Energy-Aware Resource Scheduling based on Genetic Algorithm

Ashwin Meena Meiyappan
School of Engineering and Technology
University of Washington, Tacoma, USA
Advisor: Eyhab Al-Masri

***Abstract*—Edge computing is growing rapidly due to several factors, including the proliferation of IoT devices, the increasing demand for real-time data analysis and decision-making. With the increase in the number of connected devices and the growing demand for faster data processing, it has become critical to build sustainable and efficient edge computing systems. Effective resource scheduling, an integral component of edge computing, can lead to significant energy savings, which is especially important for the edge servers particularly one that have limited resources. In this proposal, we plan to study the use of a genetic algorithm (GA) for resource scheduling in an edge computing environment with an objective to minimize energy consumption. We aim to compare the performance of our proposed genetic algorithm using CloudSim with the resource scheduling algorithm (RSA) of the simulator which is First-Come-First-Serve (FCFS). The simulation results will be used to evaluate the algorithms based on several performance metrics, including energy consumption, make-span, and QoS satisfaction.**

Keywords—edge computing, resource scheduling, genetic algorithm, energy consumption.

I. INTRODUCTION

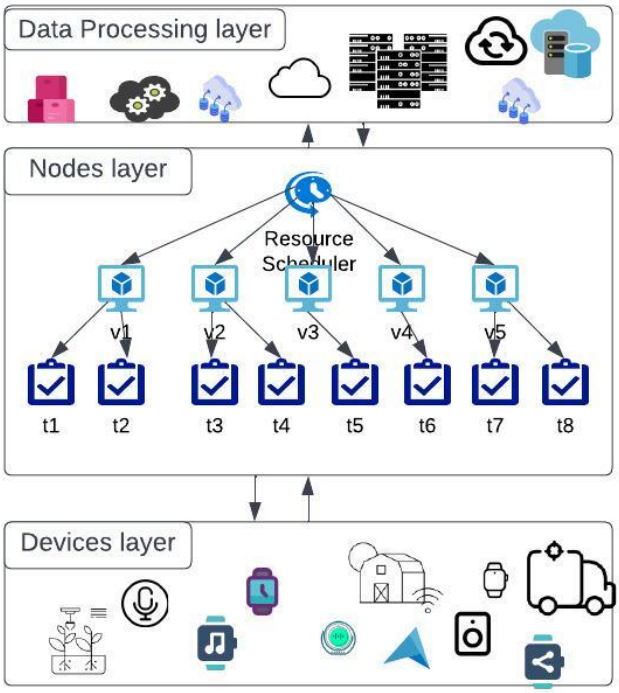
While centralized cloud-IoT solutions may offer reasonable data storage and processing capabilities for executing IoT operations, often times they are unable to meet low latency and other Quality of Service (QoS) application requirements due to network delays. Alternatively, edge computing provides support for reducing latency and improving the efficiency compared to that of traditional IoT [7].

Edge computing is an emerging distributed computing paradigm that contrasts to the traditional way of processing IoT data and executing IoT tasks, which typically take place within centralized data centre. The aim of edge computing, however, is bring compute and storage resources nearer to data sources in order to enable near real-time processing which can be very useful for latency sensitive tasks generated by IoT devices such as self-driving vehicles, industrial automation and health monitoring [13-16]. Developing sustainable edge computing systems is the need of the hour because of the growth of the Internet of Things (IoT) and the increasing number of connected devices generate massive amounts of data, which require quick processing and analysis. By processing data at the edge of the network, it is then possible to reduce the latency and network congestion that would occur if the data were sent to a centralized cloud location or a distant data center. Edge computing allows for faster data processing, lower latency, and more efficient use of network bandwidth [15].

A typical edge computing architecture consists of three layers which often include: (a) devices, (b) nodes, and (c) a data processing layer [17].

1. Devices layer: This first layer of the architecture generally includes edge devices such as sensors, smart devices, and other IoT devices. This layer is responsible for collecting and processing data at a very small scale. For example, edge devices, dependent on their capabilities, may perform functions such as initial data filtering, preprocessing, and analysis, and send only relevant data to the next layer.

2. Nodes/Gateways layer: This second layer of the edge computing architecture consists of computing nodes that are located in close proximity to local edge devices. These nodes/servers are often equipped with some computational power or are more powerful than edge devices and are capable of performing more complex computations and data analysis tasks [17]. Typically constrained edge devices can offload the tasks to edge nodes to be able to improve the processing of tasks. Edge nodes typically receive data from nearby edge devices, perform additional processing,



and analysis and may store frequently accessed data locally. Generally, edge nodes provide low-latency services to nearby edge devices.

3. Data processing layer: The third/top layer of the edge computing architecture is a data processing layer, which typically reside on the cloud and often includes computing resources that aim to provide high-level services to the edge devices or nodes. The data processing layer can be residing on a fog layer or more commonly a cloud layer and is responsible for performing complex computations and data analysis tasks that are beyond the capabilities of edge nodes or gateways. This layer receives data from the edge nodes, processes and analyzes them, and sends back relevant results to the edge nodes layer.

Because edge nodes have limited compute capabilities, and latency-sensitive applications cannot tolerate network delays, it is imperative to identify efficient scheduling algorithms for edge resources. Existing research efforts have focused primarily on a limited number of objectives such as execution time and cost while neglecting energy consumption. In addition, edge computing environments often consist of a variety of devices with different capabilities and resources such as processing power, memory, and storage. Furthermore, edge computing are highly dynamic which means the availability of resources may change frequently [18]. In addition, energy consumption is often neglected in terms of scheduling resources across edge environments [4].

Energy-aware resource scheduling in edge computing faces several challenges, including heterogeneous resources, energy consumption models, dynamic environment, trade-offs between energy efficiency and performance, QoS requirements, limited communication bandwidth, and security and privacy issues. To address these challenges, it is crucial to devise efficient and effective energy-aware scheduling algorithms that can balance energy efficiency, performance, and QoS requirements effectively.

Resource scheduling, the process of allocating the available resources to various tasks or jobs in a system, is a critical component of edge computing. Resources include CPU, memory, storage, or other computing resources. Resource scheduling aims to efficiently utilize the available resources to optimize system performance while meeting the task requirements. In an edge environment with virtualization in place, resource scheduling is the process of allocating set of tasks to a set of resources (Virtual machines). Resource scheduling is a NP-hard problem as discussed in [1,6] , there are many research efforts which have been conducted for resource scheduling in cloud computing which initially used linear algorithms but couldn't solve when there was increase in dimensions and complexity [1] since linear algorithms can become computationally expensive when dealing with large search spaces, which can result in long computation times and also can sometimes become trapped in local optima, which are suboptimal solutions that are close to the starting point but not the best possible solution but evolutionary algorithm like GA are better equipped to avoid getting stuck in local optima by exploring a wider range of potential solutions. So researchers started adopting evolutionary approaches.

In this proposal, we plan to use an evolutionary algorithm - Genetic Algorithm for resource scheduling in edge computing. Our objective or fitness function will be employed in order to reduce energy consumption. There are several research efforts which employs GA to scheduling problems in cloud computing environment and (Multi access Edge Computing) MEC as discussed in - [5, 8,9] because it is a meta heuristic technique which is generally used to generate high quality solutions to optimization and search problems as mentioned in [10]. Hence, initial efforts will be taken to work with one objective function and later efforts will be taken to adapt the algorithm with multiple fitness function to consider cost, execution time and other QoS requirements of the task.

We are studying energy usage in resource scheduling for edge computing, as the available resources at edge nodes are limited. Also it is the need of the hour to build efficient and sustainable edge computing systems. Our research is aimed at understanding how a resource scheduler can delegate tasks to the VMs in edge nodes, regardless of whether the task is partially or fully offloaded to the edge nodes layer and assuming that there exists enough resources available to complete the tasks. Our research, however, does not involve deciding whether to offload the task to the cloud layer or complete it at the edge nodes layer.

II. RESEARCH QUESTIONS

A. Identify the selection strategy to select parents from the random initial population for Genetic Algorithm.

Genetic algorithm is a problem-solving technique inspired by natural evolution. It is a way to find the best solution to a problem by mimicking the process of natural selection [10]. A selection strategy helps to determine the individuals from the current population based on fitness function for the next generation. We will identify the best selection strategy suitable for resource scheduling. In addition, there are several research efforts on evaluating the selection strategies as in [2, 19,20], which discuss the performance evaluation of a selection strategy using a genetic algorithm. Following are three popular strategies used:

1. **Roulette Wheel Selection**: In this strategy, individuals are selected randomly with probability proportional to their fitness score. The individuals with higher fitness scores have a higher chance of being selected for reproduction. This strategy is simple to implement, but can be biased towards fitter individuals.
2. **Tournament Selection**: A small subset of individuals is randomly selected from the population, and the individual with the highest fitness score is chosen for reproduction. This process is repeated until the desired number of individuals for reproduction is reached. Tournament selection is less biased towards fitter individuals and is useful when the fitness function is noisy or non-linear.
3. **Rank Selection**: Individuals are sorted based on their fitness score, and a probability of selection is assigned to each individual based on their rank in the sorted population. The individuals with higher ranks have a higher chance of being selected for reproduction. This strategy is similar to roulette wheel selection, but instead of using the actual fitness values, it uses the rank of the individuals.

Rank selection can reduce the bias towards fitter individuals seen in roulette wheel selection and is also useful when the fitness function is non-linear.

As part of our research proposal, we will investigate the use of these three popular selection strategies to identify a suitable selection method for our scheduling algorithm.

B. *Can we employ Genetic Algorithm having multiple fitness function to optimize multiple objectives including energy consumption, execution time, cost and other QoS requirements?*

We can have multiple fitness function but need to find a way to combine the fitness function. So far weighted sum looks better to consider and vary weights based on the importance of one fitness function over the other as discussed in [9]. In addition, we will investigate the use of various weighting methods or strategies and identify their impact on the overall selection and/or scheduling process.

C. *To implement/develop resource scheduling strategy that extends the functionality of CloudSim simulator which will also help for future researchers.*

CloudSim is a widely used simulation toolkit that allows researchers and developers to model and simulate cloud computing environments. It provides a set of APIs that allow users to create and configure virtualized resources, such as virtual machines, and to model the behavior of cloud computing applications and workloads. By default CloudSim implements First-Come-First-Serve (FCFS) resource scheduling algorithm using java. Hence we are planning to extend the functionality of resource scheduling algorithm in CloudSim by implementing Genetic Algorithm for resource scheduling using java.

III. RELATED WORK

Numerous research studies have focused on optimizing resource scheduling in cloud computing environments, but they have not placed a significant emphasis on minimizing latency and achieving real-time response as that was not the major concern. Similarly, some research articles on resource scheduling in edge computing have evaluated performance based solely on task execution time, without considering factors such as the use of expensive GPU-powered virtual machines or the energy consumption required for execution. We would have to dig further to assess the trade-offs between energy consumption, execution time, and cost.

This survey [4] - discusses about all the research papers which evaluated the performance of scheduling algorithms in fog computing. It lists a detailed survey of resource scheduling algorithms in fog computing, which proposes two layer static scheduling technique, cluster fog resources and etc. Also there are research papers which focus on latency and service stability but not on execution cost. So our main focus is to develop Genetic Algorithm and combine energy consumption, execution cost and execution time as objective function to schedule resources in edge computing.

Another research paper[3], discusses a resource scheduling strategy for edge computing and proposes an algorithm for selecting the appropriate edge node for a task based on whether the node has sufficient resources to complete the task or not. If the resources are insufficient, the paper presents a novel cooperation strategy to divide and merge the task results. But we cannot employ this strategy for mapping tasks to VMs.

This paper [5] - describes the use of a genetic algorithm for resource scheduling in cloud computing environments. The problem involves mapping a set of virtual machines (VMs) to physical machines. Also authors mention about how they formulated this problem as a genetic algorithm by defining a fitness function to evaluate the quality of each mapping and using crossover and mutation operators to generate new mappings, which will help us to formulate the resource scheduling problem which will involve mapping set of tasks to the available VMs.

So far, we couldn't find research efforts which employs genetic algorithm to resource scheduling problem in edge computing to optimize energy consumption at edge nodes. Hence Fig. table 1 : summarizes the research papers.

Table 1: Short survey of research papers

Paper	RS (Resource Scheduling Algorithm)	GA	Edge node Energy Consumption	Edge (E) or Cloud (C)
[1]	Yes	No	No	C
[3]	No	No	No	E
[4]	Yes	No	No	E
[5]	Yes	Yes	No	C
[6]	Yes	No	No	E
[7]	Yes	No	No	E
[9]	Yes	Yes	No	E
[11]	Yes	Yes	No	C
[21]	Yes	Yes	No	C
[18]	Yes	No	No	E
[22]	Yes	No	No	E
[23]	Yes	No	Yes	E
[24]	No	Yes	No	E

Table 1: Short survey of research papers

Paper	RS (Resource Scheduling Algorithm)	GA	Edge node Energy Consumption	Edge (E) or Cloud (C)
Our proposed Algorithm	Yes	Yes	Yes	E

IV. METHODOLOGY

Genetic algorithm (GA) is a problem-solving technique inspired by the process of natural selection and evolution. It is a type of optimization algorithm that is used to find the best solution to a problem by mimicking the process of natural selection. In genetic algorithm, a population of potential solutions is randomly generated and evaluated using a fitness function. The fitness function measures how well each individual solution solves the problem at hand. The individuals in the population then "evolve" by undergoing a process of selection, crossover, and mutation.

The selection process chooses the fittest individuals in the population to mate and create offspring, which inherit traits from their parents. And the crossover process exchanges genetic material between two parents to create new individuals, while the mutation process introduces random changes to the genetic material of an individual.

This process is repeated over multiple generations, with the fittest individuals in each generation being selected for reproduction. Over time, the population evolves towards better and better solutions to the problem, until a satisfactory solution is found. Genetic algorithms are widely used in optimization problems where the search space is large or complex, and where traditional optimization techniques may not be effective.

In general, Genetic Algorithm is commonly used for solving scheduling problems [25,26] since it can handle complex, nonlinear problems with a large number of variables and constraints. GA can find good solutions quickly, even in cases where traditional optimization methods would require significant computational resources [27]. Also GA can adapt to changes in the scheduling problem over time. For example, if new tasks or resources are added, or if the constraints change, GA can quickly adjust to the new conditions and find a new optimal solution [29]. GA also provides multiple solutions that are not necessarily the same, allowing for a broader range of options for the scheduler to choose from [28]. Overall, GA is a powerful method for solving scheduling problems that offers flexibility, efficiency, and adaptability.

Following are the general steps involved in using genetic algorithm for scheduling problem:

- We need to define the problem and identify the variables involved in the scheduling process, such as tasks, resources, time, and constraints.
- Chromosome: The chromosome is a set of genes that represent the parameters of the problem. A gene could represent a task or a resource.
- Fitness function: It will be used to evaluate the quality of each individual in the population. The fitness function is a measure of how well a particular schedule satisfies the objectives and constraints of the problem.
- Initialize the population: Create an initial population of individuals. Each individual represents a possible solution to the problem and is represented by a chromosome.
- Evaluate the fitness of each individual in the population using the fitness function.
- Select the fittest individuals in the population to be the parents of the next generation. (Step 6 - future reference)
- Perform crossover between the selected individuals to generate new offspring. Crossover involves swapping genetic material between the parents to create new individuals that inherit traits from both parents.
- Perform mutation on the offspring generated in the previous step to introduce genetic diversity into the population. Mutation involves randomly changing the genetic material of an individual.
- Evaluate the fitness of each individual in the new population using the fitness function.
- If the termination condition is not met, go back to step 6. Otherwise, select the best individual in the final population as the solution to the problem

Overview of formulating genetic algorithm for resource scheduling problem in edge computing:

- Defining the problem: Our objective is to identify the most efficient way to assign a group of tasks to VMs available at the edge node, with the aim of minimizing energy consumption. Initially, we will employ GA using a single fitness function - minimum energy consumption. However, we intend to expand the approach to incorporate multiple fitness functions, such as cost, execution time, and QoS parameters. Lets consider T as set of n tasks -$\{t_0,\ldots t_n\}$ and V as set of m available VMs $\{v_0,\ldots,v_n\}$ at edge node.
- Fitness function : Selecting combination of VMs which uses minimum energy to run a set of tasks. Fitness function will be to calculate energy spent on VM to run a task using Amdahl's law and expected execution time of tasks.

Below is the outline of the algorithm which we are planning to implement using genetic algorithm :

Algorithm 1:

Input:
Parameters for genetic algorithm and
for resource scheduling;
Output:
energy aware mapping of tasks to VMs

1: invoke Algorithm 2 to create initial population;
2: invoke Algorithm 3 to calculate fitness of every individual;
3: **repeat**
4: invoke Algorithm 4 to use selection strategy to identify the fittest individuals who will make it to the next generation.
5: invoke Algorithm 5 to generate new population using GA operators : crossover and mutation;
6: replace old population using new population;
7: again invoke Algorithm 3 to calculate fitness value of new population
8: **until** termination criteria is not met.
9: **return** a sub-optimal or near optimal solution

Initial Population :

In GA, the initial population refers to a randomly generated set of potential solutions for a given problem. The population is usually made up of a fixed number of individuals, each representing a possible solution to the problem being solved. It is a critical component of GA, as it provides the starting point for the optimization process. We are planning to consider all combinations of available VMs and mapping it to given group of tasks.

Let's consider simple example of generating initial population in genetic algorithm for resource scheduling problem. We would like to return energy aware mapping of available VMs to group of tasks.

T = { t_1, t_2, t_3, t_4, t_5 }

V = { $v_1, v_2, v_3, v_4, v_5, v_6 \ldots\ldots, v_{100}$}

Total initial population size: $^{100}C_5$ combinations

Fig. Table 2 : shows all possible initial solutions

Table 2: Example initial population

	t1	t2	t3	t4	T5
Sol 1	v4	V6	v89	V77	V56
Sol 2	V3	V19	v43	v58	V99
Sol 3	v13	V37	V51	V81	V97
Sol 4	V21	V46	V66	V71	V85

Table 2: Example initial population

	t1	t2	t3	t4	T5
….	…	….	…	….	…
Sol 75,287,520	V13	V41	v68	v55	V92

Where Sol 75,287,520 equals: $^{100}C_5$. These are all possible combinations of mapping VMs to group of tasks. In later steps, we will calculate energy spent on every VM for that particular task and combine total energy spent for all VM combination. Based on fitness function, selection strategy, some threshold and GA operators some VM combinations will pass on to next generation and rest will be ignored. Below is the generalized algorithm to generate initial population.

Algorithm 2:

Input:
set of n tasks : $t_0, \ldots, t_n$
set of m available VMs: $v_0,\ldots,v_m$
Output:
initial population
1: **repeat**
2: generate all combinations of VMs for n tasks
store it in all_combo_vms list
3: **until** all combinations of VMs are generated
4: **for** every combo in all_combo_vms **do**:
5: **for** every vm in combo **do** :
6: calculate energy_consumed by vm for task i,
7: store energy_consumed for task i
8: **return** initial population

Calculate energy consumed by task in a particular VM: We have to figure out how to theoretically measure energy consumed by vm for particular task using vm specifications.

Time complexity of algorithm 2: To generate all combinations of VMs for n tasks is exponential, let it be M. Then calculating energy consumed by all combinations of VMs is O(M * M). As part of our research, we will consider some criteria instead of taking all combinations of tasks as initial population if required to improve execution time of our algorithm.

Evaluate individuals using fitness function:

Since we are considering only one fitness function, let's calculate total energy consumed for the initial population. Otherwise later, we will devise a plan for determining which additional goals should be taken into account and how to assess them collectively.

Algorithm 3:

Input:
initial population
Output:
list of fitness value for all candidates

1: **for** every combo in all_combo_vms **do**:
2: **for** energy_consumed in combo **do** :
3: sum energy_consumed by vm for task i
4: store this sum for every combo
5: **return** list of total energy spent for every vm_combination

---

Selection strategy (Algorithm 4):

In this proposal, let's understand roulette wheel selection, but we will research and evaluate all available selection strategies to determine which one is the most effective in achieving the optimal solution.

Roulette wheel selection: In general it involves assigning a portion of a roulette wheel to each potential solution in the population, with the size of the portion proportional to the solution's fitness score. A roulette wheel is then spun, and the selected solution is the one that the wheel's pointer stops on. It gives more chances of selection to solutions with higher fitness scores, as they occupy a larger portion of the roulette wheel. If f_i is the fitness of individual i in the population, its probability of being selected is

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

After performing research of available selection strategy we will implement algorithm 4 for our proposed genetic algorithm for resource scheduling .

GA operators (Algorithm 5):

Cross over: Crossover is the process of combining two or more parent solutions to create new offspring solutions. The idea behind crossover is to exchange genetic material between parent solutions in a way that preserves and potentially improves upon the desirable characteristics of each solution. In the context of genetic algorithms, the genetic material refers to the individual components or parameters of a solution that are being optimized. The result of crossover is a set of new solutions that inherit genetic material from their parents.

Mutation : It is a random change in the genetic material of a solution. Mutation introduces new genetic material that was not present in the parent solutions and can help the genetic algorithm explore new regions of the solution space. Mutation is typically applied to a small proportion of the population, and the magnitude of the change is usually kept small to prevent drastic changes that could disrupt the overall performance of the population.

We will implement algorithm 5 after we consider how we are going to perform the above GA operators to generate new population and explore more how will this help us reach near optimal solution. Since there are scenarios where mutation might not be of great help [30].

Termination criteria:

The termination criteria of a genetic algorithm (GA) are the conditions used to decide when to stop the algorithm. Since a GA has the potential to run indefinitely, it is important to establish these criteria. Various termination criteria exist, such as a fixed number of generations, convergence to the same result over several generations, or achieving a fitness value threshold. For our proposal, we will stop the algorithm if the same results persist for 100 consecutive generations. We plan to test alternative termination criteria for the GA and choose the most effective one that leads to a solution that is close to optimal.

V. EVALUATION

We are planning to use cloudSim to simulate experiments and evaluate proposed genetic algorithm for energy aware resource scheduling. CloudSim is a Java-based framework that provides a simulation environment for modeling and simulating cloud computing environments and services. It provides a modular and extensible architecture that allows us to model various components of cloud computing systems, such as data centers, virtual machines, job scheduling policies, and resource provisioning algorithms. It provides a set of APIs that allows us to create our own simulation scenarios, configure the simulation parameters, and analyze the results. Currently, we are utilizing CloudSim to model an edge environment by choosing a data center with limited CPU, storage, and other resources. However, we may switch to FogSim in the future if necessary.

Consider, for example, a simple sample simulation on CloudSim: assume we create a datacenter that contains a single host and execute two tasks on it. The tasks will be executed within VMs that have equivalent MIPS (million instructions per second) requirements, and both tasks will require the same amount of time to finish their execution.

Algorithm:

1: initialize cloudSim package before creating any entities. And initializing cloudSim library with number of users, calendar and trace events.

2: create one datacenter object with host specification, this is the resource provider in cloudSim. We need at least one of this to run a simulation on cloudSim.

3: create broker

4: create two vms with required specifications like: vmid, mips, number of cpus, image size (MB), ram memory, bandwidth and vmm name. Add the vm object to vmlist and submit it to the broker.

5: create two tasks with properties like: id, length(number of instructions) , fileSize, outputSize, utilizationModel. Add the task to task-list and submit to the broker.

6: broker allocates two tasks to two vms. By default cloudSim uses FCFS to allocate tasks to available vms.

```
========== OUTPUT ==========
Cloudlet ID    STATUS      Data center ID    VM ID    Time    Start Time    Finish Time
    0          SUCCESS        2              0        1000       0.1         1000.1
    1          SUCCESS        2              1        1000       0.1         1000.1
```

Snippet of the output :

Explanation : Cloudlet ID is task id, both tasks (two similar tasks) were scheduled on different vms (same vm specification) by broker. Based on STATUS variable both tasks were successfully executed for 1000 units of time (Time).

In cloudSim after the VMs are created they are registered(submitted) with broker and task list is also submitted to broker. The resource scheduling algorithm is implemented in the broker entity. As mentioned earlier, broker implements FCFS algorithm to schedule tasks to available VMs. We are planning to expand the CloudSim framework's ability to execute genetic algorithm for mapping set of tasks to available vms.

Experimentation algorithm:

1: initialize CloudSim package before creating any entities. And initializing CloudSim library with number of users, calendar and trace events.

2: implement new broker entity using genetic algorithm proposed in this proposal.

3: create multiple datacenter object with host specification.

4: instantiate new broker entity

5: create multiple vms with specifications like: vmid, mips, number of cpus, image size (MB), ram memory, bandwidth and vmm name. Add all the vm objects to vmlist and submit it to the new broker entity.

6: create multiple tasks with properties like: id, length(number of instructions) , fileSize, outputSize, utilizationModel. Add all the tasks to task-list and submit to the new broker.

7: return status if the tasks were executed successfully after mapping to vms consuming less energy and execution time.

Above mentioned are the steps involved to conduct experiments in cloudSim. We will evaluate total energy spent for all combinations of VMs for given group of tasks as mentioned in table 2. Then we will compare our proposed algorithm with existing algorithms like FCFS (First Come First Serve), SJF (Shortest Job First) and others, to analyze if genetic algorithm is able to find a near optimal solution to map available VMs to tasks with minimum energy consumption. As part of our research we will evaluate energy consumed by VMs to complete a set of tasks.

VI. DELIVERABLES

Throughout the duration of project following items will be delivered:

- source code
- GitHub extension for GA in Java
- Final report with methodology, observations, results and analysis of proposed algorithm.
- Presentation of the thesis
- Experiments conducted as part of research
- IEEE Conference-like paper

VII. EDUCATIONAL STATEMENT

With my prior professional experience, I am equipped to undertake the technical experimental tasks mentioned in this thesis proposal. During the summer of 2022, I served as a software engineer intern on Amazon's ML infra team. There, I utilized AWS services to design and implement a backend API service in Java, automating the client onboarding process. The resulting service was deployed to the CI/CD pipeline. In addition, I have worked as a data analyst for nearly three years at Deloitte India. All of my previous experiences in the software industry have prepared me to write code that is production-ready, allowing me to expand the functionality of cloudSim by implementing a genetic algorithm.

The research work I will be conducting will benefit from the knowledge gained in my master's program courses, specifically TCSS 562 (cloud computing), TCSS 543 (Advanced Algorithm), and TCSS 559 (Service Computing).

VIII. PROJECT TIMELINE

Quarter 1:

Week 1 - 2:

- Explore all modules in CloudSim framework.
- Understand and conduct small experiments on CloudSim to learn how broker entity works.
- Identify ways to how to add a custom-built algorithm that overrides default scheduling technique on CloudSim.

Week 3:

- Investigating how to employ genetic algorithm and generate a good initial population for simulation.

Week 4:

- Understand all available selection strategies and find the best one which will help for solving our problem.
- Compare existing selection strategies and identify a suitable one that can be used for building an energy-aware scheduler.

Week 5 - 6:

- Build and evaluate fitness function to calculate energy consumption.

• Expand the research within the developed algorithm to include multiply objectives.

• Identify an appropriate algorithm to combine all objectives.

Week 7 - 8:

• Learn how to employ the GA operators - crossover and mutation.

• Investigate the importance of including GA operators as part of the algorithm and their impact on the overall results.

• Determine the termination criteria or an optimal solution.

Week 9 - 10:

• Implement the genetic algorithm and write unit tests.

Quarter 2:

Week 1 - 2:

• Identify all possible test cases

• Identify sample test data and a proper testbed to be used for evaluating the GA algorithm.

• Begin on conducting all planned experiments.

Week 3 - 4:

• Fine-tune the GA algorithm to improve its performance in terms of finding an optimal solutions.

• Conduct an evaluation comparing the proposed GA algorithm with existing ones such as FCFS and other heuristic algorithms.

Week 5 - 7:

• Explore iFogSim and identify if the same experiment could be conducted on iFogSim simulation environment.

• Extend the GA capability to work on that of iFogSim.

• Create a GitHub library extension for using GA in scheduling edge resources.

Week 8 - 10:

• Collect and analyze results.

• Prepare final report and summaries.

• Complete the final report and delivery material.

REFERENCES

1. Shweta Varshney, Sarvpal Singh on *"A Survey on Resource Scheduling Algorithms in Cloud Computing", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 9 (2018) pp. 6839-6845*
2. Hari Mohan Panter on *"Performance Evaluation of Selection Methods of Genetic Algorithm and Network Security Concerns" , Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license.*
3. Li, Xiaomin ; Wan, Jiafu ; Dai, Hong-Ning ; Imran, Muhammad ; Xia, Min ; Celesti, Antonio on "*A Hybrid Computing Solution and Resource Scheduling Strategy for Edge Computing in Smart Manufacturing*" IEEE transactions on industrial informatics, 2019, Vol.15 (7), p.4225-4234; IEEE
4. Khaled M. Matrouk, Kholoud Al-atoun on *"Scheduling Algorithms in Fog Computing: A Survey*" in International Journal of Networked and Distributed Computing · January 2021.
5. Jianhua Gu, Jinhua Hu, Tianhai Zhao, Guofei Sun on "*A New Resource Scheduling Strategy Based on Genetic Algorithm in Cloud Computing Environment"* JCP 2012 Vol.7(1): 42-52 ISSN: 1796-203X.
6. Xue, Fei ; Hai, Qiuru ; Dong, Tingting ; Cui, Zhihua ; Gong, Yuelu on "A deep reinforcement learning based hybrid algorithm for efficient resource scheduling in edge computing environment" Information sciences, 2022, Vol.608, p.362-374; Elsevier Inc
7. Vijayasekaran, G. ; Duraipandian, M. On "An Efficient Clustering and Deep Learning Based Resource Scheduling for Edge Computing to Integrate Cloud-IoT". Wireless personal communications, 2022, Vol.124 (3), p.2029-2044; New York: Springer US
8. Zhang, Nan ; Yang, Xiaolong ; Zhang, Min ; Sun, Yan ; Long, Keping on "A genetic algorithm-based task scheduling for cloud resource crowd-funding model". International journal of communication systems, 2018, Vol.31 (1), p.e3394-n/a; HOBOKEN: Wiley
9. Luhach, Ashish Kumar ; Ramasubbareddy, Somula ; Swetha, Evakattu ; Srinivas, T Aditya Sai on "A Multi-Objective Genetic Algorithm-Based Resource Scheduling in Mobile Cloud Computing". International journal of cognitive informatics & natural intelligence, 2021, Vol.15 (3), p.58-73
10. Genetic algorithm Wikipedia "https://en.wikipedia.org/wiki/Genetic_algorithm"
11. Y Xua, K Lia, J Hub, K Lia on "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues" - Information Sciences, 2014 - cs.newpaltz.edu
12. "https://en.wikipedia.org/wiki/Fitness_proportionate_selection"
13. "Deep Reinforcement Learning for Vehicular Edge Computing: An Intelligent Offloading System" Ning, Zhaolong ; Dong, Peiran ; Wang, Xiaojie ; Rodrigues, Joel ; Xia, Feng. ACM transactions on intelligent systems and technology, 2019, Vol.10 (6), p.1-24; ACM.
14. "Energy-Efficient Edge Computing Service Provisioning for Vehicular Networks: A Consensus ADMM Approach". Zhou, Zhenyu ; Feng, Junhao ; Chang, Zheng ; Shen, Xuemin. IEEE transactions on vehicular technology, 2019, Vol.68 (5), p.5087-5099; New York: IEEE.
15. "Edge Computing for the Internet of Things: A Case Study". Premsankar, Gopika ; Di Francesco, Mario ; Taleb, Tarik. IEEE internet of things journal, 2018, Vol.5 (2), p.1275-1284; PISCATAWAY: IEEE.
16. "Learning-Based Context-Aware Resource Allocation for Edge-Computing-Empowered Industrial IoT". Liao, Haijun ; Zhou, Zhenyu ; Zhao, Xiongwen ; Zhang, Lei ; Mumtaz, Shahid, Jolfaei, Alireza ;Ahmed, Syed Hassan ; Bashir, Ali Kashif. IEEE internet of things journal, 2020, Vol.7 (5), p.4260-4277; Piscataway: IEEE
17. "Edge-Computing-Based Intelligent IoT: Architectures, Algorithms and Applications". Liu, Xiao ; Jin, Jiong ; Dong, Fang. Sensors (Basel, Switzerland), 2022, Vol.22 (12), p.4464; Basel: MDPI AG
18. "Dyme: Dynamic Microservice Scheduling in Edge Computing Enabled IoT". Samanta, Amit ; Tang, Jianhua. IEEE internet of things journal, 2020, Vol.7 (7), p.6164-6174; Piscataway: IEEE
19. "Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms". Jinghui Zhong ; Xiaomin Hu ; Min Gu ; Jun Zhang. International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), 2005, Vol.2, p.1115-1121; IEEE
20. "Comparison of binary coded genetic algorithms with different selection strategies for continuous optimization problems". Kang-Di Lu ; Guo-Qiang Zeng ; Jie Chen ; Wen-Wen Peng ; Zheng-Jiang Zhang ; Yu-Xing Dai ; Qi Wu. 2013 Chinese Automation Congress, 2013, p.364-368; IEEE.
21. "Cloud Computing Resource Scheduling Research Based on the Improved Quantum Genetic Algorithm". Lijun Mao, Xinyan Wang, Jing Li. Journal of Computer Applications, 2013, Vol.33 (8), p.2151-2153

22. “Software-Defined Heterogeneous Edge Computing Network Resource Scheduling Based on Reinforcement Learning” . Li, Yaofang ; Wu, Bin. Applied sciences, 2023, Vol.13 (1), p.426; Basel: MDPI AG
23. “Secure and Latency-Aware Digital Twin Assisted Resource Scheduling for 5G Edge Computing-Empowered Distribution Grids”. Zhou, Zhenyu ; Jia, Zehan ; Liao, Haijun ; Lu, Wenbing ; Mumtaz, Shahid ; Guizani, Mohsen ; Tariq, Muhammad. IEEE transactions on industrial informatics, 2022, Vol.18 (7), p.4933-4943; Piscataway: IEEE
24. “Task Scheduling for Mobile Edge Computing Using Genetic Algorithm and Conflict Graphs”. Al-Habob, Ahmed A ; Dobre, Octavia A ; Armada, Ana Garcia ; Muhaidat, Sami. IEEE transactions on vehicular technology, 2020, Vol.69 (8), p.8805-8819; IEEE
25. “Transit network design and scheduling using genetic algorithm – a review”. Johar, Amita ; Jain, S S ; Garg, P K. An international journal of optimization and control, 2016, Vol.6 (1), p.9-22; Balikesir:
26. “Nurse Scheduling Using Genetic Algorithm”. Leksakul, Komgrit ; Phetsawat, Sukrit. Mathematical problems in engineering, 2014, Vol.2014, p.1-16; LONDON: Hindawi Publishing.
27. JM Johnson, Y Rahmat-Samii on “Genetic algorithm optimization and its application to antenna design”.- Proceedings of IEEE Antennas …, 1994 - ieeexplore.ieee.org
28. Katoch, S., Chauhan, S.S. & Kumar, V. A review on genetic algorithm: past, present, and future. *Multimed Tools Appl* **80**, 8091–8126 (2021). https://doi.org/10.1007/s11042-020-10139-6
29. M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," in *Computer*, vol. 27, no. 6, pp. 17-26, June 1994, doi: 10.1109/2.294849.
30. A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, and V. B. S. Prasath, “Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach,” *Information*, vol. 10, no. 12, p. 390, Dec. 2019, doi: 10.3390/info10120390.