IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-12, Dec 2019 E-ISSN: 2347-2693

Deadline Sensitive Lease Scheduling Using Hungarian Genetic Algorithm in Cloud Computing Environment

Duraksha Ali[1*], Manoj Kumar Gupta[2]

[1,2]School of Computer Science and engineering, Shri Mata Vaishno Devi University, Katra, India

[*]*Corresponding Author: 17mms003@smvdu.ac.in, Tel.: +91-9149443328*

DOI: https://doi.org/10.26438/ijcse/v7i12.715 | Available online at: www.ijcseonline.org

Accepted: 29/Nov/2019, Published: 31/Dec/2019

***Abstract*:** OpenNebula, a cloud platform handles a variety of leases employing scheduler, Haizea and majority of them are deadline-sensitive in real time. As existing Backfilling AHP model for deadline-sensitive lease scheduling suffers from lease rejection and do not scrutinize the estimations for waiting leases. In our proposed work, to overcome this pitfall we have devised Hungarian-Genetic Algorithm (HGA). Time Estimations for leases are performed using optimized Hungarian Algorithm to optimally render resources to available leases but it executes boundlessly. Thus, it's blended with Genetic Algorithm to set bounds to it by utilizing fitness function. Output of HGA is a scheduling structure with optimal lease combination which consumes minimum time. Finally HGA is compared with Backfilling AHP model and HGA schedules greater quota of leases and minimizes lease ostracism comparatively. Also proposed model works fine on increasing number of leases as computational time is not directly proportional to number of leases scheduled.

Keywords: Deadline sensitive, Resource allocation, Leases, Lease scheduling, Cloud computing

I. INTRODUCTION

Cloud computing presents a most promising, low capital distributed computing technology and eScience infrastructure on which the research association has recently ventured into [1]. It renders numerous types of services and information with a contemporary vision of Telcos and Infotech in elastic "pay-as-you-go" model which is in full swing nowadays [2].

One of the most critical issues in cloud computing is scheduling as plethora of virtualized assets are being utilized per task and Service provider needs to deliver cloud resources in best possible manner for its efficient performance i.e productive resource utilization, minimizing cost and queue waiting time [3], [4]. So the main motive of scheduling is the mapping of tasks to the suitable resources which optimizes the objective [5], [6].

The deadline is the time before which a lease, task or service must be delivered or must terminate its execution. Varied cloud resources render varied intensities of performance on the basis of various pricing models. Usually high-speed resources are costly comparatively thus there is a tradeoff between resource cost and execution time [7].

A lease is a kind of agreement in which one party concurs to deliver a collection of computational resources to other party. OpenNebula is an open source platform for cloud computing that handle and manage virtual machines, data centers and cloud resources (both local infrastructures and external pool of public cloud resources) by employing the lease scheduler, Haizea to amplify its scheduling capabilities while managing various kinds of leases including deadline sensitive leases [8], [9].From Haizea, hardware and software resources are requested by cloud user as a lease.

II. RELATED WORK

Deadline sensitive scheduling

A variety of algorithms have been enhanced to achieve the requirements of cloud computing. An advanced work performed in scheduling deadline based approaches is as follows:

Scientific workflows with task replication [10] make use of task replication to diminish the effects of performance fluctuations rendered in cloud computing environment by utilizing the budget plethora and idle resources. Motive of task replication is hiking of contingency policies to rectify obstructions arising due to the performance variabilities and incorrect assessment of task execution time so improves the cloud performance rather than fault tolerance [11].

Resource provisioning for Data-intensive applications approach [12] is pertinent to prodigious amounts of computing power so surplus cloud resources are added to the pool of private cloud infrastructure from public cloud

especially on requirement of QoS when single private organization has restricted computational capacity. It employs a hybrid cloud middleware viz. Aneka to handle the cloud bursting model to gather and deliver resources from external and local infrastructure seamlessly [13], [14]. Deadline sensitive lease scheduling using AHP [15] schedules greater number of leases within the deadline restrictions and puts a stop to lease ostracism by implementing Backfilling Algorithm incorporated with a Multi Criteria Decision Maker (MCDM) viz. AHP (Analytical Hierarchy Process). Apart from AHP, other MCDM's can also be employed like TOPSIS, PROMETHEE, VIKOR, ELECTRE, WSM, OWA etc. [16], [17], [18] present the outranking and selection methods for choosing the optimal solution among various candidates in multiple criteria analysis using PROMETHEE. AHP employs DHT (Decision Hierarchy Tree) to optimize the objective which is to schedule a lease based on the alternatives (similar leases) and criterions (parameters).

Grouped task scheduling [19] classifies tasks into a set of clusters based on several task attributes such as task size, task type, user type and task latency with the motive to minimize latency and execution time while meeting deadlines too. It works in two stages, one is to determine which cluster to be scheduled initially and other is to determine which task inside a chosen cluster to be scheduled initially.

Pair-based task scheduling [20] pairs tasks hailing from two different categories prior to task scheduling to minimize total layover time. Task mapping is performed primarily to several clouds and ultimately assigned to the most convenient cloud. It evaluates column opportunity matrix (COM) and row opportunity matrix (ROW) and utilizes lease time and converse lease time for task scheduling.

CEDA scheduling for workflow applications [21] estimates upward rank of tasks in the workflow, prefers the task of highest rank and allots to the cheapest VM instance while pondering the VM acquisition period. CEDA aims to minimize total economic cost and execution time for workflows while fulfilling deadlines and favors an already active VM instance with unconsumed period of its charge time interval sufficient to implement the task prior to its latest finish time to avoid launching new VM instances and extra cost overhead. Resource scheduling and provisioning involve the selection and provisioning of resources for scientific workflows while organizing tasks into a schedule to map them to the optimal resources within deadlines [22]. It is based on PSO (particle swarm optimization) algorithm and involves encoding of problem (solution representation) which is meant to evaluate the dimension of a particle (workflows) and fitness function is optimized based on the objective.

For Multi-tenant cloud environment, an approach of Workflow scheduling is proposed [1] that presents CWSA (Cloud-based Workflow Scheduling Algorithm) to cater the complex issue of resource management in Multi-tenancy. It maximizes resource utilization by taking advantage of idle slots thus schedules greater number of tasks within deadline restrictions leading to reduced makespan and also minimizes the tardiness, execution time of workflows and cost of workflows.

III. PRELIMINARIES

A. Introduction to Hungarian Algorithm

Hungarian algorithm (also named as Reduced Matrix method or Flood's technique) devised by Kuhn [23] provides the first effective mode of obtaining an optimum solution without comparing directly or indirectly every single option. However, the basic work of this algorithm was originally obtained from the works of König and Egerváry (Two Hungarian mathematicians) [23], [24]. It employs the concept of Matrix Reduction in which we subtract and add the suitable numbers to the cost matrix to reduce it to the opportunity cost matrix indicating corresponding penalties related to allocating any lease to a scheduler. If we are capable of turning the matrix down to a level of having not less than one element, zero in every column and row, then there is the probability of achieving optimal assignments [25], [26], [27].

B. Introduction to Genetic Algorithm

Genetic Algorithm (GA) is a category of metaheuristic technique employed to solve complex space search problems because of its capability of recognizing the global optimum [28]. GA possesses an initial population and begins with a set of feasible solutions. Every chromosome has a gene string encoding a particular solution. In GA, fittest chromosomes are chosen, amalgamating them to generate a new final robust solution. In general, the effectiveness of GA is determined by the selection of genetic operators (Selection, Crossover and Mutation) and related criterions [29]. First step is Selection operator whose main motive is to select chromosomes to generate upcoming population and the commonly applied selection method is roulette wheel in which a section of the wheel is assigned to each chromosome based on its fitness function. Then in Crossover, genes are split and amalgamated between two chosen chromosomes based on the predetermined probability. The third step, Mutation changes the values of randomly chosen genes from a chromosome based on another predetermined probability. Furthermore, the fittest chromosomes are duplicated and sent directly to next population. When GA fulfills the selected goal, it finally terminates [30], [31], [32].

GA constitutes of two key modules viz. algorithmic flows (iterative approach to produce and choose chromosomes for

achieving superior quality solutions) and chromosome representation (designing a solution). Various research studies are focusing on the growth of latest complete chromosome representation based algorithmic flows. All the solutions and the decisions pertaining to the development of a solution are modeled comprehensively and there is invertible and one-to-one mapping between solution space and chromosome space but leading to an inefficient search because of production of prodigious poor quality load imbalanced solutions. In case of incomplete chromosome representation, the mapping is not invertible thus it gives rise to a new way of producing new chromosomes from solutions called shadowing method [28].

IV. PROPOSED WORK

A. *Problem Statement*

For scheduling deadline-sensitive leases in Cloud Computing environment, an existing Backfilling Algorithm is employed to backfill the smaller leases to the idle time slots to schedule greater number of leases but this algorithm does not work well when similar kind of leases occur in cloud environment and this pitfall was removed by incorporating Backfilling Algorithm with a Multi Criteria Decision Maker (MCDM) namely AHP (Analytical Hierarchy Process). Still Backfilling Algorithm does not consider the evaluations for the leases in waiting. Thus to overcome this issue we apply Hungarian algorithm which considers the time estimations of the waiting leases as well but it operates endlessly without any bounds.

Therefore the challenge here is to devise an approach that can overcome above problem. For this we developed an algorithm called Hungarian-Genetic Algorithm (HGA) by incorporating Optimized Hungarian Algorithm with Genetic Algorithm to keep it within limits by utilizing Fitness Function.

B. Proposed Scheme (Hungarian-Genetic Algorithm)

In this section, we have proposed an approach called Hungarian-Genetic Algorithm (HGA) to enhance existing Backfilling Algorithm for scheduling deadline-sensitive leases in optimal time. Our proposed work, HGA (Hungarian-Genetic Algorithm) can be better explained in the following steps:

Step1- Initialization:
When several deadline- sensitive leases arrive in Cloud Computing environment for the purpose of scheduling then an equal number of scheduler threads are created.

Step2- Matrix Formulation:
Initially lease data is preprocessed i.e we extract the lease parameter (Execution Time) of all the leases on each scheduler thread and set the values in the form of a square matrix. We begin with providing the Square Matrix as an input to the Genetic Algorithm and set it as Initial Population.

Step3- Implementation of Optimized Hungarian Algorithm:
We implement Optimized Hungarian Algorithm (OHA) to the Initial Population for computing the Time Estimations of all the leases on the Scheduler threads and at the end of this algorithm we obtain Total Time of execution of the optimum lease combination. The working of this algorithm is explained in the section 4.5.

Step4- Set Fitness Function:
After obtaining the Total Time of execution of all the leases, set the Total Time as Fitness Function to to determine how "suitable" and "fit" a particular solution is with respect to the set objective or the problem in consideration.

Step5- Permutations:
After acquiring the Fitness Function, we perform Crossover in which we calculate the total possible Permutations of all the leases on each scheduling thread as the potential combinations for next generations. For the next generation, again we compute Time Evaluations of the Square Matrix using Optimized Hungarian Algorithm and Total Time of execution is obtained.

Step6- Comparison and Decision Making:
Now we compare this current value of Total time with that of the Fitness Function. We perform Decision Making here for the optimized solution. If current value is better than the Fitness Function (i.e if current value <= Fitness Function) then set the current value of Execution Time as new Fitness Function, otherwise discard the current value and resume the value of Fitness function. After obtaining new Fitness Function, repeat again with taking next combination as crossover for next generation, calculating Time Estimations and then comparing until 10 Generations. After 10 generations we come up with an optimized output having feasible combination with least Execution Time of all the leases, thus we can schedule greater number of deadline sensitive leases and that too within the deadline constraints.

Step7- Computation of Number of Leases Scheduled within a Deadline Generation:
For the rigid deadlines, after getting the optimized scheduling structure we can compare the optimized output (after 10 generations) with that of a set deadline generation by comparing the average number of tasks to be scheduled by which we can determine the number of leases scheduled within a set deadline generation in case if the deadlines are hard ones.

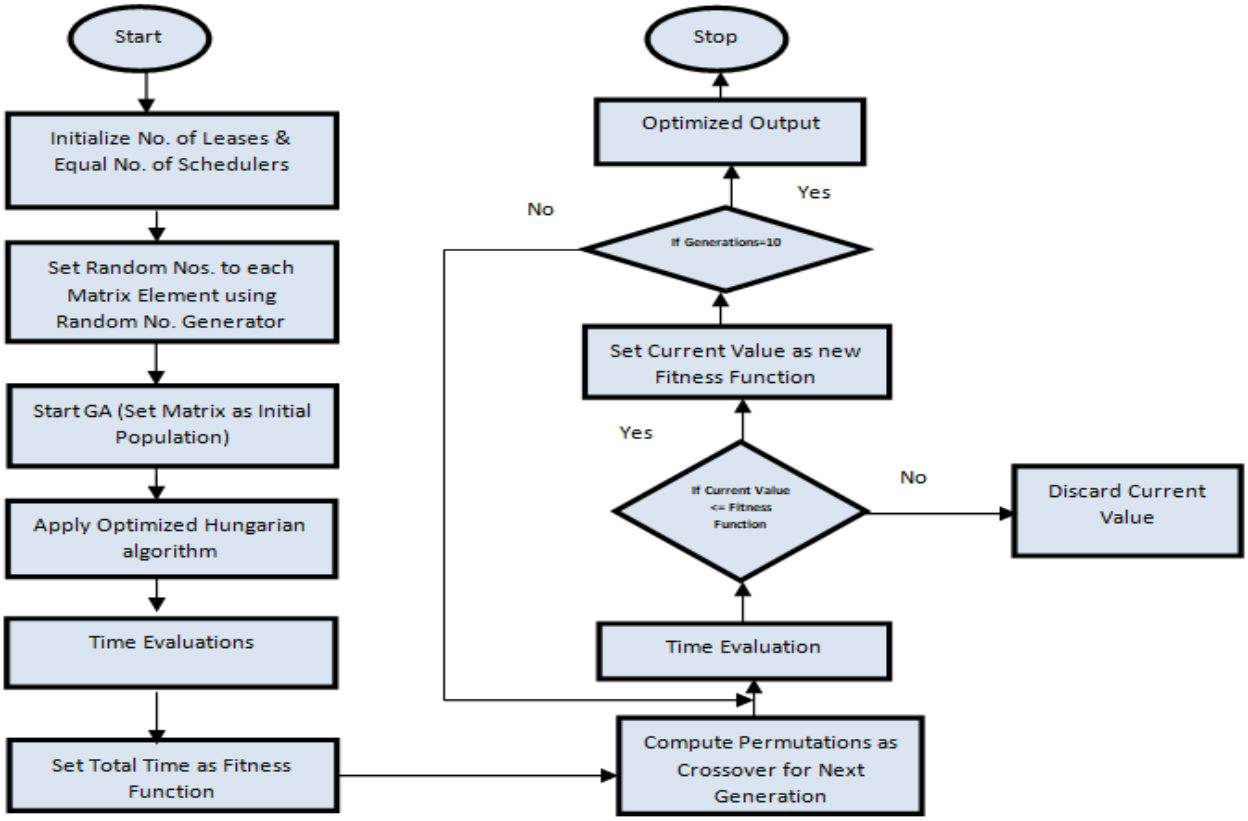

Figure 1.Flowchart of HGA

C. Flowchart

The Flowchart of the proposed approach is shown in the figure1. The algorithm begins with initializing the Number of leases say "n" and an equal number of Scheduler Threads are created say "n". Using a Random Number Generator, execution time of all the leases on all the scheduler threads is randomly set. These random numbers are placed at each matrix element constructing a Square Matrix of size n×n.

Genetic Algorithm (GA) begins with setting square matrix as Initial Population. In next step, Optimized Hungarian Algorithm is applied to find the Time Estimations of all the leases over all the scheduler threads and we obtain total Time of execution of the leases over all the Thread. In next step, set Total Time as Fitness Function as a single figure of merit to determine how close a particular design solution is to accomplishing the set goals. Further in Crossover, compute the Permutations of the number of leases available to act as a combination for the next Generations. For next Generation calculate the Time Evaluations using Optimized Hungarian Algorithm and Total Time of execution is obtained. This Current Value is compared with the Fitness Function. If Current Value is less than Fitness Function then set Current value as new Fitness Function, else discard the current value. It will now draw next possible Combination for Crossover to obtain next Generation and this process will repeatedly loop around upto ten generations. Finally an optimized solution is achieved in the form the lease combination having least Execution Time.

D. *Illustration*

In this section, we will explain the HGA (Hungarian-Genetic Algorithm) with the help of an example. We will begin with the number of leases 'n' say n=3 and automatically n=3 scheduler threads will be generated, constructing an n×n Square Matrix. By employing a Random Number Generator, we will randomly assign random numbers to each Matrix Element. Every entry in the Matrix will indicate the time of execution of a lease on a particular scheduler. Suppose we generate the following 3×3 matrix shown in figure2:

	ST1	ST2	ST3
Lease1 (L1)	4	5	4
Lease2 (L2)	4	8	7
Lease3 (L3)	7	5	5

Figure2. Matrix (*Initial Population*)

Here the lease combination initially is: [1, 2, 3] and to further obtain the combinations of Time Estimations for Leases (L1, L2, L3) over the Scheduler Threads (ST1, ST2,

ST3) we apply Optimized Hungarian Algorithm. In above figure, L1 requires 4 seconds to complete its Execution on ST1, 5 seconds of Execution Time on ST2 and an Execution Time of 4 seconds on ST3 and similarly for lease2 and lease3. In Optimized Hungarian Algorithm, initially Scheduler Thread, ST1 executes its lease then ST2 executes and finally ST3 executes its lease.

	ST1	ST2	ST3
L1	4	5+4=9	4+9=13
L2	4+4=8	8+0=8	7+4=11
L3	7+8=15	5+0=5	5+0=5

Figure3. Time Estimation of Optimized Hungarian Algorithm

Figure3. shows the Time Evaluations of leases. For lease1 (L1) to be scheduled on Scheduler Thread1 (ST1), it needs to wait for no other lease or thread thus completes its execution in just 4 seconds. For L1 to be scheduled on ST2, it has to wait for scheduler1's lease1 (ST1's L1) i.e. for 4 seconds of Waiting time shown by underlined value in figure, thus it will complete its execution in total of 9 seconds. For L1 to be scheduled over ST3, it has to wait for 9 seconds and completes execution after 13 seconds. Similarly for L2 to be scheduled over ST1, it will have to wait for L1 to complete its execution thus it will complete its execution after 8 seconds. For L2 to be scheduled over ST2, it can start its execution only after 9 seconds (Execution time of L1 over ST2) but will have to check if ST1 completed its execution or not, if ST1 already done, then it has zero Waiting time else some Waiting time. Similarly Time is estimated for all the elements using same method. Scheduled time for all the scheduler threads is: ST1 = 15, ST2 = 22, ST3 = 29 and Total Time is 66.

In case of rigid deadlines, we compare optimized scheduling output that is obtained after 10 generations with that of a set deadline generation by comparing the average number of tasks to be scheduled to determine the number of leases scheduled within a set deadline generation.

V. RESULT AND DISCUSSIONS

The proposed methodology of deadline-sensitive Lease scheduling using HGA (Hungarian Genetic Algorithm) in the cloud computing environment is deployed using the Java programming language by using NetBeans 8.0 as IDE. Proposed model uses the windows machine with a processor of Core i3 and primary memory of 4GB. Some experiments are being conducted to measure the impact of the proposed mode with the other existed technologies.

A. Experiment Number 1: Number of Leases Scheduled and rejected (when deadline is adjusted)

The proposed model uses the Hungarian and genetic algorithm to schedule the leases based on the fact of one lease assigned per single scheduler. Here the proposed system improves the Hungarian model by blending it with the traditional genetic algorithm. When the proposed model of HGA is compared with that of the basic Backfilling algorithm for scheduling the leases the results are totally above the Backfilling algorithm and we found our approach outperforming it.

Figure4 and table1 show the number of leases scheduled and rejected by Backfilling algorithm, Backfilling AHP approach and proposed HGA when deadlines are adjusted. Comparatively our proposed approach, HGA schedules all the leases when deadlines are adjusted.

Table1. Number of Leases scheduled by Existing Approaches [15] and Proposed technique HGA

Number of leases scheduled and rejected when deadline is adjusted (Soft Deadlines)							
Experiment No.	**Number of Leases**	**Backfilling Algorithm**		**Backfilling AHP Approach**		**Proposed HGA**	
		No. of Leases Scheduled	**No. of Leases Rejected**	**No. of Leases Scheduled**	**No. of Leases Rejected**	**No. of Leases Scheduled**	**No. of Leases Rejected**
1	5	4	1	5	0	5	0
2	7	6	1	7	0	7	0
3	10	8	2	10	0	10	0
4	15	12	3	15	0	15	0
5	20	16	4	20	0	20	0
6	30	26	4	30	0	30	0
7	50	43	7	50	0	50	0

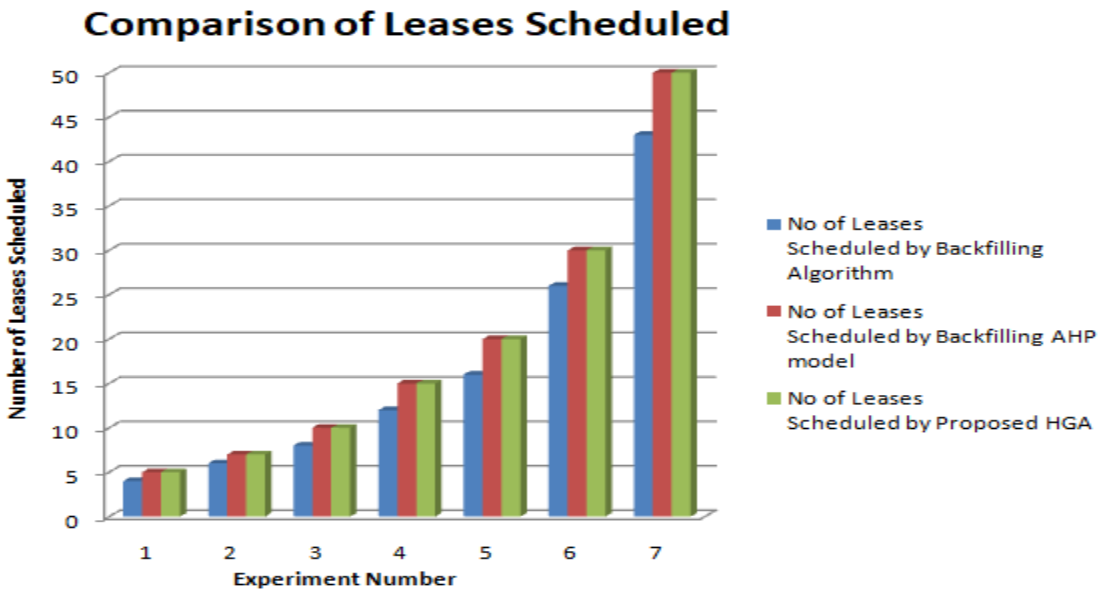

Figure4. Number of Leases scheduled by Backfilling Algorithm, Backfilling AHP approach [15] and Proposed HGA

B. Experiment Number 2: Number of Leases Scheduled and rejected (when deadline is not adjusted i.e. Rigid Deadlines)

When we compare our proposed approach of HGA with that of the basic Backfilling algorithm and Backfilling AHP model for scheduling the leases, the results are totally above them and we found our approach outperforming them in terms of number of leases scheduled when the set deadline is not adjusted.

Figure5 and table2 show the number of leases scheduled and rejected by Backfilling algorithm, Backfilling AHP approach and proposed HGA when the set deadlines are not adjusted (Rigid Deadlines).

As AHP model uses the backfilling technique where scheduling process always seek to the available resources of the past to be released, Whereas the proposed model uses the Improved Hungarian model which uses the resources systematically without colliding with the given scenario.

C. Experiment Number 3: Time Taken for Scheduling

When computation time in milliseconds is measured with that of increasing number of leases, the experiments show that the number of leases scheduled is not directly proportional to the measured computation time as shown in the figure6 and table3. This indicates that proposed model works fine on increasing of the number of leases.

Table2.Number of Leases scheduled by Backfilling Algorithm, Backfilling AHP approach [15] and Proposed HGA

Number of leases scheduled and rejected when deadline is not adjusted (Rigid Deadlines)								
Experiment No.	**Number of Leases**	**Deadline Generation**	**Backfilling Algorithm**		**Backfilling AHP Approach**		**Proposed HGA**	
			No. of Leases Scheduled	**No. of Leases Rejected**	**No. of Leases Scheduled**	**No. of Leases Rejected**	**No. of Leases Scheduled**	**No. of Leases Rejected**
1	12	6	8	4	11	1	11	1
2	20	7	8	12	10	10	13	7
3	25	7	10	15	12	13	16	9
4	30	8	11	19	13	17	17	13
5	40	6	17	23	19	21	24	16
6	44	7	23	21	27	17	31	13

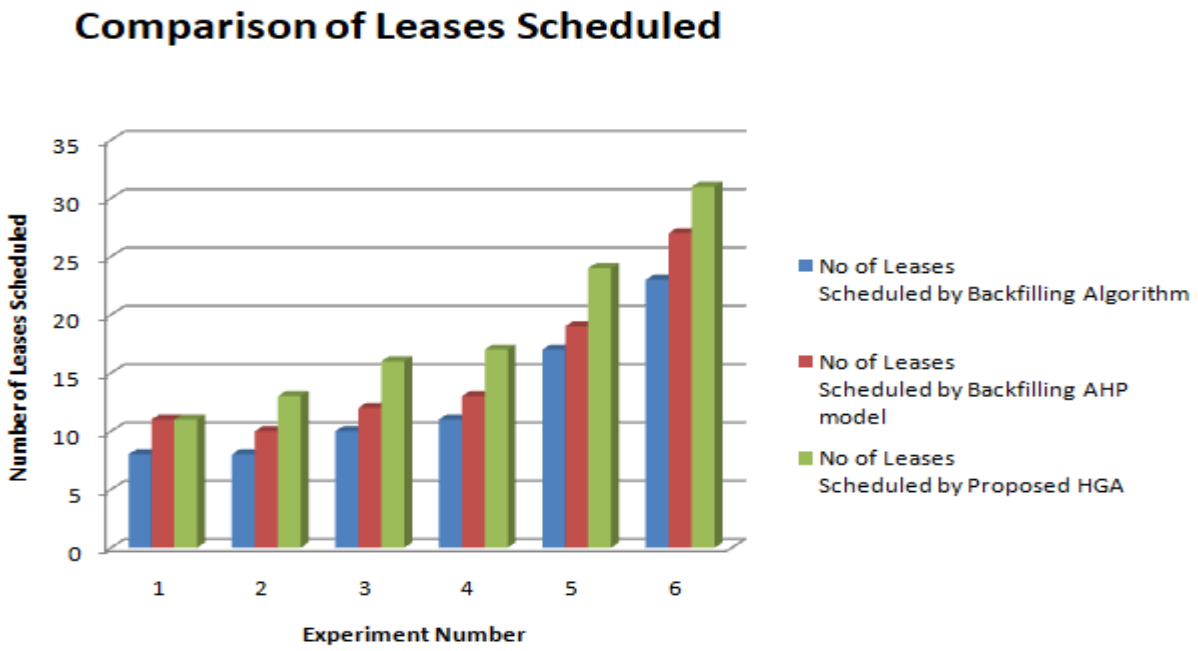

Figure5. Number of Leases scheduled by Backfilling Algorithm, Backfilling AHP approach [15] and Proposed HGA

Table3. Measured Time for Scheduling Leases

Number of Leases	Time Taken for Scheduling (in Milliseconds)
2	40
4	45
6	117
8	293
10	802

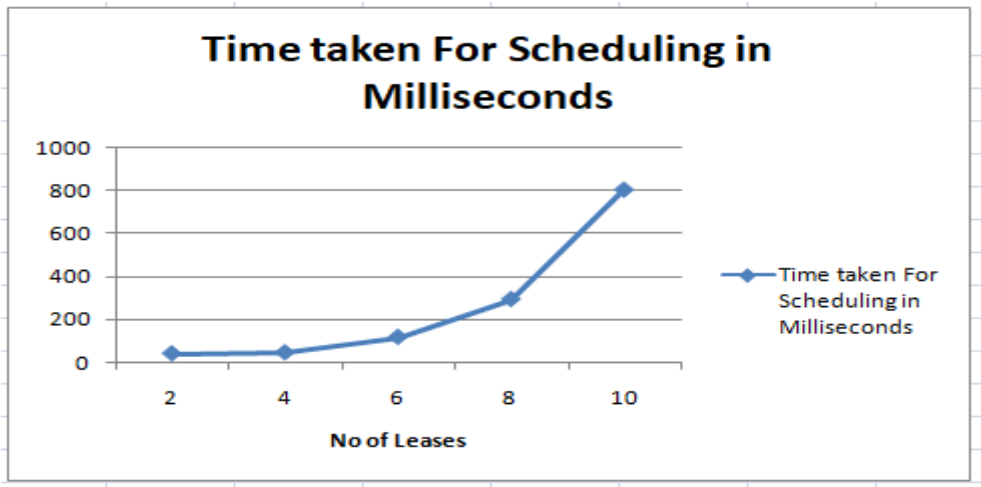

Figure6. Measured Time for the Scheduling Leases in milliseconds

VI. CONCLUSIONS AND FUTURE WORK

In this proposed work, we have devised HGA (Hungarian-Genetic Algorithm) with the key motive to lessen the execution time of deadline sensitive leases in cloud computing environment. HGA blends the Hungarian algorithm with Genetic algorithm to produce an optimal and robust approach. This approach chooses such a lease

combination which consumes least time of execution among various permutations of Lease-Scheduler mappings available. To find out the optimal lease combination, we have carried out decision making and comparison with the fitness function. The proposed HGA schedules greater number of leases as compared to the existing technologies (Backfilling AHP model and basic Backfilling Algorithm) in case of rigid deadlines.

It schedules all the leases without any lease rejection in case when deadlines are soft and found to be outperforming the basic Backfilling Algorithm by using the optimal lease combinations which consume minimum time. Also increasing the number of leases is not directly proportional to the computation time so performs well on increasing the number of leases.

For the future work, our algorithm can also work in real time in Haizea for OpenNebula. Since, in our approach we have worked only on Execution time so the future directions also include considering other parameters for further refinement.

REFERENCES

[1] Rimal, Bhaskar Prasad, and Martin Maier. "Workflow scheduling in multi-tenant cloud computing environments." IEEE Transactions on parallel and distributed systems 28.1 (2017): 290-304.

[2] Nayak, SuvenduChandan, and ChitaranjanTripathy. "Deadline based task scheduling using multi-criteria decision-making in cloud environment." Ain Shams Engineering Journal9.4 (2018): 3315-3324.

[3] Arunarani, A. R., D. Manjula, and VijayanSugumaran. "Task scheduling techniques in cloud computing: A literature survey." Future Generation Computer Systems 91 (2019): 407-415.

[4] Srinivasan, Sriramkrishnan. Cloud computing basics. Springer, 2014.

[5] Kalra, Mala, and Sarbjeet Singh. "A review of metaheuristic scheduling techniques in cloud computing." Egyptian informatics journal 16.3 (2015): 275-295.

[6] Wu, Xiaonian, et al. "A task scheduling algorithm based on QoS-driven in cloud computing." Procedia Computer Science17 (2013): 1162-1169.

[7] Arabnejad, Vahid, Kris Bubendorfer, and Bryan Ng. "Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources." Future Generation Computer Systems 75 (2017): 348-364.

[8] Sotomayor, Borja, et al. "Virtual infrastructure management in private and hybrid clouds." IEEE Internet computing 13.5 (2009): 14-22.

[9] http://haizea.cs.uchicago.edu/

[10] Calheiros, Rodrigo N., and RajkumarBuyya. "Meeting deadlines of scientific workflows in public clouds with tasks replication." IEEE Transactions on Parallel and Distributed Systems 25.7 (2014): 1787-1796.

[11] Jackson, Keith R., et al. "Performance analysis of high performance computing applications on the amazon web services cloud." 2010 IEEE second international conference on cloud computing technology and science. IEEE, 2010.

[12] Toosi, Adel Nadjaran, Richard O. Sinnott, and RajkumarBuyya. "Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka." Future Generation Computer Systems 79 (2018): 765-775.

[13] Vecchiola, Christian, et al. "Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka." Future Generation Computer Systems 28.1 (2012): 58-65.

[14] Xu, Xiangqiang, and Xinghui Zhao. "A framework for privacy-aware computing on hybrid clouds with mixed-sensitivity data." 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems. IEEE, 2015.

[15] Nayak, SuvenduChandan, and ChitaranjanTripathy. "Deadline sensitive lease scheduling in cloud computing environment using AHP." Journal of King Saud University-Computer and Information Sciences 30.2 (2018): 152-163.

[16] Zhao, Zhuo, Ying Jiang, and Xin Zhao. "SLA_oriented service selection in cloud environment: a PROMETHEE_based Approach." Computer Science and Network Technology (ICCSNT), 2015 4th International Conference on. Vol. 1. IEEE, 2015.

[17] Kaur, Kulbir, and Harshpreet Singh. "PROMETHEE based component evaluation and selection for Component Based Software Engineering." 2014 IEEE Int. Conf. on Advanced Communications, Control and Computing Technologies. IEEE, 2014.

[18] Brans, Jean-Pierre, PhVincke, and Bertrand Mareschal. "How to select and how to rank projects: The PROMETHEE method." European journal of operational research 24.2 (1986): 228-238.

[19] Ali, Hend Gamal El Din Hassan, Imane Aly Saroit, and Amira Mohamed Kotb. "Grouped tasks scheduling algorithm based on QoS in cloud computing network." Egyptian informatics journal 18.1 (2017): 11-19.

[20] Panda, Sanjaya Kumar, ShradhaSurachita Nanda, and Sourav Kumar Bhoi. "A pair-based task scheduling algorithm for cloud computing environment." Journal of King Saud University-Computer and Information Sciences (2018).

[21] Haidri, Raza Abbas, ChittaranjanPadmanabhKatti, and Prem Chandra Saxena. "Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing." Journal of King Saud University-Computer and Information Sciences (2017).

[22] Rodriguez, Maria Alejandra, and RajkumarBuyya. "Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds." IEEE transactions on cloud computing 2.2 (2014): 222-235.

[23] H.W. Kuhn , The Hungarian method for the assignment problem, Nav. Res. Logist. Q. 2 (1-2) (1955) 83–97 .

[24] Date, Ketan, and Rakesh Nagi. "GPU-accelerated Hungarian algorithms for the Linear Assignment Problem." Parallel Computing 57 (2016): 52-72.

[25] Suleiman Kabiru, Bello Malam Saidu, AbdullahiZubairu Abdul, Uba Ahmad Ali. "An Optimal Assignment Schedule of Staff-Subject Allocation ". Journal of Mathematical Finance, 2017, 7, 805-820

[26] R R Patel, T T Desai, S J Patel. "Scheduling of Jobs based on Hungarian Method in Cloud Computing". International Conference on Inventive Communication and Computational Technologies (ICICCT 2017).

[27] Disha Patel, Ms.JasmineJha. " Hungarian Method Based Resource Scheduling algorithm in Cloud Computing". IJARIIE ISSN(O)-2395-4396

[28] Lin, Chi-Shiuan, I-Ling Lee, and Muh-Cherng Wu. "Merits of using chromosome representations and shadow chromosomes in genetic algorithms for solving scheduling problems." Robotics and Computer-Integrated Manufacturing58 (2019): 196-207.

[29] Younas, Irfan, et al. "Efficient genetic algorithms for optimal assignment of tasks to teams of agents." Neurocomputing 314 (2018): 409-428.
[30] Casas, Israel, et al. "GA-ETI: An enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments." Journal of computational science 26 (2018): 318-331.
[31] Agarwal, Mohit, and Gur Mauj Saran Srivastava. "A genetic algorithm inspired task scheduling in cloud computing." 2016 International Conference on Computing, Communication and Automation (ICCCA). IEEE, 2016.
[32] Huang, Chin-Jung. "Integrate the Hungarian method and genetic algorithm to solve the shortest distance problem." 2012 Third International Conference on Digital Manufacturing & Automation. IEEE, 2012.

.

AUTHORS PROFILE

Miss. Duraksha Ali pursued Bachelors of Technology (Information Technology Engineering) from B.G.S.B University Rajouri, India in 2016 and Masters of Technology (Computer Science Engineering) from S.M.V.D University Katra, India in 2019. Her main research work focuses on cloud computing, scheduling in cloud computing, various MCDM (Multi Criteria Decision Makers) in cloud computing and cloud security.

Mr. Manoj Kumar Gupta pursued Bachelors of Engineering from CCS University Meerat in 2001, Masters of Technology from HBIT Kanpur in 2009 and pursued Ph.D from IIT Rorkee in 2014. He is currently working as Associate Professor in Department of Computer Science, S.M.V.D University Katra. He has published many research papers in esteemed journals. His main research work focuses on Data Mining Algorithms, Bio Informatics and Computational Biology, Algorithms. He has teaching experience of 13 years, Research experience of 3.5 years and 5 years of experience in Administration.

