НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
март–апрель 2023 Том 23 № 2 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
March–April 2023 Vol. 23 No 2 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
март–апрель 2023 Том 23 Номер 2

10.17586/2226-1494-2023-23-2-313-322

Hybrid JAYA algorithm for workflow scheduling in cloud

Sandeep Kumar Bothra[1]✉, Sunita Singhal[2], Hemlata Goyal[3]

[1,2,3] Manipal University Jaipur, Jaipur, Rajasthan, 303007, India

[1] bothrajain@gmail.com✉, https://orcid.org/0000-0003-0555-569X
[2] sunita.singhal@jaipur.manipal.edu, https://orcid.org/0000-0003-2462-8102
[3] hemlata.goyal@jaipur.manipal.edu, https://orcid.org/0000-0003-1344-0921

Abstract
Workflow scheduling and resource provisioning are two of the most critical issues in cloud computing. Developing an optimal workflow scheduling strategy in the heterogeneous cloud environment is extremely difficult due to its NP-complete nature. Various optimization algorithms have been used to schedule the workflow so that users can receive Quality of Service (QoS) from cloud service providers as well as service providers can achieve maximum gain but there is no such model that can simultaneously minimize execution time and cost while balancing the load among virtual machines in a heterogeneous environment using JAYA approach. In this article, we employed the hybrid JAYA algorithm to minimize the computation cost and completion time during workflow scheduling. We considered the heterogeneous cloud computing environment and made an effort to evenly distribute the load among the virtual machines. To achieve our goals, we used the Task Duplication Heterogeneous Earliest Finish Time (HEFT-TD) and Predict Earliest Finish Time (PEFT). The makespan is greatly shortened by HEFT-TD which is based on the Optimistic Cost Table. We used a greedy technique to distribute the workload among Virtual Machines (VMs) in a heterogeneous environment. Greedy approach assigns the upcoming task to a VM which have lowest load. In addition, we also considered performance variation, termination delay, and booting time of virtual machines to achieve our objectives in our proposed model. We used Montage, LIGO, Cybershake, and Epigenomics datasets to experimentally analyze the suggested model in order to validate the concept. Our meticulous experiments show that our hybrid approach outperforms other recent algorithms in minimizing the execution cost and makespan, such as the Cost Effective Genetic Algorithm (CEGA), Cost-effective Load-balanced Genetic Algorithm (CLGA), Cost effective Hybrid Genetic Algorithm (CHGA), and Artificial Bee Colony Algorithm (ABC).

Keywords
JAYA algorithm, workflow scheduling, execution cost, makespan, load balance, cloud computing

For citation: Bothra S.K., Singhal S., Goyal H. Hybrid JAYA algorithm for workflow scheduling in cloud. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2023, vol. 23, no. 2, pp. 313–322. doi: 10.17586/2226-1494-2023-23-2-313-322

УДК 004.75

Гибридный алгоритм JAYA для планирования рабочих процессов в облаке

Сандип Кумар Ботра[1]✉, Сунита Сингхал[2], Хемлата Гоял[3]

[1,2,3] Манипальский университет Джайпура, Джайпур, Раджастхан, 303007, Индия

[1] bothrajain@gmail.com✉, https://orcid.org/0000-0003-0555-569X
[2] sunita.singhal@jaipur.manipal.edu, https://orcid.org/0000-0003-2462-8102
[3] hemlata.goyal@jaipur.manipal.edu, https://orcid.org/0000-0003-1344-0921

Аннотация
Планирование рабочих процессов и предоставление ресурсов — две наиболее важные проблемы облачных вычислений. Разработка оптимальной стратегии планирования рабочих процессов в гетерогенной облачной среде чрезвычайно сложна из-за ее NP-полной природы. При планировании рабочего процесса используются различные алгоритмы оптимизации для получения пользователями качественного обслуживания (Quality of Service, QoS) от поставщиков облачных услуг. При этом поставщики услуг должны получать максимальную выгоду. Сегодня не существует такой модели, которая могла бы одновременно минимизировать время и

© Bothra S.K., Singhal S., Goyal H., 2023

стоимость выполнения работ при балансировке нагрузки между виртуальными машинами в гетерогенной среде с использованием подхода JAYA. В работе предложен гибридный алгоритм JAYA для минимизирования стоимости вычислений и времени выполнения работ при планировании рабочего процесса. Рассмотрена гетерогенная среда облачных вычислений, позволяющая равномерно распределять нагрузку между виртуальными машинами. Для достижения этих целей использованы эвристические подходы Task Duplication Heterogeneous Earliest Finish Time (HEFT-TD) и Predict Earliest Finish Time (PEFT). Длительность выполнения работ значительно сокращается благодаря HEFT-TD, основанному на таблице оптимистичных затрат (Optimistic Cost Table). Для распределения рабочей нагрузки между виртуальными машинами в гетерогенной среде использован жадный алгоритм. Жадный алгоритм назначает предстоящую задачу виртуальной машине с наименьшей нагрузкой. Рассмотрено изменение производительности, задержки завершения и время загрузки виртуальных машин. С целью проверки предложенной концепции для экспериментального анализа представленной модели использованы наборы данных Montage, LIGO, Cybershake и Epigenomics. Выполненные эксперименты показали, что рассмотренный гибридный подход превосходит более ранние алгоритмы по минимизации стоимости и времени его выполнения, такие как Cost Effective Genetic Algorithm (CEGA), Cost-effective Load-balanced Genetic Algorithm (CLGA), Cost effective Hybrid Genetic Algorithm (CHGA) и Artificial Bee Colony Algorithm (ABC).

Ключевые слова
алгоритм JAYA, планирование рабочего процесса, стоимость выполнения, время выполнения, баланс нагрузки, облачные вычисления

Ссылка для цитирования: Ботра С.К., Сингхал С., Гоял Х. Гибридный алгоритм JAYA для планирования рабочих процессов в облаке // Научно-технический вестник информационных технологий, механики и оптики. 2023. Т. 23, № 2. С. 313–322 (на англ. яз.). doi: 10.17586/2226-1494-2023-23-2-313-322

Introduction

Cloud computing is an emerging sector of computing in which a collection of resources is provided to a user as a service rather than as a product. The best thing about these services is that consumers do not need to be aware of the actual locations of the resources or the configurations of these resources that provide the needed service. A workflow submitted by a user to be executed on the cloud is a group of interdependent tasks that are represented by a Directed Acyclic Graph [1]. When workflow scheduling is done in a heterogeneous computing system, the problem becomes more complicated since the processors in the distributed environment may not be similar and require varying amounts of time to complete the same operation. Workflow scheduling and resource provisioning are two of the most critical issues in cloud computing. Developing an optimal workflow scheduling technique in the heterogeneous cloud environment is extremely difficult due to its NP-complete nature [2].

Various optimization algorithms [3] have been used to schedule the workflow so that users can receive Quality of Service (QoS) from cloud service providers. Task scheduling is essential for optimum utilization of cloud resources and also for providing end-users with a QoS [4]. Task scheduling issues come in two flavors: static scheduling and dynamic scheduling. In the static category, all task details, 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, in the dynamic category, such data is not available, and choices are made in real time. Furthermore, static scheduling refers to compile-time scheduling, and dynamic scheduling refers to scheduling at runtime [5].

Heuristic approaches are problem-dependent and frequently too greedy, resulting in their becoming stuck in a local optimum and failing to achieve an optimal solution. When there is partial information or limited computing power, a meta-heuristic is a higher-level technique or heuristic that is used to find, generate, or select a heuristic capable of offering a good solution to an optimization problem.

Due to the poor convergence rate of the meta-heuristic technique, achieving an optimal solution is difficult. As a result, a one-size-fits-all solution will not guarantee optimal resource utilization. That's why, the hybrid nature of the approach is one of the best ways to reach our goal function. So we developed a new model using the JAYA algorithm where seeding is performed using the Task Duplication Heterogeneous Earliest Finish Time (HEFT-TD) [6] and Predict Earliest Finish Time (PEFT) [7] heuristics. We made an effort in this paper to shorten the computation time and cost while still meeting the deadline.

Our goal was to create a hybrid metaheuristic strategy that was effective for decreasing processing time and costs while maintaining load balance amongst Virtual Machines (VMs) under time restrictions. The strategy used in this work employs a HEFT-TD and PEFT strategies during population initialization, which helps with cost cutting and load balancing.

The following is the rest of the article which summarizes prior research in this topic. The implementation and outcomes of the recommended method, as well as a comparison with existing methods, are discussed in further detail. This publication also includes a discussion and conclusion that recommends additional research in this area.

Related Work

The objective of our model in a cloud computing context is to improve the system turnaround time and resource usage. The number of interdependent jobs and available resources in a distributed environment varies dynamically. Various scheduling algorithms developed to obtain optimum solution, some of them heuristic and other are meta-heuristic as both methodologies are integral to maintaining optimal resource scheduling as we mentioned earlier.

Based on the Particle Swarm Optimization (PSO) technique, the authors develop a multi-objective algorithm for scheduling workflow. They use two factors in their innovative method to fulfill their goals: makespan and resource usage as well as a strict encoding technique. Despite the fact that their testing results show that their technique is more resilient than baseline approaches, they fail to account for VMs balancing [8].

The authors [9] use a Pareto distribution to allocate unused VMs in Ant Colony Optimization (ACO) to minimize computation cost and time. They also use a minimal relocation of VMs strategy to boost the effectiveness of their approach in assessing workflow computation time and price; however, as their approach is dependent on a relatively small workflow size, the performance of the algorithm is uncertain.

To schedule the workflow, the authors enhanced PSO. In order to achieve the lowest operational time and cost, they started by regulating the global and local performance of particles using a nonlinear decreasing technique of inertia weight. However, they failed to take into account the dynamic nature of the cloud computing environment [10].

An Artificial Bee Colony (ABC) based algorithm is proposed in the literature [11], with the authors emphasizing the QoS regulations and critical security principles. A hive table is kept in a data center to reduce execution cost, execution time, job migration, and VM load-balancing. They should build a hybrid strategy because the ABC approach alone is unable to manage all of these aspects.

For load balancing during workflow scheduling in the cloud, the author [12] uses the JAYA algorithm. Authors [13] suggested a modified backfilling technique for optimal cloud resource use in which they schedule jobs without a decision-maker to firmness conflicts. To lower the execution cost and duration, a vocalization of the humpback whale optimization algorithm [14] is presented. By using less energy, this technique protects the environment. Authors [15] proposed an efficient method for scheduling work in the cloud utilizing the MAPREDUCE and GA-WOA. The proposed approach consists of stages, such as feature reduction, feature selection, task separation, and task scheduling. They intended to reduce makespan, however they did not consider load balance among virtual machines.

Although the authors [16] integrated execution time and throughput in their model based on Bat algorithm, but they didn't take into account communication time which is a crucial component in reducing execution time and increasing throughput. They were likewise unconcerned about the load balance between the multiple VMs also. This obscurity is eliminated in [17] when the authors tried to optimize the resource allocation for VMs and suggested using the Bat approach to evenly distribute the workload over several VMs. In the paper [18], authors proposed a hybrid method that combined Heterogeneous Earliest Finish Time (HEFT) and Genetic Algorithm (GA) to reduce processing costs and time under budget limitations; however, they were unable to account for VM booting time. In the literature [19], a multi-objective load balancing method, which is based on GA, was proposed; in it delays in acquisition are ignored while cost and time are decreased. In the literature [20], authors described an approach utilizing JAYA in which they only concentrated on minimizing the execution cost and makespan but did not take into account the performance variance and acquisition delay of the VM as well as they did not aware regarding load balance among VMs. To address these issues, we were inspired to implement a hybrid JAYA model "HJA".

After a thorough examination of the literature, we determined the research gap that researchers' applied heuristic methods to workflow scheduling are ineffective due to the NP-hard nature of the problem. Iteration is required for metaheuristic techniques to reach an optimal solution. As a consequence, hybrid metaheuristic approaches outperform traditional metaheuristic techniques in terms of identifying the optimal solution.

Proposed Methodology

Description of JAYA

After a deep literature review, we decided to implement a hybrid model using the JAYA algorithm (HJA) which is based on the heuristics of HEFT-TD and PEFT. Since all evolutionary and swarm cognitive algorithms are unpredictable and necessitate the same governing variables, such as size of population and iteration number, we choose the JAYA [21] technique. In addition to the standard control parameters, various algorithms, such as ACO, PSO, GA, etc., also need their own algorithm-specific parameter settings, such as pheromone value, evaporation rate, cognitive acceleration constant, and crossover rate. The above-mentioned algorithms performance is greatly impacted by the right adjustment of algorithm-specific parameters. Inadequately adjusting algorithm-specific parameters either leads to the best local result or increases processing effort. There are no algorithm-specific parameters needed for JAYA. It was invented by R. Venkata Rao in 2016 [21]. It is a parameter-less algorithm that requires a few iterations to achieve an optimum solution.

$$X^{t+1}_{j,k} = X^{t}_{j,k} + r1^{t}_{j}(X^{t}_{j,best} - |X^{t}_{j,k}|) - r2^{t}_{j}(X^{\ t}_{j,worst} - |X^{t}_{j,k}|). \qquad (1)$$

New solution is obtained by applying the above mentioned eq. (1).

During the t^{th} iteration j^{th} variable of k^{th} solution is update by $X^{t+1}_{j,k}$. Random number $r1$ and $r2$ have range between 0 to 1. During the t^{th} iteration best candidate is $X^{t}_{j,best}$ and worst candidate is $X^{\ t}_{j,worst}$.

Brief introduction of HEFT-TD and PEFT

HEFT-TD [6] is based on the duplication approach, this approach can be used to reduce the cost of communication between two dependent jobs. It is predicated on the concept that communicating across dependent activities running on the same system is completely free. In order to achieve this goal and reduce the cost overhead associated with inter-task communication, this strategy duplicates predecessor tasks.

The PEFT [7] is composed of two stages: the task prioritizing phase, which establishes task priorities, and the processor selection phase, which chooses the best processor for carrying out the current job. This algorithm forecasts by computing an Optimistic Cost Table while maintaining quadratic time complexity. The sum of a node earliest start time and computation time is called the Earliest Finish Time (EFT) of a node on a particular processor.

Computation of Execution Time and Cost

In this study, we focused not just on reducing execution costs while achieving the time constraint, but also managing load among various VMs. We took into account a VMs acquisition delay and performance fluctuation, both of which are crucial factors in reducing computation time in a diversified cloud computing environment.

$$ET_{VM_k}(T_i) = \frac{Size_{T_i}}{Speed_{VM_k}}. \quad (2)$$

Here $ET_{VM_k}(T_i)$ is task execution time which we obtained by dividing the size of the task ($Size_{T_i}$) through the processing speed of the kth VM ($Speed_{VM_k}$) as mentioned in eq. (2).

$$TT_{E_{ij}} = \frac{DataFile_{T_i}}{\beta}. \quad (3)$$

Here $TT_{E_{ij}}$ is the time consumed to transfer the data between tasks scheduled in the different VMs. It is also known as communication time. It can be calculated by using the size of an output data file and average bandwidth β as mentioned in eq. (3) where E_{ij} refers to the edge between parent task T_i to child task T_j, and $DataFile_{T_i}$ is the output file of task T_i.

$$avail(VM_k) = ST_{T_i} + \left\{\frac{ET_{VM_k}(T_i)}{(1 - PerVar)}\right\}, \quad (4)$$

$avail(VM_k)$ indicates when kth VM is ready to execute new task. $PerVar$ is performance variation of VM and ST_{T_i} is the time which is estimated to start the execution as depicted in eq. (4).

$$ST_{T_i} = acq\_delay, \text{ if task is root node.} \quad (5)$$

We used eq. (5) to compute the starting execution time of root node where acq_delay refers to booting time of VM, i.e. 60 sec.

$$ST_{T_i} = \max\{avail\{VM_k\}, \max_{T_p \in pred(T_i)}\{FT_{T_p} + TT_{E_{pi}}\}\}. (6)$$

If the task is not a root node, then ST_{T_i} is computed using eq. (6) where FT_{T_i} is the time which is estimated to finish the execution.

$$FT_{T_i} = ST_{T_i} + \left\{\frac{ET_{VM_k}(T_i)}{(1 - PerVar)}\right\}. \quad (7)$$

By using the eq. (7) we computed the finishing time of task T_i.

$$If\ TET \leq D,\ TET = \max_{T_i \varepsilon W}\{FT(T_i)\}. \quad (8)$$

TET is total execution time which also included the termination delay of last VM which is executed till end of complition. TET is computed using eq. (8).

$$TEC = \sum_{n=1}^{VM_n} C_{type(VM_k)} \times \left[\frac{VM_{no\_ET} - VM_{no\_ST}}{TimeInterval}\right]. \quad (9)$$

If TET does not violate the deadline constraint (D) then Total Execution Cost (TEC) is calculate as given in eq. (9) where $C_{type(VM_k)}$ denotes the cost of execution on VM of k type, while VM_{no_ST} and VM_{no_ET} is the start and end times of VM execution, respectively.

Our experiment takes into account three types of deadline constraints: hard, medium, and soft.

$$Deadline\ D = (\alpha + 1) \times \min ET(Wi). \quad (10)$$

Above, eq. (10) used to calculate the deadline where $\min ET(W_i)$ is the sum of time to start and execute all the tasks of workflow W_i on the fastest VM, and α is a step length whose value is 0.4. The hard range is 0 to 1.2, the medium range is 1.2 to 2.8, and the soft range is 2.8 to 4.4. All above equations are taken from literature [22].

Proposed Algorithm

1. Determine the population size and termination criteria
2. Initialize one-one set of candidate solution using HEFT-TD and PEFT
3. Initialize remaining N-2 population using random technique with greedy approach
4. Compute the fitness of each candidate solutions in the population using fitness function using eq. (8) & eq. (9)
5. Find the best and worst candidate solutions
6. Apply the eq. (1) to all candidate solutions
7. Compute the Finish Time of candidate solution
8. According to eq. (10), check the deadline constraint of candidate solution
9. If any candidate solution does not satisfy deadline constraint according the eq. (8), then go to step 6 otherwise compute TEC using eq. (9)
10. If $(X^{t+1}_{j,k}) \leq (X^{t}_{j,best})$ then
11. $(X^{t}_{j,best}) = (X^{t+1}_{j,k})$
12. End If
13. Repeat steps 5 to 12 until termination criteria satisfied
14. Return best candidate solution

In our JAYA-based hybrid module, the population is initialized using three techniques. One individual is initialized using HEFT-TD, another one using the PEFT approach, and the remaining candidate solutions are generated using a random method with a greedy strategy. These approaches not only minimize the makespan but also reduce the computation cost. The fitness of each candidate is computed using eq. (8) and eq. (9), and the JAYA approach is applied using eq. (1) to find a more optimum solution. If the candidate solution satisfies the deadline constraint criteria, then the total execution cost is computed. These steps are iterated until we receive the optimum solution. Fig. 1 illustrates our model through flowchart to explain each step.

Evaluation of Performance

Experimental Environment

We included various types of workflows as benchmarks like Montage, Cybershake, LIGO, and Epigenomics, with sizes of fifty, hundred, and five hundred tasks.

Montage represents an astronomy application where the majority of nodes focus more on I/O than processing.

The “Flexible Image Transport System” (FITS) format photographs used as input in this astronomy application are output as custom sky mosaics. Cybershake is employed by “Southern California Earthquake Center” to display earthquake risks in a particular area. It produces artificial seismograms. Additionally, this process requires a lot of memory and a central processing unit. Epigenomic is bioinformatics application workflow where the majority of the nodes are CPU-intensive. Changes in human cell gene function are referred to as epigenetic state and they are mapped on a genome-wide scale by an epigenomic approach. “Laser Interferometer Gravitational-wave Observatory” is acronym standing for LIGO. This scientific workflow uses a lot of CPU power and consumes a lot of memory. Numerous things happen in the universe that causes gravitational waves which it detects.

We executed the suggested model HJA in a JAVA-based robust environment and reached at a conclusion after performing each type of workflow 30 times. We examined five different types of VMs according to requirements [22] as illustrated in Table 1. We used the Amazon Elastic Block Store (EBS) average bandwidth of 20 kbps[1]. We employed 500 candidate solutions as size of population and a highest number of iterations of 250 in our experiment [23–26].

In the Fig. 1, i is each individual candidate solution from set of total number of N solutions.

Analysis of Experimental Result

Execution Cost and Makespan Analysis

Fig. 2 and 3 show our overall comparison of the baseline and our suggested HJA. The outcome of our experiment demonstrates the resilience of our suggested model HJA. HJA is 20.16 %, 16.58 %, 14.04 %, and 2.88 % less expensive than ABC, CEGA, CLGA, and CHGA respectively.

HJA has a 39.92 %, 9.05 %, 13.91 %, and 2.82 % shorter average makespan than ABC, CEGA, CLGA, and CHGA respectively.

Deadline and Load Balance Analysis

Table 2 illustrates the hit rate and Fig. 4 demonstrates balance of load among VMs.

If a processor hasn’t any job, the load index measured value as zero, the load index increase according to the job assign to a processor.

$$VMC_i = PE_{num} \times PE_{mips}. \qquad (11)$$

[1] Amazon Elastic Block Store. Available at: https://aws.amazon.com/ebs/ (accessed: 22.07.2020).

Table 1. Details of VMs in our experimental environment

VM Types	Processing Capacity, GFLOPS	ECUs (Cores)	Memory, GB	Disk, GB	Cost /Hour, $
m1.Small	4.4	1(1)	1.7	160	0.04
m1.Large	17.6	4(2)	7.5	850	0.16
m1.Xlarge	35.2	8(4)	15	1690	0.32
c1.Medium	22	5(2)	1.7	350	0.20
c1.Xlarge	88	20(8)	7	1690	0.80

Table 2. Analysis of hit rate under deadline constraint, %

Deadline	Algorithm	Montage	Cybershake	LIGO	Epigenomics
Hard	CHGA	96.30	94.07	93.10	92.30
	ABC	78.10	77.10	79.10	79.12
	HJA	95.62	93.00	92.56	90.15
	CEGA	92.34	88.48	88.50	83.49
	CLGA	95.50	91.48	91.46	88.02
Crunch	CHGA	99.90	99.80	99.74	99.83
	ABC	82.08	80.35	81.03	82.97
	HJA	99.61	99.82	99.60	99.81
	CEGA	99.50	99.62	99.50	99.61
	CLGA	99.51	99.76	99.57	99.74
Soft	CHGA	99.89	99.80	99.77	99.81
	ABC	99.81	99.89	99.81	99.90
	HJA	83.45	82.00	85.78	86.57
	CEGA	99.68	99.70	99.61	99.50
	CLGA	99.69	99.78	99.70	99.71

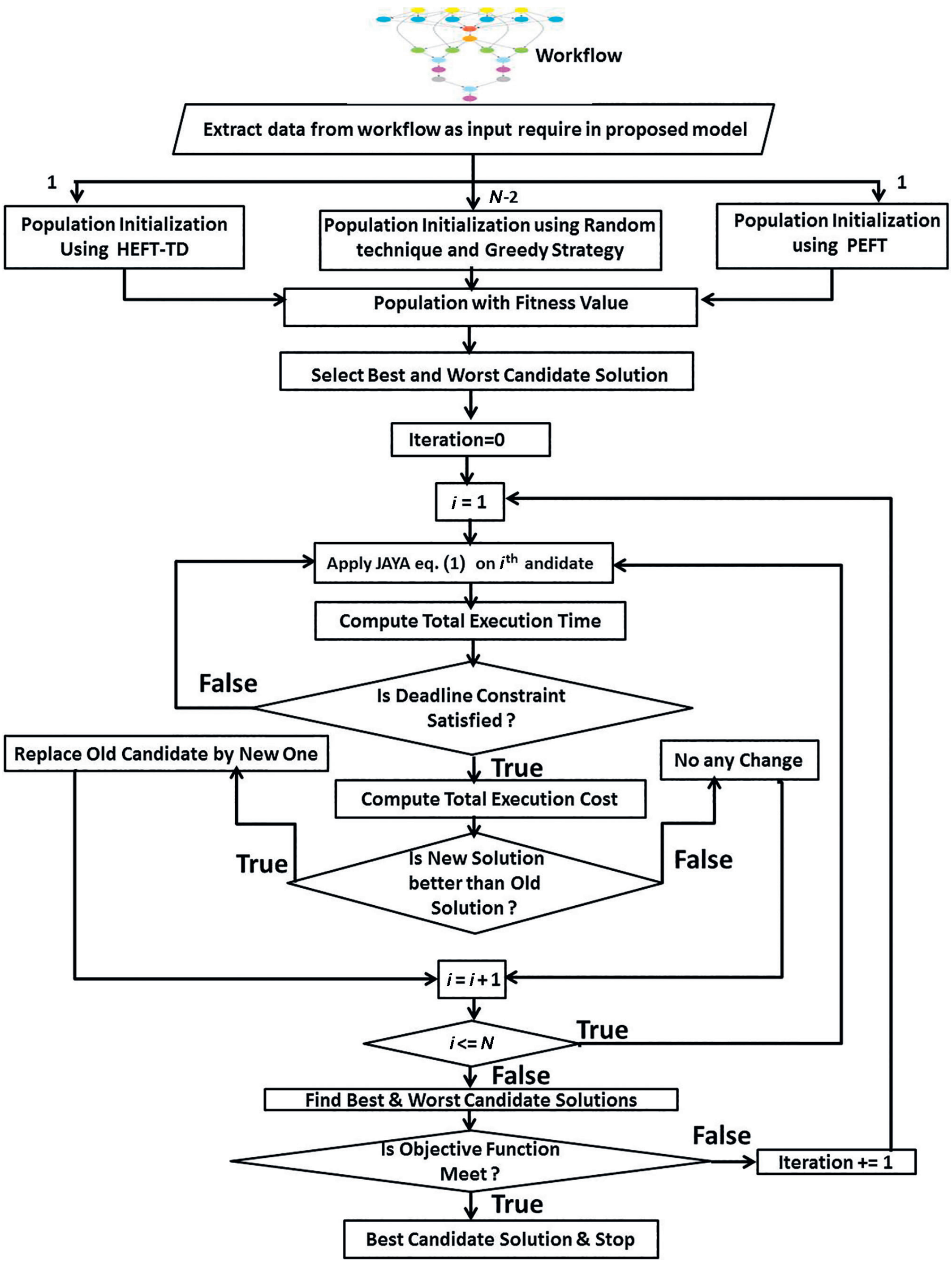
Workflow
Extract data from workflow as input require in proposed model
1
N-2
1
Population Initialization Using HEFT-TD
Population Initialization using Random technique and Greedy Strategy
Population Initialization using PEFT
Population with Fitness Value
Select Best and Worst Candidate Solution
Iteration=0
i = 1
Apply JAYA eq. (1) on i^th andidate
Compute Total Execution Time
False
Is Deadline Constraint Satisfied ?
True
Replace Old Candidate by New One
No any Change
Compute Total Execution Cost
True
Is New Solution better than Old Solution ?
False
i = i + 1
i <= N
True
False
Find Best & Worst Candidate Solutions
Is Objective Function Meet ?
False
Iteration += 1
True
Best Candidate Solution & Stop

Fig. 1. Flowchart of proposed model

We can calculate the capacity of a VMC_i through product of the number of processing elements PE_{num} available in VM_i and its execution speed in MIPS as mentioned in eq. (11). Here, PE_{num} indicates the number of processing elements in a particular VM_i.

$$VMC = \sum_{i=1}^{m} VMC_i, \qquad (12)$$

VMC indicates the execution capacity of all VMs which is equal to the sum of all VMC_i as mentioned in eq. (12) where m is the total number of VMs.

$$L_i = \frac{\sum_{j=1}^{n} T_j}{VMC_i}. \qquad (13)$$

Load L_i is computed using eq. (13). Here, T refers to task length which is divided by VMC_i, and n is the number of total tasks. Task length is expressed as in MI (Million Instruction).

$$TL = \sum_{i=1}^{m} L_i, \qquad (14)$$

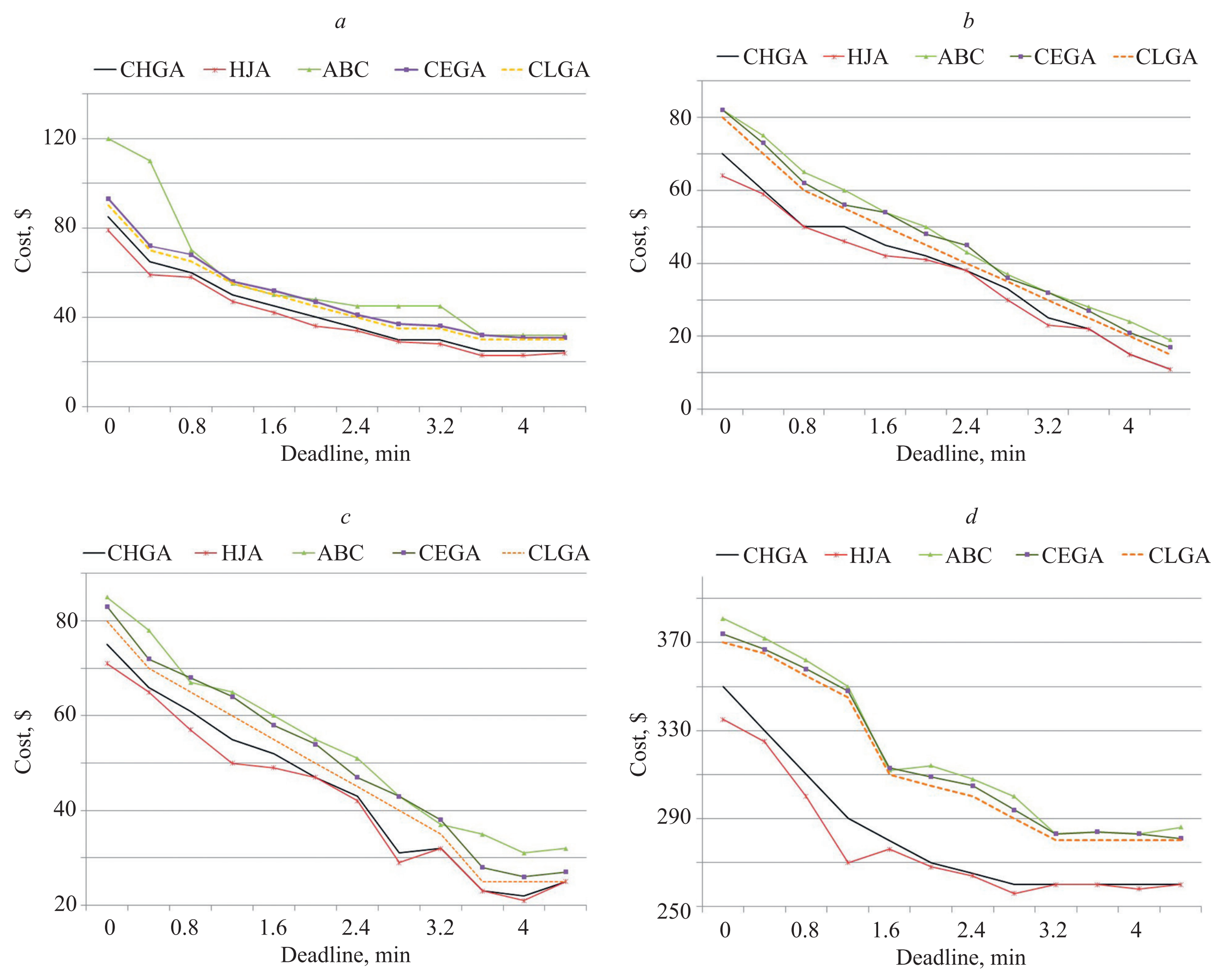

Fig. 2. Analysis of Cost: Montage Workflow (*a*); Cybershake Workflow (*b*); LIGO Workflow (*c*); Epigenomics Workflow (*d*)

Total Load (*TL*) represents the load on all VMs in a given data center which is calculated as given in eq. (14), where *i* refers to the number of VMs till *m*.

$$LCpu = \frac{TL}{VMC}. \quad (15)$$

Load capacity per unit is computed as given in eq. (15).

$$TH_i = LCpu \times VMC_i. \quad (16)$$

The threshold value TH_i of any VM_i is computed as mentioned in eq. (16).

For any VM_i, if the total load on VM_i is less than its threshold value TH_i then it is under-loaded, if it is equal to its TH_i then it is balanced, and if it is greater than its TH_i then it is over-loaded. In our experiment, we examine the variation of load on 5 VMs under the baseline algorithms as well as our proposed HJA. If load is greater than 100 percentages that means machine is overloaded; if less than 100 percentages, i.e., it is under-loaded.

Fig. 4 illustrates how our HJA outperforms competing baseline techniques in terms of load balancing across all 5 VMs. VM_1 has the highest load of 112, and VM_3 has the lowest load of 95. It indicates that VM_3 is under loaded by –5 whereas VM_1 is overloaded by +12. This demonstrates the sturdiness of the HJA model we've suggested.

Discussion

Three strategies are used to initialize the population in our hybrid module based on JAYA. The remaining potential solutions are created using a random method with a greedy strategy, and one individual is initialized using HEFT-TD, another using the PEFT methodology. These methods lower the cost of calculation while simultaneously minimizing the makespan. To arrive at a more ideal option, the fitness of each candidate is calculated and the JAYA technique is used. The overall execution cost is calculated if the candidate solution meets the deadline constraint requirements. We repeat these procedures until we find the optimum solution. The results of our experiment show how robust our proposed model HJA is. ABC, CEGA, CLGA, and CHGA are each 20.16 %, 16.58 %, 14.04 %, and 2.88 % more expensive than HJA. In comparison to ABC, CEGA, CLGA, and CHGA, respectively, HJA has shorter average makespans of 39.92 %, 9.05 %, 13.9131 %, and 2.82 %. In terms of load balancing across all 5 VMs, our HJA performs better than competing baseline

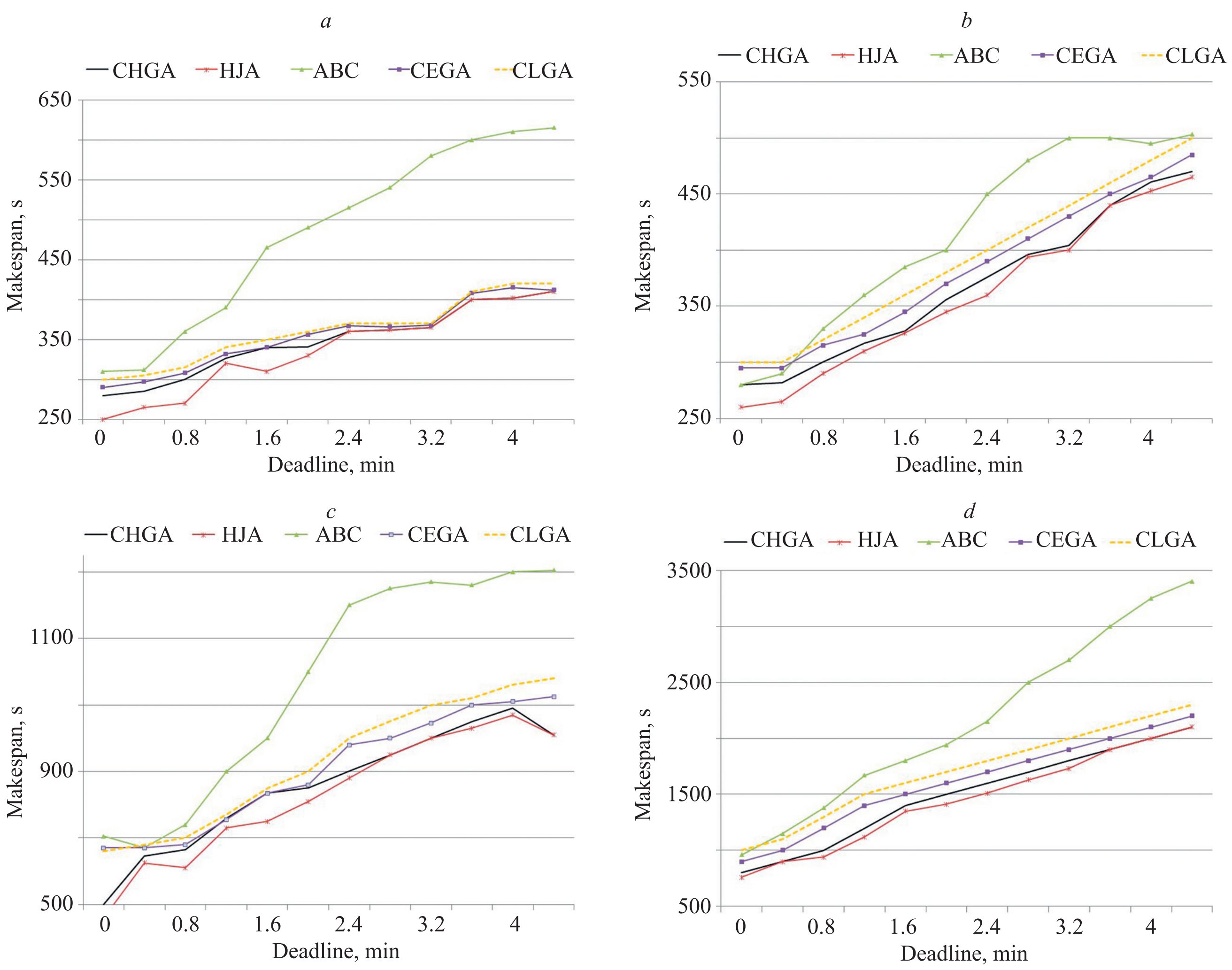

Fig. 3. Analysis of Makespan: Montage Workflow (*a*); Cybershake Workflow (*b*); LIGO Workflow (*c*); Epigenomics Workflow (*d*)

methodologies. The load on VM_1 is the highest at 112, and the load on VM_3 is the lowest at 95. It shows that VM_1 is overloaded by +12, whereas VM_3 is under-loaded by –5. This illustrates how resilient the HJA paradigm we've proposed is.

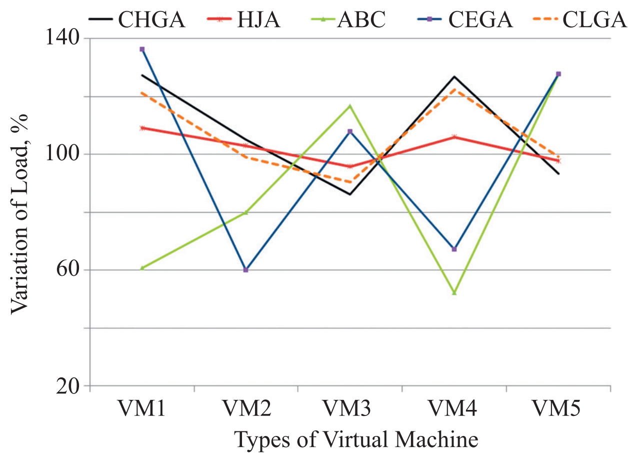

Fig. 4. Balance of load among various VMs

Conclusion and Future Direction

We revealed our meta-heuristic, cost-effective, load-balanced hybrid evolutionary strategy to schedule scientific process. We applied the HEFT-TD and PEFT approaches to minimize the execution cost of workflows and to balance the load across VMs where configuration of VMs is varying. We extensively evaluated four types of workflows which are belong to scientific domains with varying task sizes within a user-defined deadline, taking into account three parameters: makespan, computing cost, and load balance. Our experimental findings have shown that the suggested HJA algorithm outperforms the ABC, CEGA, CLGA, and CHGA in terms of computational cost and execution time as well as load balancing across virtual machines.

In the future, we would like to address the dynamic nature of workflow with varying nature of communication bandwidth and communication delays throughout data centers using the most recent metaheuristics strategies along with machine learning.

References

1. Yu J., Buyya R. A taxonomy of scientific workflow systems for grid computing. *ACM SIGMOD Record*, 2005, vol. 34, no. 3. pp. 44–49. https://doi.org/10.1145/1084805.1084814
2. Singhal S., Grover J. Hybrid biogeography algorithm for reducing power consumption in cloud computing. *Proc. of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 121–124. https://doi.org/10.1109/ICACCI.2017.8125827
3. Bothra S.K., Singhal S. Nature-inspired metaheuristic scheduling algorithms in cloud: A systematic review. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2021, vol. 21, no. 4, pp. 463–472. https://doi.org/10.17586/2226-1494-2021-21-4-463-472
4. Bhatt C., Singhal S. Anatomy of virtual machine placement techniques in cloud. *Lecture Notes in Networks and Systems*, 2022, vol. 373, pp. 609–626. https://doi.org/10.1007/978-981-16-8721-1_59
5. Singhal S., Patel J. Load balancing scheduling algorithm for concurrent workflow. *Computing and Informatics*, 2018, vol. 37, no. 2, pp. 311–326. https://doi.org/10.4149/cai_2018_2_311
6. NoorianTalouki R., Hosseini Shirvani M., Motameni H. A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms. *Journal of King Saud University — Computer and Information Sciences*, 2022, vol. 34, no. 8, pp. 4902–4913. https://doi.org/10.1016/j.jksuci.2021.05.011
7. Arabnejad H., Barbosa J.G. List scheduling algorithm for heterogeneous systems by an optimistic cost table. *IEEE Transactions on Parallel and Distributed Systems*, 2014, vol. 25, no. 3, pp. 682–694. https://doi.org/10.1109/tpds.2013.57
8. Shubham, Gupta R., Gajera V., Jana P.K. An effective multi-objective workflow scheduling in cloud computing: a pso based approach. *Proc. of the 2016 Ninth International Conference on Contemporary Computing (IC3)*, 2016, pp. 1–6. https://doi.org/10.1109/ic3.2016.7880196
9. Lal A., Krishna C.R. Critical path-based ant colony optimization for scientific workflow scheduling in cloud computing under deadline constraint. *Advances in Intelligent Systems and Computing*, 2018, vol. 696, pp. 447–461. https://doi.org/10.1007/978-981-10-7386-1_39
10. Xu R., Wang Y., Cheng Y., Zhu Y., Xie Y., Sani A.S., Yuan D. Improved particle swarm optimization based workflow scheduling in cloud-fog environment. *Lecture Notes in Business Information Processing*, 2019, vol. 342, pp. 337–347. https://doi.org/10.1007/978-3-030-11641-5_27
11. Meshkati J., Safi-Esfahani F. Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing. *The Journal of Supercomputing*, 2019, vol. 75, no. 5, pp. 2455–2496. https://doi.org/10.1007/s11227-018-2626-9
12. Mohanty S., Patra P.K., Ray M., Mohapatra S. An approach for load balancing in cloud computing using JAYA algorithm. *International Journal of Information Technology and Web Engineering*, 2019, vol. 14, no. 1, pp. 27–41. https://doi.org/10.4018/IJITWE.2019010102
13. Nayak S.C., Parida S., Tripathy C., Pattnaik P.K. Dynamic backfilling algorithm to increase resource utilization in cloud computing. *International Journal of Information Technology and Web Engineering*, 2019, vol. 14, no. 1, pp. 1–26. https://doi.org/10.4018/IJITWE.2019010101
14. Masadeh R.M.T., Sharieh A.-A.A., Mahafzah B.A. Humpback whale optimization algorithm based on vocal behavior for task scheduling in cloud computing. *International Journal of Advanced Science and Technology*, 2019, vol. 13, no. 3, pp. 121–140.
15. Sanaj M.S., Prathap P.J. An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment. *Materials Today: Proceedings*, 2021, vol. 37, pp. 3199–3208. https://doi.org/10.1016/j.matpr.2020.09.064
16. Gu Y., Budati C. Energy-aware workflow scheduling and optimization in clouds using bat algorithm. *Future Generation Computer Systems*, 2020, vol. 113, pp. 106–112. https://doi.org/10.1016/j.future.2020.06.031
17. Ullah A., Nawi N.M., Khan M.H. BAT algorithm used for load balancing purpose in cloud computing: an overview. *International Journal of High Performance Computing and Networking*, 2020, vol. 16, no. 1, pp. 43. https://doi.org/10.1504/ijhpcn.2020.110258

Литература

1. Yu J., Buyya R. A taxonomy of scientific workflow systems for grid computing // ACM SIGMOD Record. 2005. V. 34. N 3. P. 44–49. https://doi.org/10.1145/1084805.1084814
2. Singhal S., Grover J. Hybrid biogeography algorithm for reducing power consumption in cloud computing // Proc. of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2017. P. 121–124. https://doi.org/10.1109/ICACCI.2017.8125827
3. Bothra S.K., Singhal S. Nature-inspired metaheuristic scheduling algorithms in cloud: A systematic review // Научно-технический вестник информационных технологий, механики и оптики. 2021. V. 21. N 4. P. 463–472. https://doi.org/10.17586/2226-1494-2021-21-4-463-472
4. Bhatt C., Singhal S. Anatomy of virtual machine placement techniques in cloud // Lecture Notes in Networks and Systems. 2022. V. 373. P. 609–626. https://doi.org/10.1007/978-981-16-8721-1_59
5. Singhal S., Patel J. Load balancing scheduling algorithm for concurrent workflow // Computing and Informatics. 2018. V. 37. N 2. P. 311–326. https://doi.org/10.4149/cai_2018_2_311
6. NoorianTalouki R., Hosseini Shirvani M., Motameni H. A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms // Journal of King Saud University - Computer and Information Sciences. 2022. V. 34. N 8. P. 4902–4913. https://doi.org/10.1016/j.jksuci.2021.05.011
7. Arabnejad H., Barbosa J.G. List scheduling algorithm for heterogeneous systems by an optimistic cost table // IEEE Transactions on Parallel and Distributed Systems. 2014. V. 25. N 3. P. 682–694. https://doi.org/10.1109/tpds.2013.57
8. Shubham, Gupta R., Gajera V., Jana P.K. An effective multi-objective workflow scheduling in cloud computing: a pso based approach // Proc. of the 2016 Ninth International Conference on Contemporary Computing (IC3). 2016. P. 1–6. https://doi.org/10.1109/ic3.2016.7880196
9. Lal A., Krishna C.R. Critical path-based ant colony optimization for scientific workflow scheduling in cloud computing under deadline constraint // Advances in Intelligent Systems and Computing. 2018. V. 696. P. 447–461. https://doi.org/10.1007/978-981-10-7386-1_39
10. Xu R., Wang Y., Cheng Y., Zhu Y., Xie Y., Sani A.S., Yuan D. Improved particle swarm optimization based workflow scheduling in cloud-fog environment // Lecture Notes in Business Information Processing. 2019. V. 342. P. 337–347. https://doi.org/10.1007/978-3-030-11641-5_27
11. Meshkati J., Safi-Esfahani F. Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing // The Journal of Supercomputing. 2019. V. 75. N 5. P. 2455–2496. https://doi.org/10.1007/s11227-018-2626-9
12. Mohanty S., Patra P.K., Ray M., Mohapatra S. An approach for load balancing in cloud computing using JAYA algorithm // International Journal of Information Technology and Web Engineering. 2019. V. 14. N 1. P. 27–41. https://doi.org/10.4018/IJITWE.2019010102
13. Nayak S.C., Parida S., Tripathy C., Pattnaik P.K. Dynamic backfilling algorithm to increase resource utilization in cloud computing // International Journal of Information Technology and Web Engineering. 2019. V. 14. N 1. P. 1–26. https://doi.org/10.4018/IJITWE.2019010101
14. Masadeh R.M.T., Sharieh A.-A.A., Mahafzah B.A. Humpback whale optimization algorithm based on vocal behavior for task scheduling in cloud computing // International Journal of Advanced Science and Technology. 2019. V. 13. N 3. P. 121–140.
15. Sanaj M.S., Prathap P.J. An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment // Materials Today: Proceedings. 2021. V. 37. P. 3199–3208. https://doi.org/10.1016/j.matpr.2020.09.064
16. Gu Y., Budati C. Energy-aware workflow scheduling and optimization in clouds using bat algorithm // Future Generation Computer Systems. 2020. V. 113. P. 106–112. https://doi.org/10.1016/j.future.2020.06.031
17. Ullah A., Nawi N.M., Khan M.H. BAT algorithm used for load balancing purpose in cloud computing: an overview // International Journal of High Performance Computing and Networking. 2020. V. 16. N 1. P. 43. https://doi.org/10.1504/ijhpcn.2020.110258
18. Aziza H., Krichen S. A hybrid genetic algorithm for scientific workflow scheduling in cloud environment // Neural Computing and

18. Aziza H., Krichen S. A hybrid genetic algorithm for scientific workflow scheduling in cloud environment. *Neural Computing and Applications*, 2020, vol. 32, no. 18, pp. 15263–15278. https://doi.org/10.1007/s00521-020-04878-8
19. Sardaraz M., Tahir M. A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing. *International Journal of Distributed Sensor Networks*, 2020, vol. 16, no. 8. https://doi.org/10.1177/1550147720949142
20. Gupta S., Agarwal I., Singh R.S. Workflow scheduling using Jaya algorithm in cloud. *Concurrency and Computation: Practice and Experience*, 2019, vol. 31, no. 17, pp. e5251. https://doi.org/10.1002/cpe.5251
21. Rao R.V. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *International Journal of Industrial Engineering Computations*, 2016, vol. 7, no. 1, pp. 19–34. https://doi.org/10.5267/j.ijiec.2015.8.004
22. Bothra S.K., Singhal S., Goyal H. Deadline-constrained cost-effective load-balanced improved genetic algorithm for workflow scheduling. *International Journal of Information Technology and Web Engineering*, 2021, vol. 16, no. 4, pp. 1–34. https://doi.org/10.4018/IJITWE.2021100101
23. Bothra S.K., Singhal S., Goyal H. Cost effective hybrid genetic algorithm for workflow scheduling in cloud. *System Research and Information Technologies*, 2022, vol. 3, pp. 121–138. https://doi.org/10.20535/srit.2308-8893.2022.3.08
24. Golosov P.E., Gostev I.M. Cloud computing simulation model with a sporadic mechanism of parallel task solving control. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2022, vol. 22, no. 2, pp. 269–278. (in Russian). https://doi.org/10.17586/2226-1494-2022-22-2-269-278
25. Arkhipov I.V. Genetic algorithm application for multi-criteria scheduling problem. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2015, vol. 15, no. 3, pp. 525–531. (in Russian). https://doi.org/10.17586/2226-1494-2015-15-3-525-531
26. Becker M., Gatchin Y.A., Karmanovskiy N.S., Terentiev A.O., Fyodorov D.Y. Information security in cloud computing: problems and prospects. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2011, vol. 11, no. 1, pp. 97–102. (in Russian)

Applications. 2020. V. 32. N 18. P. 15263–15278. https://doi.org/10.1007/s00521-020-04878-8
19. Sardaraz M., Tahir M. A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing // International Journal of Distributed Sensor Networks. 2020. V. 16. N 8. https://doi.org/10.1177/1550147720949142
20. Gupta S., Agarwal I., Singh R.S. Workflow scheduling using Jaya algorithm in cloud // Concurrency and Computation: Practice and Experience. 2019. V. 31. N 17. P. e5251. https://doi.org/10.1002/cpe.5251
21. Rao R.V. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems // International Journal of Industrial Engineering Computations. 2016. V. 7. N 1. P. 19–34. https://doi.org/10.5267/j.ijiec.2015.8.004
22. Bothra S.K., Singhal S., Goyal H. Deadline-constrained cost-effective load-balanced improved genetic algorithm for workflow scheduling // International Journal of Information Technology and Web Engineering. 2021. V. 16. N 4. P. 1–34. https://doi.org/10.4018/IJITWE.2021100101
23. Bothra S.K., Singhal S., Goyal H. Cost effective hybrid genetic algorithm for workflow scheduling in cloud // System Research and Information Technologies. 2022. V. 3. P. 121–138. https://doi.org/10.20535/srit.2308-8893.2022.3.08
24. Голосов П.Е., Гостев И.М. Имитационная модель облачных вычислений со спорадическим механизмом управления параллельным решением задач // Научно-технический вестник информационных технологий, механики и оптики. 2022. Т. 22. № 2. С. 269–278. https://doi.org/10.17586/2226-1494-2022-22-2-269-278
25. Архипов И.В. Применение генетического алгоритма для многокритериальной задачи календарного планирования // Научно-технический вестник информационных технологий, механики и оптики. 2015. Т. 15. № 3. С. 525–531. https://doi.org/10.17586/2226-1494-2015-15-3-525-531
26. Беккер М.Я., Гатчин Ю.А., Кармановский Н.С., Терентьев А.О., Федоров Д.Ю. Информационная безопасность при облачных вычислениях: проблемы и перспективы // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2011. Т. 11. № 1. С. 97–102.

Authors

Sandeep K. Bothra — Researcher, Manipal University Jaipur, Jaipur, Rajasthan, 303007, India, sc 57279416100, https://orcid.org/0000-0003-0555-569X, bothrajain@gmail.com

Sunita Singhal — PhD, Associate Professor, Manipal University Jaipur, Jaipur, Rajasthan, 303007, India, sc 57194063002, https://orcid.org/0000-0003-2462-8102, sunita.singhal@jaipur.manipal.edu

Hemlata Goyal — PhD, Assistant Professor, Manipal University Jaipur, Jaipur, Rajasthan, 303007, India, sc 57202709665, https://orcid.org/0000-0003-1344-0921, hemlata.goyal@jaipur.manipal.edu

Авторы

Ботра Сандип Кумар — исследователь, Манипальский университет Джайпура, Джайпур, Раджастхан, 303007, Индия, sc 57279416100, https://orcid.org/0000-0003-0555-569X, bothrajain@gmail.com

Сингхал Сунита — PhD, доцент, Манипальский университет Джайпура, Джайпур, Раджастхан, 303007, Индия, sc 57194063002, https://orcid.org/0000-0003-2462-8102, sunita.singhal@jaipur.manipal.edu

Гоял Хемлата — PhD, доцент, Манипальский университет Джайпура, Джайпур, Раджастхан, 303007, Индия, sc 57202709665, https://orcid.org/0000-0003-1344-0921, hemlata.goyal@jaipur.manipal.edu

Received 25.07.2022
Approved after reviewing 11.01.2023
Accepted 14.03.2023

Статья поступила в редакцию 25.07.2022
Одобрена после рецензирования 11.01.2023
Принята к печати 14.03.2023

Работа доступна по лицензии
Creative Commons
«Attribution-NonCommercial»