# *Workflow Scheduling in Cloud Computing using Mimetic Algorithm*

**Dr B.V. RamaKrishna**
Associate professor, AI&DS Department
Vignan Institute of Technology and Science
Yadadri, Bhuvangiri (Dist), Hyderabad.

**V. Chandana Sri**
UG student, AI&DS Department
Vignan Institute of Technology and Science
Yadadri, Bhuvangiri (Dist), Hyderabad.

**P. Nithin Reddy**
UG student, AI&DS Department
Vignan Institute of Technology and Science
Yadadri, Bhuvangiri (Dist), Hyderabad.

**R. Durga Mallikarjun**
UG student, AI&DS Department
Vignan Institute of Technology and Science
Yadadri, Bhuvangiri (Dist), Hyderabad.

## ABSTRACT

Cloud computing has provided a new prospect for modern data processing because it enables scaling, elasticity, and economical management of computing resources for different services. One of the major problems in this area is efficient workflow scheduling, which comprises of allocating several interacting subtasks to virtual resources in a manner that gains optimization. This work proposes an advanced schedule based on a Mimetic Algorithm which is an hybrid algorithm including Genetic Algorithm, robust global explorations and more effective solutions through precise optimization with local Hill Climbing. This hybrid strategy enhances efficiency in both the pace and quality of results obtained. Such improvements resolve the cloud workflow scheduling problems intrinsic to its NP hard properties. It also utilizes dynamic resource allocation systems which aim for reduced execution costs and make span by dynamically aligning tasks to the distributed resources offered in the cloud. Simulations using Cloud-Sim showed that response time, cost, and resource utilization were significantly improved when compared to Genetic Algorithm. Despite of having challenges such as computational demands and scalability in large scale environments this approach marks a significant advancement in cloud workflow scheduling, showcasing great potential for future improvements.

***Key Words***: - *Cloud Computing, Metaheuristic, Hill Climbing, Resource utilization, Genetic algorithm, Optimization, Scheduling*.

## INTRODUCTION

Cloud Computing domain becoming the solution for modern era digital communication offering on-demand virtualization benefits to clients with inexpensive infrastructures [2]. The work scheduling of cloud servers is placing great challenge which involves task mappings, interdependencies management and resource planning activities [5]. The goal of workflow scheduling is to optimize the performance of cloud systems through minimal cost and maximal resource utilization in energy efficient environments [3]. Workflow scheduling falls under NP-hard problem. Various heuristic and metaheuristic algorithms identified for optimal solution efficiency [7]. However these algorithms face premature convergence and poor local search capability problem [4]. Investigations maximized techniques efficiency to meet constrained cloud workflow deadlines. SLA based resource allocation for SaaS environments maximizing the resource utilization ability [10]. Dual criteria scheduling algorithms in grid workflows improving efficiency and optimization degree in work flow processes [8].

## METHODOLOGY

The Mimetic Algorithm (MA) is integrated with the global search techniques like Genetic Algorithms (GA) and Hill Climbing which finely tune local optimization rate. This integration allows algorithms to look into wide search spaces and provide optimal solutions in variable scaling. MA improves the workflow scheduling problem in cloud computing, which is NP-hard because it requires a large meticulous assignment of tasks and resources in order to minimize cost, spawn and resource utilization. It also honors strict deadlines and SLAs in a significant way. The functioning of the MA is a process of a series of specified steps, each of them contributing to improve the ultimate scheduling outcome.

*Initialization*

The algorithm initiates with a random initial population of potential solutions, each being a specific type of the workflow tasks onto available cloud resources. The solutions are depicted such that each workflow task is connected to a VM, and consequently, the entire execution plan is constructed. Diversity of the population is very important because it enables to cover a large region of possible solutions by the algorithm. By covering extra regions of the solution space early, the algorithm minimizes the chances of premature convergence and maximizes the chances of finding a globally optimal solution.

*Fitness Evaluation*

Each population solution is evaluated by a fitness function that tries to capture the quality of scheduling. Many significant metrics are taken into account by the fitness function. They are Execution Cost-Total cost incurred while executing the tasks on cloud resources, Execution time: Cumulative time required to finish the entire workflow and Resource Utilization: The extent to which available resources are being utilized effectively. The fitness function also encompasses real- world constraints, i.e., task dependencies, deadlines, and SLAs. The constraints ensure that only realistically feasible and implementable schedules are being taken into account, and hence the resultant solutions are made more realistic and applicable.

*Selection*

For constructing the next generation, a selection is performed. This one selects the parent solutions on the basis of their fitness values. The roulette wheel method of selection is widely employed wherein the chance for a solution n to be selected is based on its fitness score. It fulfills the requirements such as use strong solutions by providing them with a greater opportunity for reproduction and also maintains the population diverse to prevent stagnation as well as the local optima traps.

*Crossover and Mutation*

After parent selection, the genetic operations involve both crossover and mutation

Crossover: This operation consists of sampling pieces of two or more parent solutions and mixing them to generate new individuals. Through the mixing of the attributes of good performing solutions, crossover encourages the creation of new and potentially improved scheduling setups. Various crossover strategies, including single-point or multi-point crossover, may be employed based on the encoding scheme. Mutation: Mutation adds random, small-scale alterations to a solution, i.e., assigning an activity to a different available VM. This operation is very important in avoiding the algorithm getting stuck in an area of the space that has a lot of suboptimal solutions. It helps in boosting the diversity of the population and enables the algorithm to search new areas of the search space, hence improving the speed and the scope of the search process.

*Local Search with Hill Climbing*

The basic difference between Mimetic Algorithm and Genetic Algorithm is the addition of a local search technique. A Hill Climbing, in our case as a secondary level of post-processing procedure for each child generated in each iteration. Once the genetic iterations have generated the child solution, it is optimized by local search. The Hill Climbing technique searches for nearby solutions step by step by making small, incremental changes. If there are any changes in an improvement according to the fitness function, it is accepted and executed; otherwise, it is rejected.

Algorithm: Hill Climbing

```
Step 1:  function HILLCLIMB(individual)
Step 2:  current ← individual
Step 3:  bestFit ← EVALUATEFITNESS(current)
Step 4:  for i = 1 to maxIters do
Step 5:  neighbor ← GENERATENEIGHBOR(current)
Step 6:  neighborFit ← EVALUATEFIT- NESS(neighbor)
Step 7:  if neighborFit > bestFit then
Step 8:  current ← neighbor
Step 9:  bestFit ← neighborFit
Step 10: end if
Step 11: end for
Step 12: return current
Step 13: end function
```

Algorithm: Genetic Algorithm

```
Step 1:  function GENETICALGORITHM(regionalList)
Step 2:  population←GENERATEPOPULATION
Step 3:  (regionalList)
Step 4:  for i = 1 to 100 do
Step 5:  population←EVOLVEPOPULATION(population)
Step 6:  end for
Step 7:  return best DC from sorted population
Step 8:  end function
Step 9:  function EVOLVEPOPULATION(population)
Step 10:  newPop ← empty list
Step 11: for all pairs in population do
Step 12: parent1, parent2 ← random selection
Step 13: child ← CROSSOVER(parent1, parent2) 13: MUTATE(child)
Step 14: Add child to newPop
Step 15: end for
Step 16: return newPop
Step 17: end function
Step 18: function MUTATE(individual)
Step 19: if random < 0.1 then
Step 20: Modify fitness by small random value
Step 21: end if
Step 22: end function
```

Algorithm: Mimetic Algorithm

```
Step 1:  function MEMETICSCHEDULER(regionalList)
Step 2:  population ← ConfigureSimulation- Panel.population
Step 3:  solution ← empty list
Step 4:  for all dc in regionalList do
Step 5:  fitness ← CALCULATEFITNESS(dc, popula- tion)
Step 6:  Add (dc, fitness) to solution
Step 7:  end for
Step 8:  Sort solution by fitness
Step 9:  return DC with max fitness
Step 10: end function
Step 11: function CALCULATEFITNESS(dcName, mem pop)
Step 12: vmFit ← 0
Step 13: for all bean in mem pop do
Step 14: if bean.DC == dcName then
Step 15: vmFit += bean.fitness
Step 16: bean.fitness -= 100 ▷ Local search tweak
Step 17: return vmFit
Step 18: end if
Step 19: end for
Step 20: end function
```

The simulation experiment conducted to show the performance metrics of the proposed Mimetic Algorithm using the tool kit know as Cloud-Sim Java tool. The tests incorporate different type of workflows of cloud infrastructure to simulate realistic environment. For bench marking purposes Mimetic Algorithm was test against the conventional algorithms like the Genetic Algorithm & Particle Swarm Optimization across different workload levels.

## RESULT ANALYSIS

*Observations*

*Execution Time Improvement:*

Simulation conducted with Mimetic Algorithm reveal a decrease of fifteen to thirty percent in workflow execution time compare to basic algorithms. This rise was due to the Mimetic algorithm local search capability and efficient solution run time optimization searching enhances convergence and task scheduling efficiency. Moreover workflow partitioning is a result of local search method improves parallel execution in distributed processing environment.

*Cost Optimization:*

Mimetic Algorithm given a outstanding performance in controlling costs in the financial aspects. The algorithm relatively optimized VM selection and cloud data center allocation as the workload, resource requirements, and pricing structures changed, which met performance requirements while controlling costs. This change improvised the resource allocations and expenditure frameworks resulting in lower operational expenditure in every term in the Genetic Algorithm.

TABLE 1. Genetic Vs Mimetic in VM Environment

| Metric(s) | Genetic Algorithm (GA) | Mimetic Algorithm (MA) |
|---|---|---|
| Average Execution Time (ms) | 71.51 | 69.74 |
| Average Cost ($) | 3.51 | 3.12 |
| Resource Utilization | 91% | 96.7% |
| VM optimization | 93% | 97.8% |

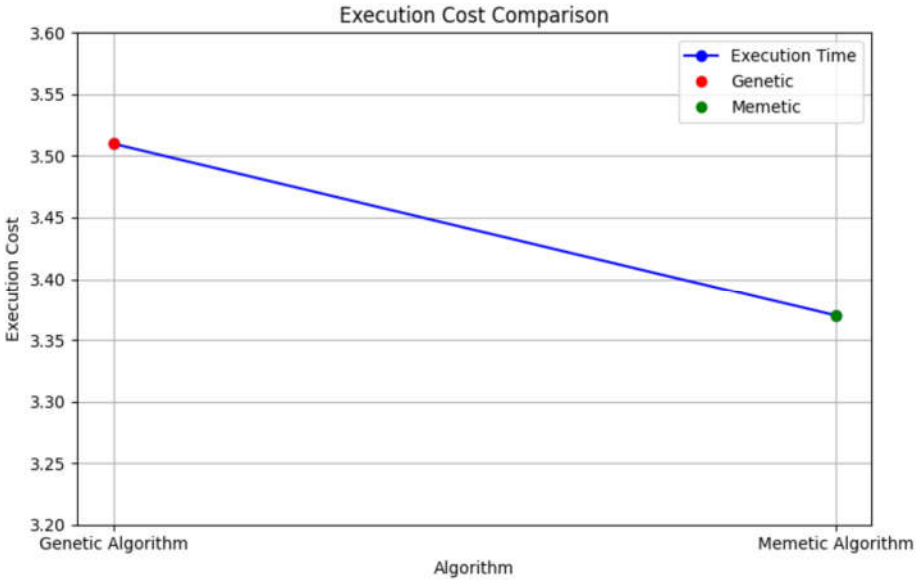


Figure1. Process Cost Optimization comparison

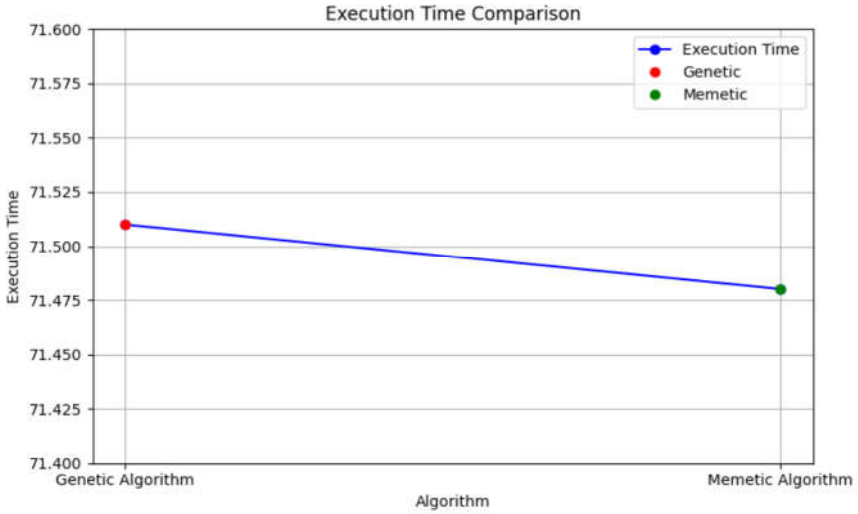


Figure2. Process Time Comparison

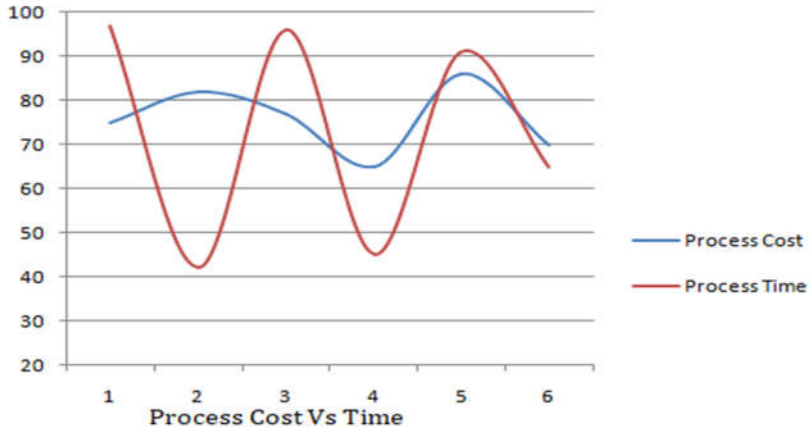


Figure3. Process Time Vs Process Cost

## CONCLUSION

The Mimetic Algorithm (MA), a hybrid Metaheuristic approach that combines the global optimization power of Genetic Algorithms (GA) with precision and delicacy of local search methods like Hill Climbing, presents a robust solution to the remove the issue of workflow scheduling in cloud computing environments. Based on the natural selection process and evolution concepts, MA brings both intelligence and adaptability to the scheduling process. Whereas Genetic Algorithms enable searching of the solution space in an extensive way, whereas Hill Climbing will search by refining single solutions in a strengthening way. This supporting enables the Mimetic Algorithm to overcome some of the usual weaknesses associated with conventional methods, including low convergence and being trapped within local optima. . In actual commercial cloud environments, workloads sometimes change in scope, and access to resources could dynamically change as well. By virtue of being adaptive, MA's scheduling property enables it to react well against such changes while making real-time adjustments to secure ongoing optimization. This adaptive attribute not only enables sustained performance but also helps support service-level agreement (SLA) compliance critical in commercial use of clouds.

## REFERENCES

[1] Prashant Shukla and Sudhakar Pandey, Energy Efficient Workflow Scheduling Algorithm for Latency-Sensitive Applications using Cloud-Fog Collaboration, VOL. 12, NO. 3, 2024.

[2] Amer Saeed; Gang Chen; Hui Ma; Qiang Fu, Genetic Algorithm with Repair Method for Deadline-Constrained IoT Workflow Scheduling in Cloud Computing, 2024.

[3] Parameshachari B.D, Usha M, Shwetha N, Mohan B R,Scheduling the Task of User in Cloud computing using Hybrid Procedure of PSO and Lion Algorithm, VOL. 10, NO. 4, 2022.

[4] Ismail M. Ali, Karam M. Sallam, Nour Moustafa, An Automated Task Scheduling Model Using Non-Dominated Sorting Genetic Algorithm II for Fog-Cloud Systems,2019.

[5] Ahmad Taghinezhad-Niar and Javid Taheri, "Security, Reliability, Cost, and Energy-Aware Scheduling of Real-Time Workflows in Compute-Continuum Environments", 7th International Conference on Future Internet of Things and Cloud (Fi-Cloud), 2019.

[6] W. Chen, J. Wu, and Z. Zheng, A Throughput Maximization Strategy for Deadline Constrained Cloud Workflow Systems, in Proceedings of the 2013 IEEE 6th International Conference on Cloud Computing, pp. 911-918, 2013.

[7] Abdulsalam Alsmady, Tareq Al-Khraishi, Wail Mardini, Hadeel Alazzam, Yaser Khamayseh, Workflow Scheduling in Cloud Computing Using Memetic Algorithm in 6th Annual China Grid Conference, pp. 3-9, 2011.

[8] Georgios L. Stavrinides, Helen D. Karatza, Cost-Effective Utilization of Complementary Cloud Resources for the Scheduling of Real-Time Workflow Applications in a Fog Environment, IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 8, pp. 1374-1381, 2011.

[9] L. Wu, S. K. Garg, and R. Buyya, SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments, in 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 195-204, 2011.

[10] R. Prodan and M. Wieczorek, Bi-Criteria Scheduling of Scientific Grid Workflows, IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 3, pp. 417-431, 2009.