QoS-Aware Efficient Tasks Scheduling in Heterogeneous Cloud Computing Environment

Junaid Hassan [a] ✉ , Zeshan Iqbal [b]

[a] Department of Computer Science, UET Taxila, 47040, Pakistan

✉, [a] junaid.hassan@students.uettaxila.edu.pk

[b] zeshan.iqbal@uettaxila.edu.pk

ABSTRACT

The management and prioritization of network traffic to ensure the efficient transmission of important data is achieved through Quality of Service (QoS) technologies and techniques. QoS facilitates allocating necessary resources and bandwidth to critical applications and services while de-prioritizing less important traffic. This is accomplished by classifying and marking packets. Task scheduling involves coordinating and managing the execution of tasks in a computer system or network, including allocating resources and determining the order in which tasks are executed. Task scheduling algorithms use priority, resource requirements, and task dependencies to determine the most efficient way to execute tasks. A heterogeneous cloud environment utilizes multiple cloud computing platforms from different vendors such as IaaS, PaaS, and SaaS to deliver services and optimize cost, performance, and scalability. The task scheduling problem in cloud computing involves effectively mapping workloads to virtual resources. The study introduces the genetic-based algorithm BGA to increase makespan and resource consumption, while SGA focuses on convergence speed. These strategies are compared with current meta-heuristic and heuristic techniques.

Keywords: Cloud Computing; Task Scheduling; Genetic Algorithm; Quality of Service; Bandwidth

1. INTRODUCTION

Cloud computing consists of providing computing facilities such as servers, storage, databases, networking, and intelligence over the Internet, also known as "the cloud". This delivery method offers benefits such as quicker innovation, adaptable resources, and cost savings through economies of scale. Users can access these services from anywhere in the world, on any device with an internet connection, and only pay for what they use [1]. This eliminates the necessity for institutions to invest in and maintain expensive hardware and infrastructure, allowing them to focus on their core business. Cloud computing has several deployment models, including public clouds, operated by third-party providers and offering resources to the general public [2]; private clouds, functioned by a sole organization for its usage; and hybrid clouds, a combination of public and private clouds.

Cloud computing technology encompasses various services, including Infrastructure as a Service, that offer virtual computing resources, such as virtual machines and storage. Platform as a Service provides a platform for developing, operating, and managing applications and services without having to create and maintain the underlying infrastructure. Software as a Service (SaaS) delivers applications over the Internet, enabling users to access software through a web browser [3]. Function as a Service (FaaS) allows running code snippets or functions in response to events without managing infrastructure. Data as a Service (DaaS) provides data-related services, including databases, big data processing, and analytics.

1.1. STAKEHOLDERS IN CLOUD

These technologies are powered by distributed computing, virtualization, and automation, making it possible for organizations to access and use large-scale computing resources as needed. Cloud services are offerings provided by cloud computing providers to their clients on the internet. Some common types of cloud services include Compute services like virtual machines, containers, and server-less computing [4, 5]. Database services like relational databases, NoSQL databases, and managed data services. Networking services like virtual private networks, content delivery networks, and domain name systems. Security services like identity and access management, data encryption, and threat protection. Analytics services like data warehousing, big data processing, and business intelligence. Artificial Intelligence and Machine Learning services, Integration and management services like API management, integration with on-premise systems, and multi-cloud management. Customers can choose and subscribe to these services as per their requirements and only pay for what they use, making it a cost-effective way to access and utilize technology [6, 7]. The cloud stakeholders and their relation are shown in Figure 1.

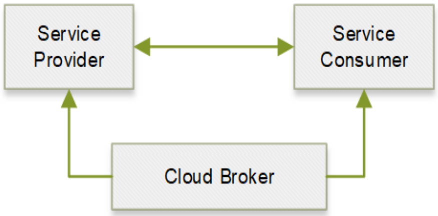


Figure 1: Clouds Stakeholders

The characteristics of cloud services include [8, 9]:

- On-demand self-service: a feature that allows customers to get computing resources whenever they need them, without needing to communicate with a human representative from the service provider.
- Broad network access: enables access to cloud services one can access it from any device as long as one has an internet connection.
- Rapid elasticity: enables customers to quickly scale up or down the number of resources they use, in response to changing demand.
- Measured service: provides usage-based billing, allowing customers to only pay for what they use.
- Scalability: one can manage larger amounts of work by adding more resources when they need them, and removing resources when they don't need them anymore.
- Reliability: ensures that services are highly available and durable, with multiple copies of data stored across multiple locations.

Cloud services are very flexible, scalable, and cost-effective. They allow organizations to concentrate on their main business deeds rather than dealing with complicated IT infrastructure.

1.2. HETEROGENEOUS DISTRIBUTED COMPUTING

Heterogeneous Distributed Computing refers to a system where different types of computing devices, such as servers, laptops, mobile devices, and edge devices, are connected and work together to solve a problem [10]. These devices have different hardware configurations, operating systems, and capabilities, making the system heterogeneous. In heterogeneous distributed computing, each device can contribute its processing power and storage, enabling the system to perform complex tasks more efficiently and quickly than a single device could [11]. The devices can be located in different locations, and connected over a network, such as the internet. Heterogeneous distributed computing is commonly used in areas like scientific research, processing, and cloud computing, where large amounts of required in time processing [12]. It provides the ability to scale processing power and storage as needed and enables organizations to take advantage of the processing power of devices across their network.

1.3 SCHEDULING ISSUES

Scheduling issues include Load balancing: - distributing workloads evenly across available resources to confirm the efficient utilization of resources. Resource allocation determines which resources should be assigned to which tasks, taking into account factors such as processing power, storage capacity, and network bandwidth [5, 8-10]. Task prioritization determines which tasks should be given priority and executed first, based on factors such as deadline, importance, and resource requirements. Resource utilization optimization maximizes the use of available resources to ensure that computing resources are used efficiently and effectively [13]. Fault tolerance ensures that the system continues to operate even in the event of resource failures or other disruptions. Resource negotiation negotiating with other systems or providers for additional resources as needed, to meet changing demands.

To address these scheduling issues, various algorithms and techniques have been developed, including static scheduling, dynamic scheduling, and meta-scheduling [14]. Scheduling issues in cloud computing virtual machines refer to the challenges of allocating and managing virtual machines (VMs) effectively in a cloud environment [15]. Some of the common scheduling issues in cloud computing VMs include [16, 17]:

- Overloading: ensuring that VMs are not overloaded with too many tasks, which can lead to reduced performance and increased downtime.
- Resource allocation: determining how much computing power, memory, and storage to give to each virtual machine (VM)
- Task prioritization: deciding which VMs should be given priority and receive the most resources, based on factors such as deadline, importance, and resource requirements.
- Energy efficiency: optimizing the utilization of computing to reduce energy usage and improve sustainability.
- Resource utilization optimization: making sure that VMs are using resources effectively and efficiently.
- Load balancing: distributing workloads evenly across available resources to avoid overloading any single resource.
- Fault tolerance: ensuring that VMs continue to operate even in the event of failures or other disruptions.
- To address these scheduling issues, various algorithms and techniques have been developed, including dynamic scheduling, resource allocation, and task prioritization algorithms.

1.4 TAXONOMY OF TASKS

A taxonomy of tasks in cloud computing virtual machines (VMs) refers to a classification system that groups tasks into categories based on common characteristics. The purpose of a task taxonomy is to provide a systematic way of understanding the different types of tasks that can be run on cloud VMs [18, 19]. Some types of schedulers in cloud environments and their techniques for task scheduling are shown in Figure 2. Batch processing: tasks that run and require a huge amount of computing resources. Examples include data processing, scientific simulations, and image rendering. Interactive applications: tasks that require rapid response times and are executed in real-time. Examples include web applications, mobile apps, and gaming. Big data processing: tasks that involve processing large amounts of data, and utilization of technology like Hadoop and Spark. Machine learning: tasks that involve training machine learning models and running predictions. High-performance computing: tasks that require high performance in a reasonable resource. Low-latency tasks: tasks that require low response times and are executed in real-time. Examples include network protocols and streaming applications. Microservices tasks that are executed as small, independent services that communicate with other services over a network. This taxonomy can be used as a basis for designing and implementing cloud computing VM scheduling algorithms and techniques, and for choosing the appropriate type of VM for a given task [20].

1.5 META-HEURISTIC ALGORITHMS

Evolutionary algorithms (EAs) are a type of computer program that helps solve optimization problems by simulating the process of natural selection and evolution, which occur in biology. Eas

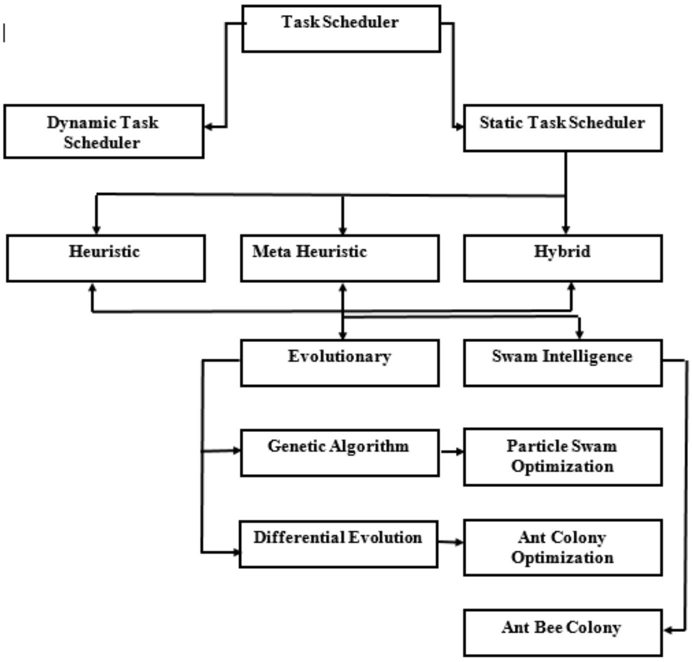


Figure 2: Some Type of Schedulers in Cloud Environment and Their Techniques for Tasks Scheduling

are used for the solution of complex issues by simulating the procedure of evolution, including reproduction, mutation, and selection. Some common characteristics of evolutionary algorithms include Population-Based: EAs operate on a population of potential solutions, called individuals, rather than a single solution. Fitness-based selection of individuals with higher fitness scores. Reproduction and variation reproduction and variation operators are used to generate new individuals, which can then be evaluated for fitness and added to the population. Repeat until Convergence is the process of selection, reproduction, and variation is repeated up to the conditions either s satisfying solution is established or an execution criterion is met [21].

EAs are being used in a large range of optimization problems, which include scheduling, resource tion, and machine learning. Some popular EA algorithms include Particle Swarm Optimization and Genetic Algorithms (GA) EAs are good for problems with large search spaces, complex constraints, and uncertain or noisy fitness functions [22].

2. RELATED WORK

A variety of strategies are under the use of cloud computing for autonomous task scheduling [23]. The chapter on the literature review extensively examines all conceivable relevant methodologies, which might be heuristic, meta-heuristic, or hybrid.

2.1 HEURISTICS ALGORITHMS

Heuristics algorithms are a type of problem-solving method that uses intuition and experience-based knowledge to find a solution. Heuristics algorithms are used for various problems, including resource allocation, load balancing, and quality of service (QoS) management. One study that focuses on heuristics algorithms for QoS management in cloud computing is [24]. The algorithm uses an adaptive mutation strategy to balance the trade-off between resource utilization and service quality. Another study focuses on heuristics algorithms for cloud computing [25]. The authors proposed an algorithm which is a hybrid heuristic that combines the GA and PSO to allocate resources in cloud computing. The algorithm uses an adaptive mutation strategy to balance the trade-off between resource utilization and service quality. Heuristics algorithms are mostly used in cloud computing for the solution of different problems, including load balancing, and QoS management. The studies discussed in this review show that heuristics algorithms can effectively optimize resource utilization and service quality in cloud computing. Heuristics algorithms are problem-solving methods that use intuition and experience-based knowledge to find solutions, making them well-suited to the dynamic and complex environment of cloud computing. One of the earliest studies that focuses on heuristics algorithms for cloud computing [26-28]. The algorithm uses the adaptive mutation strategy to balance the trade-off between resource utilization and service quality. Since then, many researchers have explored the use of heuristics algorithms in cloud computing, including load balancing and QoS management. For example [29, 30], the authors proposed an algorithm called a hybrid heuristic that combines the PSO and GA optimization to allocate resources in the field of cloud computing. In these years, the rallying on using heuristics algorithms in cloud computing to handle big data has increased. For example [31, 32], the authors proposed a heuristics algorithm for big data resource allocation in cloud computing that handle both computational resources and network resources. These algorithms used an adaptive mutation strategy to balance the trade-off between resource utilization and service quality [33]. With the increasing demand for cloud computing to handle big data, the usage of heuristics algorithms in the field of cloud computing is expected to continue to grow [34].

2.2 META-HEURISTICS ALGORITHMS

Meta-heuristics algorithms are a class of optimization algorithms that are commonly uses in the field of cloud computing to solve the problem of resource allocation and management. Meta-heuristics algorithms are flexible and scalable, making them well-suited to the dynamic and complex environment of cloud computing. One of the earliest studies that focuses on meta-heuristics algorithms for cloud computing [34-36]. One of the researchers proposed an algorithm called meta-heuristics for better allocation of all concerned resources. The algorithm combines the simulated annealing and PSO for better service quality and utilization of the resources. Results shows the performance of meta-heuristic algorithm is improving in term of utilization of the resources and service quality. The researcher proposes a hybrid meta-heuristics algorithm that combines the GA and optimization of the ant colony for allotment of the cloud computing resources. The algorithm considers the QoS and utilization of the resources, The results depicts that the proposed algorithm performance is improved as per another algorithm.

In this era, task scheduling techniques have become increasingly important in cloud computing. As cloud computing continues to evolve and become an integral part of many businesses, research on task scheduling techniques has grown. In this review, we will discuss the major advances in task scheduling techniques from 2010 to 2022 and their impact on cloud computing. In 2010, research on task scheduling techniques focused on improving the output of existing systems. Some researchers suggested methods for task scheduling that were based on the utilization of multiple cores and grid computing. These techniques were able to upgrade the performance of existing techniques by optimizing the utilization of resources and improving the scalability of the system. In 2012, research began to shift towards developing new techniques for task scheduling that could better utilize the cloud computing environment. This involved developing techniques such as resource virtualization, dynamic resource allocation, and multi-objective optimization. These techniques allowed cloud computing systems to better utilize the available resources and achieve higher levels of scalability [37]. In 2014, research began to focus on developing techniques that could improve the security of cloud computing systems. This involved developing techniques such as authentication and authorization management, secure scheduling of the tasks, and secure allocation of the resources. These techniques allowed cloud computing systems to better protect their data from malicious attackers and better manage the access of users to resources [38]. In 2016, research began to focus on developing techniques that could improve the QoS of cloud computing systems. This included developing techniques for QoS-based task scheduling, adaptive allocation of the resources, and online scheduling of the tasks. These techniques allowed cloud computing systems for better prioritize tasks based on their importance and to dynamically adjust the resources allocated to tasks based on their current demands [39]. In 2018, research began to focus on developing techniques that would allow cloud computing systems to better manage large-scale data processing. This included developing techniques such as graph-based scheduling, machine learning-based scheduling, and distributed scheduling. These techniques allowed cloud computing systems to efficiently process large amounts of data and to make good use of their available resources [39]. In 2020, research began to focus on developing techniques that would allow cloud computing systems to better manage their workloads. This included developing techniques such as self-adaptive scheduling, predictive scheduling, and application-level scheduling. These techniques allowed cloud computing systems to better predict the demands of their workloads and to automatically adjust the resources allocated to them to maximize efficiency. In 2022, research is focusing on developing techniques that could better manage the interactions between cloud computing systems and their users.

This includes developing techniques such as user-level scheduling, resource-level scheduling, and user-oriented scheduling. These techniques allow cloud computing to understand the needs of users and to allocate resources more efficiently [85].

Overall, research on task scheduling techniques has greatly improved the performance level of cloud computing over the past decade. These techniques have enabled cloud computing for better usage of its resources, improve its security, and provide better QoS for its users. As cloud computing continues to evolve, research on task scheduling techniques will continue to remain an important area of research. The latest techniques are listed in Table 1.

Table 1: Summary of the Some Latest Techniques

Ref#	Improvements	Stanchness	Flaws
[8]	Improves makespan.	Adaptive probability of crossover and mutation is Introduced.	Load is imbalanced
[27]	Reduces probability of failure and completion time.	It has multi-objective optimization.	Converges pre-maturely Load balancing not considered.
[40]	Improves makespan and resource utilization.	It has a Greedy Strategy to update vectors. The roulette wheel is used as a selection operator.	Assessed on a small dataset.
[41]	Improves load balancing.	The share of each VM is calculated based on the power and size of jobs.	Low number of jobs.
[19]	Improves makespan.	Tournament selection operation of GA is used.	Not compared with other meta-heuristics. VM Load balancing is balanced.
[42]	Reduces makespan.	Fast Convergence.	Trapped able in local-minima.
[43]	Provides load balancing and reduces makespan.	Two conflicting objectives are combined in a relation to define fitness as minimization function.	High probability of mutation which may lead to pre-mature convergence.
[44]	Improves makespan, resource utilization and Convergence.	Velocity in PSO is updated using Differential Evolution Algorithm.	Not evaluated on any big dataset.
[45]	Improves time and cost of execution.	An archive of dominating and non-dominating particles is maintained.	Fittest particle being chosen does not meet both objects because its fitness function is not unbiased. Exploration ability is limited owing to fittest range depending on multi-objective.
[17]	Reduce execution time of jobs.	LCFP and SCFP are used parallel to randomization for population Initialization.	Not considered load balancing in the objective function. Evaluated on a very small dataset.

2.3 DATASET

Two types of datasets we use in our experiments.

2.3.1 SYNTHETIC DATASET

Synthetic test dataset is a dataset created by computer algorithms or programs. The details of the dataset are shown in Table 2. The data can be generated from real-world data to represent the characteristics of the

existing dataset. Synthetic test datasets are useful for large-scale experiments or for situations when real-world data is not available. Four classes in this dataset are include that are as follow:
Class 0: Customers with high income and low spending
Class 1: Customers with low income and high spending
Class 2: Customers with high income and high spending
Class 3: Customers with low income and low spending

Table 2: Synthetic Dataset Jobs

No	Size
1.	34304
2.	37849
3.	36347
4.	30933
5.	34976
6.	38501
7.	37059
8.	36504
9.	29606
10.	34944

2.3.2 REALISTIC DATASET

A realistic dataset for scheduling of the task would add-up information about the tasks, like, name of the task, duration of the task, dependencies of the task, task priority, task resources needed, task due date, and task completion status. It would also include information about the resources available for completing the tasks, such as: resource name, resource availability, resource cost, resource location, and resource type. The dataset should also include information about the environment in which the tasks will be completed, such as: the current time, the location of the tasks, the current temperature and humidity, and any other environmental factors that may affect the completion of the tasks. Finally, the dataset should include information about the people assigned to the tasks, such as: person name, person availability, person skill set, and person work preferences. The details of the dataset are shown in Table 3.

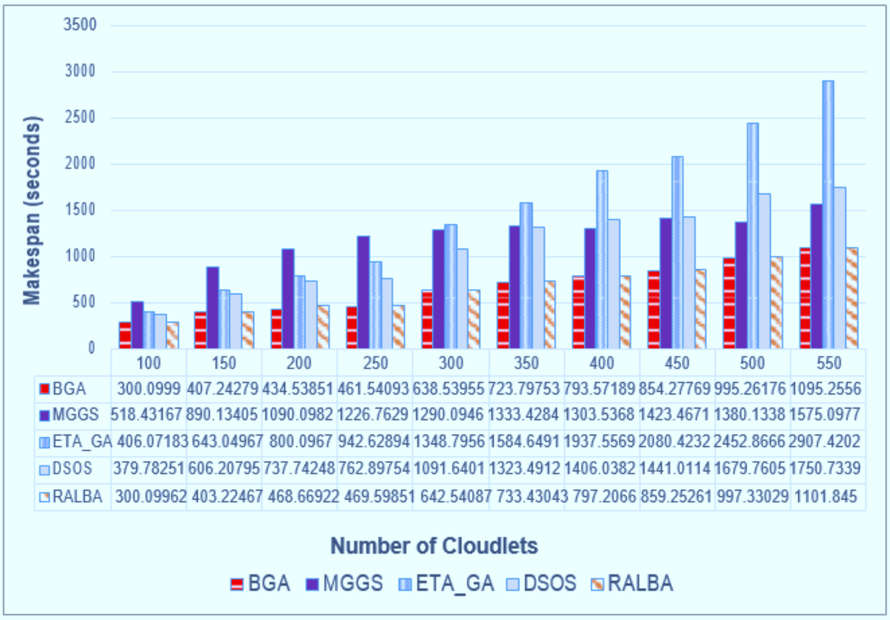
Table 3: Realistic Dataset

No	Size
1.	83000
2.	95000
3.	91000
4.	81000
5.	27500
6.	49000
7.	103000
8.	95000
9.	47000
10.	71000

This article defines the exiting problem and later on its improved solution. The data set is given below with required details. The improved solution is presented for efficient task scheduling. The motive of selecting logical reasoning and algorithms is to improve the load balancing and makespan while enhancement of speed convergence of optimization algorithm. Merging of meta-heuristic and heuristic is for improved task scheduling.

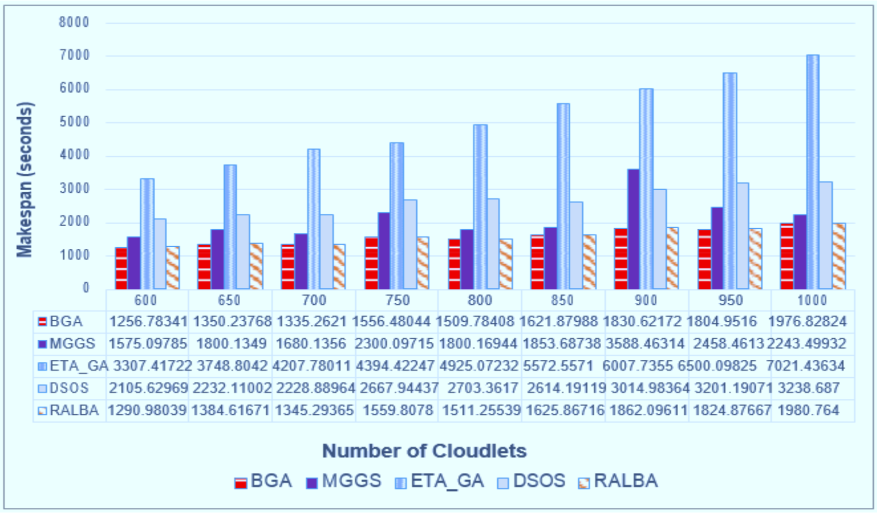
3 EXPERIMENTS AND RESULTS

At first, the experiment to establish the mutation rate of both genetic algorithms is shown. The performance of both algorithms on synthetic and realistic datasets is then shown, along with the relevant commentary. Experiments show the duration and resource use of both strategies. Because the batched sizes in both datasets varied, numerous graphs are created for each batched size. In graphs, the average of all batched for the performance indicators is provided individually as well. It is also indicated if the findings of studied methodologies have improved or declined. On the GoCJ and Synthetic datasets, the makespan of BGA is evaluated, however in the synthetic dataset, all categories, namely Normal as well left, right and uniform workload, are used in experiments. Figure depicts the makespan using the GoCJ dataset on batch sizes ranging from 100-550, with a 50-percentage difference in batched sizes. In most situations, BGA has obtained a shorter makespan. At batch sizes of 100 & 150, both algorithms are almost identical, however in additional batched sizes, BGA technique outperforms MGGS, ETA-GA, DSOS, and RALBA. The ETA-GA does not have a decent makespan as batched size increases, and its results has worsened on batches of more than 300. Figure 3-5 show the makespan using the GoCJ dataset on batched sized 600-1000. On big batch sizes, the ETA-GA dropped even further. The BGA has outperformed all benchmark approaches in terms of makespan. MGGS takes a long time on high batch sizes and improves over time, but BGA still outlasts MGGS. Figure 1 depicts the behavior of the GoCJ in terms of makespan. BGA outperformed RALBA, ETA-GA, DSOS, and MGGS in terms of average makespan by 33.1, 65.5, 40.4, and 1%, respectively.



	100	150	200	250	300	350	400	450	500	550
BGA	300.0999	407.24279	434.53851	461.54093	638.53955	723.79753	793.57189	854.27769	995.26176	1095.2556
MGGS	518.43167	890.13405	1090.0982	1226.7629	1290.0946	1333.4284	1303.5368	1423.4671	1380.1338	1575.0977
ETA_GA	406.07183	643.04967	800.0967	942.62894	1348.7956	1584.6491	1937.5569	2080.4232	2452.8666	2907.4202
DSOS	379.78251	606.20795	737.74248	762.89754	1091.6401	1323.4912	1406.0382	1441.0114	1679.7605	1750.7339
RALBA	300.09962	403.22467	468.66922	469.59851	642.54087	733.43043	797.2066	859.25261	997.33029	1101.845

Figure 1: Makespan on GoCJ 100 to 550



	600	650	700	750	800	850	900	950	1000
BGA	1256.78341	1350.23768	1335.2621	1556.48044	1509.78408	1621.87988	1830.62172	1804.9516	1976.82824
MGGS	1575.09785	1800.1349	1680.1356	2300.09715	1800.16944	1853.68738	3588.46314	2458.4613	2243.49932
ETA_GA	3307.41722	3748.8042	4207.78011	4394.42247	4925.07232	5572.5571	6007.7355	6500.09825	7021.43634
DSOS	2105.62969	2232.11002	2228.88964	2667.94437	2703.3617	2614.19119	3014.98364	3201.19071	3238.687
RALBA	1290.98039	1384.61671	1345.29365	1559.8078	1511.25539	1625.86716	1862.09611	1824.87667	1980.764

Figure 2: Makespan on GoCJ 600 to 1000

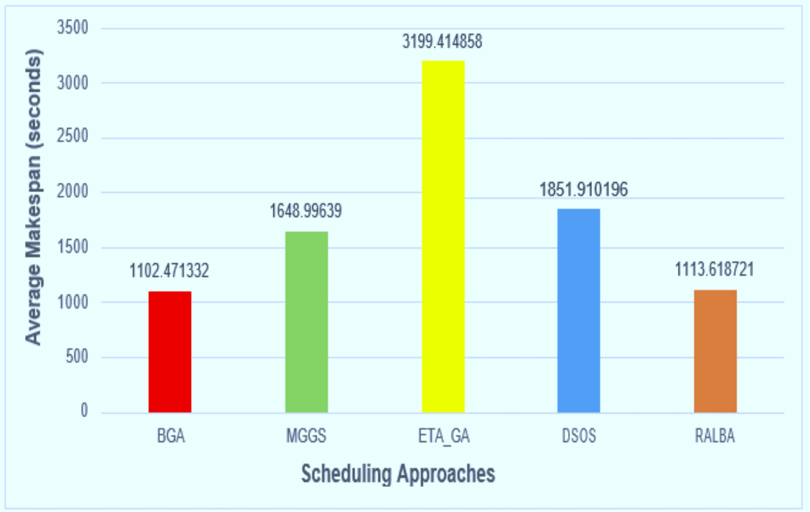
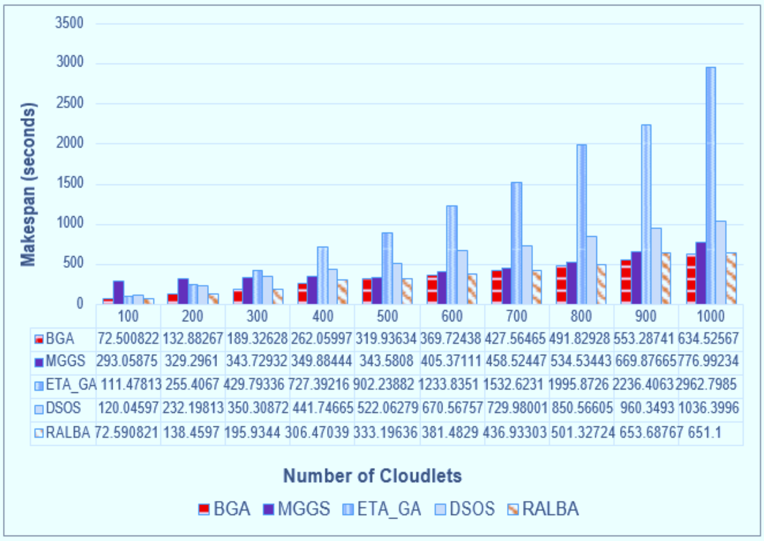


Figure 3: Average Makespan on GoCJ

It is calculated for altogether of synthetic datasets with batched sizes ranging from 100-1000 and a difference of hundred in batched sizes. Figure 6-9 show the results on left-skewed dataset, makespan is usually superior to other approaches. RALBA has a superior makespan for batch sizes of 100 & 1000, whereas BGA has a better makespan overall. On average, the BGA outperformed the DSOS, RALBA, MGGS, and ETA-GA by 27.3, 71.9, 40.5 and 4.6%, correspondingly.



	100	200	300	400	500	600	700	800	900	1000
BGA	72.500822	132.88267	189.32628	262.05997	319.93634	369.72438	427.56465	491.82928	553.28741	634.52567
MGGS	293.05875	329.2961	343.72932	349.88444	343.5808	405.37111	458.52447	534.53443	669.87665	776.99234
ETA_GA	111.47813	255.4067	429.79336	727.39216	902.23882	1233.8351	1532.6231	1995.8726	2236.4063	2962.7985
DSOS	120.04597	232.19813	350.30872	441.74665	522.06279	670.56757	729.98001	850.56605	960.3493	1036.3996
RALBA	72.590821	138.4597	195.9344	306.47039	333.19636	381.4829	436.93303	501.32724	653.68767	651.1

Figure 4: Makespan on Normal

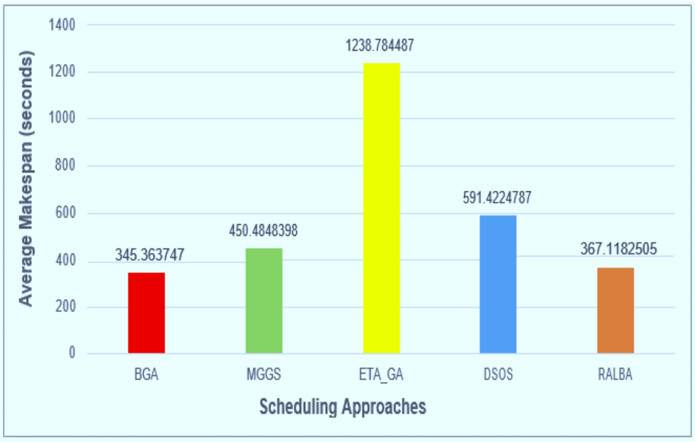


Figure 5: Average Makespan on Normal

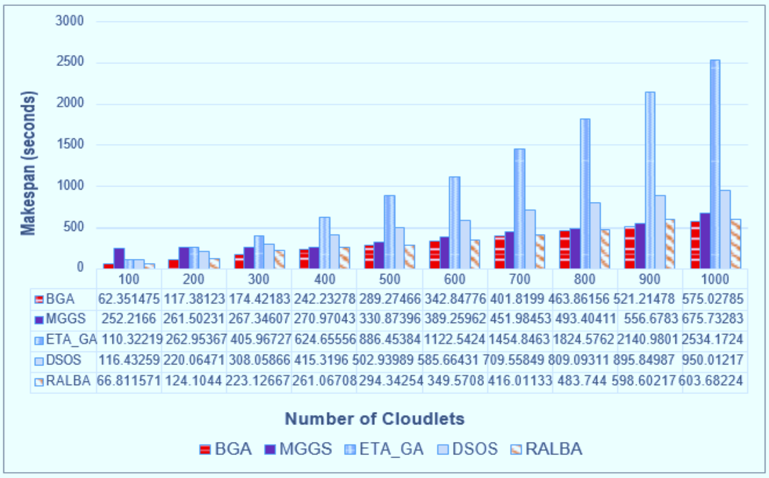


Figure 6: Average Makespan on Normal

The makespan is calculated for altogether of synthetic datasets with batched sizes ranging from 100-1000 and a difference of hundred in batched sizes. Figure 9 depicts the study of makespan on makespan on the left-skewed dataset, is usually superior to other approaches. RALBA has a superior makespan for batch sizes of 100 & 1000, whereas BGA has a better makespan overall. The graph depicts the regular of all batched sizes. The algorithm outperformed the DSOS, RALBA, MGGS, and ETA-GA by 27.3, 71.9, 40.5 and 4.6%, correspondingly. Figure 10 depicts the makespan analysis on a right-skewed dataset.

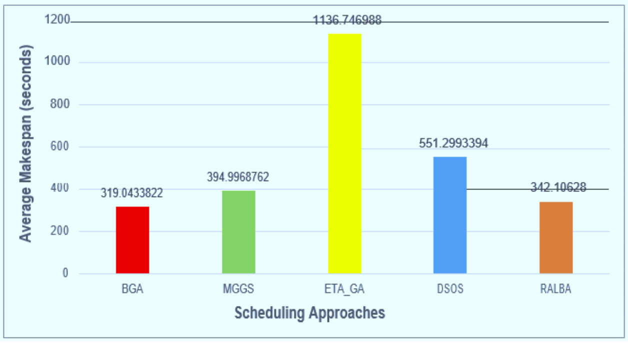


Figure 7: Makespan on Uniform Dataset

4.1 ARUR

The ARUR are a metric used to calculate resources consumption. BGA's improvement is evaluated first on both datasets (GoCJ and Synthetic). Figure 7 showing the ARUR of proposed and other approaches on the both dataset for batched sizes ranging from 100 to 550, with a batch size alteration of 50. On batch sizes 100 and 200, RALBA has a decent ARUR, however BGA excels on larger batched sizes. Figure 11 showing the ARUR analysis on the GoCJ dataset with batch sizes ranging from 600 to 1000. Figure 11 shows that BGA improves ARUR by 30, 80.4, 83.1, and 0.5% when compared to RALBA, ETA-GA, DSOS, and MGGS. On the GoCJ dataset, the BGA offers adequate load balancing for high batch sizes.

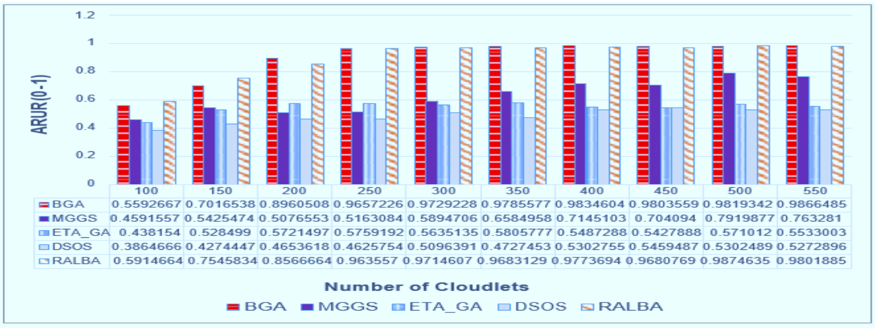
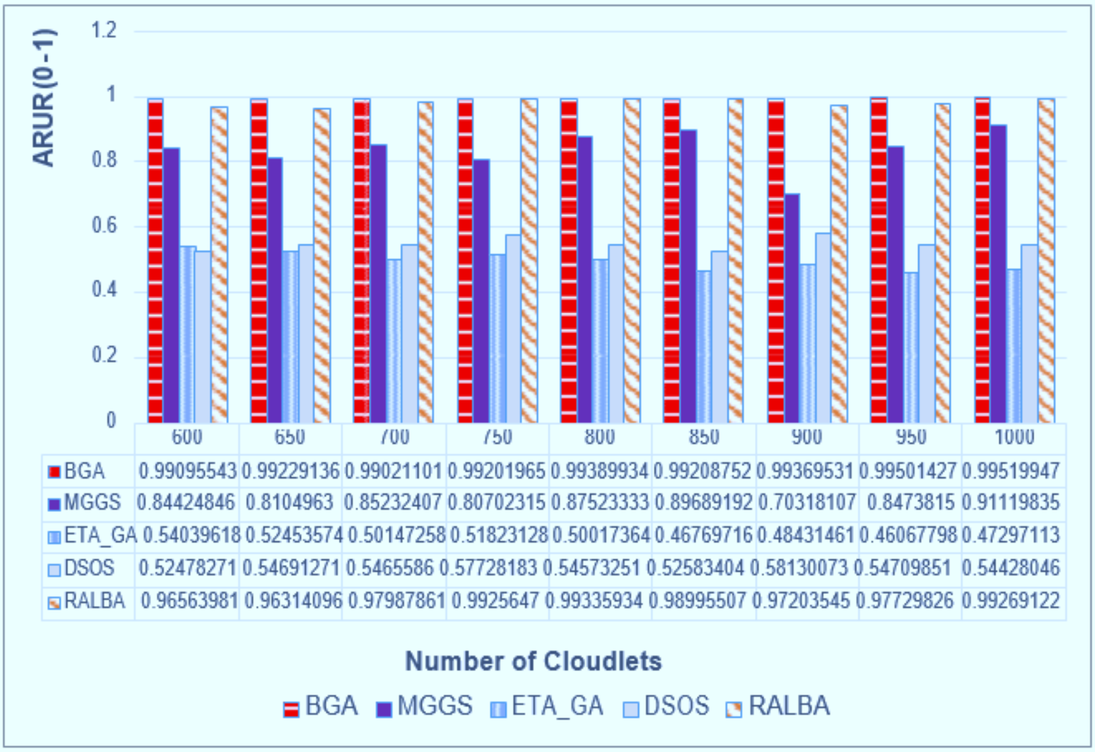


Figure 8: ARUR on GoCJ 100 to 550

The examination on a left-skewed dataset for ARUR demonstrates that BGA increases resource consumption considerably on high batch sizes. Below figures shows the ARUR for batched sizes 100 - 1000, whereas, it is clear that BGA outperforms RALBA, ETA-GA, DSOS, and MGGS in terms of ARUR by 15.2, 77, 60.6, and 5.4%, respectively.

Figures 11 and 12 show the behaviour of BGA and further approaches on a homogenous datasets. BGA improves across all batched sizes. In comparison to DSOS, ETA-GA, MGGS, and RALBA, the percentage of improvement in makespan is 19.2, 71.9, 42.1, and 6.7%, respectively.



	600	650	700	750	800	850	900	950	1000
BGA	0.99095543	0.99229136	0.99021101	0.99201965	0.99389934	0.99208752	0.99369531	0.99501427	0.99519947
MGGS	0.84424846	0.8104963	0.85232407	0.80702315	0.87523333	0.89689192	0.70318107	0.8473815	0.91119835
ETA_GA	0.54039618	0.52453574	0.50147258	0.51823128	0.50017364	0.46769716	0.48431461	0.46067798	0.47297113
DSOS	0.52478271	0.54691271	0.5465586	0.57728183	0.54573251	0.52583404	0.58130073	0.54709851	0.54428046
RALBA	0.96563981	0.96314096	0.97987861	0.9925647	0.99335934	0.98995507	0.97203545	0.97729826	0.99269122

Figure 9: ARUR on GoCJ 600 to 1000

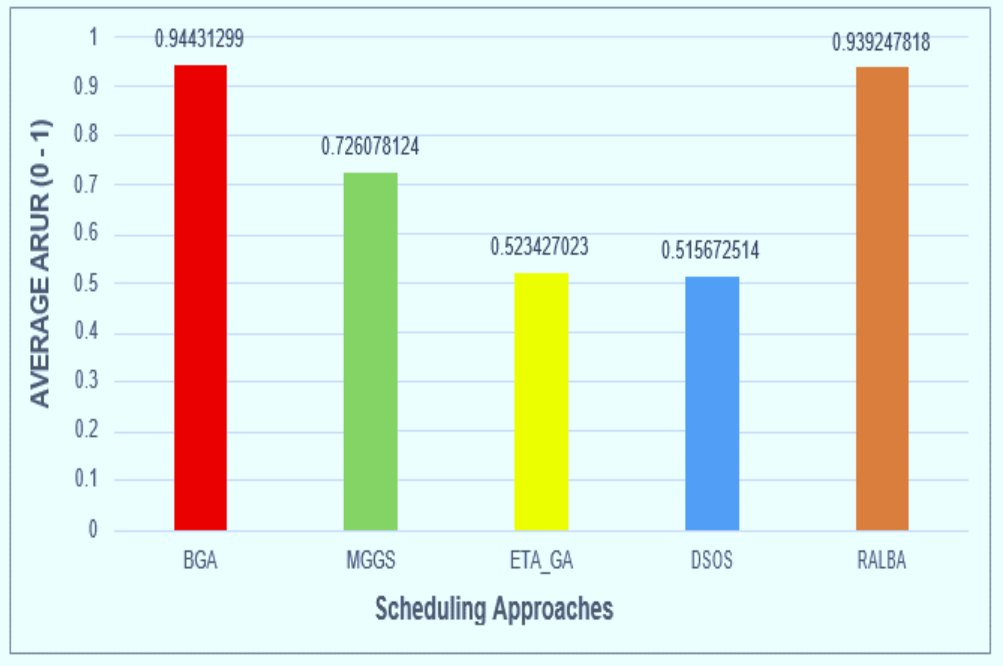
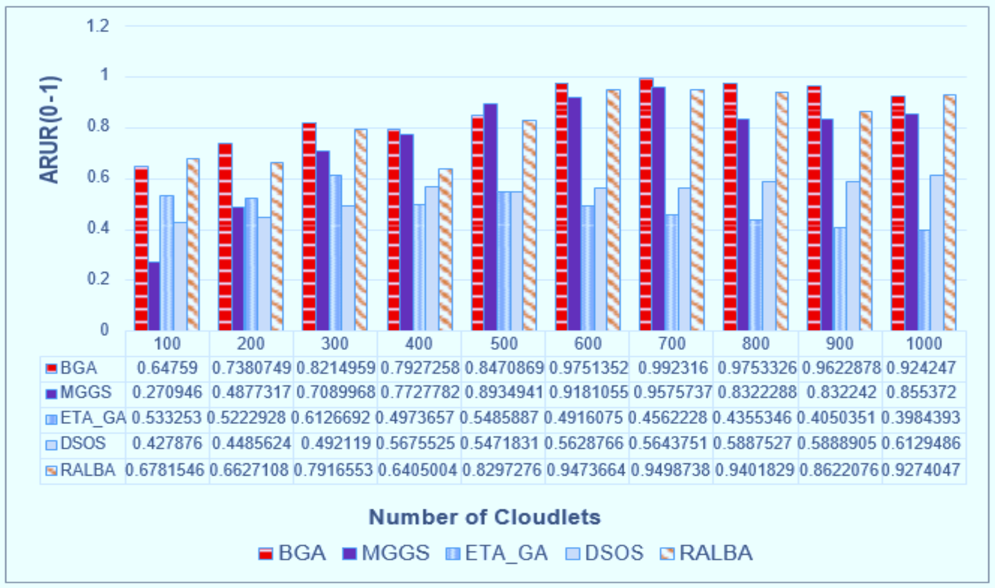


Figure 10: Average of ARUR on GoCJ



	100	200	300	400	500	600	700	800	900	1000
🟥 BGA	0.64759	0.7380749	0.8214959	0.7927258	0.8470869	0.9751352	0.992316	0.9753326	0.9622878	0.924247
■ MGGS	0.270946	0.4877317	0.7089968	0.7727782	0.8934941	0.9181055	0.9575737	0.8322288	0.832242	0.855372
▥ ETA_GA	0.533253	0.5222928	0.6126692	0.4973657	0.5485887	0.4916075	0.4562228	0.4355346	0.4050351	0.3984393
□ DSOS	0.427876	0.4485624	0.492119	0.5675525	0.5471831	0.5628766	0.5643751	0.5887527	0.5888905	0.6129486
▨ RALBA	0.6781546	0.6627108	0.7916553	0.6405004	0.8297276	0.9473664	0.9498738	0.9401829	0.8622076	0.9274047

Figure 11: ARUR on Left Skewed

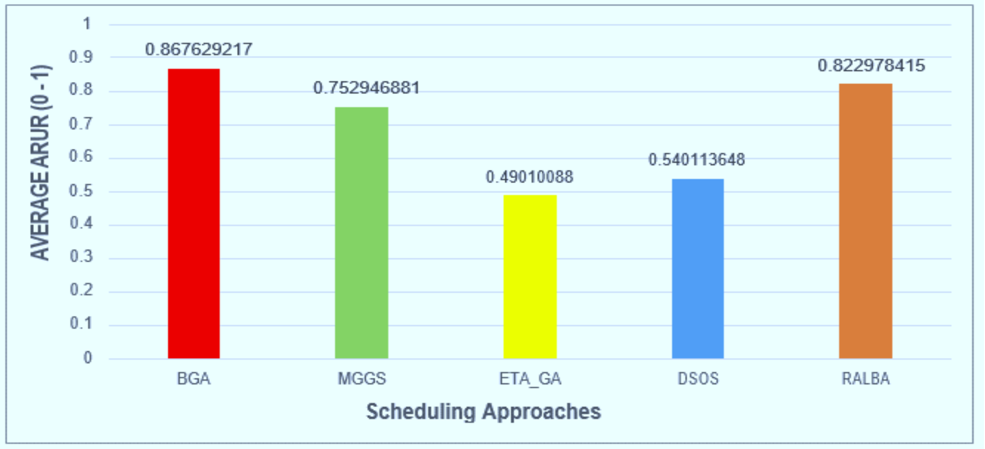
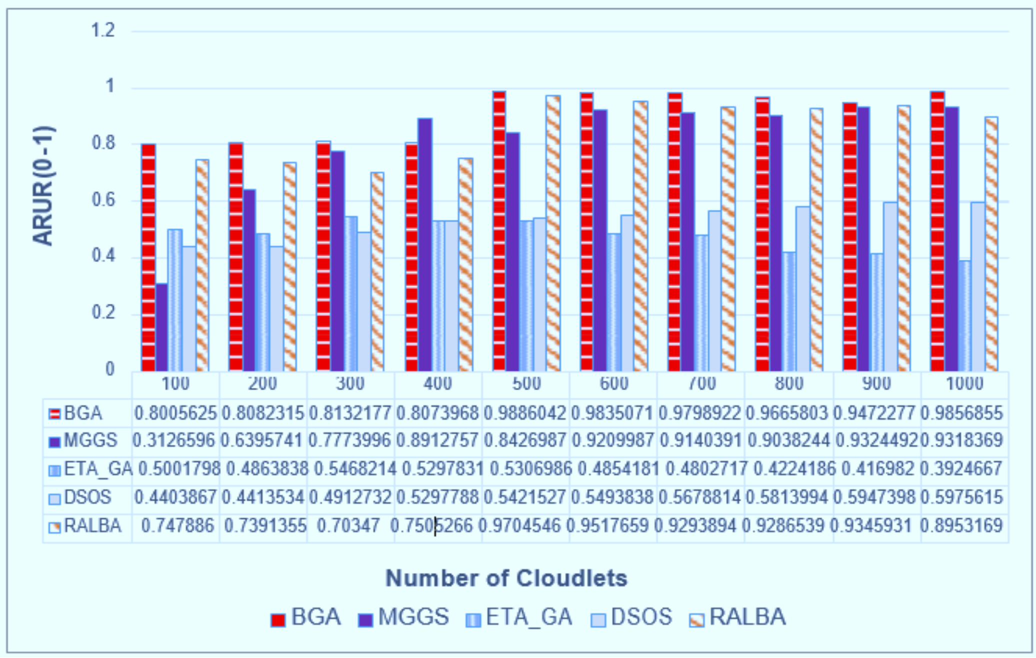


Figure 12: Average of ARUR on Left Skewed



	100	200	300	400	500	600	700	800	900	1000
▤ BGA	0.8005625	0.8082315	0.8132177	0.8073968	0.9886042	0.9835071	0.9798922	0.9665803	0.9472277	0.9856855
■ MGGS	0.3126596	0.6395741	0.7773996	0.8912757	0.8426987	0.9209987	0.9140391	0.9038244	0.9324492	0.9318369
▥ ETA_GA	0.5001798	0.4863838	0.5468214	0.5297831	0.5306986	0.4854181	0.4802717	0.4224186	0.416982	0.3924667
□ DSOS	0.4403867	0.4413534	0.4912732	0.5297788	0.5421527	0.5493838	0.5678814	0.5813994	0.5947398	0.5975615
▨ RALBA	0.747886	0.7391355	0.70347	0.7505266	0.9704546	0.9517659	0.9293894	0.9286539	0.9345931	0.8953169

Figure 13: ARUR on Right Skewed

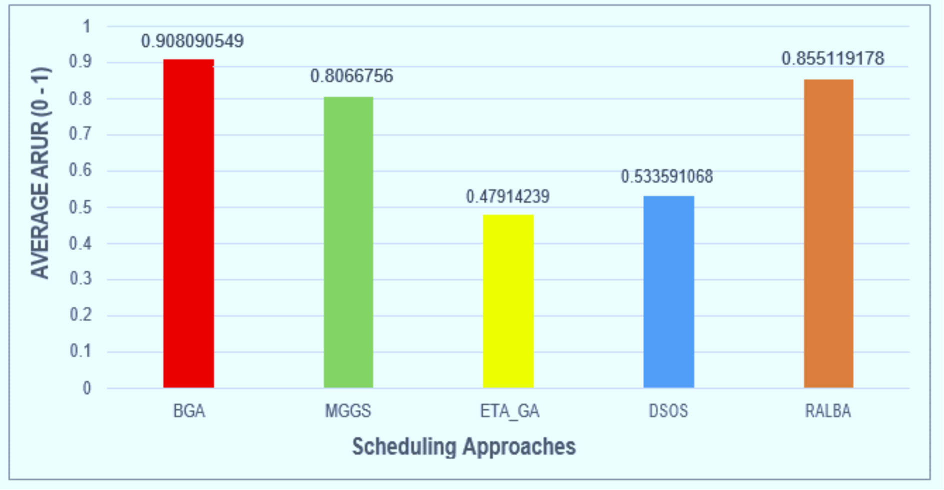
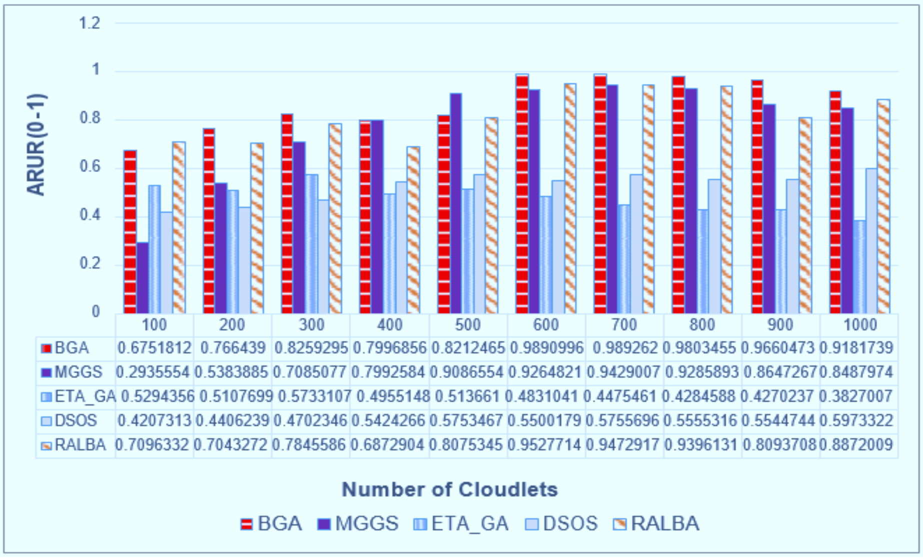


Figure 14: Average of ARUR on Right Skewed



	100	200	300	400	500	600	700	800	900	1000
BGA	0.6751812	0.766439	0.8259295	0.7996856	0.8212465	0.9890996	0.989262	0.9803455	0.9660473	0.9181739
MGGS	0.2935554	0.5383885	0.7085077	0.7992584	0.9086554	0.9264821	0.9429007	0.9285893	0.8647267	0.8487974
ETA_GA	0.5294356	0.5107699	0.5733107	0.4955148	0.513661	0.4831041	0.4475461	0.4284588	0.4270237	0.3827007
DSOS	0.4207313	0.4406239	0.4702346	0.5424266	0.5753467	0.5500179	0.5755696	0.5555316	0.5544744	0.5973322
RALBA	0.7096332	0.7043272	0.7845586	0.6872904	0.8075345	0.9527714	0.9472917	0.9396131	0.8093708	0.8872009

Figure 15: ARUR on Normal

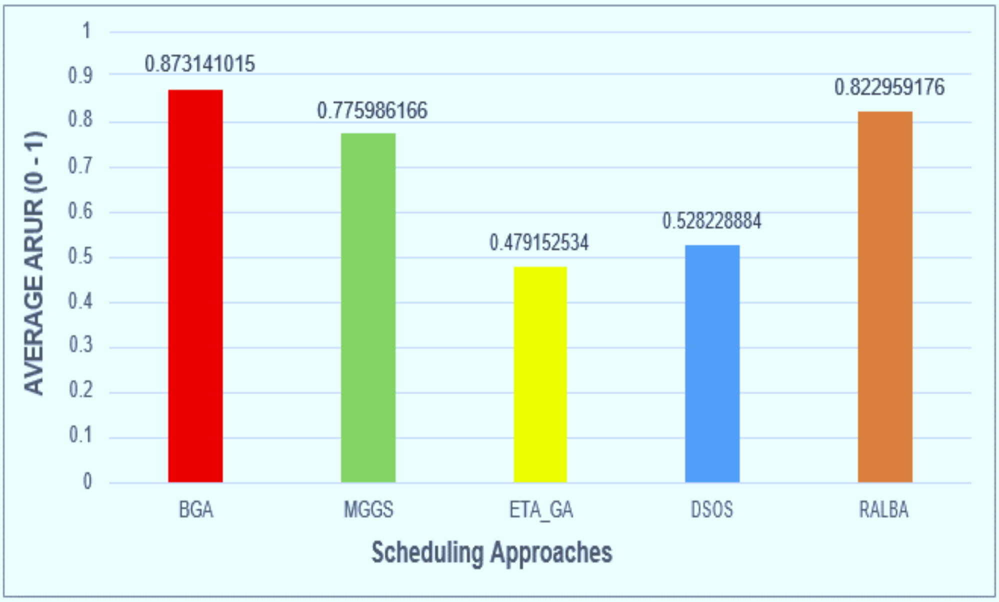
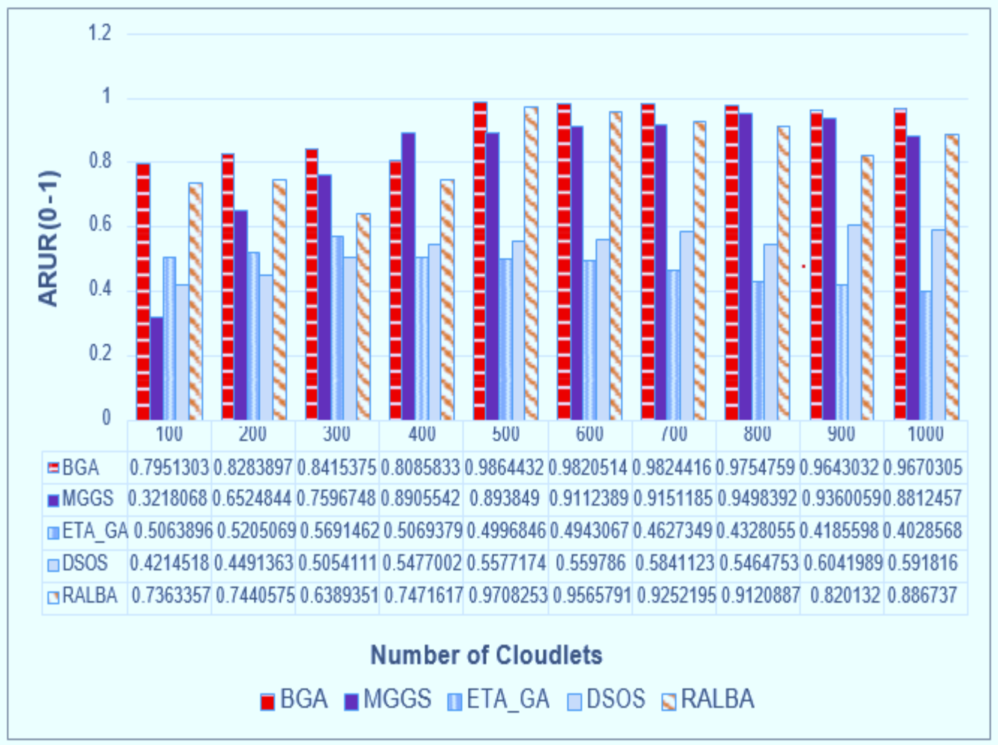


Figure 16: Average of ARUR on Normal



	100	200	300	400	500	600	700	800	900	1000
BGA	0.7951303	0.8283897	0.8415375	0.8085833	0.9864432	0.9820514	0.9824416	0.9754759	0.9643032	0.9670305
MGGS	0.3218068	0.6524844	0.7596748	0.8905542	0.893849	0.9112389	0.9151185	0.9498392	0.9360059	0.8812457
ETA_GA	0.5063896	0.5205069	0.5691462	0.5069379	0.4996846	0.4943067	0.4627349	0.4328055	0.4185598	0.4028568
DSOS	0.4214518	0.4491363	0.5054111	0.5477002	0.5577174	0.559786	0.5841123	0.5464753	0.6041989	0.591816
RALBA	0.7363357	0.7440575	0.6389351	0.7471617	0.9708253	0.9565791	0.9252195	0.9120887	0.820132	0.886737

Figure 17: ARUR on Uniform

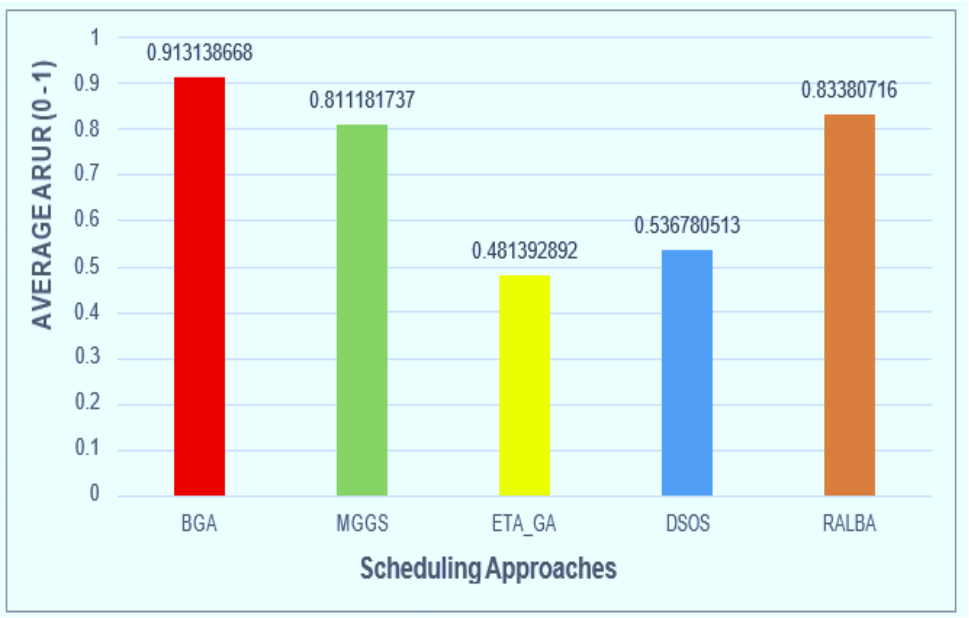
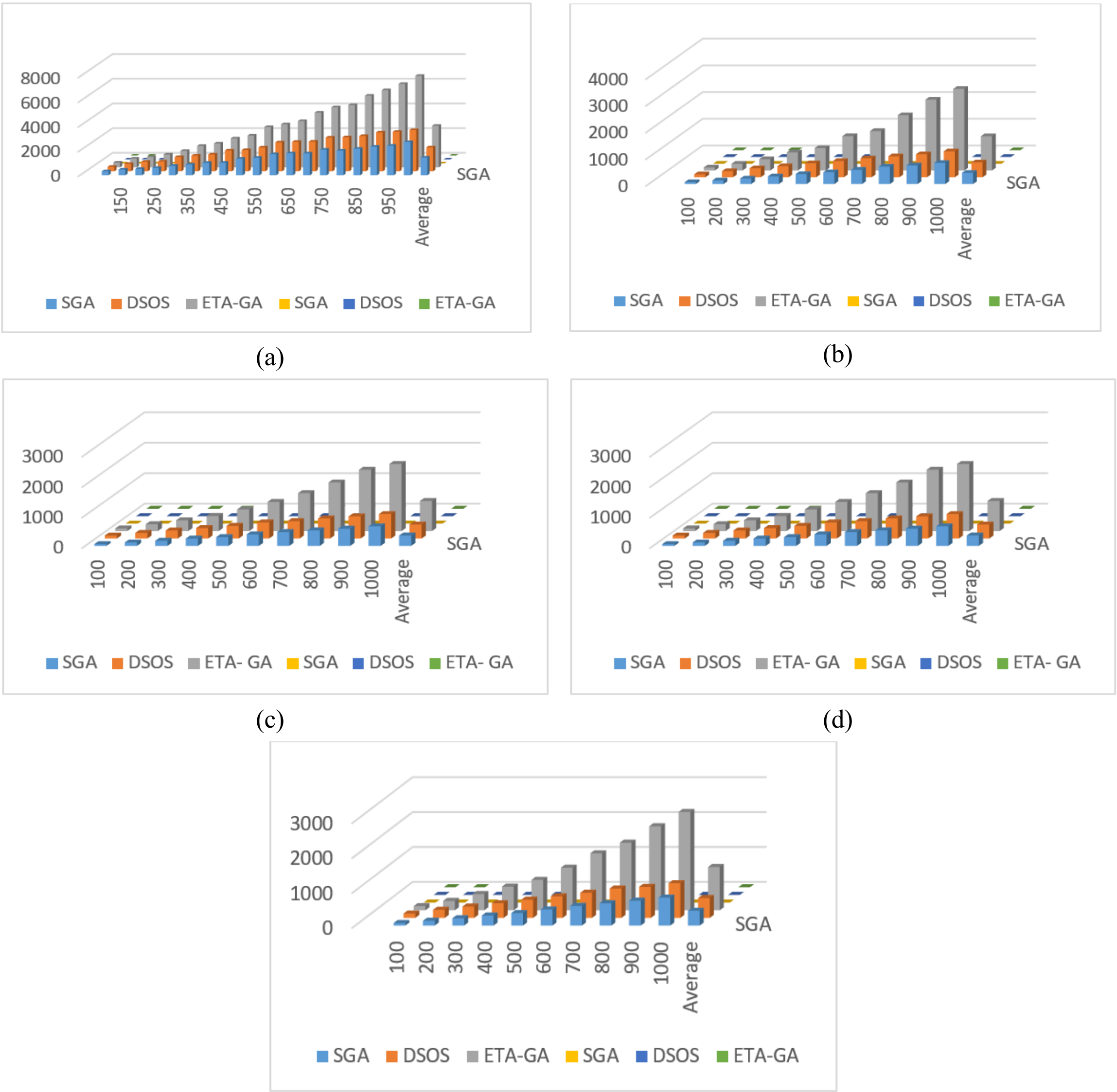


Figure 18: Average of ARUR on Uniform

4.2 SGA

The operation of the SGA has previously explained in literature review. SGA is useful when a meta-heuristic scheduler is necessary, the key focus here is improving GA's convergence speed. To employ this algorithm is previously covered in previous Section. Only meta-heuristics and SGA may be compared for convergence analysis. The fitness value is often compared on correctly applied approach improves through iterations. Because the provided approach is independent, and the demonstration of fitness value differs from that of previous methods. As a result, the approaches cannot be associated in terms of fitness. SGA's fitness function is intended to enhance load balance. The primary goal of SGA is to accelerate GA convergence.



(a)

(b)

(c)

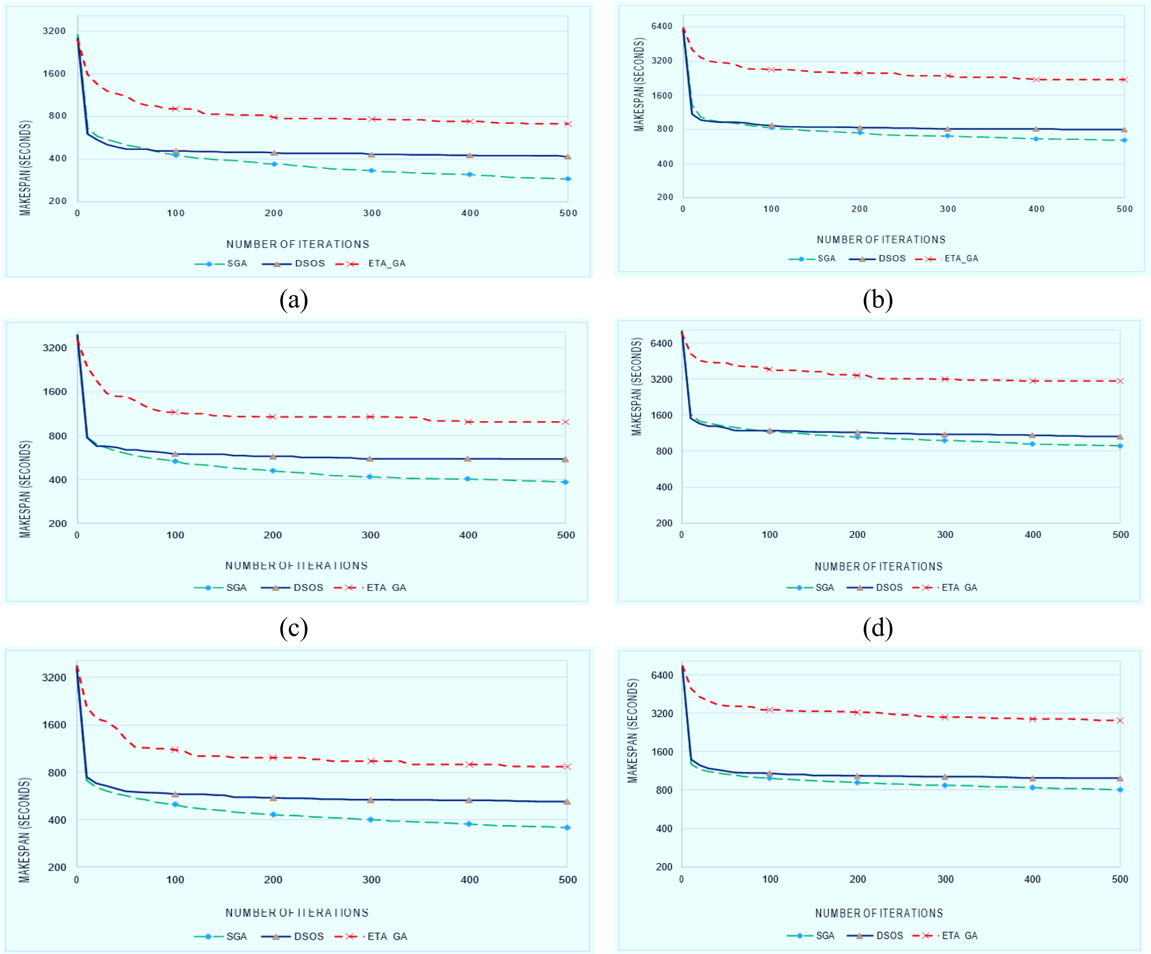
(d)

(e)

Figure 21: Results Graphs: (a) ARUR and Makespan of GoCJ, (b) ARUR and Makespan on Left Skewed (c) Makespan and ARUR on Right Skewed, (d) Makespan and ARUR on Normal, (e) Results of Uniform

A. Makespan

Figures 26-35 demonstrations the makespan's convergence rate across a batch of 500 jobs on the given dataset. It demonstrates that proposed algorithm has a faster convergence rate than ETA-GA and DSOS. Figure 24 depicts the makespan on jobs of 1000 of the GoCJ. On huge batch sizes, ETA-GA meets relatively slowly, other side SGA is the quickest of all.



(a) (b)

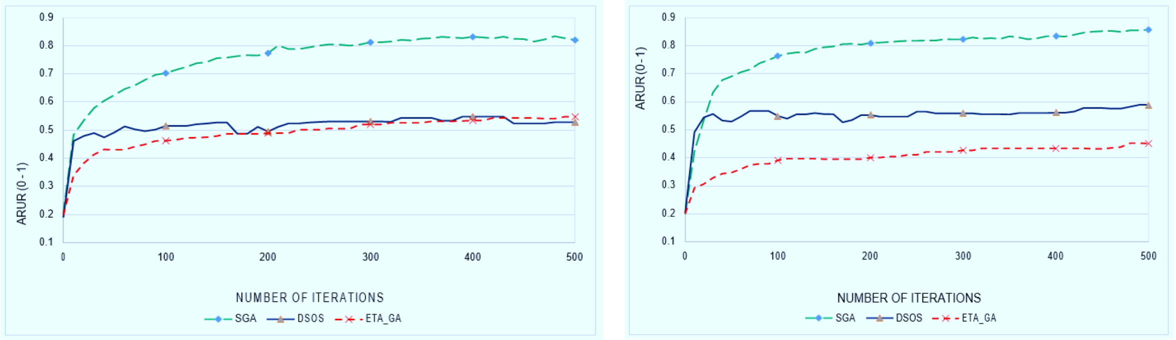
(c) (d)

(e) (f)

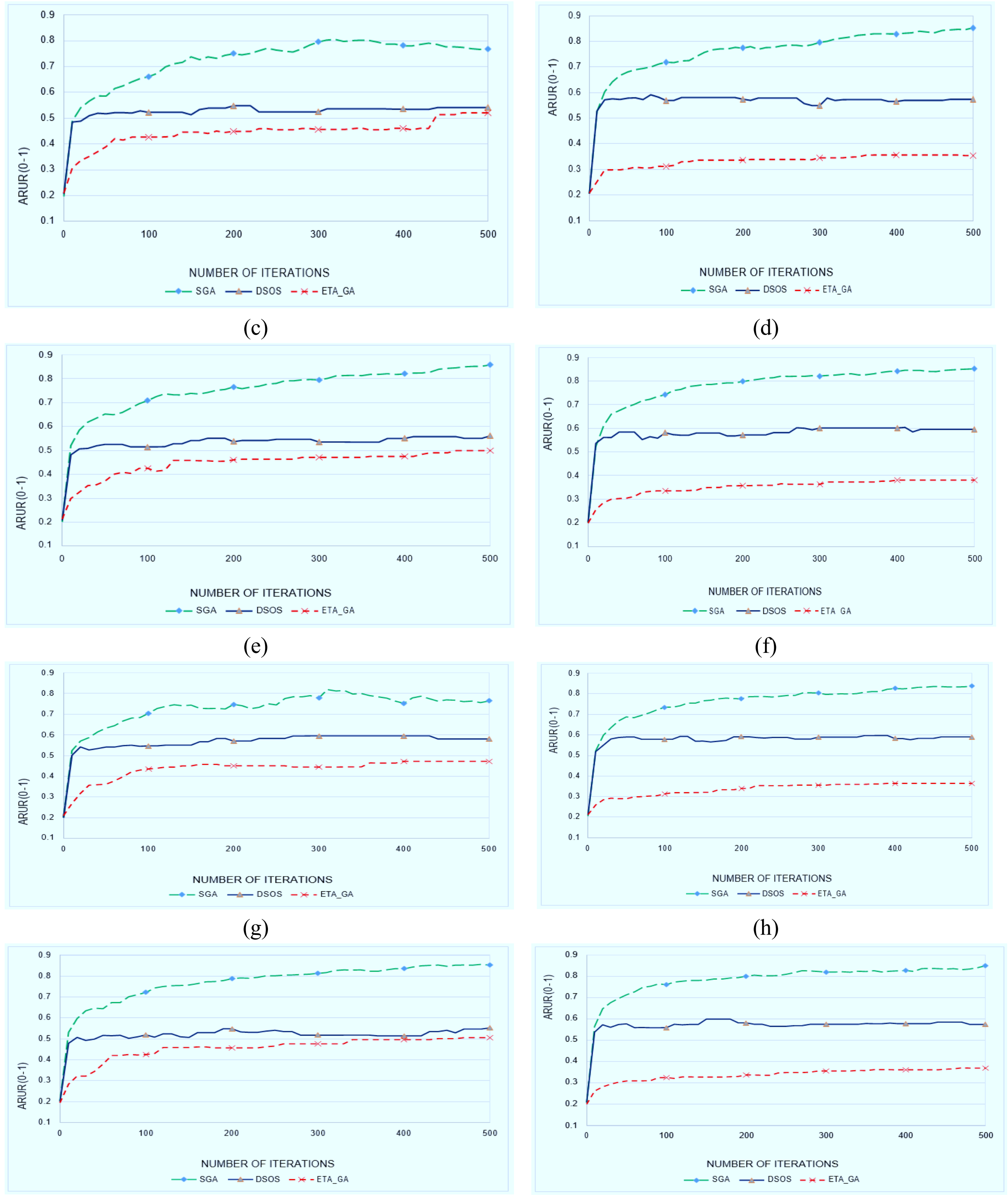
Figure 22: Makaspan Results: (a) On Right Skewed of 500 Jobs, (b) on Right Skewed of 500 Jobs, (c) on Normal of 500 Jobs, (d) on Normal of 500 Jobs, (e) on Uniform of 500 Jobs, (f) on Uniform of 500 Jobs

B. ARUR

Figures 36-45 depict ARUR analysis on GoCJ datasets for batches of 500 - 1000, respectively. SGA outperforms ARUR considerably. SGA converges quicker than other approaches on typical datasets for batches of 500 and 100, as illustrated in Figures. SGA beats ARUR on the even dataset at batched size 500, however DSOS and ETA-GA are quite close. The difference in ARUR value is substantial in batch size 1000, and the SGA converges quickly once more. Figures 39 demonstrate this.



(a) (b)



(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

Figure 23: ARUR Results (a) On GoCJ of 500 Jobs, (b) On GoCJ of 500 Jobs (c) On Left Skewed of 500 Jobs (d) On Left Skewed of 500 Jobs (e) On Right Skewed of 500 Jobs (f) On Right Skewed of 500 Jobs (g) On Normal of 500 Jobs (h) On Normal of 500 Jobs (i) On Uniform of 500 Jobs (j) On Uniform Dataset of 500 Jobs

5. SIGNIFICANCE AND NOVELTY OF PROPOSED WORK

The significance of the proposed work lies in its comprehensive exploration of task scheduling optimization, the introduction of a genetically based algorithm, the emphasis on meta-heuristic approaches, and the evaluation against existing techniques. The focus on QoS criteria and the use of diverse datasets contribute to the practical relevance and novelty of the research. The proposed work holds significant importance and novelty in the field of cloud computing for several reasons:

5.1 OPTIMIZATION OF TASK SCHEDULING

The work addresses a crucial challenge in cloud computing, specifically the optimization of task scheduling. Efficiently allocating workloads to virtual resources is a fundamental concern due to the diversity of professions and their unique resource requirements.

5.2 HEURISTIC AND META-HEURISTIC SCHEDULERS

The study explores the use of heuristic and meta-heuristic schedulers for mapping independent jobs. This is essential given the variety of alternative mappings required to cater to different professions and resource needs.

5.3 GENETICALLY BASED ALGORITHM

The introduction of a genetically based algorithm adds novelty to the research. This algorithm aims to enhance both makespan and resource consumption, indicating a holistic approach to task scheduling optimization.

5.4 ALGORITHM COMPARISON AND EVALUATION

The proposed work compares the genetically based algorithm with the SGA (second algorithm) and several existing heuristic techniques. This comparative analysis provides insights into the strengths and weaknesses of different scheduling strategies.

5.5 FOCUS ON CONVERGENCE SPEED

The consideration of convergence speed in the SGA highlights a nuanced approach to algorithm design. This is crucial in real-world applications where the efficiency of scheduling algorithms can significantly impact system performance.

5.6 META-HEURISTIC ADOPTION EMPHASIS

The work strongly emphasizes the importance of adopting meta-heuristic approaches. Meta-heuristics, with their ability to explore a vast solution space, are recognized as valuable tools for addressing the complexities of cloud task scheduling.

5.7 QOS CRITERIA

The thesis concentrates on two critical Quality of Service (QoS) criteria, namely makespan and resources used. This focus on QoS criteria is essential for defining optimal mappings and ensuring that the proposed scheduling strategies align with performance objectives.

5.8 SYNTHETIC AND REALISTIC DATASETS

To validate the functionality of the proposed methodologies, the use of both synthetic and realistic datasets enhances the robustness and applicability of the research findings to practical scenarios.

6. CONCLUSION AND FUTURE WORK

In the field of cloud computing, the problem of task scheduling necessitates the effective plotting of workloads to virtual resources. Because of diversity of professions and needed resources, there are several alternative mappings. To map independent jobs, heuristic and meta-heuristic schedulers are used. The meta-heuristic has the capacity to search the vast universe of promising results. The genetically based algorithm introduced in study in order to increase the makespan and needed resource consumption. The second algorithm (SGA), is concerned with convergence speed. The strategies given are compared to several current both heuristic techniques. Other genetic optimization based approaches are assessed in conjunction with the described methodologies as well. To learn how the provided methodologies function, synthetic and realistic datasets are used.

- For optimum scheduling, heuristic, meta-heuristic, and hybrid approaches are thoroughly studied.
- The importance of meta-heuristic adoption is strongly emphasised.
- The effectiveness of a genetically based evolutionary strategy in meta-heuristics is demonstrated.

- This thesis focuses on two of the most important QoS criteria, namely makespan and resources used, in order to define optimal mapping.

Future work in the field of cloud computing task scheduling could build upon the current research by addressing the following areas:

- Investigate the potential of hybrid algorithms that combine the strengths of heuristic, meta-heuristic, and genetically based approaches. This exploration could lead to more robust and adaptable scheduling strategies.
- Extend the research to handle dynamic workloads in real-time. Developing algorithms that can adapt to changing workload conditions and resource availability would enhance the practical applicability of the scheduling system.
- Integrate energy efficiency as a key criterion for task scheduling. Future work could focus on algorithms that not only optimize makespan and resource usage but also minimize energy consumption, contributing to environmentally sustainable computing.
- Explore the integration of machine learning techniques to predict workload patterns and resource demands. This could enable more proactive and anticipatory scheduling strategies, improving overall system efficiency.
- Conduct extensive scalability testing to evaluate the performance of the proposed algorithms in handling large-scale cloud environments. This would ensure that the scheduling strategies remain effective as the scale of cloud systems continues to grow.
- Implement the proposed algorithms in a real-world cloud computing environment to assess their practical feasibility and performance. This could involve collaboration with industry partners to deploy and evaluate the algorithms in operational cloud platforms.
- Investigate methods for allowing users to customize the scheduling algorithms based on their specific requirements and priorities. Providing a level of customization could enhance the adaptability of the system to diverse user needs.
- Integrate security and privacy considerations into the scheduling algorithms. Future work could explore methods for ensuring secure and private task scheduling, especially in multi-tenant cloud environments.
- Explore the compatibility of the proposed algorithms with different cloud platforms and architectures. Ensuring cross-platform adaptability would make the scheduling strategies applicable in a variety of cloud computing environments.
- Establish a continuous benchmarking process to monitor the performance of scheduling algorithms over time. This would involve updating algorithms based on evolving workload patterns and technological advancements.

7. REFERENCE

[1] J. Logeshwaran, "The control and communication management for ultra dense cloud system using fast Fourier algorithm," ICTACT Journal on Data Science and Machine Learning, vol. 3, no. 2, pp. 281-284, 2022.

[2] C. Surianarayanan and P. R. Chelliah, "Essentials of Cloud Computing." Springer, 2019.

[3] S. Kolb, "On the Portability of Applications in Platform as a Service." University of Bamberg Press, 2019.

[4] M. Abdullahi, M. A. Ngadi, S. I. Dishing, S. i. M. Abdulhamid, and M. J. Usman, "A survey of symbiotic organisms search algorithms and applications," Neural computing and applications, vol. 32, pp. 547-566, 2020.

[5] M. Abdullahi, M. A. Ngadi, S. I. Dishing, and S. i. M. Abdulhamid, "An adaptive symbiotic organisms search for constrained task scheduling in cloud computing," Journal of ambient intelligence and humanized computing, vol. 14, no. 7, pp. 8839-8850, 2023.

[6] Y.-L. Lee, S. N. Arizky, Y.-R. Chen, D. Liang, and W.-J. Wang, "High-availability computing platform with sensor fault resilience," Sensors, vol. 21, no. 2, p. 542, 2021.

[7] J. Hassan and Z. Iqbal, "QoS-Aware Efficient Tasks Scheduling in Heterogeneous Cloud Computing Environment," University of Wah Journal of Computer Science, vol. 5, no. 1, 2023.

[8] A. Hussain, M. Aleem, A. Khan, M. A. Iqbal, and M. A. Islam, "RALBA: a computation-aware load balancing scheduler for cloud computing," Cluster Computing, vol. 21, pp. 1667-1680, 2018.

[9] R. Priyadarshini, M. Alagirisamy, N. Rajendran, A. K. Marandi, and V. V. Patil, "Minimization of Makespan and Energy Consumption in Task Scheduling in Heterogeneous Cloud Environment," International Journal of Intelligent Systems and Applications in Engineering, vol. 10, no. 2s, pp. 276–280-276–280, 2022.

[10] H. Ning, Y. Li, F. Shi, and L. T. Yang, "Heterogeneous edge computing open platforms and tools for internet of things," Future Generation Computer Systems, vol. 106, pp. 67-76, 2020.

[11] T. S. Alnusairi, A. A. Shahin, and Y. Daadaa, "Binary PSOGSA for load balancing task scheduling in cloud environment," arXiv preprint arXiv:1806.00329, 2018.

[12] F. A. Saif, R. Latip, M. Derahman, and A. A. Alwan, "Hybrid meta-heuristic genetic algorithm: Differential evolution algorithms for scientific workflow scheduling in heterogeneous cloud environment," in Proceedings of the Future Technologies Conference, 2022: Springer, pp. 16-43.

[13] R. Pradhan and S. C. Satapathy, "Energy Aware Genetic Algorithm for Independent Task Scheduling in Heterogeneous Multi-Cloud Environment," 2022.

[14] M. Raushan, A. K. Sebastian, M. Apoorva, and N. Jayapandian, "Advanced load balancing min-min algorithm in grid computing," in Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI-2018), 2020: Springer, pp. 991-997.

[15] S. Supreeth, K. Patil, S. D. Patil, S. Rohith, Y. Vishwanath, and K. Prasad, "An efficient policy-based scheduling and allocation of virtual machines in cloud computing environment," Journal of Electrical and Computer Engineering, vol. 2022, 2022.

[16] R. Pradhan and S. C. Satapathy, "Particle Swarm Optimization-Based Energy-Aware Task Scheduling Algorithm in Heterogeneous Cloud," in Communication, Software and Networks: Proceedings of INDIA 2022: Springer, 2022, pp. 439-450.

[17] Z. Mohamad, A. A. Mahmoud, W. Nik, M. A. Mohamed, and M. M. Deris, "A genetic algorithm for optimal job scheduling and load balancing in cloud computing," International Journal of Engineering & Technology, vol. 7, no. 3, pp. 290-294, 2018.

[18] N. P. Sodinapalli, S. Kulkarni, N. A. Sharief, and P. Venkatareddy, "An efficient resource utilization technique for scheduling scientific workload in cloud computing environment," IAES International Journal of Artificial Intelligence, vol. 11, no. 1, p. 367, 2022.

[19] R. Ghafari, F. H. Kabutarkhani, and N. Mansouri, "Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review," Cluster Computing, vol. 25, no. 2, pp. 1035-1093, 2022.

[20] D. K. Shukla, D. Kumar, and D. S. Kushwaha, "An efficient tasks scheduling algorithm for batch processing heterogeneous cloud environment," International Journal of Advanced Intelligence Paradigms, vol. 23, no. 1-2, pp. 203-216, 2022.

[21] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," Computers & Industrial Engineering, vol. 137, p. 106040, 2019.

[22] J.-B. Wang et al., "A machine learning framework for resource allocation assisted by cloud computing," IEEE Network, vol. 32, no. 2, pp. 144-151, 2018.

[23] Y. Lu and X. Xu, "Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services," Robotics and Computer-Integrated Manufacturing, vol. 57, pp. 92-102, 2019.

[24] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A load balancing algorithm for the data centres to optimize cloud computing applications," IEEE Access, vol. 9, pp. 41731-41744, 2021.

[25] N. Thakkar and R. Nath, "Performance Analysis of Min-Min Max-Min and Artificial Bee Colony Load Balancing Algorithm in Cloud Computing," IJACS, vol. 7, no. 4, 2018.

[26] K. Dubey and S. C. Sharma, "A hybrid multi-faceted task scheduling algorithm for cloud computing environment," International Journal of System Assurance Engineering and Management, vol. 14, no. Suppl 3, pp. 774-788, 2023.

[27] F. Ebadifard and S. M. Babamir, "A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment," Concurrency and Computation: Practice and Experience, vol. 30, no. 12, p. e4368, 2018.

[28] A. Halty, R. Sánchez, V. Vázquez, V. Viana, P. Pineyro, and D. A. Rossit, "Scheduling in cloud manufacturing systems: Recent systematic literature review," 2020.

[29] G. F. Da Silva, F. Brasileiro, R. Lopes, F. Morais, M. Carvalho, and D. Turull, "QoS-driven scheduling in the cloud," Journal of Internet Services and Applications, vol. 11, pp. 1-36, 2020.

[30] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: a big picture," Journal of King Saud University-Computer and Information Sciences, vol. 32, no. 2, pp. 149-158, 2020.

[31] R. Gulbaz, A. B. Siddiqui, N. Anjum, A. A. Alotaibi, T. Althobaiti, and N. Ramzan, "Balancer genetic algorithm—A novel task scheduling optimization approach in cloud computing," Applied Sciences, vol. 11, no. 14, p. 6244, 2021.

[32] T. Prem Jacob and K. Pradeep, "A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization," Wireless Personal Communications, vol. 109, pp. 315-331, 2019.

[33] T. Wang, P. Zhang, J. Liu, and M. Zhang, "Many-objective cloud manufacturing service selection and scheduling with an evolutionary algorithm based on adaptive environment selection strategy," Applied Soft Computing, vol. 112, p. 107737, 2021.

[34] K. Maryam, M. Sardaraz, and M. Tahir, "Evolutionary algorithms in cloud computing from the perspective of energy consumption: A review," in 2018 14th international conference on emerging technologies (ICET), 2018: IEEE, pp. 1-6.

[35] M. Abd Elaziz, S. Xiong, K. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," Knowledge-Based Systems, vol. 169, pp. 39-52, 2019.

[36] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, and A. Abraham, "Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing," Neural Computing and Applications, vol. 30, pp. 1845-1863, 2018.

[37] M. Baghel, S. Agrawal, and S. Silakari, "Survey of metaheuristic algorithms for combinatorial optimization," International Journal of Computer Applications, vol. 58, no. 19, 2012.

[38] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," Information sciences, vol. 305, pp. 357-383, 2015.

[39] A. Hameed et al., "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," Computing, vol. 98, pp. 751-774, 2016.

[40] K. Etminani and M. Naghibzadeh, "A min-min max-min selective algorihtm for grid task scheduling," in 2007 3rd IEEE/IFIP International Conference in Central Asia on Internet, 2007: IEEE, pp. 1-7.

[41] R. NoorianTalouki, M. H. Shirvani, and H. Motameni, "A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 8, pp. 4902-4913, 2022.

[42] M. Abdullahi and M. A. Ngadi, "Symbiotic organism search optimization based task scheduling in cloud computing environment," Future Generation Computer Systems, vol. 56, pp. 640-650, 2016.

[43] A. Hussain, M. Aleem, M. A. Islam, and M. A. Iqbal, "A rigorous evaluation of state-of-the-art scheduling algorithms for cloud computing," IEEE Access, vol. 6, pp. 75033-75047, 2018.

[44] X. Wu, M. Deng, R. Zhang, B. Zeng, and S. Zhou, "A task scheduling algorithm based on QoS-driven in cloud computing," Procedia Computer Science, vol. 17, pp. 1162-1169, 2013.

[45] E. S. Alkayal, N. R. Jennings, and M. F. Abulkhair, "Efficient task scheduling multi-objective particle swarm optimization in cloud computing," in 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops), 2016: IEEE, pp. 17-24.