A Hybrid Job Scheduling Approach for Cloud Computing Environments: On the Usage of Heuristic and Metaheuristic Methods

Abhijith Remesh
Very Large Business Application Lab
Otto von Guericke University, Magdeburg
abhijith.remesh@ovgu.de

Abdulrahman Nahhas
Very Large Business Application Lab
Otto von Guericke University, Magdeburg
abdulrahman.nahhas@ovgu.de

Andrey Kharitonov
Very Large Business Application Lab
Otto von Guericke University, Magdeburg
andrey.kharitonov@ovgu.de

Klaus Turowski
Very Large Business Application Lab
Otto von Guericke University, Magdeburg
klaus.turowski@ovgu.de

Abstract

The Information Technology Industry has been revolutionized through Cloud Computing. The process of job scheduling is an integral part in cloud computing and developing novel scheduling strategies are relevant. Different scheduling heuristics performs differently in terms of different metrics, and their performance also depends on the nature and amount of tasks. Hence, an adaptive scheduling mechanism with better optimization potential posses great significance. This paper proposes a hybrid task scheduling approach which combines genetic algorithm with several generic scheduling heuristics for achieving better optimization potential and adaptability in processing large-scale workloads. This approach aims to optimize the performance metrics namely makespan, average flow time, throughput, and average waiting time. The approach is modelled in CloudSimPlus framwork and the experimental results indicate that the proposed hybrid approach consistently outperforms the individual heuristics in terms of the stated metrics irrespective of the workload scale. It is also observed that the optimization potential tends to increase as the workload scale becomes heavier and using flow time as the objective function produces complementary effects on the other metrics.

Keywords: cloud computing, task scheduling, scheduling heuristics, metaheuristics, genetic algorithm, performance metrics.

1. Introduction

Cloud computing enables on-demand and convenient access to computing resources like hardware, software, and platforms over the internet as utility services. This enables a large number of users to configure and utilize these resources based on their respective requirements (Aladwani, 2020). These virtualized computing resources are dynamically scalable and are billed based on the pay-per-use business model (Jain & Upadhyay, 2017). The performance of cloud computing is determined by resource allocation and appropriate scheduling (Venu, 2020). Any unit of work that is to be performed within a certain amount of time can be regarded as a task. Task scheduling is an NP-complete problem and thus, a good task scheduling approach is expected to map the tasks to specific VMs efficiently based on specific performance criteria (Mathew et al., 2014). Scheduling is the practice of allocating tasks among resources in a finite amount of time. The tasks are scheduled in different ways, to optimize certain objective functions (Kalra & Singh, 2015). In the cloud computing environment, the process of scheduling happens at the host level wherein VMs are assigned among hosts known as VM allocation, and at the VM level wherein the tasks are mapped among the VMs known as task scheduling (Soltani et al., 2017). This paper focuses on the field of task scheduling among VMs.

The scientific community has been proposing several scheduling strategies for the last two decades. There exist generic scheduling heuristics and their improved versions, generic metaheuristic optimization-based scheduling heuristics, and eventually, hybrid scheduling strategies wherein scheduling heuristics and metaheuristic optimization methods are combined. Different scheduling heuristics exhibit different behaviour for different metrics. Furthermore, a specific scheduling heuristic may perform better in terms of one metric and may perform worse in terms of another metric and the performance metrics can also be conflicting or complementing in nature. In most of hybrid scheduling strategies discussed in the scientific community, only a few scheduling heuristics are considered. Hence, this paper proposes a hybrid task scheduling strategy that aims to incorporate a number of scheduling heuristics within a metaheuristic optimization method and intends to observe how this strategy would perform based on certain performance metrics. In other words, a metaheuristic optimization approach is used to optimize the sequence or order in which multiple scheduling heuristics must run considering certain performance metrics. Most of the existing scheduling approaches in the scientific community are evaluated on smaller problem instances comprising a fewer number of tasks. However, the performance of scheduling on large-scale workloads is relevant and requires investigation. Hence, an adaptive job scheduling mechanism on large-scale workloads is significant. This paper intends to evaluate the proposed hybrid task scheduling approach using large-scale problem instances so as to examine its optimization potential while increasing the workload scale. This paper intends to translate and apply the hybrid design discussed in the research paper (Nahhas et al., 2021) into a task scheduling problem. This hybrid design is translated by combining several scheduling heuristics and a metaheuristic optimization technique where the metaheuristic optimization is applied for running these heuristics. This approach is evaluated in terms of certain performance metrics using large-scale workload traces. Thus, this paper intends to address and investigate the following research questions:

1. How does the proposed hybrid task scheduling approach perform on large-scale task scheduling problems in comparison to individual baseline scheduling heuristics ?
2. Which objective values are relevant for task scheduling in the cloud computing environment ?

The first section provides an introduction to the cloud task scheduling research problem, specifying the motivation behind the proposed model and addresses the relevant research questions. The second section is the literature research which discusses the prominent scheduling heuristics, metaheuristic and hybrid scheduling approaches discussed in the domain of cloud task scheduling. The third section presents the conceptual model of our proposed hybrid task scheduling strategy in detail and the fourth section discusses the evaluation of our approach through comparative analysis of the experimentation results. The fifth section points out the conclusion and future prospects of the research work.

2. Related work

Over the last two decades, the scientific community produced numerous contributions in the context of task scheduling in the cloud computing environment. The related work has been conducted and studied in two phases. In the first phase, the research articles evaluating and comparing several generic scheduling heuristics in terms of certain performance metrics are focused. In the second phase, the research articles proposing hybrid scheduling methods which combines metaheuristic optimization with generic scheduling heuristics are discussed.

2.1. Heuristics techniques

This section mainly describes the relative performance analysis of different scheduling heuristics in terms of certain performance metrics in the cloud computing environment. Heuristics are problem-solving techniques for finding an approximate solution within a given time period. However, the solutions obtained from heuristic methods are often satisfactory and may not be optimal (Chen, 2018). The main purpose of conducting this literature research is to obtain certain base-line scheduling heuristics and prominent task scheduling-oriented performance metrics.

Jemina Priyadarsini and Lawrence (2014) compared the performance of Max-Min and Min-Min heuristic in terms of makespan and it is observed that Max-Min heuristic achieves better makespan when compared with Min-Min heuristic. Sharma et al. (2017) performed a relative performance analysis of Min-Min and Max-Min heuristics with respect to makespan. The authors concluded that the Max-Min heuristic outperforms the Min-Min heuristic for makespan when the large-sized tasks are greater in number than the short-sized tasks and vice-versa. This implies that size of the tasks has an effect on the performance of heuristics. Madni et al. (2017) compared the performance of six heuristics namely First Come First Serve (FCFS), Minimum Completion Time (MCT), Minimum Execution Time (MET), Max-Min, Min-Min, and Sufferage in terms of cost, degree of imbalance, makespan, and throughput. The experiments conducted in both homogeneous and heterogeneous environments using large-scale workload concluded that Min-Min heuristic performed better than other heuristics, followed by Max-Min and Sufferage and their performance varied based on the considered workload. Sindhu and Mukherjee (2011) presented two scheduling heuristics namely Longest Cloudlet Fastest Processing Element (LCFP) and Shortest Cloudlet

Fastest Processing Element (SCFP). In LCFP, the larger tasks are mapped to resources with high computational capacity whereas in SCFP, the shorter tasks are assigned to resources with high computational capacity. Experiments were conducted with respect to makespan whose results indicated that these heuristics exhibited similar performances at a smaller number of tasks and LCFP exhibited better performance relatively at larger number of tasks. This again implies that the scale of the workload is a matter of concern. Streit (2002) proposed a self-tuning job scheduler known as dynP job scheduler with a dynamic switching policy to switch the scheduling policy during the run time. The authors developed a simple and advanced decider based on the scheduling policies namely FCFS, SJF, LJF and observed that the advanced dynamic decider achieved better performance relative to the simple one based on the performance metrics like utilization, average response time. This indicates the advantage produced by dynamic switching of heuristics. Aladwani (2020) compared the scheduling heuristics namely FCFS, Shortest Job First (SJF), and Max-Min in terms of total waiting time and total finish time (makespan). The paper concludes that among all the heuristics, SJF heuristic performs the best in terms of total waiting time and makespan. Bandaranayake et al. (2020) proposed a new scheduling strategy called Total Resource Execution Time Aware Algorithm (TRETA) with the main goal of minimizing the makespan. The proposed approach suggest that the total execution time of computing resource is a crucial factor for identifying an optimal schedule. This scheduling strategy is compared and evaluated against the other heuristics like Min-Min, Max-Min, FCFS and MCT in terms of the performance metrics makespan, degree of imbalance and throughput using the large real-world NASA workload logs. Experimental results depicted that the proposed approach shows a significant amount of improvement in the aforementioned metrics relatively. These two research works indicate that different scheduling heuristics perform differently in terms of different metrics.

All these above research works indicate that the performance of the scheduling heuristics depends on the considered performance metrics and the nature of the workload. This justifies the need for an adaptive scheduling mechanism involving a number of scheduling heuristics that would optimize different metrics. Consequently, certain base-line scheduling heuristics and task scheduling-oriented performance metrics are identified from the conducted literature research.

2.2. Metaheuristic and Hybrid Techniques

Meta-heuristic is an iterative approach that guides a subordinate heuristic, thus exploring and exploiting search space with the aim of finding optimal solutions (Christiansen & Fagerholt, 2009). This section mainly discusses some of the research articles which leveraged metaheuristic and hybrid techniques for task scheduling optimization in the cloud environment.

Abdi et al. (2014) proposed a modified Particle Swarm Optimization (PSO) in which SCFP heuristic is combined with the standard PSO algorithm. The standard PSO generates the initial population randomly whereas SCFP heuristic is used to generate the initial population in this hybrid approach. It was noticed that the proposed approach minimized the makespan when compared with standard GA and PSO. Dasgupta et al. (2013) proposed a novel load balancing strategy based on genetic algorithm (GA) which must be able to adjust itself as per the dynamic workload situations. The main objective of the proposed strategy is to ensure improved load balance among the resources and to minimize the makespan. Experimentation results indicated that the proposed approach performed better as opposed to FCFS, Round Robin, Stochastic Hill Climbing (SHC). Kaur and Verma (2012) proposed a modified GA that is combined with SCFP and LCFP heuristics. Unlike the standard GA where the initial population is generated randomly, the modified GA generates the initial population with LCFP and SCFP heuristics. Experiment results indicated that the modified GA showed relatively better performance under heavy loads in terms of average makespan and execution cost. Verma and Kumar (2012) proposed an improved version of GA in which the Min-Min, Max-Min heuristics are combined with standard GA such that the initial population is generated with Min-Min and Max-Min heuristics hoping to produce better solutions which in turn provides better future generations on the application of cross over and mutation. The experimental results depicted that the makespan of improved GA is less than that of the standard GA. Singh and Kalra (2014) proposed a modified GA where the initial population has been generated using the Enhanced Max-Min heuristic. It was observed that this proposed modified GA outperforms the other modified GA where the initial population was generated with LCFP and SCFP heuristics as discussed in (Kaur & Verma, 2012) and the other modified GA where the initial population has been generated with Min-Min and Max-Min heuristic methods as discussed in (Verma & Kumar, 2012). Nahhas et al. (2021)

presented a hybrid approach that uses both heuristics and metaheuristics approach so as to optimize the VM allocation problem considering resource utilization. These authors suggest that a hybrid load management strategy using heuristics and GA-based optimization approaches consumes less power in data centers when compared to a specific or generic approach. The authors considered various VM allocation policies for the hybrid approach which were encoded as integers in the chromosomes of the GA-based optimization model such that each chromosome represents a specific combination of VM allocation policies. The paper concludes that the proposed hybrid approach results in a significant reduction in energy consumption, appreciable improvement in VM migrations, and a slight increment in SLA violations as opposed to individual VM allocation policies.

From the related work, prominent baseline scheduling heuristics, metaheuristic optimization methods and relevant performance metrics are identified. Certain scheduling heuristics identified in the literature research are used as input heuristics in our hybrid approach as they are widely used in different research articles. The GA-based optimization model discussed in (Nahhas et al., 2021) is adopted for optimization in our approach. It is also evident that most of the research articles evaluated their scheduling approaches on smaller problem instances and only a few research articles utilized large-scale workloads for evaluating their respective scheduling strategies. Also, the research paper (Bandaranayake et al., 2020) that utilized large-scale workloads for evaluating their approach against prominent scheduling heuristics in terms of certain metrics is identified as the reference paper for comparison. Hence, the cloud infrastructure setup, performance metrics, heuristics and large-scale workload discussed in this paper will be used for evaluating our proposed hybrid task scheduling strategy.

3. Conceptual Hybrid Scheduling Model

This section presents the proposed conceptual hybrid model to optimize the task scheduling problem in the cloud computing environment. This hybrid design is adopted from the (Nahhas et al., 2021) where it is applied and evaluated on the VM placement problem. This paper translates this design into the context of task scheduling as a hybrid task scheduling approach. This hybrid approach uses a metaheuristic optimization technique to optimize and determine the sequence in which several scheduling heuristics must run.

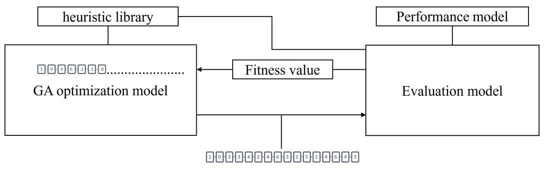

Figure 1. Hybrid scheduling model

The hybrid task scheduling approach enriches the adaptive component of the hybrid load management framework described in (Nahhas et al., 2019). Nahhas et al. (2021) applied the hybrid design on the VM placement problem in which VM allocation policies were used as inputs to the GA-based optimization whereas in our approach, different task scheduling heuristics are used as inputs to the model. Hence, our proposed approach also performs optimization through GA with the objective of identifying better combination of heuristics for efficient task scheduling. This GA-based optimization drives towards finding better combination of scheduling heuristics as a certain number of generations pass by. Figure 1 depicts the abstract representation of the hybrid scheduling model which consists of heuristic library, performance model, GA optimization model and evaluation model. The heuristic library contains the list of heuristics encoded as integers, performance model contains the task scheduling oriented performance measures. The GA optimization model and evaluation model work in tandem, based on the fitness value, thereby driving the optimization process. The GA optimization model generates a population of chromosomes or solution candidates. Each chromosome or solution candidate follows integer encoding in which each integer represents a specific scheduling heuristic which will be discussed in detail in the following section. This population gets evaluated with respect to a objective or fitness function generations after generations in a continuous manner, thereby optimizing towards better solution candidates. This approach needs to be evaluated on large-scale problems so as to validate its adaptive nature and also to investigate the performance of this approach on small-scale and large-scale task scheduling problems. This is done to understand how the proposed approach would perform when subjected to heavier and lighter workloads.

3.1. Genetic Algorithm-based Optimization Model

Based on the conducted literature review, GA-based optimization methods are applied prominently and produced effective results when combined with other generic heuristic approaches. GA offers high-quality solution candidates for optimization and search problems through genetic operations like parent selection, cross-over, and mutation. In this technique, a population of possible solution candidates also known as chromosomes or individuals undergoes evolution over many generations to arrive at better solutions for an optimization problem. Hence, as the evolution progresses, the population of solution candidates gets evolved toward better solution candidates. The basic unit of GA is a chromosome or a solution candidate. A chromosome itself is a collection of genes. In our approach, each gene corresponds to a specific scheduling heuristic, and thus, a chromosome represents a collection of several scheduling heuristics. These scheduling heuristics are encoded as integers from 0 to 7 and since the length of the solution candidate or chromosome is defined as 24, a solution candidate contains 24 integers randomly placed where each integer represents a certain scheduling heuristic. These scheduling heuristics will be switched one after another based on a switching interval which is set as 25s in our approach. When the incoming set of tasks is not able to be completed within the duration of length of the chromosome (24*25s), then the remaining set of tasks will roll back to the beginning of the chromosome. However, the length of the solution candidate, the switching interval, population size and generation size can be regarded as hyper-parameters of the optimization model and these parameters are defined based on several trail and error experimentation.

3.2. Scheduling heuristics for the optimization model

As discussed in the literature review, the following scheduling heuristics are used as inputs to the GA-based optimization model as they are relatively prominent and widely used across different research works.

- First Come First Serve (Aladwani, 2020)
- Shortest Job First (Aladwani, 2020)
- Longest Job First (Streit, 2002)
- Shortest Cloudlet Fastest Processing Element (Sindhu & Mukherjee, 2011)
- Longest Cloudlet Fastest Processing Element (Sindhu & Mukherjee, 2011)
- Min-Min (Sharma et al., 2017) (Bandaranayake et al., 2020)
- Max-Min (Sharma et al., 2017) (Aladwani, 2020)
- Random

3.3. GA-based optimization process

1. As the first step, the GA generates an initial population of N individuals in a random manner. N corresponds to the population size, and N is set as 10. The length of each individual is set as 24. Both these attributes forms hyper-parameters
2. A fitness function should be defined based on the task scheduling problem. This fitness function evaluates each individual of the population in every generation and attaches a fitness value to the individuals
3. Parent Selection process happens based on these fitness values through elitism and parent selection. In elitism, the elite individuals in the population having the best fitness values are directly moved to the next generation. The elite count is set as 3. In tournament selection, the rest of the parent individuals are selected through tournament selection whose count is set as 4.
4. The selected parent individual undergoes any of the cross-over operations namely single-point crossover, random crossover, two-point crossover, and uniform cross-over to produce offsprings. The crossover probability must be lower than the predefined crossover rate for the crossover to occur. The crossover rate is set as 0.5.
5. The generated offsprings then undergo a mutation process wherein the genes within an offspring get exchanged among each other. The mutation probability must be lower than the predefined mutation rate for the mutation to take place. The mutation rate is set as 0.4.
6. The best individual in each generation is identified whose fitness value is compared with previous best fitness value. If its fitness value is better than the previous one, the best fitness value is updated.
7. The process from 2 to 6 continues until the termination condition is reached. The termination condition in our approach is the generation size which is set as 50. The values for different hyper-parameters are defined based on several trial and error experimentation.

3.4. Mathematical representation of the problem

This section describes the scheduling problem through mathematical representation as shown below.

- Let D represents the datacenter and Let DB represents the datacenter broker who manages the datacenter.
- Let C_i = { C_1, C_2 , C_3 , C_4 , ...C_n } indicates the tasks or cloudlets that are incoming to the data center D.
- Let H_i = { H_1, H_2 , H_3 , H_4 , ...H_n } represents the hosts present in the datacenter.
- Let VM_i = { VM_1, VM_2 , VM_3 , VM_4 , ...VM_n } represents the available VMs across all the hosts in the datacenter
- Let T_i denotes the time slices based on which different scheduling heuristics are switched from one to another. This implies that each scheduling heuristic will run for this time slice after which the next scheduling heuristic is selected.
- Let SH_i = { SH_1, SH_2 , SH_3 , SH_4 , ...SH_n } denote the set of scheduling heuristics (i = 0 to 7). Any of the element from this set is assigned to the datacenter broker DB for every T_i time slice.
- Let P represents the population of individuals whose strength is determined by the population size, N.
- Let I_i = { I_1, I_2 , I_3 , I_4 , ...I_n } corresponds to the set of individuals in the population P. Each individual in the population is also called a chromosome or a solution candidate. The length of each chromosome must be predefined which is set as 24.
- Each individual I is a collection of genes represented as { $G_1, G_2, G_3, G_4, G_5, G_6$ ......G_{24} } with a length of 24 as previously stated. Thus, each individual contains 24 genes or elements. Each gene corresponds to an element from SHi. In other words, each gene of the individual corresponds to a particular scheduling heuristic.
- Let FV_i = { FV_1, FV_2 , FV_3 , FV_4 , ...FV_n } represents the set of fitness values of all individuals of the population P. The fitness value of each individual in the population is determined by computing the fitness function.
- Let E_i $(i \in 1, 2, ..n)$ represents the elite individuals of the population P where n = elite count.
- Let TI_i $(i \in 1, 2, ..n)$ represents the tournament individuals of the population P where n = PopSize - elite count.
- Let SI_i = { SI_1, SI_2 , SI_3 , SI_4 , ...SI_n } represents the set of possible solution individuals or solution candidates obtained after the optimization of the GA evolving through many generations. The GA aims to obtain an optimal solution candidate $s \in SI_i$ based on a fitness function. This solution candidate is expected to minimize certain performance metrics.

3.5. Performance metrics

From the literature review conducted, several task scheduling oriented performance metrics were identified such as makespan, total waiting time, throughput, flow time. The objective of the hybrid task scheduling approach is to minimize these performance metrics.

Makespan (in seconds) is defined as the time taken to complete the last task in the set of tasks (finish time of the last task) (Bandaranayake et al., 2020). In other words, it can be defined as the total time taken to finish the execution of a set of jobs or a workload.

$$Makespan = FinishTime_i, \ i = last\ task$$

The throughput defines the number of tasks executed per unit time interval (Bandaranayake et al., 2020).

$$Throughput = \frac{Numberoftasks}{Makespan}$$

Total waiting time (in seconds) is the sum of waiting time of a set of tasks (Aladwani, 2020).

$$Totalwaitingtime = \sum_{n=1}^{totaltasks} WaitingTime$$

$$Averagewaitingtime = \frac{Totalwaitingtime}{No.oftasks}$$

Flow time (in seconds) is defined as the sum of completion time of a set of tasks (Sindhu & Mukherjee, 2011) (Kalra & Singh, 2015).

$$flowtime = \sum_{n=1}^{totaltasks} CompletionTime$$

$$Averageflowtime = \frac{flowTime}{No.oftasks}$$

The objective of the proposed solution is to reduce makespan, average flow time, average waiting time and consequently, increase throughput of the scheduling process. These metrics are considered for evaluating the proposed hybrid task scheduling approach. The fitness function can be framed based on single objective value or multi-objective values. In this paper, the fitness function or the objective function is framed based on single objective value, flow time so as to minimize flow time in the process of scheduling and to observe the consequent effect on other metrics. In fact, other objective functions were also framed based on other metrics and their different kinds of combinations. But, the respective empirical analysis gave better optimization results for objective function based on flow time. Moreover, the study of empirical analysis cannot be included in this paper due to the limited space. Thus, objective or fitness function was framed based on flow time.

4. Experimentation, Results and Analysis

This section describes the experimental setup used for the evaluation of the hybrid task scheduling approach. The individual scheduling heuristics and the proposed hybrid GA-based task scheduling approach are subject to run on the experimental setup to evaluate their comparative performance. The computation results from these simulation experiments enables us to answer the stated research questions.

4.1. Experimental Setup

Table 1. Cloud Infrastructure

Cloud Infrastructures	Value
No. of Datacenters	1
No. of Hosts	20
VmScheduler	Space Shared
No. of VMs	20
CloudletScheduler	Space Shared
VM PEs	128
VM RAM	1024
VM Bandwidth	1000
VM Storage	100000
VM MIPS	1000 - 4000
Cloudlets	NASA workload traces

The cloud infrastructure configuration used for the experiment is referenced from the research paper (Bandaranayake et al., 2020). Our cloud infrastructure comprises a datacenter with 20 hosts, 20 VMs are created across all these 20 hosts such that one host will contain one VM and both the host and VM processing capacity ought to be the same. Four instance types of VMs with different MIPS capacities are provisioned across the hosts. The workload log used for evaluation are referenced from the research paper (Bandaranayake et al., 2020) which is originally taken from the parallel workload archive (Feitelson, 2005). This log contains three months' worth of data as tasks from NASA Ames Research Center's 128-node iPSC/860. It includes basic information about jobs describing their number of nodes, run time, start time, and user information. While modelling in CloudSimPlus, the cloudlet's length corresponds to the job's run time and the cloudlet's required cores correspond to the job's number of nodes. This NASA workload log contains around 18239 jobs which has been batched at different sizes up to 18239 so that it enables us to comprehend how the performance of the proposed approach gets impacted while increasing the workload size and how the hybrid approach would behave at lighter and heavier workload sizes. Experiments are conducted on servers with hardware specifications of Intel Core i5 3570 @ 3.40GHz CPU, 16 GB RAM and storage capacity of 512 GB HDD and 128 GB SSD.

The heuristics FCFS, Min-Min and Max-Min were evaluated in CloudSim Plus considering the same cloud infrastructure setting, the same performance metrics, and the same NASA workload benchmark used in the referenced paper. It is observed that the results showed more or less resemblance with that of the referenced paper (Bandaranayake et al., 2020) for all the performance metrics. This enables us to evaluate our proposed hybrid scheduling strategy in comparison to the heuristics FCFS, Min-Min, Max-Min and author's approach. Moreover, other heuristics namely SJF, LJF, Random, SCFP, LCFP are also considered.

4.2. Evaluation and Comparative analysis of the results

It is already discussed that the proposed hybrid task scheduling approach uses GA as the metaheuristic for optimization and uses flow time as the objective function. This section evaluates the proposed hybrid task scheduling approach against other individual approaches namely FCFS, Random, SJF, LJF, SCFP, LCFP, TRETA, Min-Min and Max-Min in terms of makespan, throughput, average waiting time and average flow time. The heuristics FCFS, Max-Min, Min-Min are used for comparison because the reference paper (Bandaranayake et al., 2020) from which cloud infrastructure setup is adopted, also uses the same for comparison. Furthermore, other heuristics such as SJF, LJF, SCFP, LCFP, Random are also considered. The evaluation is done by comparing on each metric separately, firstly with makespan, then by throughput, followed by average waiting time and average flow time at different workload sizes. This is mainly done to investigate the performance of the hybrid task scheduling approach while increasing the workload size and to observe its performance on lighter and heavier workloads. It is to be noted that TRETA approach is considered for comparison while evaluating on makespan, throughput and is excluded for comparison while evaluating on average waiting time, average flow time as TRETA approach has been only tested on makespan and throughput.

4.3. Evaluating by comparing makespan

Figure 2 shows the graphical plot which depicts the makespan produced by hybrid GA approach and other individual approaches at different workload sizes. The horizontal axis indicates the number of tasks while the vertical axis indicates the makespan in seconds. It is pretty evident from the plot that the proposed hybrid GA approach outperforms other approaches thereby providing lesser makespan values. It is also noted that the rate of increase of makespan for the hybrid GA approach while increasing the workload is relatively less when compared to individual heuristics. This justifies that the hybrid GA approach consistently outperforms all other individual heuristics and TRETA approach in terms of makespan for all workload sizes. The approach also shows stable performance and better optimization potential when the workload amount is increased.

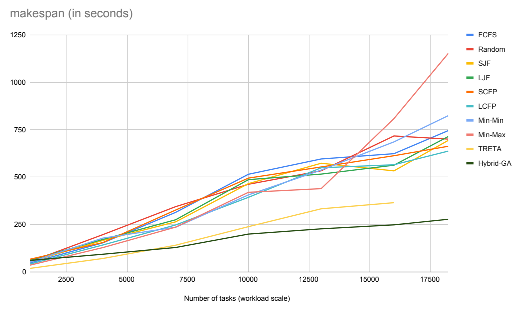

Figure 2. Makespan of hybrid GA approach against other heuristics

4.4. Evaluating by comparing throughput

Figure 3 represents the throughput produced by the proposed hybrid GA approach and other individual approaches at different workload sizes. The horizontal axis indicates the number of tasks while the vertical axis indicates the throughput. It is apparent from the plot that hybrid GA approach produced higher throughput as opposed to other individual heuristics and also, shows better performance in terms of throughput even when the workload amount is increased. It is to be also noted that the throughput shows a steady increase for the hybrid GA approach when the workload amount is uniformly increased as opposed to other individual heuristics and TRETA approach. Thus, it is a clear indication that the hybrid GA approach perform consistently and produce better optimization as the workload becomes heavier.

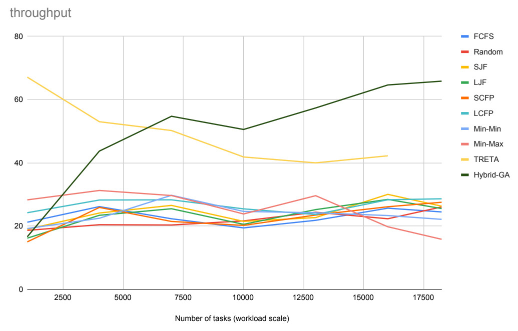

Figure 3. Throughput of hybrid GA approach against other heuristics

4.5. Evaluating by comparing average waiting time

Figure 4 represents the average waiting time produced by the proposed hybrid GA approach and other individual heuristics at different workload sizes. The horizontal axis indicates the number of tasks while the vertical axis indicates the average waiting time in seconds. The plot clearly indicates that there has been a significant reduction in the average waiting time for the proposed hybrid GA approach as opposed to the other individual heuristics. On increasing the workload, while the individual heuristics produced a significant increase in the average waiting time, the hybrid GA approach relatively produced a slight increase in the average waiting time.

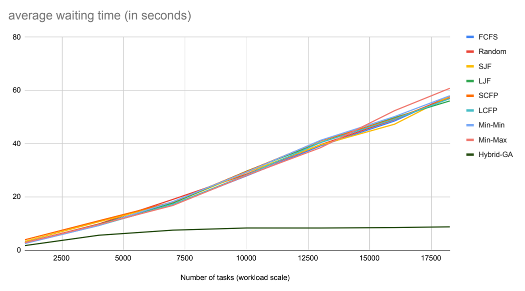

Figure 4. Average waiting time of hybrid GA approach against other heuristics

4.6. Evaluating by comparing average flow time

Figure 5 represents the average flow time produced by the proposed hybrid GA approach and each of the individual heuristics. The horizontal axis indicates the number of tasks while the vertical axis indicates the average flow time in seconds. The plot clearly indicates that there has been a significant reduction in the average flow time for the proposed hybrid GA approach as opposed to the other individual heuristics similar to the previous plot. The rate of increase in average flow time is relatively less for the hybrid approach when compared to the other approaches. This justifies the fact that the proposed hybrid approach produce better optimization potential as the workload becomes heavier.

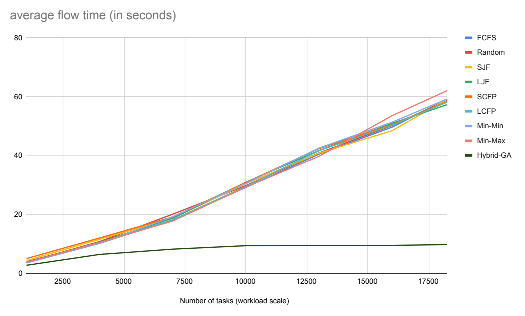

Figure 5. Average flow time of hybrid GA approach against other heuristics

These comparative results justifies the fact that hybrid GA approach consistently outperforms other individual heuristics in terms of the performance metrics namely makespan, throughput, average waiting time and average flow time at both small-scale and large-scale workloads. Furthermore, hybrid GA approach outperforms TRETA approach in terms of makespan and throughput.

5. Conclusion and Future Prospects

In this paper, a hybrid GA-based task scheduling approach with the intention of optimizing the performance metrics namely makespan, throughput, average flow time and average waiting time has been presented. Initially, literature research has been conducted to identify prominent scheduling heuristics, task scheduling-oriented performance metrics, large-scale workload and a reference paper for comparison-based evaluation. The experiments were modelled in CloudSim Plus simulation framework using large-scale workload. Based on the experimental results, it can be concluded that the hybrid GA-based scheduling approach outperforms the individual scheduling heuristics thereby providing better values for makespan, throughput, average waiting time, and average flow time in a consistent manner. Flow time is used as the objective function in our approach. Among the metrics, it is observed that average flow time and average waiting time has been significantly reduced in the proposed approach. This implies that flow time has good complementary effects on other metrics. The proper tuning of hyper-parameters of GA is significant for its performance. Randomly generating the initial population of the GA has a slight adverse effect on the quality of its final solutions where there exists a scope for improvement. Another scope of improvement is the incorporation of more scheduling heuristics in the GA. The switching interval parameter which facilitates the switching among heuristics, is to be reduced while running lighter workloads and is to be increased while running heavier workloads. The hybrid GA approach has been evaluated on independent tasks where there is no precedence among the tasks. Hence, there exists the scope to evaluate the approach on dependent tasks. These form some of the potential areas where there exist scope for improvement and where further research can be conducted.

Therefore, this paper answers the addressed research question stated earlier as follows; the proposed hybrid task scheduling approach consistently outperforms the individual scheduling approaches at small-scale and large-scale workloads. In fact, the proposed hybrid task scheduling approach exhibits stable scheduling performance and better optimization potential on large problem instances. Several performance metrics like makespan, average flow time, throughput, average waiting time can be considered as significant objective values in the context of cloud task scheduling and optimizing on flow time produces complementary effects on other metrics.

References

Abdi, S., Motamedi, S. A., Sharifian, S., et al. (2014). Task scheduling using modified pso algorithm in cloud computing environment. *International conference on machine learning, electrical and mechanical engineering*, *4*(1), 8–12.

Aladwani, T. (2020). Types of task scheduling algorithms in cloud computing environment. *Scheduling Problems-New Applications and Trends*, 145–152.

Bandaranayake, K., Jayasena, K., & Kumara, B. (2020). An efficient task scheduling algorithm using total resource execution time aware algorithm in cloud computing. *2020 IEEE International Conference on Smart Cloud (SmartCloud)*, 29–34.

Chen, J. (2018). *Heuristics*. https://www.investopedia.com / terms / h / heuristics . asp (accessed: 15.08.2021)

Christiansen, M., & Fagerholt, K. (2009). Maritime inventory routing problems. *Encyclopedia of optimization*, *2*, 1947–1955.

Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K., & Dam, S. (2013). A genetic algorithm (ga) based load balancing strategy for cloud computing. *Procedia Technology*, *10*, 340–347.

Feitelson, D. (2005). *Parallel workloads archive*. https://www.cs.huji.ac.il/labs/parallel/workload/ (accessed: 15.04.2021)

Jain, A., & Upadhyay, A. (2017). Cloud scheduling using meta heuristic algorithms. *International Journal of computer sciences and engineering*, *5*(10), 132–139.

Jemina Priyadarsini, R., & Lawrence, D. L. A. (2014). Performance evaluation of min-min and max-min algorithms for job scheduling in federated cloud. *International Journal of Computer Applications*, *99*(18), 47–54.

Kalra, M., & Singh, S. (2015). A review of metaheuristic scheduling techniques in cloud computing. *Egyptian Informatics Journal*, *16*(3), 275–295.

Kaur, S., & Verma, A. (2012). An efficient approach to genetic algorithm for task scheduling in cloud computing environment. *International Journal of Information Technology and Computer Science*, *4*(10), 74–79.

Madni, H., Shafie, A. L., Abdullahi, M., Abdulhamid, S., & Usman, M. (2017). Performance comparison of heuristic algorithms for task scheduling in iaas cloud computing environment. *PLoS ONE*, *12*, 1–26.

Mathew, T., Sekaran, C., & Jose, J. (2014). Study and analysis of various task scheduling algorithms in the cloud computing environment. *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*, 658–664.

Nahhas, A., Bosse, S., Pohl, M., & Turowski, K. (2019). Toward an autonomic and adaptive load management strategy for reducing energy consumption under performance constraints in data centers. *Proceedings of the 9th International Conference on Cloud Computing and Services Science*, 471–478.

Nahhas, A., Cheyyanda, J. T., & Turowski, K. (2021). An adaptive scheduling framework for the dynamic virtual machines placement to reduce energy consumption in cloud data centers. *Proceedings of the 54th Hawaii International Conference on System Sciences*, 878–888.

Sharma, N., Tyagi, S., & Atri, S. (2017). A comparative analysis of min-min and max-min algorithms based on the makespan parameter. *International Journal of Advanced Research in Computer Science*, *8*(3).

Sindhu, S., & Mukherjee, S. (2011). Efficient task scheduling algorithms for cloud computing environment. *High Performance Architecture and Grid Computing*, 79–83.

Singh, S., & Kalra, M. (2014). Scheduling of independent tasks in cloud computing using modified genetic algorithm. *2014 International Conference on Computational Intelligence and Communication Networks*, 565–569.

Soltani, N., Soleimani Neysiani, B., & Barekatain, B. (2017). Heuristic algorithms for task scheduling in cloud computing: A survey. *International Journal of Computer Network and Information Security*, *9*(8), 16–22.

Streit, A. (2002). A self-tuning job scheduler family with dynamic policy switching. *Workshop on Job Scheduling Strategies for Parallel Processing*, 1–23.

Venu, G. (2020). Task scheduling in cloud computing: A survey. *International Journal for Research in Applied Science and Engineering Technology*, *8*(5), 2258–2266.

Verma, A., & Kumar, P. (2012). Independent task scheduling in cloud computing by improved genetic algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering*, *2*(5), 111–114.