only streamlines operations but also paves the way for self-optimizing and self-adapting systems in cloud workflow management [19, 20].

Meta-heuristic algorithms are key to workflow scheduling within cloud computing due to the inherent complexity of task allocation. These algorithms, by their nature of adaptive and efficient search strategies, offer an effective way to navigate the vast solution space, addressing the NP-hard nature of scheduling problems [21]. They enable the optimization of resource allocation, contributing significantly to reduced execution times, minimized costs, and improved overall efficiency in cloud-based workflow management. Inverted Ant Colony Optimization (IACO) represents a deviation from the conventional ACO algorithm, which is a metaheuristic derived from the foraging behavior of real ants. In ACO, ants construct solutions by probabilistically choosing paths in a graph based on pheromone trails and heuristic information. The pheromone trails reflect the attractiveness of edges in the graph, and ants deposit pheromones on the paths they traverse. Over time, paths with higher pheromone concentrations become more attractive to other ants, leading to the emergence of high-quality solutions. The IACO algorithm introduces a novel concept to the ACO framework called "inversion." In the traditional ACO, the pheromone trail is reinforced for successful paths, and it is evaporated gradually to encourage exploration. However, in IACO, the pheromone trail on the best path (i.e., the path with the highest desirability) is reduced instead of increased during the pheromone update process. This reduction is referred to as "inversion". The core idea behind the inversion mechanism in IACO is to enhance exploration capabilities. By reducing the pheromone level on the best path, the algorithm encourages ants to explore alternative routes rather than always favoring the currently best-known path. This helps in diversifying the search space and prevents the algorithm from getting stuck in local optima. For workflow scheduling in cloud computing, IACO is applied to find an optimized allocation of tasks to VMs, aiming to lower the total execution time and overall costs. In this context, the graph represents the task dependency graph, and ants traverse paths by assigning tasks to available virtual machines. The main contributions of the study can be summarized as follows:

- Adaptation of traditional ACO algorithm with an inversion mechanism to enhance exploration capabilities and prevent local optima convergence.
- Improvement in the allocation of tasks to VMs, leading to reduced total execution time and minimized costs in cloud-based workflow scheduling.
- Establishment of a pioneering approach that sets a potential benchmark for optimization in cloud computing, influencing future research and practical implementations.

II. RELATED WORK

Choudhary, et al. [22] combined the Heterogeneous Earliest Finish Time (HEFT) heuristic and the Gravitational Search Algorithm (GSA) for workflow scheduling. The GSA is a powerful meta-heuristic that imitates the law of gravity to search for optimal solutions, while HEFT is a widely used heuristic that schedules tasks based on their earliest finish times on heterogeneous resources. One of the key contributions of their work is the introduction of a new factor called "cost time equivalence," which enhances the realism of the bi-objective optimization process. By considering the monetary cost ratio (MCR) and the schedule length ratio (SLR) as performance metrics, they compare the proposed algorithm's performance with existing algorithms. To validate their results, rigorous experiments are conducted over various scientific workflows. They demonstrate the effectiveness of their proposed algorithm by comparing it with standard GSA, Hybrid Genetic Algorithm (HGA), and HEFT. Statistical tests, such as Analysis of Variance (ANOVA), are utilized to validate the results. The simulation results consistently show that the proposed approach outperforms the existing algorithms in terms of both makespan and cost optimization. The algorithm's effectiveness is demonstrated across different workflow scenarios, providing robust evidence of its superiority over the compared algorithms.

Elsherbiny, et al. [23] introduced a novel algorithm that extends the Intelligent Water Drops (IWD) algorithm, a nature-inspired optimization method, to optimize the scheduling of workflows in cloud computing environments. The suggested algorithm is applied and incorporated into the workflow simulation toolkit, allowing for comprehensive testing in various simulated cloud environments with different cost models. The results of the experiments demonstrate that the proposed IWD-based algorithm outperforms classical workflow scheduling algorithms in terms of both performance and cost. They conducted a thorough comparison with several well-known scheduling algorithms. In most situations, the proposed IWD-based algorithm exhibited noticeable enhancements in terms of both performance and cost, outperforming the alternative algorithms. This showcases the effectiveness and efficiency of the IWD-based approach in optimizing workflow scheduling for cloud computing environments.

Ismayilov and Topcuoglu [24] address the intricate challenge of dynamic workflow scheduling within the context of a Dynamic Multi-Objective Optimization Problem (DMOP). This dynamic aspect arises from two primary sources: resource failures (manifested as software or hardware faults) and the inherent variability in the number of objectives during the execution of workflows in real-world cloud computing scenarios. To surmount this intricate problem, the authors propose an innovative prediction-based dynamic multi-objective evolutionary algorithm named NN-DNSGA-II. This algorithm ingeniously combines the capabilities of an artificial neural network with the NSGA-II algorithm, allowing it to make informed predictions concerning the evolving objectives and subsequently adapt its strategies accordingly. The study also involves the adaptation of five prominent non-prediction-based dynamic algorithms from the existing literature, with the overarching goal of addressing the dynamic workflow scheduling dilemma. The NN-DNSGA-II algorithm is thoughtfully designed to encompass six distinct objectives within the scheduling process. It aims to minimize critical aspects such as makespan, cost, energy consumption, and degree of imbalance while simultaneously maximizing attributes like reliability and utilization. To assess its efficacy, the authors conducted comprehensive empirical studies employing real-world applications sourced from the Pegasus workflow management system. This rigorous evaluation entails a range of metrics tailored for DMOPs characterized by unknown true Pareto-optimal fronts. Metrics include considerations such as the number of non-dominated solutions, Schott's spacing, and the Hypervolume indicator. The findings derived from the empirical investigation reveal the remarkable performance of the NN-DNSGA-II algorithm. It consistently outperforms alternative algorithms across various scenarios, underlining its supremacy in effectively managing dynamic workflow scheduling imbued with multiple objectives and unknown true Pareto-optimal fronts.

Mangalampalli, et al. [25] introduced a novel workflow-scheduling mechanism that incorporates task priorities to schedule tasks onto appropriate virtual resources efficiently. The Whale Optimization Algorithm (WOA) was used as the methodology to model this algorithm. Extensive simulations were conducted using the workflow simulator to evaluate the proposed mechanism's performance. The mechanism was compared against existing algorithms, including PSO, CS, ACO, and GA. The simulation results revealed significant improvements in makespan, migration time, and energy consumption when using the proposed mechanism. These improvements indicate the effectiveness of the WOA-based workflow-scheduling approach in optimizing task scheduling in cloud computing environments. By considering task priorities, the proposed mechanism is able to make more informed and efficient scheduling decisions, leading to reduced makespan (total execution time), migration time (task relocation between resources), and energy consumption. These improvements are crucial for enhancing the overall performance and resource utilization in cloud-based workflow management.

Zeedan, et al. [26] introduced an innovative approach termed Enhanced Binary Artificial Bee Colony-based Pareto Front (EBABC-PF) for optimizing workflow scheduling in cloud computing environments. The proposed approach involves a sequence of strategic steps aimed at achieving efficient task scheduling. The initial step of the approach involves task prioritization using the HEFT algorithm. HEFT organizes tasks based on their earliest finish times across heterogeneous resources, thereby establishing a prioritized sequence. Subsequently, an initial solution is constructed using the Greedy Randomized Adaptive Search Procedure (GRASP), a constructive metaheuristic approach renowned for its optimization capabilities. The core task scheduling phase is executed through the utilization of the enhanced Binary Artificial Bee Colony (BABC) algorithm. This modified version of the BABC algorithm integrates several enhancements specifically targeted at refining the local search process. The process incorporates circular shift and mutation operators, which are applied to the population's food sources while considering the improvement rate. These enhancements contribute to augmenting the algorithm's search capacity and effectiveness. The proposed EBABC-PF approach is simulated and implemented using WorkflowSim, an extension of the CloudSim tool designed to manage workflows within cloud environments. To assess its performance, the approach is rigorously compared against a range of other scheduling algorithms, which include HEFT, Deadline Heterogeneous Earliest Finish Time (DHEFT), Non-dominated Sort Genetic Algorithm (NSGA-II), and the standard BABC algorithm. This comparative analysis is conducted across diverse task sizes and benchmark workflows. The simulation results obtained exhibit the exceptional efficiency of the proposed EBABC-PF approach across multiple performance metrics. It notably outperforms the alternative algorithms in terms of makespan (total execution time), processing cost, and resource utilization. This finding underscores the approach's effectiveness in optimizing workflow scheduling within cloud computing environments, rendering it a superior choice for this intricate task.

III. Proposed Approach

A. Problem Statement

In the domain of cloud computing, workflow scheduling involves the representation of workflows as Directed Acyclic Graphs (DAGs), denoted as $G = (V, E)$. V refers to a collection of vertices, each representing an individual task within the workflow. E, on the other hand, denotes the set of edges signifying task dependencies. In this setup, tasks must be executed in a specific order, where parent tasks precede the execution of their child tasks. Fig. 1 offers a concrete illustration of task dependencies spanning from T_1 to T_{10}. Serving as the root node, T_1 takes the lead as the first task to be executed. Once T_1 is finished, tasks T_2 and T_3, located on the first tier of the DAG, are initiated. In a parallel manner, once task T_2 is accomplished, tasks T_4 and T_5 are set into motion.

Additionally, the execution of task T_6 is contingent upon the completion of task T_3, thus establishing T_3 as the necessary precursor to T_6. Scientific workflows encompass a specialized class of workflows extensively utilized across a range of scientific fields, such as astronomy, biology, and gravitational waves, among others. Prominent instances of practical scientific workflows include SIPHT, LIGO, Epigenomics, CyberShake, and Montage, all meticulously cataloged by the Pegasus project. The structural depiction of these scientific workflows is presented in Fig. 2. The scheduling of workflows can be perceived as a mapping function, allocating numerous interdependent tasks to available virtual machines. A sample mapping is illustrated in Fig. 3, demonstrating the allocation of n tasks to m VMs. In such instances, when employing a brute force algorithm, there emerge $m*n$ potential combinations. Consequently, the intricacy of workflow scheduling is acknowledged, and achieving a solution within polynomial time is not attainable. Consequently, the pursuit of a nearly optimal resolution to the workflow scheduling predicament proves advantageous and attainable through the assistance of meta-heuristic algorithms.

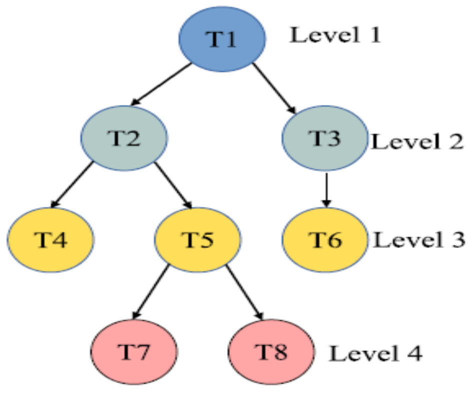

Fig. 1. Task dependencies in workflow scheduling

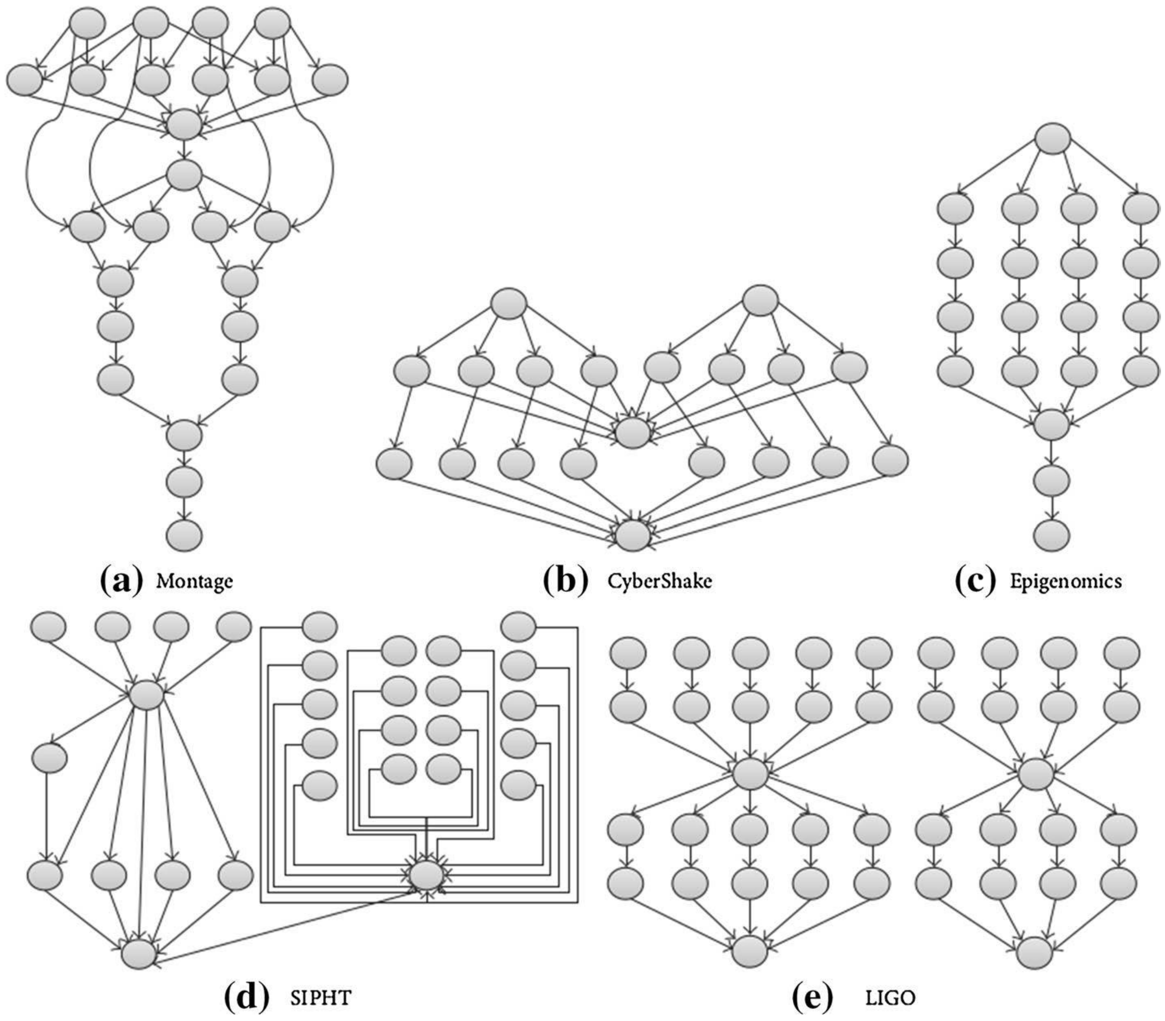

Fig. 2. The structural depiction of scientific workflows.

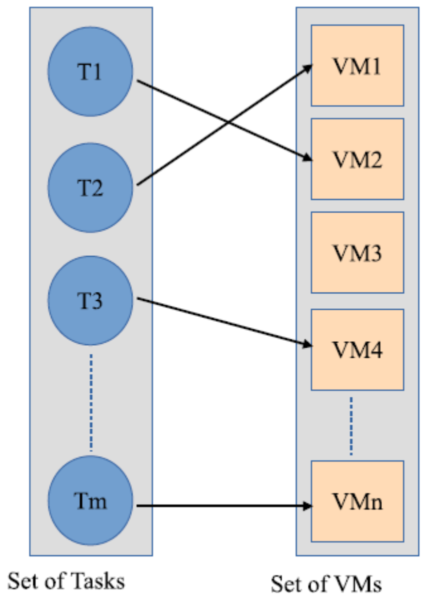

Fig. 3. Task mapping model.

B. Fitness Function

The optimization algorithm aims to enhance specific parameters within the fitness function. In this context, we have employed two distinct fitness functions, labeled *F1* and *F2*, which are the focal points of optimization through the proposed algorithm. The initial fitness function, *F1*, encompasses a synthesis of the Total Execution Time (TET), whereas the subsequent fitness function, *F2*, is constructed from the Total Execution Cost (TEC). The precise formulations of TET and TEC are outlined in Eq. (1) and Eq. (5), respectively.

$$Fitness\ (F_1) = Total\ Execution\ Cost\ (TEC) \tag{1}$$

The total execution time, often referred to as the makespan, signifies the longest duration taken by tasks within the workflow to reach completion. But it quantifies the time necessary for accomplishing all tasks distributed among various VMs. The mathematical expression to compute the makespan of the workflow can be deduced from Eq. (2), where CT_i symbolizes the completion time of task T_i within the workflow. The completion time of a task encompasses its entire execution duration, and in cases where task dependencies exist, the waiting time of preceding tasks is also considered. The calculation of completion time is presented in Eq. (3). The waiting time for task T_i is established as the utmost completion time among all predecessor tasks within the workflow, as delineated in Eq. (4). Furthermore, the execution time of task T_i on jth VM can be evaluated using Eq. (5). Here, SZ_{Task} signifies the size of task T_i, quantified in million instructions (MI), $Num\ (PE_j)$ represents the count of cores allocated to jth VM, and PE_{Unit} denotes the magnitude of each core in Millions of Instructions per Second (MIPS).

$$TET_W = \max\{CT_t | 1,2,\ldots,m\} \tag{2}$$

$$CT_i = \begin{cases} ET_i & if\ pred(T_i) = \emptyset \\ WK_i + ET_i & if\ pred(T_i) \neq \emptyset \end{cases} \tag{3}$$

$$WK_i = \begin{cases} 0 & if\ pred(T_i) = \emptyset \\ \max(CT_i) & if\ pred(T_i) \neq \emptyset \end{cases} \tag{4}$$

$$ET_{i,j} = \frac{SZ_{Task}}{Num(PE_j) \times PE_{Unit}} \tag{5}$$

Eq. (6) delineates the process for calculating TEC. The TEC for the ith VM is derived by calculating the disparity between the Least End Time (LET) and the Least Start Time (LST) of that specific VM. The cost associated with the ith VM, denoted as *C[i]*, is uniformly set at 1 across all VMs. To provide further clarity, Eq. (7) and Eq. (8) elucidate the mechanics behind computing the LET and LST, respectively. LET pertaining to the ith VM corresponds to the highest execution time among all tasks executed on that particular VM. Conversely, LST is determined as the minimum execution time of tasks in progress on the ith VM.

$$Total\ Execution\ Cost\ (TEC) = \sum_{i=0}^{VM} C[i] \times (LET[i] - LST[i] \tag{6}$$

$$LET[i] = \max(ET[i]) \tag{7}$$

$$LST[i] = \min(ET[i]) \tag{8}$$

C. Proposed Algorithm

ACO algorithms harness a populace of ants to collaboratively address optimization challenges, navigating graphs to discover paths of minimal cost while upholding specific constraints. The behavior of these ants encompasses two distinct groups: a smaller ensemble lays down pheromone trails, while the other contingent diligently tracks these trails, reinforcing them while circumventing impulsive moves. As time elapses, the potency of these trails diminishes, leading to a waning allure for the ants. Let *G = (V, E)* symbolize the graph that underpins the optimization conundrum, where *V* denotes vertices and *E* signifies edges. On this graph, viable pathways correspond to potential resolutions for the optimization predicament. While in pursuit of the shortest path, the ants deposit pheromones along their journey, cultivating an enduring memory of the exploration process. Furthermore, heuristic values might be attributed to the graph's edges, derived from antecedent knowledge or real-time feedback, exerting influence over the ants' conduct.

The decision-making process of the ants is probabilistic in nature and relies on their memory, the constraints of the problem, and the ant-routing table, a localized data structure housing pheromone trails and heuristic values. Pheromone updates transpire through two distinct mechanisms: an online step-by-step update involving the deposition of pheromone by the ant while traversing an edge and an online delayed update, which involves adjusting pheromone trails after discovering a solution and retracing the path in reverse. Additional processes for updating pheromone trails include daemon actions and pheromone evaporation. Daemon actions, while discretionary, enable ants to execute actions that are beyond their individual capacities, often involving centralized actions. Pheromone evaporation entails a gradual reduction in the strength of pheromone trails over time. This mechanism prevents the convergence toward suboptimal regions and promotes the exploration of novel areas within the graph.

Algorithm. 1. Generic ACO algorithm

```
Initialize
While stop criteria are not met do
    For all ant $a$ in $A$ do
        Position a in startNode
    End for
    Repeat
        For all ant $a$ in $A$ do
            Choose nextNode
            Pheromone$_{(currentNode,nextNode)}$+=Update
        End for
    Until every ant has a solution
    For all edge $e$ in $B$ do
        Pheromone$_e$+=Deposit
    End for
    For all edge $e$ in $E$ do
        Pheromone$_e$-=Evaporation
    End for
End while
```

Eq. (9) calculates the probability of an ant selecting a particular path. It depends on the number of pheromones$\tau_{i,j}$, present on the path between nodes i and j and the reverse of the distance ($\eta_{i,j}$) between these two nodes. The parameters α and β are control parameters that influence the relative importance of pheromones and distance in the probability calculation.

$$p_{i,j}^{k} = \frac{[\tau_{i,j}]^{\alpha}[n_{i,j}]^{\beta}}{\sum_{l=j_i}[\tau_{i,l}]^{\alpha}[n_{i,l}]^{\beta}} \tag{9}$$

The ants act like scouts, searching for food (appropriate service corresponding to user demands) in the environment. Once they find food, they return to their nests, dropping pheromones on the trails they have traversed. These pheromones serve as a form of communication for other ants, indicating the quality of the path. The pheromone trail amount can either increase when ants deposit pheromone or decrease over time due to pheromone evaporation. Eq. (10) calculates the pheromone evaporation rate, where $1-\omega$ is the pheromone declining rate.

$$\tau_{i,j} = (1-\omega) \times \tau_{i,j} + \sum_{r=1} \Delta\tau^{r} i,j \tag{10}$$

In the ACO algorithm, other ants tend to follow paths with a high amount of pheromone, as it indicates better solutions. There are two main approaches to updating the pheromone trails. The first approach involves selecting the best-so-far solutions (iteration best) and using them to update the pheromone matrices for each objective. These best solutions represent the most promising paths found so far in the search process. The second approach revolves around gathering and storing non-dominated solutions in an external set. Only the solutions in this non-dominated set are allowed to update the pheromone trails. Non-dominated solutions are those that cannot be improved in one objective without worsening at least one other objective. This approach helps maintain a diverse set of optimal solutions. Once the ants find food, they return to their nests, dropping pheromones on the trail they traveled. This pheromone serves as a signal for other ants to explore the same path, thus collectively reinforcing good solutions. Ants make local decisions based on their observations and the information available in their local environment. Instead of directly communicating with each other, ants use indirect forms of communication, which is referred to as "stigmergy." The pheromones left by ants' act as a form of stigmergic communication, guiding other ants to explore the most promising paths. Over time, the pheromones evaporate, which allows the algorithm to explore new areas and avoid convergence to sub-optimal solutions. The rate of pheromone evaporation is higher when returning to the nest takes a longer time, promoting the exploration of alternative paths. At intersections in the graph, each ant chooses one of the branches to continue its path. Ants tend to select shorter branches to return home faster, resulting in more pheromone accumulation on these shorter paths.

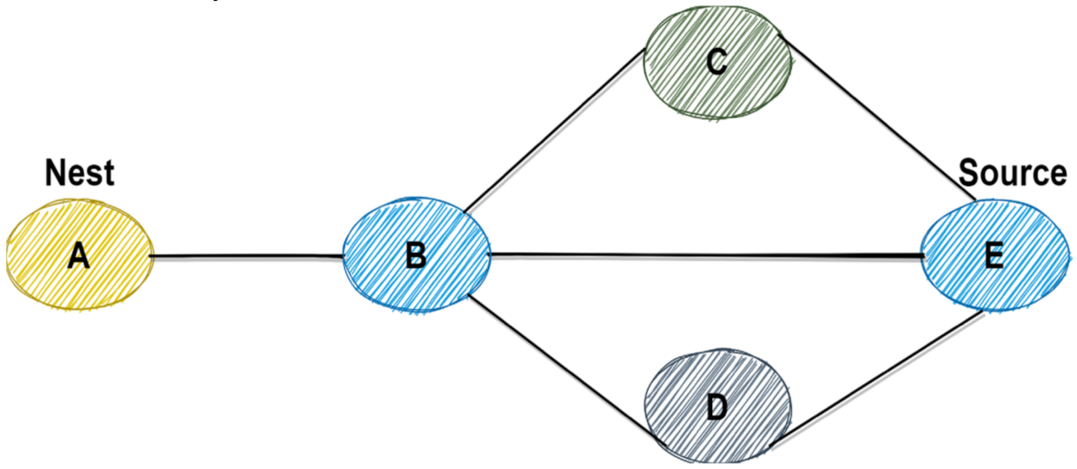

Fig. 4. Path selection by ants.

In the presented scenario (see Fig. 4), the ants encounter three paths: $A \rightarrow B \rightarrow C \rightarrow E$, $A \rightarrow B \rightarrow E$, and $A \rightarrow B \rightarrow D \rightarrow E$. Initially, due to the shorter distance, the shortest path $A \rightarrow B \rightarrow E$ will have a higher pheromone density. As the ants make their choices, they select paths based on the pheromone densities present on the branches. Consequently, most ants will opt for the shortest path, $A \rightarrow B \rightarrow E$. The main focus of the article is on the IACO algorithm for workflow scheduling in cloud computing. This algorithm combines the principles of ACO with a constrained optimization approach, which is suitable for solving single-objective discrete optimization problems. By leveraging pheromone communication and making local decisions based on observed pheromone densities, the IACO algorithm effectively guides ants to converge toward the shortest and most feasible paths in the search space.

Indeed, the IACO method exhibits several significant differences in each ant's behavior compared to the traditional ACO. The main distinctions between ACO and IACO are as follows:

- Pheromone functionality: In the ACO, the pheromones produced by ants serve only the function of attraction. Pheromones act as a positive signal, guiding ants to explore paths with higher pheromone densities. In contrast, the pheromones in the IACO method serve a dual purpose. Besides attraction, they also act as a form of repulsion. This repulsive nature of pheromones helps to alleviate the pressure on certain paths or nodes. It discourages ants from over-converging on a specific route or node.
- Role of nodes: In the ACO, nodes do not directly influence the algorithm. The focus lies primarily on paths and the pheromones present on those paths as important parameters.
- IACO Method: In the IACO, nodes play a significant role as influencing parameters. They store the number of pheromones present. If the pheromone count exceeds a specific value, the node acts as a blockade, preventing other ants from passing through it. This approach is designed to reduce pressure on certain nodes and is directly linked to the pheromone value.
- Consideration of route capacity: ACO does not explicitly consider the maximum capacity on a path when updating pheromones. In the IACO, the pheromone's evaporation process takes into account both the length of the route and the maximum capacity of the route. This consideration helps to ensure that the algorithm avoids routes that may become congested due to excessive pheromones.

These differences in behavior and parameterization between ACO and IACO allow the IACO method to address complex optimization problems more effectively, especially in scenarios where path congestion and capacity constraints are crucial considerations. By combining both attractive and repulsive properties of pheromones and incorporating node-based mechanisms, IACO offers an enhanced approach for finding optimal or near-optimal solutions in constrained optimization problems.

IV. PERFORMANCE EVALUATION

In the experiments conducted to verify the performance of the proposed method, we used the WorkflowSim toolkit. The implementations of strategies were implemented in Java and run on a computer with an Intel Core i5 processor running at 2.8 GHz and equipped with 4 GB of RAM. The WorkflowSim toolkit is an extension of CloudSim, designed specifically to simulate an environment for executing scientific workflows. In pursuit of an equitable comparison of outcomes, the simulations involving the proposed approach and other established methods were orchestrated under identical conditions. This systematic parity in experimentation conditions serves to objectively gauge the performance and efficacy of the proposed approach in contrast to alternative methodologies. The experimental trials were conducted utilizing three distinct real-world workflow applications, each hailing from diverse scientific domains. Noteworthy among these is the Montage workflow applied within the realm of astronomical physics. Furthermore, the Cybershake workflow was harnessed to analyze earthquake hazards, while the Ligo workflows were instrumental in the quest for gravitational wave detection.

The configuration of VMs is based on the specifications of Amazon EC2 instances, which are commonly used in cloud computing environments. Within the simulation model, it is assumed that the storage capacity of every VM is generously sufficient to host all the allocated tasks. Nonetheless, the mean bandwidth linking distinct virtual machines exhibits variability across three distinct scenarios: 5 Gb/s, 10 Gb/s, and 25 Gb/s. The variation in bandwidth represents different network capacities and performance levels that can affect task execution and communication between VMs. Additionally, the virtual machine preparation time is taken into consideration during the simulations. This preparation time represents the overhead required to set up and configure a VM before it can start executing tasks. The temporal interval required for VM preparation is modeled to fluctuate within the range of 1 second to 1.5 seconds. During the evaluation process of the proposed IACO algorithm, a comprehensive comparison is conducted against three alternative algorithms: standard ACO, FR-MOS, and PEFT-ACO algorithms.

In the first series of experiments, the algorithms were evaluated independently for each objective, i.e., cost and makespan. The experiments were performed on three different types of workflows: Montage, Cybershake, and Ligo, with varying numbers of tasks (100 and 300). The results averaged over 100 executions for each workflow type, are presented in Table I. It is observed that the traditional ACO algorithm performs poorly in both the total cost and makespan across all three workflow types. On the other hand, the IACO algorithm is superior to both PEFT-ACO and FR-MOS regarding cost and makespan across all workflow types. The discernible distinction in performance is especially pronounced with regard to makespan. This indicates that IACO is more efficient and effective in exploring the solution space and finding globally optimal or near-optimal solutions. Fig. 5 and Fig. 6 provide visualizations of the average results for makespan and cost when the number of tasks is set to 500.

TABLE I. OBTAINED RESULTS FOR COST AND MAKESPAN

Workflow	Tasks count	IACO		FR-MOS		PEFT-ACO		ACO	
		Cost ($/h)	*Make span (s)*	*Cost ($/h)*	*Make span (s)*	*Cost ($/h)*	*Make span (s)*	*Cost ($/h)*	*Make span (s)*
Montage	100	14.1	468.2	15.9	614.7	25.6	625.8	52.2	654.3
	300	9.2	584.7	13.8	679.4	20.1	693.5	30.1	708.3
Cybershake	100	21.8	489.5	40.7	513.7	57.7	578.5	70.2	695
	300	19.8	627.1	32.1	659.4	43.9	701.8	53.7	872.4
Ligo	100	31.5	559.1	46.9	609.5	69.7	695.8	78.7	710.8
	300	26.9	697.4	42.8	823.4	64.4	888.1	77.2	896.7

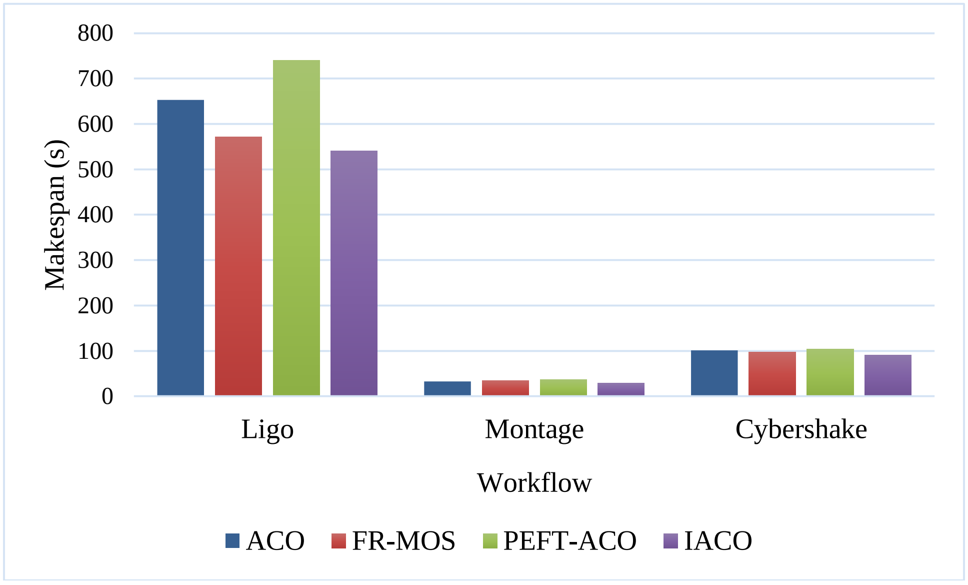

Fig. 5. Makespan comparison.

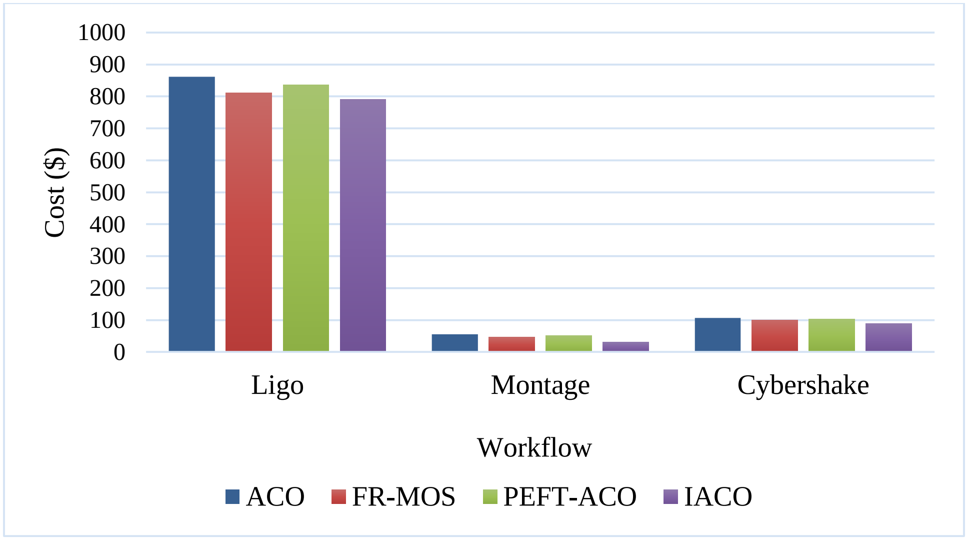

Fig. 6. Cost comparison.

V. CONCLUSION

Cloud computing has emerged as a revolutionary concept in the field of distributed systems, providing efficient and scalable solutions for high-performance and distributed computing needs. Its dynamic nature, virtual resource provisioning, and pay-per-use model have contributed to its widespread popularity among organizations and research laboratories. In this context, workflow scheduling plays a crucial role in optimizing the execution of applications in cloud environments. Workflows model processes as a sequence of steps, and scheduling involves assigning each task of the workflow to an appropriate processing resource while adhering to specific workflow rules and constraints. This paper proposed a novel workflow scheduling approach called Inverted ACO (IACO). The IACO algorithm leverages the principles of the ACO algorithm but introduces unique modifications to improve its performance in cloud-based workflow scheduling scenarios. The experiments were carried out utilizing instances from the Amazon EC2 cloud platform, encompassing three distinct real-world workflow types originating from various scientific domains.

In the comparative analysis, IACO was assessed against established methodologies in the field, namely standard FR-MOS, PEFT-ACO, and ACO algorithms. The results of the experiments show that IACO outperforms all other algorithms in most tests, particularly in terms of the trade-off between makespan and cost. This work could explore how the IACO algorithm adapts dynamically to varying cloud environments, taking into account factors like workload fluctuations and cloud structure diversity. Additionally, exploring the integration of IACO with machine learning or other metaheuristic approaches could expand its efficiency and applicability. Moreover, investigating the scalability of IACO in hybrid cloud settings or its adaptability to real-time workflow changes could significantly advance its practical implementation. Furthermore, assessing the robustness of IACO against various cloud constraints, including resource limitations or network latency, would contribute to a more comprehensive understanding of its performance under diverse cloud scenarios.

REFERENCES

[1] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Computing, pp. 1-24, 2021.

[2] M. Hosseinzadeh et al., "A Hybrid Service Selection and Composition Model for Cloud-Edge Computing in the Internet of Things," IEEE Access, vol. 8, pp. 85939-85949, 2020.

[3] R. K. Tiwari and R. Kumar, "G-TOPSIS: a cloud service selection framework using Gaussian TOPSIS for rank reversal problem," The Journal of Supercomputing, vol. 77, no. 1, pp. 523-562, 2021.

[4] S. K. Panda and P. K. Jana, "An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems," Cluster Computing, vol. 22, no. 2, pp. 509-527, 2019.

[5] V. Sundararaj, "Optimal task assignment in mobile cloud computing by queue based ant-bee algorithm," Wireless Personal Communications, vol. 104, no. 1, pp. 173-197, 2019.

[6] X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing," Journal of Ambient Intelligence and Humanized Computing, pp. 1-12, 2020.

[7] B. Cao, Z. Sun, J. Zhang, and Y. Gu, "Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 6, pp. 3832-3840, 2021.

[8] P. Behrouz, H. Vahideh, and A. A. Aghaei, "Service discovery in the Internet of Things: review of current trends and research challenges," Wireless Networks, vol. 26, no. 7, pp. 5371-5391, 2020.

[9] A. Sheeba and B. Uma Maheswari, "An efficient fault tolerance scheme based enhanced firefly optimization for virtual machine placement in cloud computing," Concurrency and Computation: Practice and Experience, vol. 35, no. 7, p. e7610, 2023.

[10] H. Vahideh, P. Behrouz, P. K. A. Asghar, and A. Ghaffari, "Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques," The International Journal of Advanced Manufacturing Technology, vol. 105, no. 1-4, pp. 471-498, 2019.

[11] V. Hayyolalam, B. Pourghebleh, M. R. Chehrehzad, and A. A. Pourhaji Kazem, "Single-objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends," Concurrency and Computation: Practice and Experience, vol. 34, no. 5, p. e6698, 2022.

[12] B. Pourghebleh and V. Hayyolalam, "A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things," Cluster Computing, pp. 1-21, 2019.

[13] P. He, N. Almasifar, A. Mehbodniya, D. Javaheri, and J. L. Webber, "Towards green smart cities using Internet of Things and optimization algorithms: A systematic and bibliometric review," Sustainable Computing: Informatics and Systems, vol. 36, p. 100822, 2022, doi: https://doi.org/10.1016/j.suscom.2022.100822.

[14] S. N. H. Bukhari, J. Webber, and A. Mehbodniya, "Decision tree based ensemble machine learning model for the prediction of Zika virus T-cell epitopes as potential vaccine candidates," Scientific Reports, vol. 12, no. 1, p. 7810, 2022.

[15] T. Gera, J. Singh, A. Mehbodniya, J. L. Webber, M. Shabaz, and D. Thakur, "Dominant feature selection and machine learning-based hybrid approach to analyze android ransomware," Security and Communication Networks, vol. 2021, pp. 1-22, 2021.

[16] B. M. Jafari, M. Zhao, and A. Jafari, "Rumi: An Intelligent Agent Enhancing Learning Management Systems Using Machine Learning Techniques," Journal of Software Engineering and Applications, vol. 15, no. 9, pp. 325-343, 2022.

[17] M. Sadi et al., "Special Session: On the Reliability of Conventional and Quantum Neural Network Hardware," in 2022 IEEE 40th VLSI Test Symposium (VTS), 2022: IEEE, pp. 1-12.

[18] J. Webber, A. Mehbodniya, Y. Hou, K. Yano, and T. Kumagai, "Study on idle slot availability prediction for WLAN using a probabilistic neural network," in 2017 23rd Asia-Pacific Conference on Communications (APCC), 2017: IEEE, pp. 1-6.

[19] R. Singh et al., "Analysis of Network Slicing for Management of 5G Networks Using Machine Learning Techniques," Wireless Communications and Mobile Computing, vol. 2022, 2022.

[20] S. Habib, S. Aghakhani, M. G. Nejati, M. Azimian, Y. Jia, and E. M. Ahmed, "Energy management of an intelligent parking lot equipped with hydrogen storage systems and renewable energy sources using the stochastic p-robust optimization approach," Energy, p. 127844, 2023.

[21] S. Mahmoudinazlou, A. Alizadeh, J. Noble, and S. Eslamdoust, "An improved hybrid ICA-SA metaheuristic for order acceptance and scheduling with time windows and sequence-dependent setup times," Neural Computing and Applications, pp. 1-19, 2023.

[22] A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, "A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing," Future Generation Computer Systems, vol. 83, pp. 14-26, 2018.

[23] S. Elsherbiny, E. Eldaydamony, M. Alrahmawy, and A. E. Reyad, "An extended intelligent water drops algorithm for workflow scheduling in cloud computing environment," Egyptian informatics journal, vol. 19, no. 1, pp. 33-55, 2018.

[24] G. Ismayilov and H. R. Topcuoglu, "Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing," Future Generation computer systems, vol. 102, pp. 307-322, 2020.

[25] S. Mangalampalli, G. R. Karri, and G. N. Satish, "Efficient Workflow Scheduling algorithm in cloud computing using Whale Optimization," Procedia Computer Science, vol. 218, pp. 1936-1945, 2023.

[26] M. Zeedan, G. Attiya, and N. El-Fishawy, "Enhanced hybrid multi-objective workflow scheduling approach based artificial bee colony in cloud computing," Computing, vol. 105, no. 1, pp. 217-247, 2023.