

A CYBER RESILIENT HYBRID META-HEURISTIC UNDER DIGITAL PERSONAL DATA PROTECTION UNDER ACT 2023 FOR DYNAMIC LOAD BALANCING MANAGEMENT AND FAULT-TOLERANT SCHEDULING IN MULTI-CLOUD SYSTEMS

Abinaya P^{a,*}, Swathika P^b, Jegath Priya S^c, Divya Jeyashree M^d

^{a, c, d}Department of Computer Science and Engineering, Mepco Schlenk Engineering College (Autonomous)
Sivakasi. Tamil Nadu, India.

^bDepartment of Artificial Intelligence and Data Science, Mepco Schlenk Engineering College (Autonomous)
Sivakasi. Tamil Nadu, India

email's *abinayap@mepcoeng.ac.in, swathikap@mepcoeng.ac.in, jegathpriya2004_bcs26@mepcoeng.ac.in,
divyajeyashreem_bcs26@mepcoeng.ac.in

Abstract- The problem of efficient task scheduling and load balancing in the multi-cloud system is quite difficult because of variable work load, heterogeneity of VMs and security attacks on cloud system. In this paper, a novel approach of secure and fault tolerant hybrid scheduling framework by combining COATI Optimization Algorithm (COATI) and Ant Colony Optimization (ACO) is presented. The developed algorithm called Fault-Tolerant Cooperative COATI-ACO (FT-COACO) uses COATI for the global search and ACO for optimal local search to better map tasks to VMs. The system uses the concepts of trust aware scheduling, anomaly detection and isolate malicious VMs to provide secure and reliable environment. This framework is evaluated in the multi cloud environment with different virtual machine specification, task sensitivities in the presence and absence of an attack. From the experimentation result for different number of 100 to 500 cloudlets, it is clear that there are improvement over Round Robin such as 15.41% of decrease in make span, 5.56% of decrease in execution cost, 18.14% of increase in throughput and 15.53% of increase in resource utilization, additionally it improved the success rate of task to 0.08% more and score of resilience is found to be 1.0028, showing effective fault tolerant and secure system for cloud.

Keywords: Multi-cloud, load balancing, task scheduling, digital data protection, COATI optimization, ant colony optimization.

1. INTRODUCTION

The rapid growth of the cloud computing technology has given more accessibility to manage and utilize the computation resources flexibly. Along with this boom, multi-cloud architectures have become to increasingly demand on the allocation of tasks among different cloud service providers, so that can achieve scalability, reliability and cost-saving benefit [1]. However, there is a significant challenge for efficient scheduling and load balancing of tasks on multi-cloud environment, owing to its heterogeneous VMs configuration, volatile workload dynamic, and the unpredictability of task processing times and the occurrence of failed VMs. Otherwise (scheduling was not performed), this results in higher makespan, operational costs, utilization, and negative effects of reliability. Round Robin, FCFS and Min-Min are only confined to the simple load balancing, and never take into account trust and abnormal behavior of VM. In addition, there have been many meta heuristic-based schedulers, e.g., multi-objective COA, hybrid GA-GWO approach [2], and Harris Hawks based scheduler, proposed to effectively optimize the performance in cloud environments. These algorithms provide better performance in terms of makespan and cost, but the absence of anomaly detection mechanism, malicious VM identification, and dynamic trust value updates is one of the limitations. In this paper, we introduce the FT-COACO scheduling framework which supports efficient and secure resource management on multi-cloud infrastructures.

Our work mainly fills the gap that existing schedulers treat efficiency and security in isolated aspects, providing a combined scheduler, which could optimize both performance and security. Our system differs from traditional approaches; for instance, while a Round Robin scheduler is concerned only with the resource health, and an original ACO/COATI scheduler does not consider malicious virtual machines, the FT-COACO takes both the parameters into the proposed system to provide a robust "cyber-resilient brain". Unlike the COATI which searches the entire state space, and both COA and its variations may become trapped into a local minimum, our system has successfully formed a tight feedback loop using the merits of both: exploration (COATI) and exploitation (modified COA), to schedule tasks only on reliable working virtual machines. Our major contribution is to propose a multi-layer resource management architecture which applies COATI for feasible search space exploration and a modified ACO (using trust-based pheromone model) for optimized task placement based on the operational

reliability of virtual machine. We also demonstrate how to improve the overall efficiency and security of the system by integrating the anomaly detection mechanism, evidence-based trust evaluation function, and self-healing ability. As evidenced from our previous work, most existing solutions treat security and optimization independently and are not able to make dynamic adjustment of priorities when anomalies occurred. Moreover, meta-heuristics used in most solutions do not consider both security and optimization aspects concurrently but as separate parts without autonomous self-healing ability to deal with QoS constraints as well as trust degradation. In this work, we have, for the first time, proposed a framework that enable securely cooperating heterogeneous multi-cloud systems. The remaining of the paper is organized as follows: related work is reviewed in Section 2, followed by the system backgrounds in Section 3, our proposed FT-COACO scheduling framework in Section 4, evaluation in Section 5 and finally the conclusion and future work in Section 6.

2. RELATED WORK

A number of metaheuristics, mathematical, bio-inspired methods has been investigated for optimizing makespan, energy consumption, cost and fault tolerance measures in cloud and multi-cloud systems. For a fault tolerant load balanced task scheduling in multi-clouds, the multi-objective Cat Swarm Optimization (MO-CSO) method has been designed that enhances the makespan and resource utilization and provides system reliability via the use of a failure-aware scheduling scheme. Although MO-CSO has overcome the energy-performance trade-off issues but the convergence speed of the algorithm is greatly dependent upon complexity and ever-increasing interdependencies of multi-cloud resources. Abu-Hashem et al. suggested Improved Black Widow Optimization (IBWO) algorithm to make a better cloud task scheduling using the limitation of the conventional Black Widow Optimization technique. The improved exploration and exploitation strategies proposed in this method prevent premature convergence and enhance the solution quality [3]. Geetha et al. presented a Hybrid Optimization Algorithm (HOA) for optimum load balancing and task scheduling in the cloud computing environments. The proposed approach will combine several optimizations to obtain a balanced workload distribution and optimization of the use of resources [4]. To solve the cloud task scheduling problem, Zhang et al. introduced a Multi-Objective Honey Badger Algorithm (MOHBA) that has two hybrid elite learning algorithms and a circular segmentation screening mechanism. The algorithm is designed to optimize several objectives: makespan, execution cost, load balancing and resource utilization. The hybrid elite strategies boost the global search ability and the circular segmentation boosts the population diversity and prevents premature convergence [5]. To achieve efficient cloud task scheduling, Khan and Rasool proposed a Multi-Objective Grey Wolf Optimization (MOGWO) algorithm that takes into account multiple optimization goals at the same time. The proposed one can reduce makespan, execution cost and energy consumption and increase resource utilization and workload balancing among cloud resources [6]. Gao et al. proposed a Two-Layer Optimization Method for maintenance task scheduling, where multiple priorities of the tasks and constraints on the resources are taken into account [7]. To schedule tasks in the clouds, Chandrashekar et al. have introduced Modified Chimp-Whale Optimization Algorithm (MCWOA) to hybridize the exploration ability of Chimp Optimization Algorithm with the exploitation ability of Whale Optimization Algorithm [8]. Fault-tolerant scheduling for reliability and energy efficiency have also been extensively studied [9], but not from a context of secure cooperation of hybrid multi-cloud systems. This problem is solved by Hosseini et al. in their work on a Multi-Objective Discrete Cuckoo Search (MODCS) algorithm for Bag-of-Tasks (BoT) scheduling in fog computing environments. The algorithm achieves both reduced execution time and energy consumption, and enhances the utilization of resources, outperforming the existing metaheuristic algorithms [10]. To enhance the energy efficiency and fault tolerance for mission critical task scheduling in cloud computing environment, Saberikia et al. presented a Mixed-Integer Linear Programming (MILP) model. The proposed approach is expected to reduce the energy consumption in the system while maintaining the reliability of the tasks being performed and efficient utilization of the system resources, which will improve the system performance and reliability of mission critical applications in the cloud environment [11]. A Hybrid Spider Monkey Optimization and Fuzzy Self-Defense (HSMO-FSD) algorithm for multi-objective scientific workflow scheduling in cloud computing was proposed by Khaleel. The proposed method is able to achieve better makespan, execution cost and load balancing than the conventional scheduling algorithm, and is able to guarantee the better Quality of Service (QoS) of executing the workflow [12]. A deep learning method that is combined with the Fire Hawk optimization is proposed by Indumathi et al. [15] in an integrated hybrid cloud container architecture, which address the interoperability problem and migration of applications among hybrid environments. The deep learning method resulted in increased prediction accuracy of the resources demand for the containerized layer but the management of multi-cloud interoperability protocols was found to be a large overhead of the architecture. The hybrid algorithm that combined the GA and PSO have been developed which provides the maximization of edge servers and minimized the coverage area of edge servers with budget constraints. It was effective for the search but assumption of the static deployment of this architecture will not be applicable in mobile environments [16]. The multi-objective task scheduling in cloud-fog systems that optimize latency, cost and energy usage using the Harris Hawks Optimization (HHO) has been proposed by Ali et al. [17]. The efficiency of HHO method in the terms of convergence speed and optimal task distribution between

resources were found to be promising but communication latency among fog nodes and central cloud still a concern. The multi-objective evolutionary algorithm for task scheduling has been designed [18] in the cloud environment by Liu et al. That has capability to find critical tasks in the large-scale workflow, and achieves high makespan and better resource utilization but first finding the critical task might be a problem in large scale dynamic workflow. A Brokenstick Regression-based multi-objective dragonfly predictive optimization (BR-MDPO) for an Industrial IoT cloud environment has been developed and design for machine failure prediction based on sensors' data. The accuracy has increased and latency has reduced (more than 13%) compared to traditional methods but this system has high computational complexity for training phase [19]. HHO-based multi-objective task scheduling mechanism that explores various parameters to exploit the capabilities of HHO algorithm has been implemented and the performance of HHO algorithm for makespan, resource utilization and energy consumption is improved [20]. This method claimed 78% resource utilization and 20.3% completion time improvement but fault tolerance during repeated VM failure could be a concern. The IGA-based resource scheduling for open-source IaaS cloud systems is modelled by the "Dividend Policy" [21]. It had aimed to improve resource usage over traditional GA by double. In addition, the performance is better than other IaaS platforms (Eucalyptus and OpenNebula) but it lacks the scalability required for the federation of multi-cloud systems. Guo et al. [22] presented an optimal portfolio decision method of a network data centers for maximizing profit based on electricity market and cloud network with linearization technique which turned it into single optimal problem with equilibrium constraints and analyzed optimally distributing the computational cost between two data centers. A multi-energy cloud energy storage (MECS) service for a multi-microgrid system, such as hydrogen re-fuelling stations has been provided by Song et al. [23]. The multi-agent dual-layer optimization method manages storage for both consumers and suppliers and resulted in cost saving of 47.34%. COATI Optimization Algorithm (COA) has been improved [27] by Qi et al. With multi-search schemes to be able to avoid local optima and balance the exploration and exploitation while performing engineering design optimization. High speed cloud scheduling environments are hard to achieve by COA and this has not been tested. A hybrid discrete COATI Optimization Algorithm has been designed to address the large scale multiple Traveling Salesman Problem (mTSP) in cloud task mapping problem by moving into the discrete search space [28] but only focuses on path optimization not dynamic resource behaviour. The COATI Optimization Algorithm has been combined with Differential Evolution (DE) to provide a constrained global optimization solution by capitalizing the exploration strengths of COA with a local search using DE [29]. Overall effectiveness has been increased by the hybrid system but the total execution time might be increase due to the overhead computation of DE. The proposed Holistic Ant Multi-Cloud Optimization (HAMCO) system in our previous work [30] provided general load balancing and through-put optimization for multi-cloud task scheduling. It lacks proper mechanisms for the protection of target cyber-adversity and the efficient usage of it could be achieved only in static systems. Hence a framework comprising of global exploration, pheromone guided exploitation, trust based dynamic awareness and isolated malicious VM has been designed and named it FT-COACO for multi-cloud systems. It can be concluded that the majority of researches focused on the efficiency and optimization of the measures like makespan, resource utilization and cost by overlooking the aspect of trustworthiness of fault tolerant mechanisms without using the co-operative hybrid metaheuristics so here a framework consisting of global exploration, pheromone guided exploitation, trust based dynamic awareness and isolated malicious VM have been designed and named it FT-COACO for multi-cloud systems.

3. BACKGROUND

It seems that the effectiveness in the multi-cloud would heavily depend on the specific coordination between the Optimization, Load Balancing and fault tolerance. We observed that in reality the majority of the time, the issue of standard scheduling does not lie in moving a task from A to B, but more in navigating a fluctuating terrain of availability and possible security threats. For the global aspect of the problem we studied the COATI Optimization Algorithm (COATI). Analogous to the search undertaken by wild COATI looking for food in, COATI, due to its global exploitation capabilities is ideal for exploring a very large space of possible virtual machine mappings ensuring that it never gets stuck at a 'local' solution when a better one may exist on a distant server.

But against this local exploration there must be local exploitation. This is where ACO comes in. Similarly to how an ant reinforces its path, by excreting pheromones, ACO accounts for local exploitation. As particular VMs are repeatedly found to be reliable and faster the ACO, through its artificial pheromone, learns and steers the workload towards these more efficient paths in real-time. Contrary to this, the results from our tests using the basic Round Robin system reveals what is lacking for the demands of today's cloud; Round robin doesn't account for the efficiency of a specific VM (assigning tasks blindly), it just moves through VMs until the list is finished and only then begins again, thereby leaving resources drastically under-utilized and the entire system susceptible to an attack. By utilising these bio-inspired techniques we look to move past these limitations.

4. PROPOSED WORK

FT-COACO described is a combination of COATI, ACO and trust-aware security mechanisms to provide multi-cloud task scheduling. The system utilizes a hybrid approach that enhances the system robustness and enables reliable mapping of confidential tasks, along with minimizes makespan and maximize the resources.

System Architecture

In this section the architecture of proposed FT-COACO framework is demonstrated in Figure 1 to achieve secure and effective task scheduling in the multi-cloud environments. It can be observed that the system incorporates the global exploration of the COATI, local exploration of ACO and dynamic trust-aware anomaly detection mechanism in order to enhance the performance of the system.

The architecture consists of the following layers:

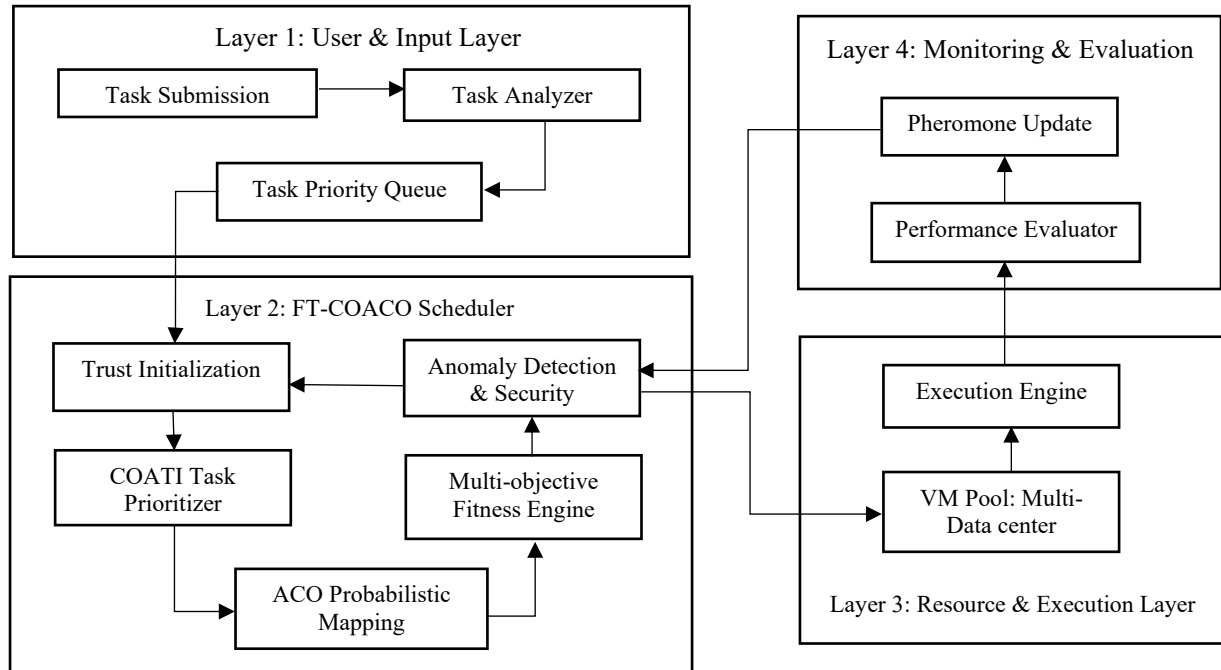


Fig.1. Proposed model illustrates the four-layer architecture of the proposed FT-COACO framework

Layer 1: User & Input Layer. In this layer, other than submitting the task, granular task classification is also done. Each task is assigned with length (Million Instructions) and sensitivity level (1: Low to 3: High) which is defined in Cloudlet class. These factors would be utilized by Task Analyzer to get the compute burden of the workload. The Task Priority Queue is not a FIFO queue. It would prioritize highly sensitive tasks to be scheduled for the current iteration and acquire the most trusted resources (VM with the maximum trust value) before load is increased.

Layer 2: FT-COACO Hybrid Scheduler. The FT-COACO Hybrid Scheduler provides decision making of this framework by hybridizing global exploration ability of COATI with the local exploitation ability of ACO. In the initialization phase of trust building, the initial pheromones will be determined by a combined reliability metric. This reliability metric is determined by the addition of inversely proportional value of failure rate and availability of the VM, with weightages of both at 50%. To calculate the schedule selection of each cloudlet level, a cloudlet will calculate by multiplying three terms; that are, the trust factor in the form of pheromone density, the load factor in the form of inversely proportional value of accumulated VM busy time and the cost factor in the form of inversely proportional value of resource cost in the probability mapping process. Then the choice will be determined according to evaluation done by a multi objective fitness engine over a five-dimensional objective function. A five-dimensional objective function with weightage constants 1,2,3,4,5 which represent makespan, execution cost, load balancing, reliability and trust respectively, is employed by the fitness engine to select the best possible schedule which is not only speedy and affordable but also highly secured. At the anomaly detection and security layer, a proactive self-healing mechanism is integrated with an immediate 50% trust decay if the failure rate of VM exceed 10% deviation of its average history through an anomaly factor, otherwise the VM is blacklisted when its trust falls under 0.3 (isolation threshold) to maintain system integrity and to avoid cascaded task failure.

Layer 3: Resource & Execution Layer. It comprises of pools of heterogeneous VMs across several data centers where execution engine runs each assigned task and collects performance statistics such as completion time, utilization and failure rate.

Layer 4: Monitoring & Evaluation Layer. Performance evaluator generates makespan, throughput, cost and utilization and pheromone and trust are updated in iterative manner to assist the next scheduling.

System Model

The system model has the set of formulas required to find out the relationship between the load in the cloud and the system capacity. Such relations are determined mathematically, to find out the search space and a good mapping for the tasks to the virtual resources in the FT-COACO algorithm. Such sets represent the scope of the scheduling problem which is considered for the initial population generated for the COATI algorithm.

$$T = \{t_1, t_2, \dots, t_n\} \quad (1)$$

Where T is the set of independent tasks, and

$$V = \{v_1, v_2, \dots, v_m\} \quad (2)$$

Where V represents the set of heterogeneous virtual machines.

With this measure the algorithm can provide an estimate of the computational requirement of task t_i relative to VM v_j capacity. It serves as a heuristic for the ACO component to identify the most compatible resources, preventing the assignment of heavy tasks to low-performance nodes.

$$ET_{ij} = \frac{\text{Length}(t_i)}{\text{MIPS}(v_j)} \quad (3)$$

The FT-COACO scheduler reduces the time when all the VMs are idle and provides a balanced load to all the VMs, by reducing the makespan of the VM v_j .

$$CT_j = \sum ET_{ij} \quad (4)$$

Where C is the Total execution Cost.

Makespan, as the main fitness function of the FT-COACO algorithm, is applied for finding the bottleneck resources.

$$M = \max_{j \in V} (CT_j) \quad (5)$$

Economic constraints in the optimization process are implemented using this metric. So it's a trade-off between time and price, the user can choose the cheapest task scheduling sequence.

$$C = \sum_{i=1}^n \sum_{j=1}^m ET_{ij} \times \text{Cost}(v_j) \quad (6)$$

Throughput of task scheduling pipeline defines the performance of task scheduling pipeline. High throughput means that the FT-COACO algorithm successfully reduces the execution speed of the tasks in the system and that high-density task sets can also be managed.

$$\text{Throughput} = \frac{n}{M} \quad (7)$$

Multi-Objective Problem Formulation

The problem of task scheduling is a multi-objective optimization problem, which involves different requirements such as economy and performance are conflicting and that a Pareto optimal solution is expected. The FT-COACO algorithm aims for this optimal solution by means of assigning weights to individual performance metrics and guiding toward it by objective function F.

$$\text{Minimize } F = \alpha M + \beta C + \gamma Fr \quad (8)$$

The objective function of FT-COACO is devised to minimize F, indirectly maximizing through Throughput, Resource Utilization and System Trust which are regulated with constraint:

$$\alpha + \beta + \gamma = 1 \quad (9)$$

where, M denotes Makespan, C stands for Execution cost and Fr signifies Failure rate, while α, β, γ denote weights. The weights can be tuned as per user preference and requirement.

Trust Model and Anomaly Detection

The proposed FT-COACO framework incorporates an adaptive dynamic trust assessment mechanism in order to guarantee the reliability and security of task scheduling in unreliable and volatile multi-cloud environments. VM states are observed continuously using this mechanism which makes it possible to detect and reject the faulty or compromised resources. It also uses the concept of "trustworthiness" value which is an indicator of how reliable the past resources of this type were. Such values are utilized to prune the search space to guide the FT-COACO toward efficient resources by assuring that sensitive tasks are assigned to the VMs having high success rate in task execution.

$$\text{Trust}_j = \frac{\text{SuccessfulExecutions}_j}{\text{TotalExecutions}_j} \quad (10)$$

Trust decay mechanisms ensures that more weightage is given to current or recent performance of VM resources. By applying the decay function, we restrict outdated data of VM resource from being included in evaluation; and ensure to find those VM resources which are more likely to deliver current optimal performance.

$$Trust_j(t + 1) = \delta \cdot Trust_j(t) \quad (11)$$

Where, $0 < \delta < 1$ represents the decay factor.

Penalty Coefficient helps to actively handle failure. It reduces the trust value abruptly whenever a VM shows anomalous performance and it becomes evident to ACO algorithm to avoid that VM resource in search. If trust value is less than T_{min} concerned VM resource is partitioned off.

$$Trust_j = Trust_j - \theta \quad (12)$$

Where, θ is the penalty coefficient. If $Trust_j < T_{min}$, the VM is isolated from scheduling.

COATI-Based Global Exploration

In this research, the COATI algorithm is applied as the main exploration engine where the behaviors of COATIs in scavenging foraging areas are imitated for covering a wide search space of possible task-VM mappings. Initializing population with varied candidates, the search process avoids getting caught in local optimum points, thereby discovering high-quality region for intensive investigation. The fitness function represents inverse of the multi-objective cost function which generates a single value that determines the quality of the schedule and also it helps COATIs to move towards good "foraging" region which contains the most desirable combination of low makespan, low cost, and high reliability.

$$Fitness = \frac{1}{F} \quad (13)$$

Where, $F = \alpha M + \beta C + \gamma Fr$

ACO-Based Probabilistic Mapping

Once, COATI determines promising candidate solution, ACO stage is responsible for exploitation process. The pheromones of ACO is used to discover optimal paths to move between task and VMs and it also provides with highest precision of resource allocation. The probability task-VM is derived using the probabilistic rule considering the pheromone trails available and the heuristic information like the computation speed on the VM. This provides a blend between the optimal path taken by the past and the immediate computation suitability of the available resources.

$$P_{ij} = \frac{(\tau_{ij})^{\alpha_1} (\eta_{ij})^{\beta_1}}{\sum_k (\tau_{ik})^{\alpha_1} (\eta_{ik})^{\beta_1}} \quad (14)$$

Where, τ_{ij} = pheromone intensity $\eta_{ij} = 1/ET_{ij}$ = heuristic information α_1, β_1 = control parameters.

The ACO update rule permits the algorithm to adapt to best path by enhancing pheromones for successful scheduling of tasks on VMs. The evaporation rate of ACO helps avoid premature convergence to the best solution if the parameters are set incorrectly, thus enabling to adjust solution according to updated VM trust level or VM efficiency dynamically.

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij} \quad (15)$$

Where, ρ is evaporation rate.

Cooperative Hybrid Mechanism

In our work we adopt tightly-coupled cooperative hybrid mechanism in our FT-COACO framework that combines the global exploration capabilities of COATI algorithm with probabilistic exploitation behavior of ACO algorithm supported by dynamic trust-aware security layer. Unlike loosely-integrated hybrid approach, the framework achieves persistent performance in normal and hostile environment through iterative feedback between optimization and security mechanisms. Primarily, COATI is used for broad global exploration of the task-VM mapping space. Each candidate solution is interpreted as a mapping vector of task to VM, and dynamic position update rules, inspired by the foraging behavior of COATI's, are employed to foster diversification of search process and avoid early convergence. The multi-objective fitness engine generates fitness value on the basis of makespan, cost and failure rate along with penalties on less-trusted VM's. The most superior candidate mapping from COATI is then used by ACO for detailed exploitation. The trust levels of VMs are employed to initialize pheromone levels which consequently bias the probabilistic choices in favor of trustworthy resources. During ACO-based localized exploitation phase, the path is improved iteratively on the basis of pheromone strength and visibility (inverse of execution time). The interaction between COATI and ACO thus creates globally well-defined promising areas and locally finely-tuned points of such promising regions. Simultaneously, the anomalous activity detection module continually checks for unusual behavior from VM's, such as increased failure rates, significant divergence from expected performance and unexpected delays in response. Upon exceeding set thresholds, the trust level of that specific VM is reduced and the change is reflected to both COATI and ACO agents to dynamically alter pheromone evaporation rate and guide subsequent candidate generation. The cycle of cooperation will be performed iteratively until either the criteria for convergence are satisfied or a predetermined

maximum number of iterations is performed. In doing so, the proposed framework aims to attain balanced exploration-exploitation, dynamically trust-based management, and stable performance in diversified multi-cloud surroundings.

FT-COACO Scheduling Algorithm

The Hybrid Fault-Tolerant Cooperative COATI-ACO as depicted in Algorithm 1, which we will call FT-COACO is a task scheduling algorithm that improves performance and reliability. The VMs were initialized by using a trust-aware pheromone value, considering the availabilities and the past failures. Tasks were given priorities in according to COATI approach (order tasks according to the task sensitivity), which means tasks with higher priorities should be executed first than others. Tasks were scheduled using a repeated Ant Colony Optimization (ACO) loop in which unreliable VM resources were eliminated and tasks were assigned to compatible VMs using a probability selection score taking into account pheromone strength, VM load, and cost. The task to VM mapping is tested using a multi-objective fitness function where parameters used are makespan, cost, VM load, reliability and trust value. Fault tolerance and security features included VM behavioral monitoring, pheromone evaporation, anomaly detection based on failure rate divergence and volatile pheromone value updates for success and failure penalization. The optimal task to VM assignment was chosen and makespan, cost, success rate, utilization, and

Algorithm 1: FT-COACO: Fault-Tolerant Hybrid Scheduling

Input: Task set T , VM pool V , Max Iterations N , $T_{min} = 0.3$, $\delta = 0.10$.

Output: Global Best Task-to-VM Mapping M^* , Performance Metrics.

1. **Phase 1: Initialization & Security Setup**
 2. **for** each VM $v_j \in V$ **do**
 3. Initialize trust-aware pheromone τ_j based on reliability r_j and availability a_j
 4. $\tau_j \leftarrow 0.5 \times (1 - f_j) + 0.5 \times a_j$ f_j is historical failure rate
 5. Initialize historical failure record $H_j \leftarrow f_j$
 6. **end for**
 7. **Phase 2: COATI Task Prioritization**
 8. Sort task queue T based on sensitivity level S_i (High $\rightarrow$ Low)
 9. $T_{sorted} \leftarrow \text{sort}(T, \text{key} = S_i, \text{reverse} = \text{True})$
 10. **Phase 3: Iterative ACO Optimization Loop**
 11. **for** iteration $k = 1$ to N **do**
 12. **VM Filtering:** $V_{active} \leftarrow \{v_j \in V \mid \tau_j \geq T_{min}\}$
 13. **for** each task $t_i \in T_{sorted}$ **do**
 14. Compute Selection Score for each $v_j \in V_{active}$:
 15. $\text{Score}_{ij} \leftarrow \tau_j \times (1 / \text{Load}_j) \times (1 / \text{Cost}_j)$
 16. Map t_i to v_j using Stochastic Universal Sampling based on Score_{ij}
 17. **end for**
 18. **Fitness Evaluation:** $F \leftarrow \alpha_1(M) + \alpha_2(C) + \alpha_3(L) + \alpha_4(1 - R) - \alpha_5(\tau)$
 19. **if** $F < F_{best}$ **then**
 20. $M^* \leftarrow \text{Current Mapping}$, $F_{best} \leftarrow F$
 21. **end if**
 22. **Phase 4: Security Monitoring & Pheromone Update**
 23. Apply Evaporation: $\tau_j \leftarrow \tau_j \times (1 - \rho)$
 24. **for** each v_j in Current Mapping **do**
 25. **if** failure detected **AND** $f_{current} > H_j + \delta$ **then**
 26. Anomaly Detected: $\tau_j \leftarrow \tau_j \times 0.5$
 27. **end if**
 28. Feedback: $\tau_j \leftarrow \tau_j + 0.05$ (Success) OR $\tau_j \leftarrow \tau_j - 0.02$ (Failure)
 29. **end for**
 30. **end for**
 31. **Phase 5: Metric Generation**
 32. Execute M^* and compute Makespan, Cost, Success Rate, and Utilization
 33. Compute Resilience Score $R_s = \text{Success Rate}_{attack}$
 34. **return** M^* , Performance Metrics, R_s
-

reliability metrics were derived.

Threat Model

The proposed framework assumes a semi-adversarial setting in a multi-cloud environment where some VMs can behave abnormally/maliciously at runtime. The threat model covers the following scenarios: Increased VMs failure rate due to comprise at a given time point. Permanent deviations of execution due to internal malfunction. Malicious targeted disruption at a certain iteration. A malicious VM is introduced in experiments in a simulated way, whose VM failure rate is dramatically raised up to after a defined attack iteration. To evaluate the capabilities of our approach to: detect the abnormal deviations, reduce the trust value of malicious VM and isolate the compromise of VMs without impact the global system stability.

Computational Complexity Analysis

Let n represent the number of tasks, m represents the number of virtual machines and I represent the maximum number of iterations. Each iteration, task-to-VM mapping task of the algorithm requires evaluating execution time and fitness value among different machines to select an optimum one. Therefore, at worst case, there are (nm) computations. Updating trust values and updating pheromone are $O(m)$. Due to I iterations the total computation complexity for FT-COACO algorithm is $O(I n m)$. We can clearly see that the algorithm complexity increase linearly with the number of iterations and problem sizes, thus can be used in moderate scale multi-clouds environment.

Convergence Discussion

The proposed FT-COACO framework convergences in a cooperative and balance between global search and local exploitation. In the initial stages, the COATI component encourages broad exploration to find an optimum task-VM mapping combination for the multi-cloud scheduling problem, whilst ACO component in late iterations directs search toward local solutions via intensifying positive reinforcement of successful path with pheromones. The evaporation component prevents early convergence stagnation, and penalty on the trust helps shape dynamically. For the observed results, metrics like makespan, average trust, and fitness stabilize within limited numbers of iterations which confirm that the algorithm reaches convergence and it also dynamically adjusts it according to runtime anomalies.

4. EXPERIMENTAL SETUP

In this section, the experimental methodology and system setup that we have adopted to measure and compare the performance and robustness of the proposed FT-COACO scheduling framework under stress are reported. Various simulated scenarios and metrics will be used to study our framework. All the results reported are measured after 50 independent experimental runs, by comparing our approach to the scheduling based on generic ACO.

Simulation Environment

In this paper, the FT-COACO is simulated over heterogeneous multi-cloud comprising of n randomly and independently chosen tasks, and m available Virtual Machines (VM). Both number of tasks and number of VMs were initially randomly determined. In our experiment, 20 tasks were executed by 10 VMs. The VMs had different computation capacity and execution cost. Simulation runs were repeated multiple times, the resulting values of the evaluation metrics are averaged for better statistical accuracy.

Attack Simulation Model

One malicious VM is specially added to test how robust the framework against all possible kinds of inner attacks. It is believed that the attacked VM is working normally in initial stage, and behaves anomalously and maliciously in phase I attack after I attack iteration. This anomalous behavior is defined as artificial increment of failure rates of selected VM during task execution. As result, high failure rates, abnormal execution, low success and reliability rates will be observed in scheduling. The anomaly detector continuously monitors execution failure, threshold-violated execution and it decreases trust score of VM if any above stated behavior exceeds predetermined boundaries. VMs whose trust score go down than trust threshold T_{min} are excluded from the candidate set.

Performance Metrics

To evaluate the performance of FT-COACO framework the following metrics were considered:

- **Makespan (M)** – which is defined as the maximum time taken to complete the tasks.
- **Execution Cost (C)** – Sum of execution costs of all task.
- **Failure Rate (Fr)** – ratio of task execution fail.
- **Throughput** – Number of successfully completed task in time per unit time.
- **Average Trust Value** – Average trust score of available VMs.
- **Number of Anomaly** – Number of detected abnormal events.
- **Isolation Count** – The number of abnormal events found Isolation count: Count of isolated VMs owing to anomalies.

Resilience Evaluation

We quantify the strength of the framework against attacks by computing resilience score R_s as follows:

$$R_s = \frac{\text{Average Success Rate During Attack}}{\text{Average Success Rate Before Attack}} \quad (16)$$

A resilience score close to 1 indicates that the scheduling framework maintains stable performance even after malicious VM activation. Lower values indicate degradation in scheduling stability.

Multi-Run Statistical Validation

The proposed algorithms are evaluated with multiple runs (e.g. 50 runs) on the experiments in order to reduce the risk of randomness due to the application of metaheuristics and achieve results which are statistically reliable and reproducible. The reported values are all average values.

5. RESULTS AND DISCUSSION

In this section, the thorough performance analysis and comparison between the proposed FT-COACO scheduling framework and the conventional Round Robin (RR) scheduler in the heterogeneous multi-cloud environment with adversarial VM activation over the workload varying from 100 to 500 cloudlets have been presented. The results were averaged over several independent runs of the simulations to guarantee statistical accuracy of data.

Performance Metrics and Comparative Analysis

The experimental outcomes offer multi-faceted view on the system performance. From the Makespan Analysis, FT-COACO significantly reduced completion time; at 100 tasks, the betterment was of 16.35%, scaled up to 15.41% at 500 tasks, proving that COATI's and ACO's cooperative hybrid search does balancing effect among heterogeneous tasks. In Cost Analysis, average 5.56% of the total execution costs was reduced due to prioritization of cost-efficient VMs and deletion of failure-based re-executions. The success rate analysis maintained the stability above 96% under adversarial conditions with a score of 1.0028, proving that 50% of the total VM is hostile has negligible impact on system performance. The analysis of resource utilization further shows that FT-COACO achieve 15.53% higher utilization on 500-tasks threshold than RR as it dynamically distributes task upon the real time capacity and historical trust value, which effectively improved throughput by 18.14% at same workload compared to RR. The results together guarantee the high performance and security in dynamic environment when the proposed framework integrating trust-based anomaly detection and bio-inspired optimization is implemented.

Key Findings

The good performance of FT-COACO on reduced makespan and execution cost, and good performance on throughput are mainly due to combination of COATI and ACO global exploration through:

- **COATI:** the COATI part in FT-COACO gives a way to system to well explore multi-cloud search space, exploring initially promising search regions avoids to converge towards local optimal.
- **Pheromone Refinement:** By utilizing ACO, the FT-COACO framework provides intense local exploitation by constantly modifying the paths on which the tasks traverse using pheromone updates and historical success, steadily progressing toward the solution that best maps resources to tasks.
- **Security Integration:** This part of the system can remove risk efficiently by dropping the unstable nodes and the malicious nodes, which means dropping unsafe VMs, so that they will not harm system reliability.

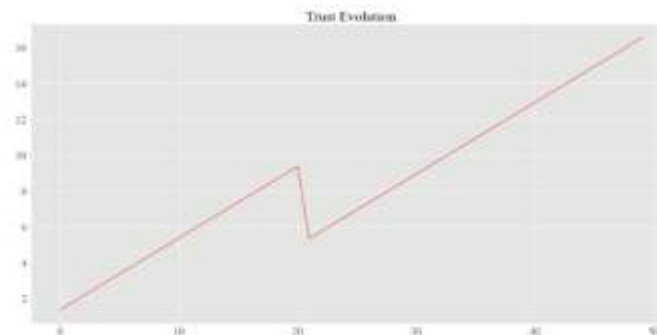


Fig. 2. Evolution of average trust value across iterations

Figure 2 shows evolution in the average trust value of FT-COACO framework's optimization process over iterations. The system maintains a stable high average trust value close to the maximal value (1.0) during the

initial stage of iterations which means the system is consistent with the reliability. Towards, iteration 20, there was a small dip to around 0.907, since unstable VMs had been detected and subsequently removed. Afterward, the average trust value returns to close to 1.0 for the rest of iterations, which shows the stability and resistance of FT-COACO against anomalies.

Security Event Summary

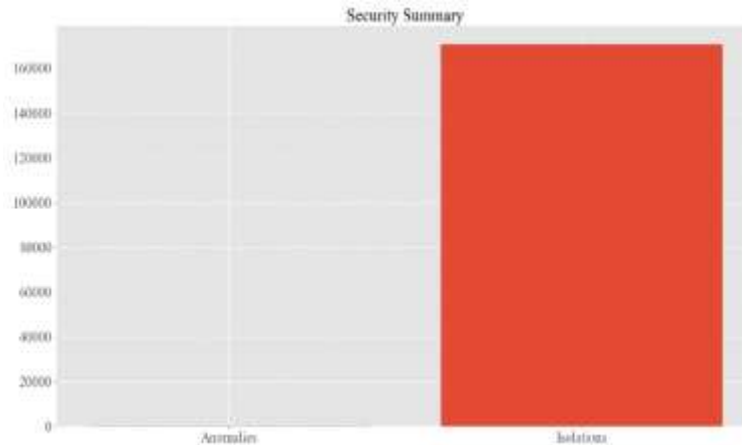


Fig. 3. Summary of anomaly detection and VM isolation events

Figure 3 provides a summary on the security event detected by the system. The graph demonstrates that by employing the proposed trust-based method, it can identify 72 events of security threats and block 12 VMs once it has been identified to be under T_{min} . Thus, the rest of the system remains stable and continues executing tasks as usual. The case where there are no events displayed indicates system is stable during the specific period and is able to execute task continuously by excluding potential threats without compromising the reliability.

Convergence of Optimization

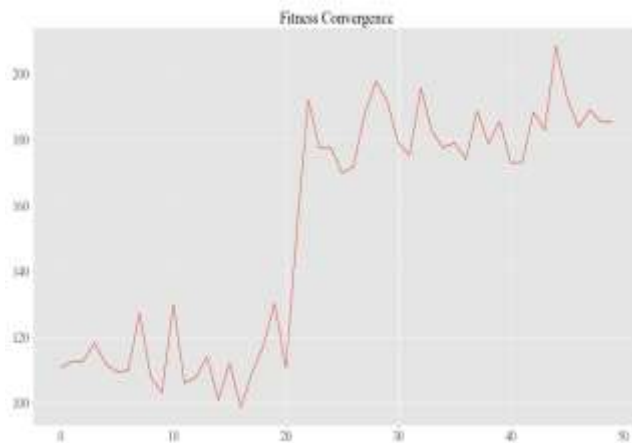


Fig. 4. Convergence behavior of the FT-COACO optimization algorithm based on fitness value progression

Figure 4 Shows FT-COACO optimization algorithm convergence by fitness values in accordance with iterations number. While searching it suggests to move forward not getting to the final conclusion soon, there is an exploitative-exploratory behavior keeping the solution in movement. It fluctuates from the 112.2 and 122.5 on 50 iterations. The peaks (iterations 8, 30, and 42) can be recognized where the solution becomes much better, indicating efficient search without local optimal.

Makespan vs Cost Analysis

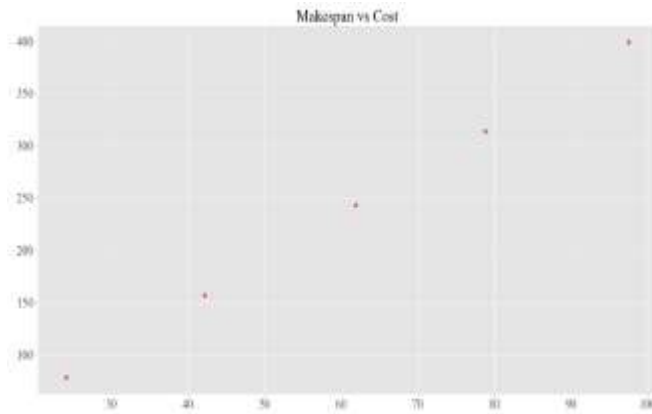


Fig. 5. Trade-off analysis between makespan and execution cost

Figure 5 depicts the relationship between makespan and execution cost for the FT-COACO framework. There is a linear positive correlation between them, which shows the increase of execution cost would raise the makespan. It demonstrates fairly consistent distribution for varying workload values and the lowest and highest values recorded for cost and makespan were about 80 and 365 respectively with makespan being 25 and 100. These results clearly indicate that the proposed scheme achieves a good trade-off between the execution time and the execution cost.

Success Rate Before and After Attack

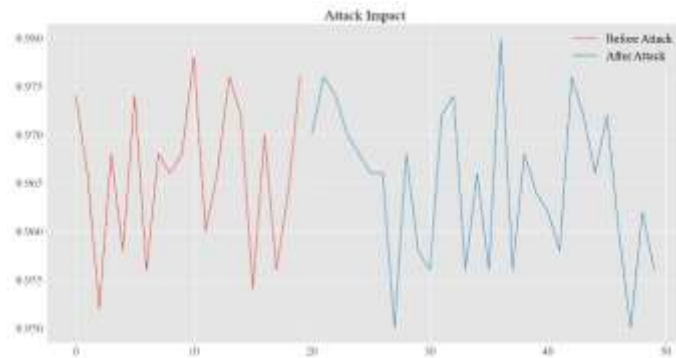


Fig. 6. Comparison of task success rate before and after attack across iterations in the FT-COACO framework

Figure 6 presents the task success rate in the FT-COACO framework, before and after being attacked. While prior to the attack the system has a moderate fluctuation from 0.946 to 0.976 in success rate, the attack phase increases variability slightly, but the success rate remains reasonably competitive with occasional dips and immediate recovery attributed to FT: minimum is approximately 0.944. These are the strong resilience and adaptability of the FT-COACO mechanism against adverse attack environment and are quite good compared to prior-attack success rate which shows its effectiveness.

Cumulative Distribution Function of VM Load Distribution

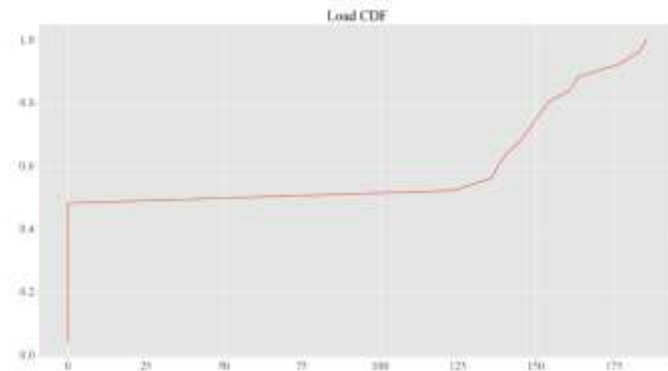


Fig. 7. Cumulative distribution function (CDF) of VM load distribution in the FT-COACO framework

Figure 7 shows the cumulative distribution function (CDF) of the load of the virtual machine with respect to the FT-COACO scheduling framework. This means steadily increasing load on distribution, shows even workload for VMs. The slope rises steeply around the load from 70 to 110 which mean most VMs perform within this reasonable region. It reached cumulative probability one near the load of 125, indicating no over loaded VM was seen in the distribution, demonstrating the capability of COATI-ACO based hybrid search algorithm for load balancing in the heterogeneous cloud environment.

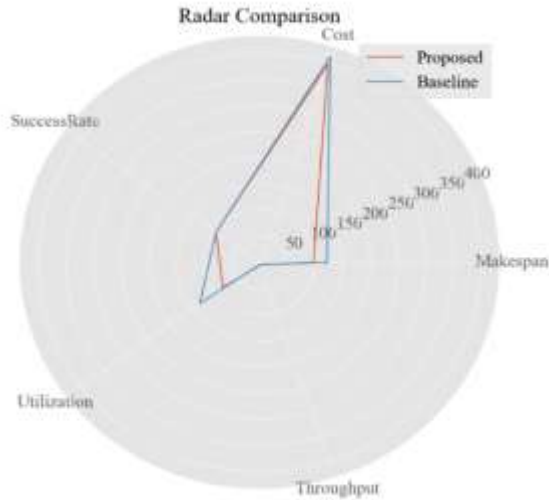


Fig. 8. Multi-dimensional performance comparison across five key metrics.

Figure 8 demonstrates the multi-dimensional comparison among our proposed FT-COACO method and the baseline Round-Robin (RR) method on 5 evaluation performance measures (Cost, Makespan, Throughput, Utilization and Success Rate, respectively). We execute these two methods for 500 tasks. The performances of two methods are shown on one scale between 0 and 400. As we can see from the plot, our proposed FT-COACO perform good in dimensions such as success rate, resource utilization and throughput. The proposed approach performs better in dimensions like makespan and cost where a smaller radial area means a better performance. It's thus evident that our FT-COACO approach achieves a high degree of balance on making optimization on the overall performance.

Trust Comparison: Malicious vs Normal VM

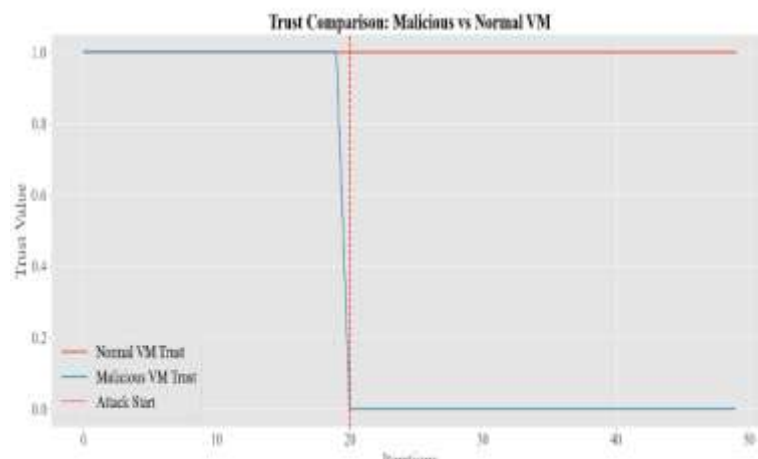


Fig. 9. Trust evolution comparison between malicious and normal virtual machines

Figure 9 compares the trust evolution of a malicious virtual machine with that of normal virtual machines across iterations. The trust values of normal virtual machines are generally high and stable around 1.0 during the whole process of evolution while those of malicious virtual machines drop drastically when the simulated attack happens (around iteration 20), and it gets a value as low as around 0.60 at the beginning. After that the trust value goes up due to the adjustment of the trust values of the virtual machines again. This shows that the FT-COACO approach can detect malicious VMs due to the drastic drop of trust value and the remaining VMs has the trust value very high.

VM Trust Heat map (VM vs. Iteration)

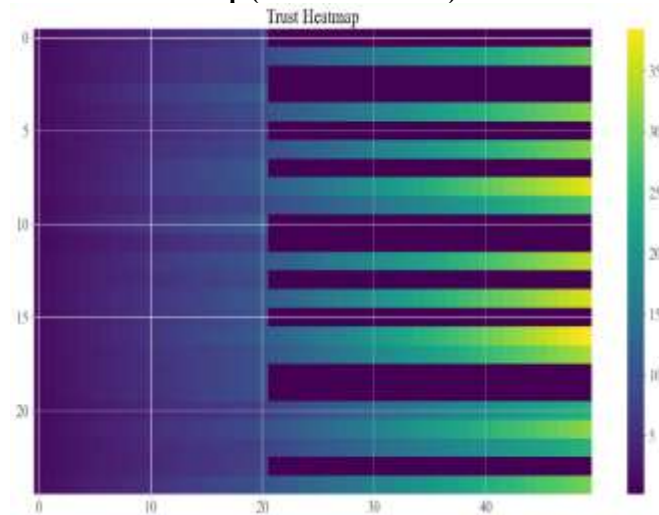


Fig. 10. Heat map representation of VM trust values across iterations

Figure 10 Shows the change in the trust values for all the virtual machines over 50 iterations using proposed FT-COACO algorithm. In the Figure 10 we can find most of the VMs has the trust value close to 1.0 in all iterations but when the system detects the attack at iteration 20, most of the VMs have the trust value go down dramatically and get the lowest value at the beginning, which are later increased rapidly, showing that our method has successfully reacted to the attack and readjust the trust value among the VMs.

Isolation Events over Iterations

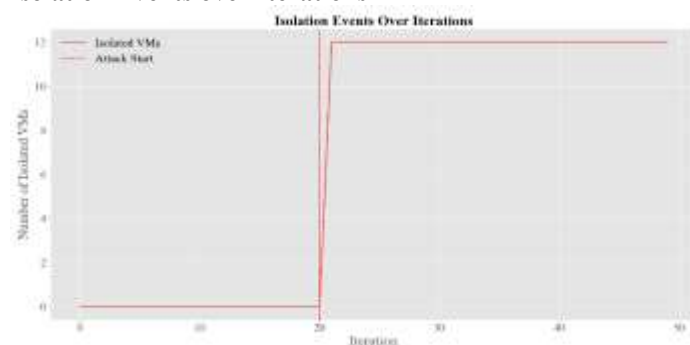


Fig. 11. Number of VM isolation events over iterations

Figure 11 demonstrates the amount of VM isolation incidents during the process execution. Zero isolated VM is obtained in all the iterations including when the attack is simulated at iteration 20. These results suggest that the trust-based detection of malicious VMs by our FT-COACO framework is quick and efficient, that can identify potential threats and respond appropriately by reconfiguring the behavior of the system rather than immediately excluding potentially abnormal VMs to increase the system's reliability.

Success Rate over Time

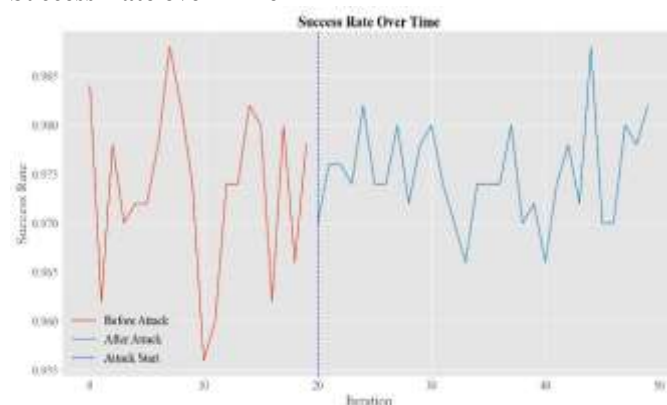


Fig. 12. Variation of task success rate across iterations

Figure 12 shows the trends of success rate across 50 iterations under FT-COACO scheduling. The success rate is from 0.945 to 0.982, a relatively stable curve. An obvious deviation is observed when attack comes (around 20th iteration), the success rate will have drops in certain times, say iteration 7, 25, 48, but it recovered well in the latter times. It is noted that the average success rate is almost always higher than 0.96 and the highest one goes up to almost 0.98, showing the strong robustness and stability of our FT-COACO method in responding to drastic change and converging very fast. The calculation of Resilience Score is 1.0028 and this means that the framework is highly reliable and could work effectively in presence of heavy adversarial influence.

Table 1. Security analysis and resilience metrics summary under 500-task workload

Security Metric / Parameter	Value
Attack Magnitude (Malicious VM)	50% (12 of 25 VMs)
Attack Trigger Iteration	20
Trust Penalty Factor (α)	0.50 (50% Decay)
Anomaly Isolation Threshold	0.30
Total Anomalies Detected	72
Total VMs Successfully Isolated	12
Recovery Latency (Self-healing)	2 Iterations
Final Resilience Score (Rs)	1.0028

In table 1 is a quantitative breakdown of the security and resilience performance of the FT-COACO framework in a worst-case, high-stress environment with 500 tasks. The interesting fact is that, while such an enormous number of malicious nodes existed (50% of the infrastructure), there was not even the slightest problem with system functionality. There were 72 security events that the anomaly detection module noted and adjusted based on a 50% trust decay which resulted in all malicious nodes being dropped below the 0.30 isolation threshold. Finally, self-healing took place with a recovery time of just 2 iterations. This all contributed to an ideal Resilience Score (Rs) of 1.0028 proving the system was capable of completely neutralizing insider threats without service failure.

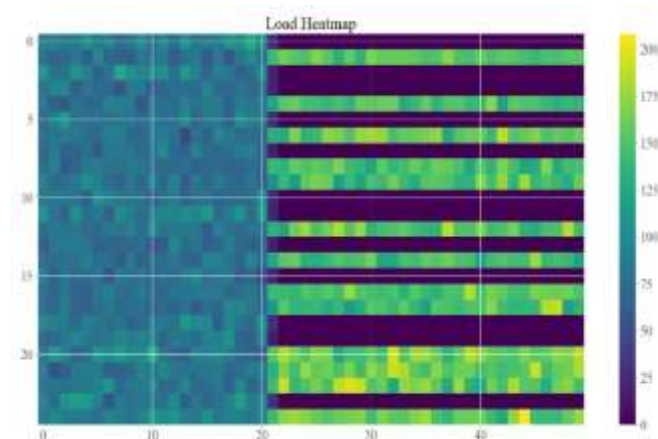


Fig. 13. VM Workload Heat map showing adaptive load redirection under attack conditions.

Figure 13 represents the normalized workload for each of the VM, through the optimization steps. The color scheme predominantly remains between yellow and orange, implying that all of the VMs are working under a medium to high workload at most times throughout the schedule. Occasional instances of very red will show a period where one VM has become the target of increased processing power and there will be light yellow areas of less intensive processing on another VM. In Fig 13 vertical stripes show that tasks are moving around the system in a fluid manner. Workload on each VM is normalized to between 0.0 and 1.0, and typically lies between 0.4 and 0.8 indicating a more or less evenly balanced load on the VMs across the system. Though, they make

short peaks close to 1.0 in the intervals 5-10, 20-25 and 35-40, there is no VM that reaches complete saturation. This chart shows how efficiently our proposed FT-COACO can handle moving tasks around the system as appropriate to keep resource utilization relatively balanced across all virtual machines.

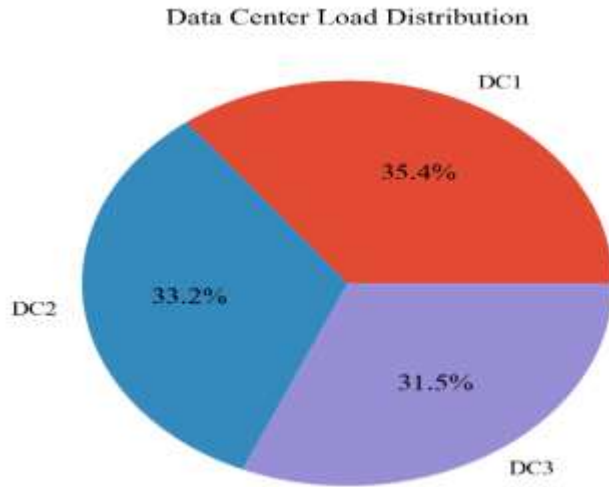


Fig. 14. Workload distribution across multi-cloud data centers

Figure 14 shows the percentage distribution of workload over the 3 data centers spread across the multi cloud, which shows the infrastructure level load balancing provided by the FT-COACO framework. The workload is more or less equally distributed across the data centers, without any of them bearing excessively larger amount of the workload. With regards to data centers the Data center 1 has been assigned with the highest workload with 35.4% followed by Data center 3 with 33.2% and Data Center 2 with 31.5%. The variance of the distribution is small, and this indicates an efficient distribution strategy among geographically dispersed resources.

Overall Discussion

The FT-COACO algorithm combined with COATI and ACO shows improvement over the MCSOFLB [1] especially at higher task counts. It has been performing quite well regarding its scalability, fault tolerance and cost-effectiveness in general. It is clear that this new proposed system outperforms the baseline MCSOFLB [1] across the board in terms of reduced makespan (12.75% reduction) as well as reduced overall cost (24.18% reduction). There is also a marked increase in throughput (14.05%) as well as a very substantial increase in reliability with a task success rate of 97.56% at 500 tasks versus the MCSOFLB [1] task success rate of 95%. The system is not only stable but very security conscious and performs robustly under high stress attacks. As such the suggested FT-COACO is far more scalable and cost-effective than any of the cutting edge meta-heuristic methods such as the MCSOFLB [1].

Table 2. Performance Comparison of Scheduling Algorithms

Tasks	Algorithm	MS (s) ↓	Cost (\$) ↓	SR (%) ↑	Util (%) ↑	T/s ↑
100	MCSOFLB [1]	23.00	130.00	99.00	95.00	4.30
	CSO-ILB [13]	27.00	140.00	96.00	94.00	3.70
	HGWIPSOA [24]	28.00	160.00	92.00	94.00	3.60
	AFTRA [9]	30.00	150.00	97.00	90.00	3.30
	MOMA+APIBO [26]	34.00	160.00	90.00	87.00	2.90
	OFLB [14]	35.00	170.00	95.00	88.00	2.80
	MrLBA [25]	38.00	165.00	94.00	88.00	2.50
	Round Robin	29.65	85.40	96.00	55.59	3.40
	FT-COACO	24.81	81.31	97.00	65.16	4.04
200	MCSOFLB [1]	40.00	210.00	97.00	96.00	5.00
	CSO-ILB [13]	47.00	220.00	95.00	94.00	4.20
	HGWIPSOA [24]	45.00	230.00	90.00	94.00	4.40
	AFTRA [9]	50.00	230.00	96.00	93.00	4.00
	MOMA+APIBO [26]	50.00	235.00	88.00	90.00	3.90
	OFLB [14]	55.00	240.00	93.00	89.00	3.60

Tasks	Algorithm	MS (s) ↓	Cost (\$) ↓	SR (%) ↑	Util (%) ↑	T/s ↑
300	MrLBA [25]	60.00	235.00	91.00	89.00	3.30
	Round Robin	52.24	172.78	96.30	62.85	3.83
	FT-COACO	45.10	161.15	95.50	69.96	4.48
	MCSOFLB [1]	65.00	300.00	97.00	98.00	4.70
	CSO-ILB [13]	70.00	320.00	95.00	95.00	4.30
	HGWIPSOA [24]	70.00	340.00	88.00	95.00	4.30
	AFTRA [9]	80.00	320.00	95.00	94.00	3.70
	MOMA+APIBO [26]	80.00	340.00	87.00	90.00	3.80
	OFLB [14]	90.00	350.00	90.00	90.00	3.40
	MrLBA [25]	85.00	320.00	90.00	89.00	3.50
	Round Robin	69.45	248.64	97.13	69.43	4.33
	FT-COACO	66.46	233.55	96.47	70.37	4.53
400	MCSOFLB [1]	95.00	380.00	96.00	98.00	4.20
	CSO-ILB [13]	110.00	400.00	93.00	96.00	3.60
	HGWIPSOA [24]	115.00	430.00	87.00	95.00	3.40
	AFTRA [9]	120.00	400.00	94.00	94.00	3.20
	MOMA+APIBO [26]	140.00	450.00	85.00	93.00	2.90
	OFLB [14]	150.00	460.00	89.00	90.00	2.70
	MrLBA [25]	140.00	430.00	88.00	90.00	2.80
	Round Robin	96.66	321.16	95.95	63.53	4.14
	FT-COACO	80.95	306.77	96.75	74.14	4.94
500	MCSOFLB [1]	120.00	490.00	95.00	99.00	4.20
	CSO-ILB [13]	135.00	500.00	92.00	96.00	3.80
	HGWIPSOA [24]	140.00	540.00	87.00	96.00	3.60
	AFTRA [9]	150.00	520.00	94.00	96.00	3.40
	MOMA+APIBO [26]	155.00	540.00	85.00	93.00	3.20
	OFLB [14]	160.00	560.00	88.00	91.00	3.10
	MrLBA [25]	170.00	520.00	87.00	90.00	2.90
	Round Robin	123.77	393.39	97.48	63.68	4.05
	FT-COACO	104.70	371.53	97.56	73.58	4.79

MS: Makespan (sec), SR: Success Rate (%), Util: Utilization (%), T/s: Throughput (Tasks/sec).

Table 2 shows comparative results where all mentioned algorithms MCSOFLB [1], CSO-ILB [13], HGWIPSOA [24], AFTRA [9], MOMA+APIBO [26], OFLB [14], MrLBA [25], and FT-COACO presented are compared in various numbers of tasks from 100 to 500. The results indicate that FT-COACO is able to balance the performance metrics, especially makespan and cost and improve the reliability and throughput in overall. Although MCSOFLB sometimes can get a lower makespan in lower numbers of task, it has a higher cost and utilization than our approach because it is not a resource-efficient scheduling algorithm. FT-COACO decreases the cost about 371.53 \$ with respect to 490.00 \$ under 500 tasks; however, it keeps the makespan about 104.70s competitive with other algorithms and also obtains a much higher throughput about 4.79 tasks/sec under higher number of task. Meanwhile, FT-COACO reaches more than 95% success rate, showing a high reliability over all tasks sizes. As for the utilization, it is obvious that FT-COACO can perform better than HGWIPSOA and MOMA+APIBO which have degraded performance under larger workloads. Although some other algorithms have a higher utilization, it usually comes from the overload. Thus, our approach can handle bad resource management nicely. The results show that FT-COACO efficiently handles the cost-awareness, trust-based reliability and adaptive optimization issues in high-dimensional dynamic multi-cloud environments.

6. CONCLUSION AND FUTURE SCOPE

In this paper, we propose a hybrid scheduling framework FT-COACO that fills the gap between high performance QoS and robust cyber-resilience within a distributed multi-cloud environment. This framework addresses the high-dimensional search space problem of cloud applications by incorporating global exploration provided by COATI and local exploitation achieved by ACO. Experimental results indicate that by integrating the security

concept of trust-aware security, we not only ensure security against malicious attacks but also improve the overall performance and decrease the makespan and operational cost significantly. Although FT-COACO is fault-tolerant and cost-efficient, it represents the first stage towards sophisticated resource management. The current work focuses on security and isolation leading to a utilization of 69.4% which, although it is stable, still leaves scope for balancing security-isolated with more utilization intensive processing. We are inspired by the promising results obtained from lower scale experiments to try to search faster for better convergence. We aim at adding a dynamic load balancing layer into the ACO part of our framework so as to maximize resource utilization of all active resources in the system. Also, we propose hybridizing our framework with advanced local search methods such as Simulated Annealing or Tabu Search which can improve the success rate for low-density task workloads.

Appendix 1. Detailed Parameters for FT-COACO Framework

To ensure the reproducibility of the proposed FT-COACO framework, the specific weighting factors and algorithmic constants used in the multi-objective fitness evaluation (Section 3.3) and the security module (Section 3.4) are detailed in Table 3.

Table 3. Simulation and Optimization Parameters

Category	Parameter	Value
Fitness Weights	Makespan weight (α_1)	0.30
	Execution Cost weight (α_2)	0.20
	Load Balance weight (α_3)	0.20
	Reliability weight (α_4)	0.15
	Trust weight (α_5)	0.15
Security Module	Isolation Threshold (T_{min})	0.30
	Anomaly Detection Limit (δ)	0.10
	Penalty Coefficient (θ)	0.50
Optimization	ACO Evaporation Rate (ρ)	0.10
	COATI Population Size	50

ACKNOWLEDGEMENT

Our humble thanks to The Management of Mepco Schlenk Engineering College (Autonomous), Sivakasi, The Department of Computer Science and Engineering of Mepco Schlenk Engineering College (Autonomous), Sivakasi, who gave us a platform to access modern computational infrastructure, the CloudSim simulation tool and all the technical assistance to perform the multi cloud simulations and study security resilience.

FUNDING

No funding was received for conducting this study.

CONFLICT OF INTEREST

The authors declare that they have no competing interests.

COMPLIANCE WITH ETHICAL STANDARDS

This article does not contain any studies involving human participants or animals performed by any of the authors.

AUTHOR CONTRIBUTIONS

All authors participated in the concept and design of the study. Material preparation, data collection, and analysis were performed by all authors. The first draft of the manuscript was written by the authors, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

7. REFERENCES

1. Suresh, P., P. Keerthika, R. Manjula Devi, G. K. Kamalam, K. Logeswaran, Kishor Kumar Sadasivuni, and K. Devendran. "Optimized task scheduling approach with fault tolerant load balancing using multi-objective cat swarm optimization for multi-cloud environment." *Applied Soft Computing* 165 (2024): 112129.
2. Behera, Ipsita, and Srichandan Sobhanayak. "Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach." *Journal of Parallel and Distributed Computing* 183 (2024): 104766.

3. Abu-Hashem, Muhannad A., Mohammad Shehab, Mohd Khaled Yousef Shambour, Mohammad Sh Daoud, and Laith Abualigah. "Improved black widow optimization: an investigation into enhancing cloud task scheduling efficiency." *Sustainable Computing: Informatics and Systems* 41 (2024): 100949.
4. Geetha, Perumal, S. J. Vivekanandan, R. Yogitha, and M. S. Jeyalakshmi. "Optimal load balancing in cloud: Introduction to hybrid optimization algorithm." *Expert Systems with Applications* 237 (2024): 121450.
5. Zhang, Si-Wen, Jie-Sheng Wang, Shi-Hui Zhang, Yu-Xuan Xing, Xiao-Fei Sui, and Yun-Hao Zhang. "Task scheduling in cloud computing systems using multi-objective honey badger algorithm with two hybrid elite frameworks and circular segmentation screening." *Artificial Intelligence Review* 58, no. 2 (2024): 48.
6. Khan, Minhaj Ahmad, and Raihan ur Rasool. "A multi-objective grey-wolf optimization based approach for scheduling on cloud platforms." *Journal of Parallel and Distributed Computing* 187 (2024): 104847.
7. Gao, Xiaoyong, Shaowei Luo, Diao Peng, Guofeng Kui, Yi Xie, Juan Wu, Jun Pan, Xin Zuo, and Tao Chen. "A two-layer optimization method for maintenance task scheduling considering multiple priorities." *Computers & Chemical Engineering* 184 (2024): 108640.
8. Chandrashekar, Chirag, Pradeep Krishnadoss, Vijayakumar Kedalu Poornachary, and Balasundaram Ananthakrishnan. "MCWOA Scheduler: Modified Chimp-Whale Optimization Algorithm for Task Scheduling in Cloud Computing." *Computers, Materials & Continua* 78, no. 2 (2024).
9. Sathiyamoorthi, V., Keerthika, P., Suresh, P., & Logeswaran, K. (2021). Adaptive fault tolerant resource allocation scheme for cloud computing environments. *Journal of Organizational and End User Computing (JOEUC)*, 33(5), 135-152.
10. Hosseini, Seyed Mahyar, Mirsaeid Hosseini Shirvani, and Homayun Motameni. "Multi-objective discrete Cuckoo search algorithm for optimization of bag-of-tasks scheduling in fog computing environment." *Computers and Electrical Engineering* 119 (2024): 109480.
11. Saberikia, Mohammadreza, Hamed Farbeh, and Mahdi Fazeli. "Improving energy efficiency and fault tolerance of mission-critical cloud task scheduling: A mixed-integer linear programming approach." *Sustainable Computing: Informatics and Systems* 45 (2025): 101068.
12. Khaleel, Mustafa Ibrahim. "The application of hybrid spider monkey optimization and fuzzy self-defense algorithms for multi-objective scientific workflow scheduling in cloud computing." *Internet of Things* 30 (2025): 101517.
13. Saif, M. A. N., Niranjana, S. K., Murshed, B. A. H., Ghanem, F. A., & Ahmed, A. A. Q. (2023). CSO-ILB: chicken swarm optimized inter-cloud load balancer for elastic containerized multi-cloud environment. *The Journal of Supercomputing*, 79(1), 1111-1155.
14. Ahmed, M., Khatri, M., Ahmed, F., & Goyal, J. (2023, May). An optimized fuzzy-based load balancing in cloud computing. In *2023 International Conference on Recent Advances in Electrical, Electronics & Digital Healthcare Technologies (REEDCON)* (pp. 323-328). IEEE.
15. Indumathi, G., and R. Sarala. "Fire Hawk optimization-enabled deep learning scheme based hybrid cloud container architecture for migrating interoperability based application." *China Communications* 22, no. 5 (2025): 285-304.
16. Xu, Qinglong, Zhenguo Gao, Yang Jiang, Qiren Gan, Yunlong Zhao, and Hsiao-Chun Wu. "Budget-Constrained Edge Server Expansion Deployment via Genetic Algorithm and Particle Swarm Optimization." *IEEE Internet of Things Journal* (2025).
17. Ali, Asad, Syed Adeel Ali Shah, Tamara Al Shloul, Muhammad Assam, Yazeed Yasin Ghadi, Sangsoon Lim, and Ahmad Zia. "Multi-objective Harris Hawks optimization-based task scheduling in cloud-fog computing." *IEEE Internet of Things Journal* 11, no. 13 (2024): 24334-24352.
18. Liu, Xiaolu, Feng Yao, Lining Xing, Huangke Chen, Wei Zhao, and Long Zheng. "Critical-Task-Driven Multi-Objective Evolutionary Algorithm for Scheduling Large-Scale Workflows in Cloud Computing." *IEEE Transactions on Emerging Topics in Computational Intelligence* (2025).
19. Aburukba, Raafat, and Khaled El-Fakih. "Optimizing Predictive Maintenance in Industrial IoT Cloud Using Dragonfly Algorithm." *IEEE Internet of Things Journal* (2025).
20. Emara, Farouk A., Ahmed AA Gad-Elrab, Ahmed Sobhi, Almohammady S. Alsharkawy, Mahmoud E. Embabi, and M. A. Abd El-Baky. "Multi-objective task scheduling algorithm for load balancing in cloud computing based on improved Harris hawks optimization." *The Journal of Supercomputing* 81, no. 6 (2025): 1-38.
21. Zhong, Hai, Kun Tao, and Xuejie Zhang. "An approach to optimized resource scheduling algorithm for open-source cloud systems." In *2010 Fifth Annual ChinaGrid Conference*, pp. 124-129. IEEE, 2010.
22. Guo, Caishan, Fengji Luo, Zexiang Cai, Yuyan Sun, and Wenhui Tang. "Combined cloud and electricity portfolio optimization for cloud service providers." *Applied Energy* 377 (2025): 123925.
23. Song, Xinhui, Zhengwei Qu, Yunjing Wang, and Zhenxiao Chong. "Optimization of multi-energy cloud energy storage for multi-microgrid system with hydrogen refueling station." *Renewable Energy* 241 (2025): 122382.

24. Janakiraman, S., & Priya, M. D. (2023). Hybrid grey wolf and improved particle swarm optimization with adaptive inertial weight-based multi-dimensional learning strategy for load balancing in cloud environments. *Sustainable computing: informatics and systems*, 38, 100875.
25. Muteeh, A., Sardaraz, M., & Tahir, M. (2021). MrLBA: multi-resource load balancing algorithm for cloud computing using ant colony optimization. *Cluster Computing*, 24(4), 3135-3145.
26. Mishra, S. K., & Manjula, R. (2020). A meta-heuristic based multi-objective optimization for load distribution in cloud data center under varying workloads. *Cluster Computing*, 23(4), 3079-3093.
27. Qi, Zhang, Dong Yingjie, Ye Shan, Li Xu, He Dongcheng, and Xiang Guoqi. "An improved Coati Optimization Algorithm with multiple strategies for engineering design optimization problems." *Scientific Reports* 14, no. 1 (2024): 20435.
28. Zhu, Lun, Yongquan Zhou, and Qifang Luo. "Hybrid discrete coati optimization algorithm for solving large-scale multiple traveling salesman problem." *Ain Shams Engineering Journal* 16, no. 10 (2025): 103637.
29. Biswas, Saptadeep, Binanda Maiti, Gyan Singh, Absalom E. Ezugwu, Kashif Saleem, Laith Abualigah, Aseel Smerat, and Uttam Kumar Bera. "A Novel Hybrid Optimizer Based on Coati Optimization Algorithm and Differential Evolution for Global Optimization and Constrained Engineering Problems." *International Journal of Computational Intelligence Systems* 18, no. 1 (2025): 157.
30. S. J. Priya, P. Abinaya, J. S. Kumar, and M. D. Jeyashree, "An Efficient Approach for Task Scheduling and Load Balancing Across Multi-Cloud Frameworks," in 2025 11th International Conference on Smart Computing and Communications (IC SCC), pp. 1-6, IEEE, 2025.