UDC 519-62
DOI: 10.20535/SRIT.2308-8893.2022.3.08

COST EFFECTIVE HYBRID GENETIC ALGORITHM FOR WORKFLOW SCHEDULING IN CLOUD

SANDEEP KUMAR BOTHRA, SUNITA SINGHAL, HEMLATA GOYAL

Abstract. Cloud computing plays a significant role in everyone's lifestyle by snugly linking communities, information, and trades across the globe. Due to its NP-hard nature, recognizing the optimal solution for workflow scheduling in the cloud is a challenging area. We proposed a hybrid meta-heuristic cost-effective load-balanced approach to schedule workflow in a heterogeneous environment. Our model is based on a genetic algorithm integrated with predict earliest finish time (PEFT) to minimize makespan. Instead of assigning the task randomly to a virtual machine, we apply a greedy strategy that assigns the task to the lowest-loaded virtual machine. After completing the mutation operation, we verify the dependency constraint instead of each crossover operation, which yields a better outcome. The proposed model incorporates the virtual machine's performance variance as well as acquisition delay, which concedes the minimum makespan and computing cost. One of the most astounding aspects of our cost-effective hybrid genetic algorithm (CHGA) is its capacity to anticipate by creating an optimistic cost table (OCT) while maintaining quadratic time complexity. Based on the results of our meticulous experiments on some real-world workflow benchmarks and comprehensive analysis of some recently successful scheduling algorithms, we concluded that the performance of our CHGA is melodious. CHGA is 14.58188%, 11.40224%, 11.75306%, and 9.78841% cheaper than standard Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Cost Effective Genetic Algorithm(CEGA), and Cost-Effective Load-balanced Genetic Algorithm (CLGA), respectively.

Keywords: cloud computing, cost effective, genetic algorithm, metaheuristic algorithm, predict earliest finish time, Workflow scheduling.

INTRODUCTION

Cloud computing is a buzzword in the current era, which provides a very elastic 'pay as you go' model[1]. Dr. Raj Kumar Buyya says, "A cloud is a kind of parallel and distributed system made up of a number of linked, virtualized computers that are constantly provided and shown as one or more unified computing resources in accordance with service-level agreements negotiated between the service provider and customers" [2]. On the basis of physical location and distribution, various deployment models are available now. Task scheduling is critical for maximizing the use of cloud resources as well as providing end users with quality of service (QoS) [3].

Static scheduling and dynamic scheduling are two different sorts of task scheduling issues. In the static category, all task characteristics, including the costs of computation and communication for each activity as well as how those activities relate to one another, are known in advance. However, the dynamic category makes such information unavailable and makes judgments at runtime [4]. Furthermore, static scheduling refers to compile-time scheduling, and dynamic scheduling refers to scheduling at runtime.

Heuristic-based and guided random search-based algorithms are the two types of static scheduling algorithms that are most frequently used.

Heuristic-based algorithms deliver approximate, frequently excellent results because of their polynomial time complexity [5]. Similar to heuristic-based algorithms, guided random search-based algorithms provide approximations, but the results' quality may be increased by including more rounds, which raises the cost of the methods [6].

Dependent tasks are represented as a workflow, which is a set of nodes and edges, with each node representing a job and each edge representing follow-up dependence [7]. Workflow is scheduled and executed by the workflow management system where tasks are scheduled and provisioned to virtual machines [8]. Various researchers are engaged themselves to resolve the problem of resource scheduling in cloud. Many tasks have been completed using the heuristic approach; however it is not very excellent owing to its problem-dependent aspect, which is that it is unable to provide a globally optimum solution. As a result, the researcher prefers to use a meta-heuristic technique. Due to its task-independent character, the meta-heuristic method delivers a global optimal solution. The researcher's ultimate objective is to maximize cloud resource usage while lowering costs for cloud's end users [9], [10].

The majority of quadratic time complexity list-based scheduling algorithms just evaluate the current task when allocating a task to a processor. Although it is a low-cost method, it does not examine what comes before the current job, which could lead to poor decisions in some cases. Lookahead [11] is an example of an algorithm that analyses the impact on child nodes, but it raises the time complexity to the fourth order. As a result, we used the PEFT approach in our proposed model "cost-effective hybrid genetic algorithm" (CHGA). One of PEFT's most amazing features is its ability to predict by making an OCT with optimistic costs while preserving quadratic time complexity.

Motivation. Following a comprehensive review, we motivated to resolve a research gap where many parameters, such as virtual machine (VM) performance variation, booting time, and shutdown time, as well as load balancing across VMs and minimize execution time in parallel using heuristics approaches, are not effectively addressed.

Objective. The goal of this research is to arrange the tasks of workflow in such a way that it reduces not only computation costs but also processing time while maintaining load balance among virtual machines. Our objective was to create a hybrid meta-heuristic technique for reducing processing time and expense while maintaining load balance across virtual machines under time constraint. During population initialization in genetic algorithm, we employed predict earliest finish time (PEFT) approach, which is significant for decreasing the makespan. We also took into account the time it takes for a virtual machine (VM) to boot up and performance fluctuations, both of which have an effect on computation time and execution cost. This represents the novelty of our model. To keep the load balanced among the virtual machines, we employed a greedy method [10] in our proposed model "cost-effective hybrid genetic algorithm" (CHGA).

The remaining sections of the paper are structured as follows. The second section goes through some background information. Sections III and IV contain problem definitions and details of our proposed model, respectively. The performance review may be found in Section V. Section V consists of two subsections: result analysis and discussion. Finally, Section VI draws the paper toward its conclusion.

RELATED WORK

Comprehensive work has been done by us on various meta-heuristic algorithms in literature [9], some of them are genetic algorithm (GA) [12], ant colony optimization (ACO) [13], particle swarm optimization (PSO) [14], artificial bee colony algorithm (ABC) [15] etc.

The RDPSO (Revised Discrete PSO) technique employed in[16] involves a greedy adaptive search procedure to establish the swarm particle, followed by the computation of local best and global best. It focuses on achieving the lowest execution cost, but load balancing across virtual machines is not provided.

In literature [17] , researchers designed a PSO-based algorithm to minimize execution cost as well as makespan and compared it with the Best Resource Selection (BRS) algorithm, but they didn't take into account dependent tasks in scheduling approach. This deficiency is removed by [18], where ACO is applied to the workflow. They used an approach ant strategy: front ant and back ant. Their study took into account pre-execution time and a pheromone threshold value, but they did not mimic a different type of scientific workflow.

Researchers in Dynamic Objective-based GA (DOGA) [19] reduced the cost of workflow execution and reached a result that was comparable to PSO, but they ignored the booting time factor and the load balancing approach. Authors provided a GA-based technique in the literature [20], where cost and time span are both reduced within a user-defined deadline. This paper was not based on real world workflow, which is accomplished by [21].

A multi-objective PSO approach with a weighted linear transform fitness function is presented in the literature [22] and they conclude that their proposed algorithm is better than genetic algorithms, but they consider only makespan and resource utilization as parameters, not other parameters like execution cost, load distribution, etc. The outcome of their experiment is not very trustworthy due to the limited size of their workflow.

A new approach SACO Slave ACO(SACO) [23] proposed a slave-ant concept where two techniques are used: diversification and reinforcement. These techniques escape slave ants from long paths. Their experiment didn't consider heterogeneous resources or load balance concepts. Multi Objectives ACO(MO-ACO) [24] addresses this flaw by presenting an approach for scheduling jobs in a cloud context that considers load balancing with cost and time but ignores dependent tasks in the cloud.

The Greedy-Ant-based ACO [25] approach uses forward and backward dependency techniques to build transition probability. To allocate the virtual machine, they used a greedy strategy. They compared their meta-heuristic model with a heuristic that has a high level of scarceness in their research.

In the suggested GA [26], VMs are grouped based on their capacity to shorten the time it takes for a procedure to complete. Before clustering the VM, they considered cost computing to make this approach more successful. They did not include the VM termination delay in their study, and they also did not examine the load balancing idea.

In the literature [27], authors focused on function optimization using improved genetic algorithms, whereas machine learning concepts are included with GA [28].

In order to reduce makespan and cost, authors presented a HEFT-ACO technique [29] that is based on the heterogeneous earliest end time (HEFT) and ACO, but they did not integrate the idea of load balancing across virtual machines.

In research [10], authors focused on balancing the load among virtual machines to increase performance. To achieve this, a greedy seeding strategy was applied with the genetic algorithm, but there was no efficient heuristic approach to reduce the makespan and cost.

Following a comprehensive review, we observed a research gap where many parameters, such as virtual machine (VM) performance variation, booting time, and shutdown time, as well as load balancing across VMs and minimize execution time in parallel using heuristics, are not effectively addressed.

Our goal was to develop a hybrid meta-heuristic approach for processing time and cost reduction in a time-constrained situation while maintaining load balance across virtual machines. To accomplish this, we used the PEFT strategy during population initialization, which helps to reduce the makespan. The ability of PEFT to anticipate by building an optimistic cost table (OCT) while preserving quadratic time complexity is one of its most amazing features. We also took into account the time it takes for a VM to boot up and performance fluctuations, both of which have an effect on computation time and execution cost. To keep the load balanced among the virtual machines, we employed a greedy method in our proposed model CHGA.

PROBLEM DEFINITION

Minimization of computing costs and makespan of scientific workflow with balancing the loads among virtual machines is the main motto of our proposed Cost Effective Hybrid Genetic Algorithm (CHGA), which works under a user-defined deadline constraint. A simple workflow is depicted in Fig. 1, and its corresponding encoding is represented in Fig. 2.

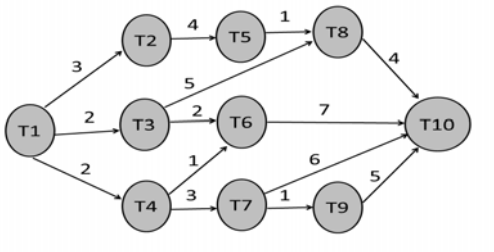

Fig. 1. Example of Workflow

Encoding of Chromosome

OrderOfTask	1	2	3	4	5	6	7	8	9	10
Task	T1	T4	T3	T6	T7	T2	T9	T5	T8	T10
VM	1	3	3	4	2	2	5	4	1	5
VmType	3	2	2	2	1	1	1	2	3	1

Fig. 2. Encoding of workflow depicted in Fig. 1

In a heterogeneous cloud computing environment, variation in the performance of VMs and booting time delays are two main factors that impact the makespan of the scientific workflow. That's why we considered both of the above-mentioned parameters in our proposed model. Schedule is illustrated as

$S = \{VM_{SET}, Map, TET, TEC\}$, Where VM_{SET} a virtual machine pool, and Map denotes the selection of an appropriate virtual machine to perform a task. Total Execution Time and Total Execution Cost, respectively, are abbreviated as TET and TEC. We generate the value 0%–24% randomly as a performance variation, and the acquisition delay is assumed to be 1 minute for each VM. We defined the problems to achieve our objectives. If TET violates the deadline constraint, then TEC is not computed, otherwise it will be computed.

THE PROPOSED HYBRID GENETIC ALGORITHM

Description of the CHGA

We explained CHGA step-by-step here.

Step 1. During population initialization, the chromosome is encoded in the same way as in the meta-heuristic technique provided by [25]. OrderOfTask, Task, VM, and VmType are four fields that are used to encode a chromosome.

If the total population is N, then $(N-1)$ is initialized using a random technique and the remaining is using PEFT. PEFT is described in section IV B.

Step 2. During the population initialization, we employed a greedy technique [10] to balance the load among several virtual machines, as illustrated through flowchart in this paper. This strategy assigns the new task to those VMi which have minimum load at that time. Compute Load Li on a VMi:

$$L_i = \frac{\sum_{j=1}^{n} T_j}{VMC_i}.$$

Step 3. Now compute the fitness of each candidate.

Step 3.1: Calculate the execution and transfer times for all of the individual's tasks.

Divide the` task's size $Size_{T_i}$ by the virtual machine's processing speed $Speed_{VM_k}$ to find the task's execution time $ET_{VM_k}(T_i)$

$$ET_{VM_k}(T_i) = \frac{Size_{T_i}}{Speed_{VM_k}}.$$

The size of an output data file $\mathrm{DataFile}_{T_i}$ and the typical bandwidth β may be used to compute the communication time $TT_{E_{ij}}$:

$$TT_{E_{ij}} = \frac{DataFile_{T_i}}{\beta}.$$

If a task is appear as root task or all parent tasks are on the same VM then communication time is zero.

Step 3.2. Calculate Execution start time ST_{T_i} and finish time $\mathrm{F}T_{T_p}$ now. ST_{T_i} is an estimated time to start the execution. It is equal to acquisition delay if the task is appear as root node, otherwise

$$ST_{T_i} = \{Max\{Avail\{VM_k\}, Max_{T_p}\}FT_{T_p} + TT_{E_{pi}}\}.$$

Here $Avail(VM_k)$ is the time of VMk when it is ready to execute a new task and the VM's performance variation is denoted by PerVar. $\mathrm{F}T_{T_p}$ indicates completion time of parent's task.

$$Avail(VM_k) = \left\{ ST_{T_i} + \left\{ \frac{ET_{VM_k}(T_i)}{(1 - PerVar)} \right\} \right\}.$$

FT_{T_i} is the time indicates to finish the execution.

$$FT_{T_i} = \left\{ ST_{T_i} + \left\{ \frac{ET_{VM_k}(T_i)}{(1 - PerVar)} \right\} \right\}.$$

Step 3.3. Now compute TET and TEC as given below:

$$\text{If } TET \le D,\ TET = \{FT(t_i)\},$$

$$TEC = \sum_{n=1}^{VM_n} C_{type(VM_k)} * \left[\frac{\text{VMno\_ET} - \text{VMno\_ST}}{TimeInterval} \right].$$

Equation (1) to Equation (8) are from literature [19], [20].

Step 4. After computing the fitness of the chromosome, the tournament-based algorithm is used to select the best two individuals for further crossover.

Step 5. A two-point crossover is used as depicted in Fig. 3.

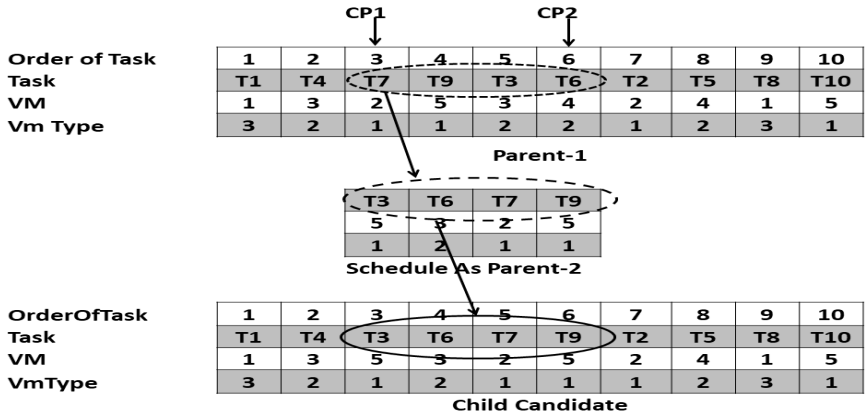

Fig. 3. Crossover Operation

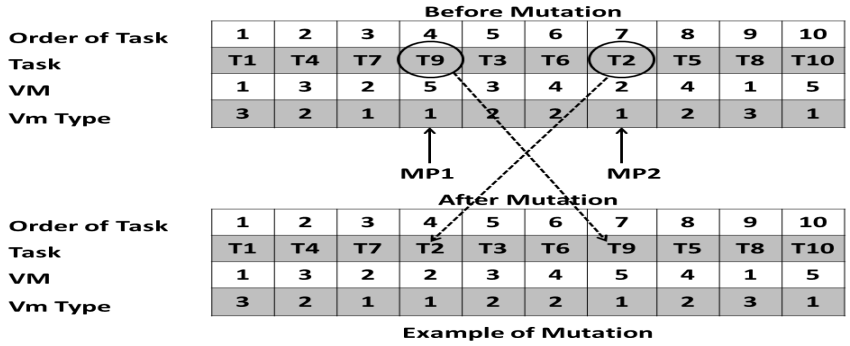

Fig. 4. Mutation Operation

Step 6. Apply mutation operations as illustrated in Fig. 4. Now check the dependency constraint on it.

If a new individual follows the dependency constraint, then it is accepted, otherwise it is discarded.

Step 7. If the fittest solution meets our objectives under the user-defined constraint, then stop the iteration; otherwise, continue it from step 3 after replacing the least fit candidate with the better new solution.

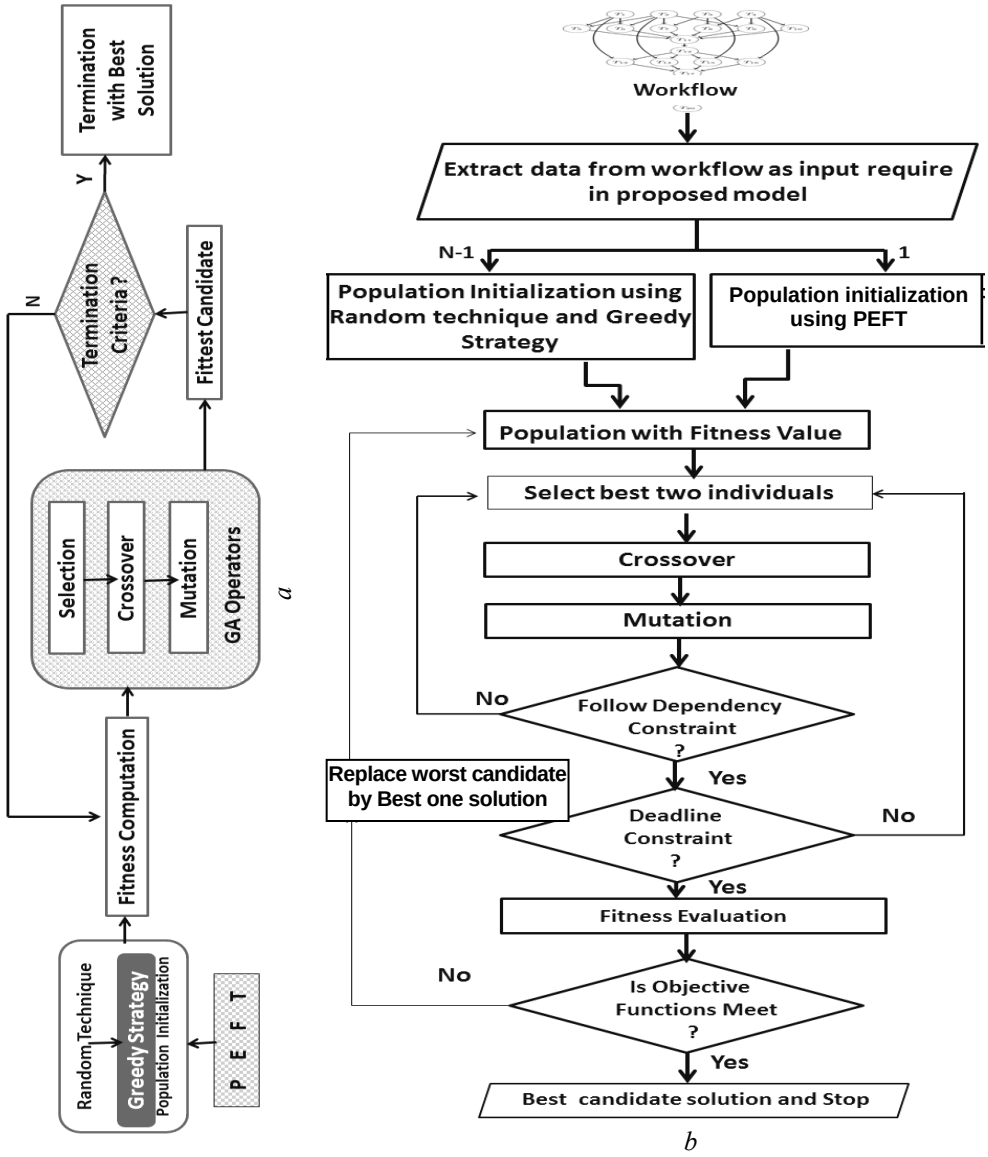

Fig. 5. Block Diagram of Proposed model CHGA (*a*); flowchart of Proposed model CHGA (*b*)

A picture is worth a thousand words, that's why we depict our proposed model CHGA through a block diagram and flowchart, as shown in Fig. 5, *a* and Fig. 5, *b* respectively.

A Glance on PEFT

The PEFT [30] consists of two stages: a task prioritization phase that identifies priority of task and a VM selection step that determines the optimum VM for executing the present job. Both stages are centered on OCT. By computing an OCT and retaining quadratic time complexity, this algorithm can forecast. Earliest Fin-

ish Time (EFT) of a node *n* on processor *p* is sum of earliest start time and computation time of a node *n* on processor *p*. Illustrated in Fig. 6.

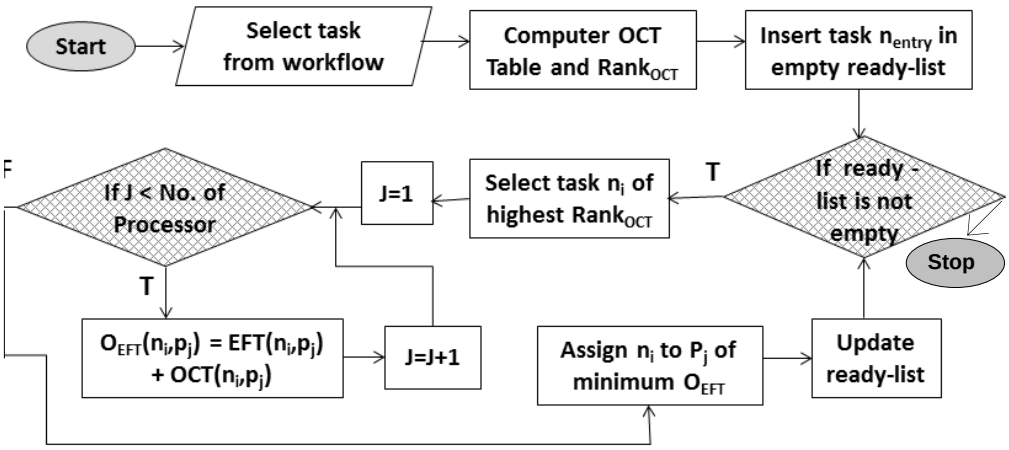

Fig. 6. PEFT Strategy

PERFORMANCE EVALUATION

Baseline Algorithm

In the current era, ACO and PCO are buzzwords. Both meta-heuristic algorithms are inspired by the natural process of resolving NP-hard problems like optimization. That's why we used them as baseline algorithms as they contributed to solving the same problem addressed here. Except these we used CEGA [20] and CLGA [10] as baseline algorithms.

Pheromone-based communication in an ant is to find the best solution. Initially, all the routes have the same probability of selection, i.e., 'no bias' due to the same or no pheromone. A local update rule is applied when the ant constructs the route, i.e., solution. Longer pathways vaporise or disintegrate more quickly than shorter ones do. Shorter pathways therefore accumulate more pheromones over time. Pheromone's quantity is responsible for indirect communication, which is known as 'stigmergy' [18]. When all the ants have completed their routes, then a global update operation is performed. Now the selection of the path is biased and the best ant is allowed to update the pheromone by the pseudo-random-proportional rule [18]. We can understand ACO from Fig. 7 and PSO from Fig. 8.

Particle Swarm Optimization was first introduced by Kennedy and Eberhart [22]. In this instance, swarm stands for the population, and particle for a potential solution. Each particle is first assigned a random coordinate. The objective function, or the distance between the particle's present position and the food, is used to evaluate performance. PBEST indicates the local best position of a particle, whereas refers to the velocity constant. By updating the velocity and position of the particle, a global optimum solution can be achieved. We keep this process going until we get our objective or reach our maximum iterations [22]. This is depicted in Fig. 8. As baseline algorithms we used ACO, PSO, CEGA [20] and CLGA [10].

```
Input: m, α, β, ρ, τ
Output: P_BEST
    InitPheromone(τ)
    P_BEST ⟵ NULL
    While StopCondition() do
        S_iter ⟵ ϕ
        For i=1 to m do
            S_i ⟵ ConstructSolution(τ)
            If S_i is valid output then
                S_i ⟵ LocalSearch(S_i)
                If ( (Obj_f(S_i )<Obj_f(P_BEST)) OR (P_BEST = NULL))
                    P_BEST ⟵ S_i
                end
                S_iter ⟵ S_iter ∪ {S_i}
            end
        end
        UpdatePheromone(τ, S_iter , P_BEST)
    end
Return P_BEST
```

Fig. 7. ACO Algorithm

```
Input: C_1, C_2, N
Output: G_BEST
    S ⟵ InitParticle()
    P_BEST ⟵ NULL
    While StopCondition() do
        For each particle P_i in S do
            // Update Local Best
            If f(P_i ) < f(P_BEST) then
                P_BEST ⟵ P_i
            end
            G_BEST ⟵ UpdateGlobalBest()
        end
        For each particle P_i in S do
            UpdateVelocity()
            UpdatePosition()
        end
    end
Return G_BEST
```

Fig. 8. PSO Algorithm

Experimental Setup

We used four types of scientific workflows: Montage, Cybershake, LIGO, and Epigenomics as benchmarks, where the size of the workflow is 50 nodes, 100 nodes, and 500 nodes approximately.

We have implemented the proposed model CHGA in a JAVA-based robust environment and concluded the result after executing each type of workflow 30 times. The accuracy of the obtained result varies by about ±5. As mentioned in Table 1, we considered 5 types of VMs as specification [31]. We assumed 20 kbps average bandwidth as proposed by Amazon Elastic Block Store (EBS) [32]. A thorough analysis of the literature [33],[34] is beneficial in deciding on various parameters.

There are 3 levels of deadline constraints: hard, crunch, and soft, which are considered in our experiment.

Deadline

$$D = (\beta + 1) \min\ ET\ ()W_i\ .$$

For hard deadline range of α : $0 \le \alpha \le 1.2$.

For crunchy deadline range of α : $1.2 < \alpha \le 2.8$.

For soft deadline range of α : $2.8 < \alpha \le 4.4$.

Here α indicates step length, whose value is 0.4.

Table 1. Configuration of VM in our practical approach

VM Types	m1.Small	m1.Large	m1.Xlarge	c1.Medium	C1.Xlarge
Processing Capacity (GFLOPS)	4.4	17.6	35.2	22	88
ECUs (Speed)	1	4	8	5	20
Cores	1	2	4	2	8
Memory (GB)	1.7	7.5	15	1.7	7
Disk(GB)	160	850	1690	350	1690
Cost /Hr. ($)	0.04	0.16	0.32	0.2	0.8

RESULT AND ANALYSIS

Evaluation of Deadline Constraint

Our suggested CHGA is evaluated and compared using baseline algorithms in order to fulfill our goal within a user-defined deadline, as depicted in Table 2 and Fig. 9. The hit rate of our proposed CHGA is better than that of other baseline algorithms, which represents its robustness. The capacity of PEFT to predict the impact of scheduling the all children task of the current parent task reduced the makespan of workflow and improved the hit rate of CHGA.

Table 2. Analysis of Hit Rate based on deadline

Deadline	Algorithm	Montage	Cybershake	LIGO	Epigenomics
Hard	CHGA	96.3002	94.0645	93.0989	92.3004
	ACO	53.9809	58.2309	52.0051	57.4506
	PSO	69.0989	67.9882	68.0898	69.1216
	CEGA	92.3433	88.4844	88.5034	83.4908
	CLGA	95.5022	91.4788	91.4602	88.0223
Crunch	CHGA	99.8956	99.8002	99.7444	99.8288
	ACO	72.0989	73.0899	71.9004	74.0112
	PSO	79.0767	80.3503	81.0302	78.9704
	CEGA	99.5011	99.6202	99.5002	99.6055
	CLGA	99.5067	99.7601	99.5676	99.7388
Soft	CHGA	99.8876	99.7909	99.7708	99.8092
	ACO	78.0998	76.0038	77.0902	78.2312
	PSO	83.4534	82.0034	85.7801	86.5709
	CEGA	99.6767	99.7022	99.6081	99.5003
	CLGA	99.6878	99.7803	99.7003	99.7099

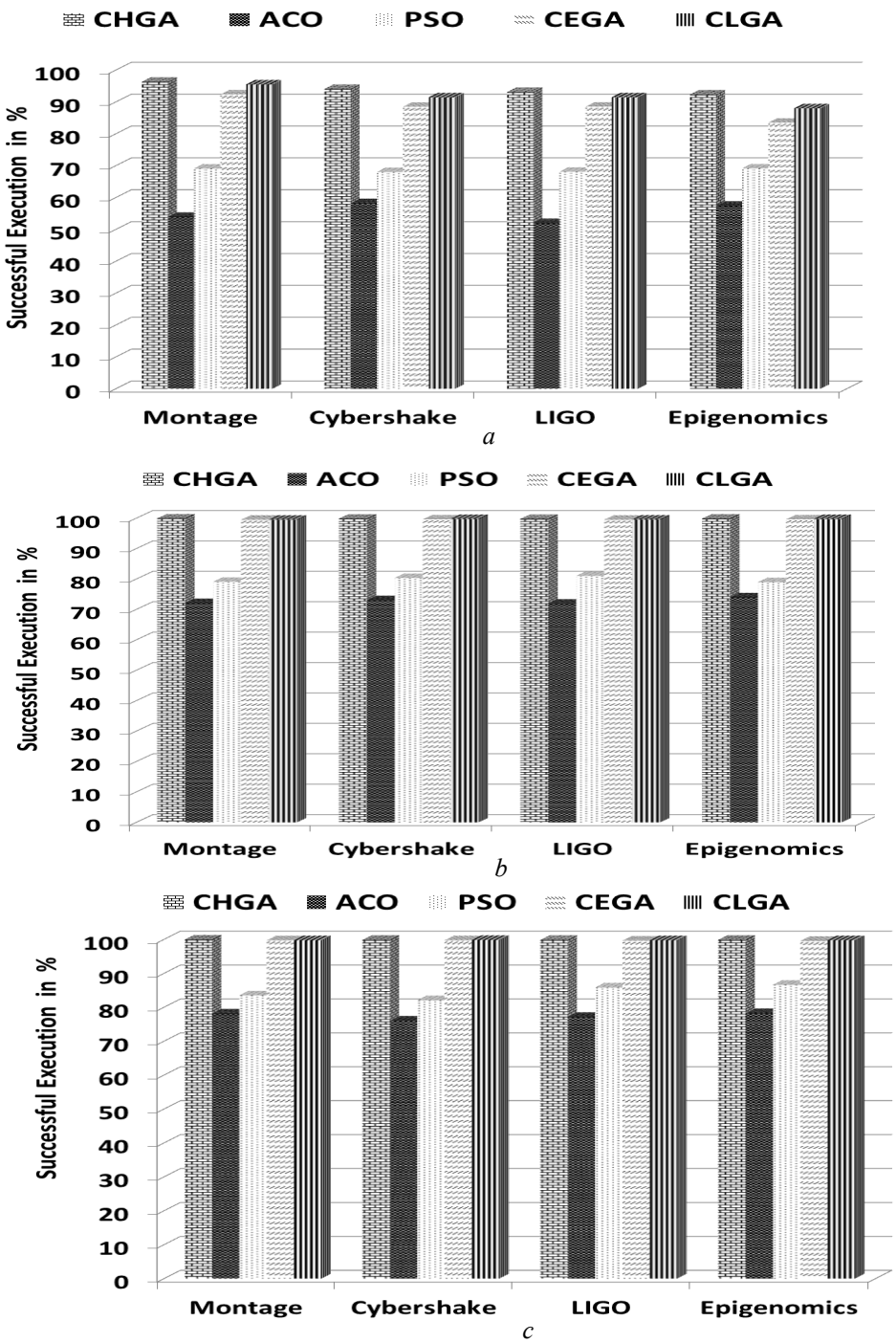

Fig. 9. Analysis under: Hard Deadline Constraint (*a*); Crunch Deadline Constraint (*b*); Soft Deadline Constraint (*c*)

Load-Balance Evaluation

Greedy strategy plays an important role in load balance. Finding a virtual machine with a low load is important before we allocate a task T_i to an individual. In order

to manage the load balance, we map the task T_i with VM_i using the greedy technique after identifying VM_i with minimum load.

The capacity of VMC_i can be calculated as given by Equation, where the number of processing elements is $\mathrm{PE}_{\mathrm{num}}$:

$$\mathrm{VMC}_i = \mathrm{PE}_{\mathrm{num}} \times \mathrm{PE}_{\mathrm{mips}} .$$

All VMC_i are collectively known as Virtual Machine Capacity (VMC), and m is the total number of VM_s :

$$\mathrm{VMC} = \sum_{i=1}^{m} \mathrm{VMC}_i .$$

Load L_i on a VM_i is as Equation.

Total load TL is as Equation

$$\mathrm{TL} = \sum_{i=1}^{m} L_i .$$

Load capacity per unit is LC_{pu} as Equation

$$LC_{pu} = \frac{\mathrm{TL}}{\mathrm{VMC}} .$$

Threshold value TH_i is as Equation

$$\mathrm{TH}_i = LC_{pu} \times \mathrm{VMC}_i .$$

The threshold value TH_i is compared with the load of VM_i to determine the status of VM_i, i.e., under-loaded, balanced or over-loaded. The result of our experiment shows that with ACO, VM1 is overloaded by +82% and VM3 is under-loaded by -58%. In contrast, with PSO, VM5 is overloaded by + 69% and VM4 is under-loaded by -63%, as shown in Fig. 10. Our model CHGA exhibited better load-balance compare to ACO, PSO, and CEGA, which denotes the robustness of CHGA. When we used the proposed CHGA, VM4 was overloaded by +26%, while VM3 was under-loaded by –12%. Illustrated in Fig. 10.

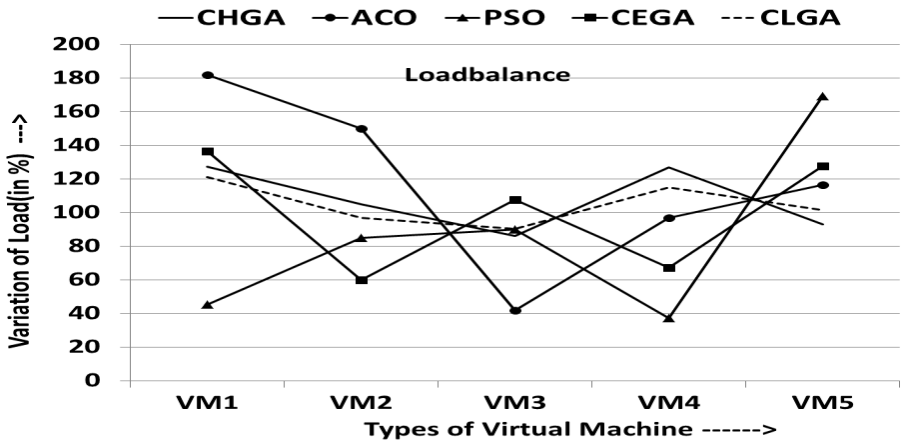

Fig. 10. Comparatively analysis of Load Balance

Cost and Makespan Evaluation

Our holistic comparison between the baseline and our proposed CHGA is depicted in Fig. 11 – 14. The obtained result of our experiment indicates the robustness of our proposed model CHGA. CHGA is 14.58188%, 11.40224%, 11.75306%, and 9.78841% cheaper than standard ACO, PSO, CEGA, and CLGA, respectively. CHGA's average makespan is 34.73619%, 31.48127%, 5.71553%, and 9.73710% lower than standard ACO, PSO, CEGA, and CLGA, respectively.

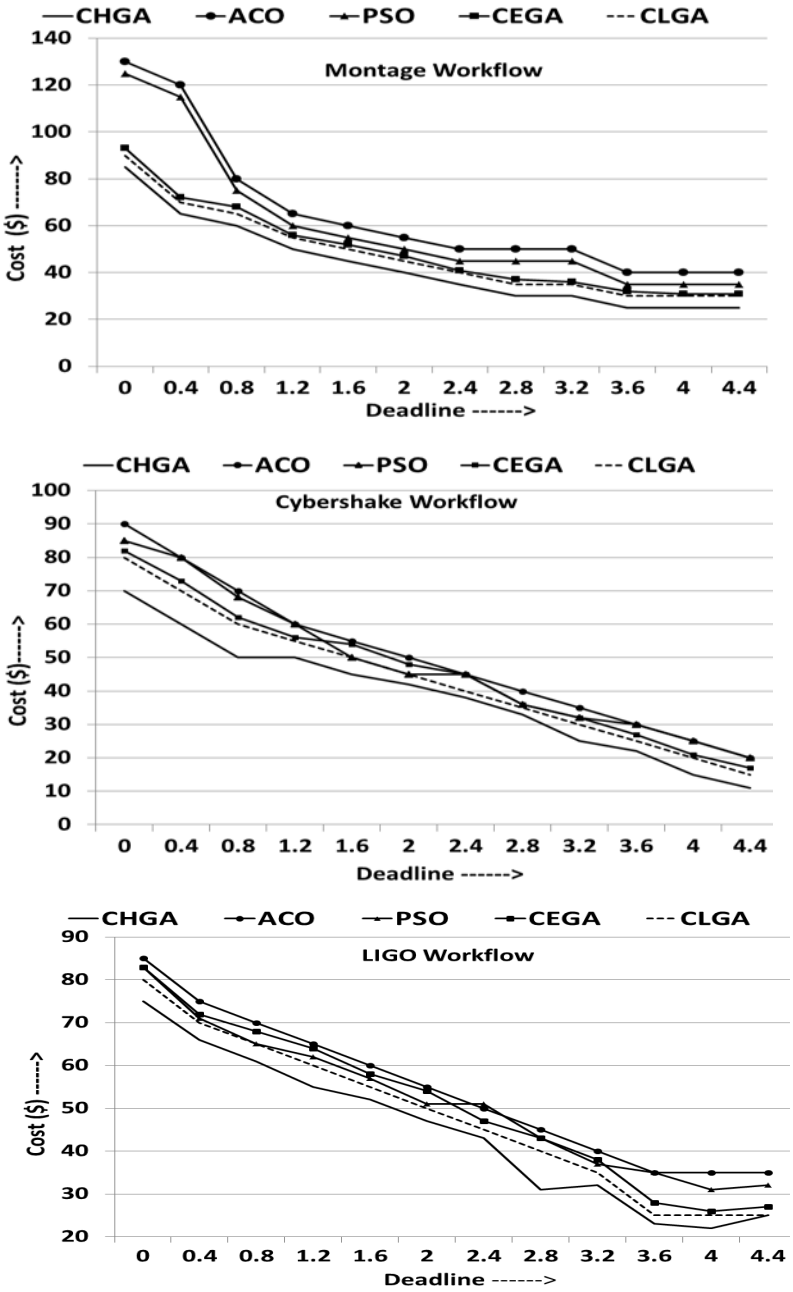

Fig. 11. Comparatively analysis of Cost. *Began*

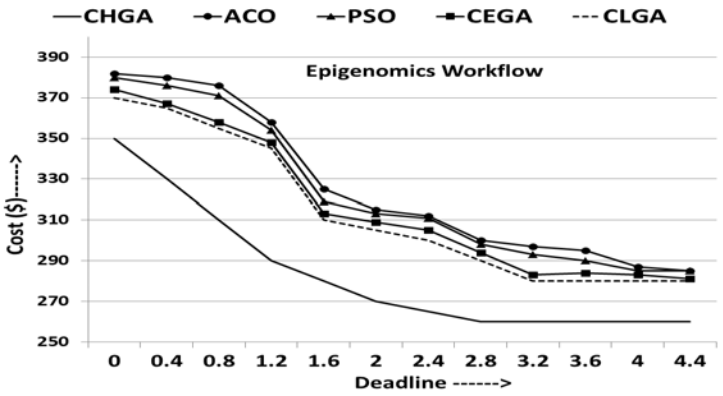

Fig. 12. Comparatively analysis of Cost. *Continued*

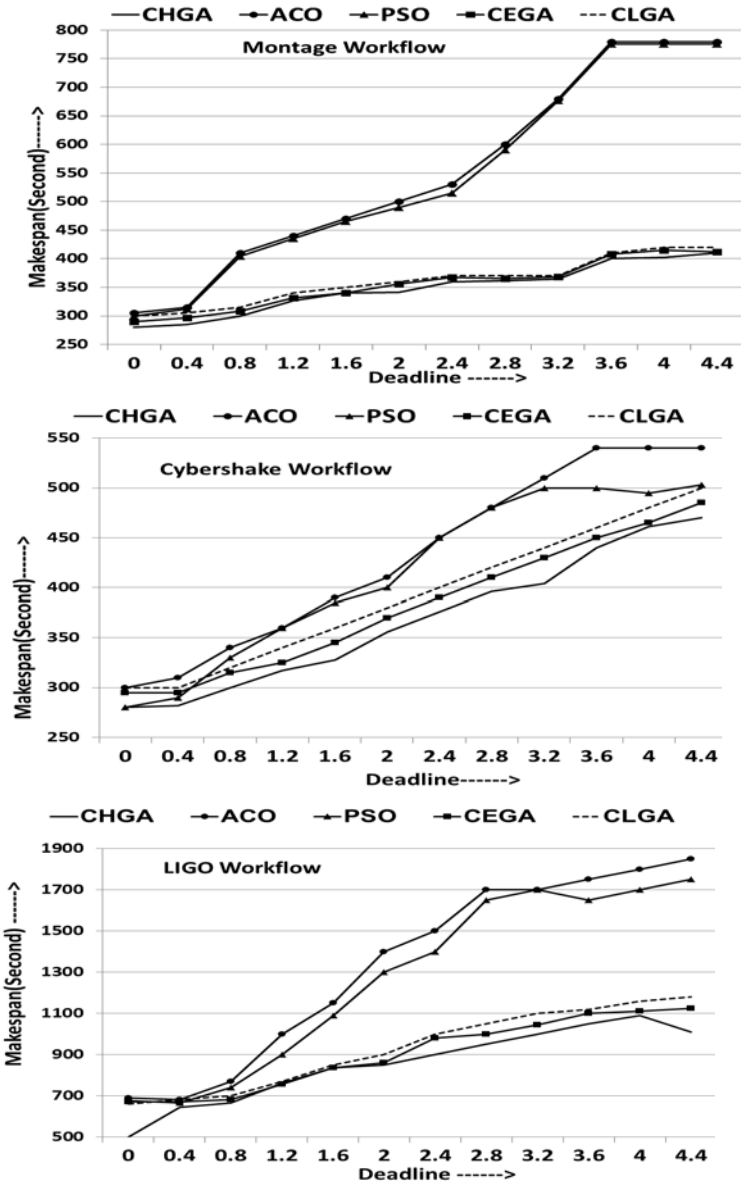

Fig. 13. Comparatively analysis of Makespan. *Began*

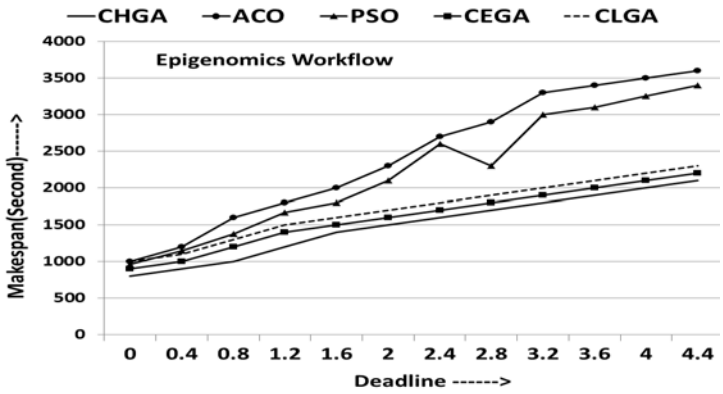

Fig. 14. Comparatively analysis of Makespan. *Continued*

The capacity of PEFT to predict the impact of scheduling the all children task of the current parent task reduced the makespan of workflow and improved the execution cost in term of minimization in our proposed model. The obtained result of our experiment indicates the robustness of our proposed model CHGA. CHGA is 14.58188%, 11.40224%, 11.75306%, and 9.78841% cheaper than standard ACO, PSO, CEGA, and CLGA, respectively. CHGA's average makespan is 34.73619%, 31.48127%, 5.71553%, and 9.73710% lower than standard ACO, PSO, CEGA, and CLGA, respectively.

DISCUSSION

Because the best schedules take into account both the gain in a sequence of tasks as well as the immediate gain in processing time, we observed that the best meta-heuristic schedules could not be achieved if we adhered to the conventional strategy of selecting processors based only on current task execution time, so we used the PEFT strategy during population initialization, which helps to reduce the makespan. Its capacity to predict the impact of scheduling all child tasks of the current parent task This attribute allows one to make the perfect decision when selecting the perfect virtual machine. We also took into account the time it takes for a VM to boot up and performance fluctuations, both of which have an influence on computation time and execution cost. These statements are verified by the obtained results of our experiments, which indicate the robustness of our proposed model CHGA. CHGA is 14.58188%, 11.40224%, 11.75306%, and 9.78841% cheaper than standard ACO, PSO, CEGA, and CLGA, respectively. CHGA's average makespan is 34.73619%, 31.48127%, 5.71553%, and 9.73710% lower than standard ACO, PSO, CEGA, and CLGA, respectively.

We also applied the greedy strategy during the initialization of the population, which plays an important role in load balancing among VMs. When we used the proposed CHGA, VM4 was overloaded by +26%, while VM3 was underloaded by -12%, which shows our model CHGA is better in load-balancing compared to ACO, PSO, and CEGA.

CONCLUSIONS AND FUTURE WORK

To schedule scientific workflow, we introduced our meta-heuristic, cost-effective, load-balanced hybrid evolutionary method. To balance the load among VMs in a

heterogeneous environment, an effective encoding approach with a greedy strategy is used. We also employed the PEFT technique to make our algorithm more cost-effective. Under a user-defined deadline, we considered three parameters: makespan, computation cost, and load balance, and rigorously tested four types of scientific workflows with varied task sizes. Our experimental results proved that the proposed CHGA algorithm's performance is better than the ACO, PSO, CEGA, and CLGA in minimizing the computing cost and execution time as well as balancing the load among virtual machines. CHGA is 17.48570%, 15.30489%, 11.75306%, and 9.78841% cheaper than standard ACO, PSO, CEGA, and CLGA, respectively. CHGA's average makespan is 34.73619%, 31.48127%, 5.71553%, and 9.73710% lower than standard ACO, PSO, CEGA, and CLGA, respectively. In the future, we will consider the dynamic nature of workflow with the latest meta-heuristic algorithms like Cuckoo search, Firefly, Lion, and Jaya, etc.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

REFERENCES

1. S. Ibrahim, B. He, and H. Jin, "Towards pay-as-you-consume cloud computing," *Proc. 2011 IEEE Int. Conf. Serv. Comput. SCC 2011*, pp. 370–377. doi: 10.1109/SCC.2011.38.
2. R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Futur. Gener. Comput. Syst.*, 25(6), pp. 599–616, 2009. doi: 10.1016/j.future.2008.12.001.
3. A.S. Kulik, A.G. Chukhray, and O.V. Havrylenko, "Information technology for creating intelligent computer programs for training in algorithmic tasks. Part 1: mathematical foundations," *System Research & Information Technologies*, no. 4, 2021. doi: 10.20535/SRIT.2308-8893.2021.4.02
4. S. Singhal and J. Grover, "Hybrid biogeography algorithm for reducing power consumption in cloud computing," *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017*, pp. 121–124. doi: 10.1109/ICACCI.2017.8125827.
5. S. Singhal and J. Patel, "Load balancing scheduling algorithm for concurrent workflow," *Comput. Informatics*, 37(2), pp. 311–326, 2018. doi: 10.4149/cai_2018_2_311.
6. G. Dalin and V. Radhamani, "IRIAL-an improved approach for VM migrations in cloud computing," *International Journal of Advanced Technology and Engineering Exploration, 2018 July 1*, 5(44), pp. 165–171.
7. J. Yu and R. Buyya, "A taxonomy of workflow management systems for Grid computing," *J. Grid Comput.*, 3(3–4), pp. 171–200, 2005. doi: 10.1007/s10723-005-9010-8.
8. D. Malhotra, "An adaptive threshold policy for host overload detection in cloud data centre," *International Journal of Advanced Technology and Engineering Exploration, 2021 Oct 1*, 8(83), pp. 1315–1335.
9. S.K. Bothra and S. Singhal, "Nature-inspired metaheuristic scheduling algorithms in cloud: A systematic review," *Sci. Tech. J. Inf. Technol. Mech. Opt.*, 21(4), pp. 463–472, 2021. doi: 10.17586/2226-1494-2021-21-4-463-472.
10. S.K. Bothra, S. Singhal, and H. Goyal, "Deadline-constrained cost-effective load-balanced improved genetic algorithm for workflow scheduling," *Int. J. Inf. Technol. Web Eng.*, 16(4), pp. 1–34, 2021. doi: 10.4018/IJITWE.2021100101.

11. L.F. Bittencourt, R. Sakellariou, and E.R.M. Madeira, "DAG scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm," *Proc. 18th Euromicro Conf. Parallel, Distrib. Network-Based Process, PDP 2010*, pp. 27–34. doi: 10.1109/PDP.2010.56.
12. V.M. Sineglazov, K.D. Riazanovskiy, and O.I. Chumachenko, "Multicriteria conditional optimization based on genetic algorithms," *System Research & Information Technologies*, no. 3, pp. 89–104, 2020. doi: 10.20535/SRIT.2308-8893.2020.3.07
13. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical Computer Science*, 344, pp. 243–278, 2005. doi: 10.1016/j.tcs.2005.05.020.
14. S.K. Patel and A.K. Sharma, "Improved PSO based job scheduling algorithm for resource management in grid computing," *International Journal of Advanced Technology and Engineering Exploration, 2019 May 1*, 6(54), pp. 152–161.
15. D. Karaboga and B. Akay, "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems," *Appl. Soft Comput. J.*, 11(3), pp. 3021–3031, 2011. doi: 10.1016/j.asoc.2010.12.001.
16. Z. Wu, Z. Ni, and L. Gu, "A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling," *Proceedings of the 2010 International Conference on Computational Intelligence and Security*, pp. 184–188. doi: 10.1109/CIS.2010.46.
17. S. Pandey, L. Wu, S.M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," *Proc. Int. Conf. Adv. Inf. Netw. Appl. AINA, 2010*, pp. 400–407. doi: 10.1109/AINA.2010.31.
18. F. Engineering, "Scheduling Workflow in Cloud Computing Based on Ant Colony Optimization Algorithm," *2013 Sixth International Conference on Business Intelligence and Financial Engineering (BIFE)*, pp. 57–61. doi: 10.1109/BIFE.2013.14.
19. Z. Chen and K. Du, "Deadline Constrained Cloud Computing Resources Scheduling for Cost Optimization based on Dynamic Objective Genetic Algorithm," *Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE, 2015*, pp. 708–714.
20. J. Meena, M. Kumar, and M. Vardhan, "Cost Effective Genetic Algorithm for Workflow Scheduling in Cloud Under Deadline Constraint," *IEEE Access*, 4, pp. 5065–5082, 2016. doi: 10.1109/ACCESS.2016.2593903.
21. M. Mollajafari and H.S. Shahhoseini, "A cost-optimized GA-based heuristic for scheduling time-constrained workflow applications in infrastructure clouds using an innovative feasibility-assured decoding mechanism," *J. Inf. Sci. Eng.*, 32(6), pp. 1541–1560, 2016. doi: 10.1688/JISE.2016.32.6.8.
22. Gupta, V. Gajera, P.K. Jana, and I.S. Member, "An Effective Multi-Objective WorkflowScheduling in Cloud Computing: A PSO based Approach, "*Ninth International Conference on Contemporary Computing*", 2016.
23. Y. Moon, H. Yu, J.M. Gil, and J. Lim, "A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments," *Human-centric Comput. Inf. Sci.*, 2017. doi: 10.1186/s13673-017-0109-2.
24. A. You, M.A.Y. Be, and I. In, "Task scheduling based on ant colony optimization in cloud environment," *AIP Conference Proceedings*, vol. 1834, iss. 1, 040039, 2017. doi: 10.1063/1.4981635.
25. B. Xiang, B. Zhang, and L. Zhang, "Greedy Ant: Ant Colony System-Inspired Workflow Scheduling for Heterogeneous Computing," *IEEE Access*, 2017. doi: 10.1109/ACCESS.2017.2715279.
26. S.K. Sonkar and M.U. Kharat, "Load prediction analysis based on virtual machine execution time using optimal sequencing algorithm in cloud federated environment," *Int. J. Inf. Technol.*, 11(2), pp. 265–275, 2019. doi: 10.1007/s41870-019-00282-1.
27. C. Yan, M.X. Li, and W. Liu, "Application of improved genetic algorithm in function optimization," *J. Inf. Sci. Eng.*, 35(6), pp. 1299–1309, 2019. doi: 10.6688/JISE.201911_35(6).0008.

28. Al-Azzawi, "Evaluation of Genetic Algorithm Optimization in Machine Learning," *J. Inf. Sci. Eng.*, 36(2), 2020.
29. A. Belgacem and K. Beghdad-Bey, "Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost," *Cluster Computing*, 25(1), pp. 579–595, 2022.
30. H. Arabnejad and J.G. Barbosa, "List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table," *IEEE Transactions on Parallel and Distributed Systems*, 25(3), pp. 682–694, 2014.
31. Z. Hill and M. Humphrey, "A quantitative analysis of high performance computing with Amazon's EC2 infrastructure: The death of the local cluster?" *Proc. IEEE/ACM Int. Work. Grid Comput.*, pp. 26–33, 2009. doi: 10.1109/GRID.2009.5353067.
32. *Amazon Elastic Block Store*. Available: https://aws.amazon.com/ebs/. Accessed 22 July, 2020.
33. M. Naderan, "Review methods for breast cancer detection using artificial intelligence and deep learning methods," *System Research & Information Technologies*, no. 1, 2021. doi: 10.20535/SRIT.2308-8893.2021.1.08
34. I.V. Beyko, J.V. Spivak, and O.V. Furtel, "Generalized solutions of optimal control problems," *System Research & Information Technologies*, no. 4, 2020, pp. 104–114. doi: 10.20535/SRIT.2308-8893.2020.4.08.

Received 15.08.2022

INFORMATION ON THE ARTICLE

Sandeep Kumar Bothra, ORCID: 0000-0003-0555-569X, Manipal University Jaipur, (Raj.), India, e-mail: bothrajain@gmail.com

Sunita Singhal, ORCID: 0000-0003-2462-8102, Manipal University Jaipur, (Raj.), India, e-mail: sunita.singhal@jaipur.manipal.edu

Hemlata Goyal, ORCID: 0000-0003-1344-0921, Manipal University Jaipur, (Raj.), India,e-mail: hemlata.goyal@jaipur.manipal.edu

ЕКОНОМІЧНО ЕФЕКТИВНИЙ ГІБРИДНИЙ ГЕНЕТИЧНИЙ АЛГОРИТМ ПЛАНУВАННЯ РОБОЧОГО ПРОЦЕСУ В ХМАРІ / Сандіп Кумар Бовра, Суніта Сінгхал, Хемлата Гоял

Анотація. Хмарні обчислення відіграють значну роль у способі життя кожного, щільно пов'язуючи спільноти, інформацію та торги по всьому світу. Розпізнавання оптимального рішення для планування робочих процесів у хмарі є складною сферою через його NP-жорсткий характер. Запропоновано гібридний метаевристичний економічно ефективний збалансований за навантаженням підхід до планування робочого процесу в гетерогенному середовищі. Модель ґрунтується на генетичному алгоритмі, інтегрованому з прогнозом найбільш раннього часу фінішу (PEFT), щоб мінімізувати makepan. Замість призначення завдання випадковим чином на віртуальній машині застосовуємо жадібну стратегію, яка відводить завдання на віртуальну машину з найменш завантаженим. Після завершення операції мутації перевіряємо обмеження залежності замість кожної операції кросовера, що дає кращий результат. Запропонована модель включає в себе дисперсію продуктивності віртуальної машини, а також затримку придбання, яка поступається мінімальній вартості makepan і computing. Одним з найбільш приголомшливих аспектів економічно ефективного гібридного генетичного алгоритму (CHGA) є його здатність передбачати, створюючи оптимістичну таблицю витрат (OCT), зберігаючи квадратичну складність часу. На основі результатів ретельних експериментів над деякими показниками робочого процесу в реальному світі та всебічного аналізу деяких нещодавно успішних алгоритмів планування отримано висновок, що продуктивність запропонованої CHGA є мелодійною.

Ключові слова: хмарні обчислення, економічно вигідні, генетичний алгоритм, метагевристичний алгоритм, прогнозування раннього часу оброблення, планування робочого процесу.