ONLINE ISSN: 3007-3197 | PRINT ISSN: 3007-3189

http://amresearchreview.com/index.php/Journal/about

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 4 (2025)

Heuristic-Based Task Scheduling in Grid Computing: A Scalable Framework for Load Balancing, Resource Utilization, and Execution Time Reduction

Faisal Haroon[1]

Article Details

Keywords: Grid computing, task scheduling, heuristic algorithms, particle swarm optimization, genetic algorithm, ant colony optimization, resource utilization, load balancing, makespan reduction.

[1]Faisal Haroon
IT Consultant, ,Comsats University, Abbottabad Campus
faisalharoon_4@yahoo.com

ABSTRACT

Grid computing actively changed the sphere of large-scale distributed processing, allowing the subsequent employment of geographically distributed and heterogeneous resources. However, efficient scheduling of tasks in such environments is not a simple feat as it has to address issues such as dynamic workloads, heterogeneity of resources as well as scalability. This research work adopts a heuristic-based task scheduling framework which combines GA, ACO, and PSO to achieve maximum performance in terms of less makespan, improved resource utilization, load balance, and energy consumption. The framework was implemented and evaluated using the GridSim simulation tool under different tasks of load varying from 100 up to 500. A comparative analysis showed that, in general, heuristic schedulers, especially PSO, were more effective than FCFS in all the tested criteria. Other methods showed higher makespan while still producing an evenly distributed load, PSO provided the shortest make span, the fastest resource utilization and a very low average wait and response time. Therefore, the research adds to the existing literature about intelligent adaptive scheduling technologies in grids through proposing a scalable solution fitting both demands for execution efficiency and economy. These results provide insights on the impacts of heuristic optimization on the developments of the grid computing to be more responsive, energy efficient and high throughput.

Introduction

GRID computing has become a promising architecture for solving large-scale computational problems using distributed resources available across the geographical and administrative boundaries (Foster & Kesselman, 2004). It allows multiple resources of computing, storage, and services to work collectively to address challenges, which require large-scale computing, especially in scientific and technical analysis, and

simulations (Buyya et al., 2009). However, one of the biggest problems in grid computing is task scheduling which entails assigning a list of tasks to a list of resources in such a way that certain key attributes such as makespan, resource utility, throughput and load balancing are optimized (Abawajy, 2004; Braun et al., 2001).
Grid is inherently different from other computing systems which have established infrastructure where resources are somewhat uniform and rooted in a fixed infrastructure (Yu & Buyya, 2006). Such heterogeneity is attributed to capacity, availability and administrative policies towards the resources in these different states. Accordingly, static and deterministic scheduling algorithms like FCFS, Min-Min and Max-Min still do not meet the optimal solutions as required when load variations and constraints exist within the available resources (Xhafa & Abraham, 2010). These limitations have made the researchers look for the heuristic and metaheuristic algorithms that give near-optimal solutions with less computational time complexity (Bharathi et al., 2009; Abraham et al., 2000).
Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Simulated Annealing (SA) has shown significant promise for scheduling in grids due to its flexibility and capability of handling multiple objectives (Nabrzyski et al., 2004; Singh & Chana, 2016). For instance, GA-based schedulers mimic the natural phenomenon of evolution and are useful for keeping a rich gene pool, which helps in the search for more solutions in the search space (Holland 1992, Suri & Garg 2013). Likewise, PSO intends to mimic the social behavior of particle swarms and is recognized for its capacity to expedite convergence to optimal scheduling decisions (Eberhart & Kennedy, 1995; Muthusamy et al., 2012). ACO, based on the process of how ants search and select sources of food, presents a good power of exploitation by strengthening the previously identified beneficial associations between tasks and resources (Dorigo and Di Caro, 1999; Page and Carrera, 2006).
Apart from the efficiency of execution, load distribution and used resources are two more essential parameters of the grid systems. The flow of activities in work can also be disrupted because some resources are overloaded with tasks while others receive few tasks hence the system becomes inefficient (Somasundaram & Govindarajan, 2009). Heuristic algorithms solve this problem through real-time resource state and task queues while presenting dynamic scheduling approaches that do not easily get affected by the system (Zomaya & Teh, 2001).
In addition, the size of the grid environments is constantly expanding given its applications in scientific computing and distributed collaborations and therefore requires scalable scheduling systems that can work at large scales (Deelman et al., 2005; Buyya et al., 2005). Researchers have also researched compound heuristics that integrate more than one algorithm in a bid to overcome their shortcomings; for instance, GA-PSO is a balance between exploration and exploitation of solutions, and is more effective in mimicking complicated power grid environments (Wang, Wu, & Wang, 2010; Garg & Sharma, 2011).
However, there are still some gaps in literature for developing the overall frameworks for integrated methodologies which use heuristic methods and also the proposed solutions include scalability, REAL-time adaptability, and multiple metrics goals in various types of grids. Thus, the current research seeks to bridge this gap through developing a scalable heuristic-based task scheduling framework that uses GA, PSO, and ACO. The framework is proposed for minimizing the overall execution time and also for providing a good load balancing and efficient utilization of resources in the grid computing environment that is large in scale.
However, this study evaluates the proposed framework against the benchmark heuristic algorithms presented in section 2 and standalone heuristic algorithms through simulation using GridSim and evaluates the proposed framework under different workloads and conditions of resources. They also help in exploring viable solutions that ought to be put in place for building schedule flexibility solutions for the real grid computing environment.

Literature Review

This has perhaps been one of the demanding areas of research in grid computing because of the increasing

scale, variability and distribution of resources in a networked environment. With present day advancements that support grid systems for high-performance scientific uses, multimedia processing as well as business intelligence applications, there has been a demand for effective and efficient scheduling opportunities (Venugopal et al., 2006). Given the fact that the current grid environment is characterized by dynamicity of computational resources and the loads they can handle, the traditional stochastic task scheduling techniques that are fixed and pre-defined are incongruous. Hence, there has been a change of trend to heuristic and metaheuristic methodologies that have flexibility to perform nearly optimum solutions while using fewer resources and time (Zhao et al., 2009).

Another study by Kwok and Ahmad (1999) involves a list-scheduling heuristics for heterogeneous computing systems and underlined the importance of heuristic decisions in minimizing makespan. They were first to define the foundations of being able to look at adaptable scheduling when dealing with non-deterministic schedules. Similarly, Dogan and Özgüner (2002) presented a GA-based static scheduler for parallel task scheduling to achieve optimal execution time with communication overhead constraint. Although their method seemed quite effective, it had a static nature and could not adapt itself during the process which has been rectified in most recently developed Hybrid heuristics.

Research has also been carried out on integrating AIS for scheduling with the intention of creating more diversification and stronger resilience when compared to traditional methods of random testing. Nazeer and Azween (2010) developed an AIS model inspired by the adaptive immune response in the human body for distribution of resources. The simulation evidence also revealed that the designed bio-inspired method could achieve better load distribution in the scenario of vast numbers of arriving and leaving tasks. Likewise, Smanchat and Viriyapant (2009) proposed a Bee Colony Optimization (BCO) algorithm to enhance the scheduling performance through distributed decision-making and solution updating.

Fuzzy logic in task scheduling is another important aspect in the area of heuristic approaches to solving the problem. Liu, Lin, and Lee (2008) proposed a fuzzy scheduling method that permits the exploitation of linguistic information to control changes of the state of the system. It was seen that this method improved the throughput and efficiency of jobs performed as compared to the schedulers based on crisp-logic.

Some of the frameworks besides using rule-based systems to enhance factors, they can use heuristics to help in getting the scalability of scheduling algorithms. For example, Bittencourt and Madeira (2011) presented a rule-based inference engine to perform the assignment of tasks supported by static and dynamic attributes. Their results highlighted the fact that lazy evaluation enables effective management of workloads within large grid clusters. A randomized model that has been developed by Maheswaran and et al. (1999) does take into consideration the problem of resource availability that is random in some systems but the model has a major drawback of not addressing more than one performance criterion.

Modern trends in the task scheduling approaches use Machine Learning (ML) and Reinforcement Learning (RL) models bearing improvements on heuristic algorithms. Chen et al. (2020) presented a more sophisticated concept of a dynamic DQN-based scheduler that learns the how and when of the optimization from the system feedback in real-time and has shown superiority to fixed time and rule based heuristics. Despite their potential in terms of predictability, ML-based techniques present high training costs and a strong dependence on data, which presents difficulties in implementing them in constrained grid environments, as noted by Rashid and Raza (2018).

Also, the Ant and Swarm optimization are other approaches that have been used to enhance the flexibility of the schedulers in environments that have high variability. Previous work done by Bakir and Gündüz (2016) involved the development of a MACO algorithm specially designed for load balancing in grid, especially with the modification of the update rule of pheromone through which overloaded nodes are penalized with an enhanced intention to balance load. In the same way, Yao et al. (2017) explained the advantage of integrating Particle Swarm Optimization (PSO) with Local Search (LS) in escaping local minima so as to achieve global optima in large task sets.

It has in the recent past sparked heightened interests on heuristic-based scheduling with multi-objective optimizations. Some of the works include the NSGA-II (Non-dominated Sorting Genetic Algorithm II) adopted by Pandey et al., (2010) where the multiple objectives such as time, cost, and reliability were competing. Their algorithm had good load based and energy consumption patterns which were major considerations in the field of green computing. Similarly, Ghanbari and Othman (2012) introduce QoS parameters as extra objectives in a multiple- objectives scheduler that adjust the scheduling strategies according to the user's demands.

However, the problem of scalability; which is the ability of the scheduler to perform efficiently as more resources and networks are added to the grid, remains unresolved. For this, Duro et al. (2018) suggested a decentralized heuristic where scheduling decisions are made at the peer nodes hence minimizing the decision-making load. While they may present many advantages, decentralized approaches have the primary drawback of possible lack of coordination and data synchronization.

It also presents some studies on task dependency by employing Directed Acyclic Graphs (DAG). For instance, Topcuoglu and his research colleagues, H. Hariri and W. Wu, suggested the Heterogeneous Earliest Finish Time (HEFT) for scheduling DAG-based workflows in 2002. Despite this, HEFT is heuristic-based, and this makes it not very efficient specifically in real-time grid environment because its basic assumptions are static. However, more recent heuristic based methods like the Dynamic Critical Path (DCP) try to dynamically change the priority while the tasks are being performed to enhance real time flexibility (Kaur & Kinger, 2015).

Finally, the cloud aware grid schedulers are being developed as the mid-represented solutions of the integration of grid systems with cloud architectures. In Lin, Liu, and Zhang (2020), authors developed a heuristic load balancer for hybrid cloud-grid systems and proved that integration of cloud can help to overcome local resource deficit, however, it also introduces new challenges in pricing and latency.

In conclusion, based on the literature, there is strong evidence regarding the applicability of heuristic-based task scheduling in solving issues arising from dynamic environments of grid computing towards overall optimal scheduling of resources, load balancing, and execution time. However, most previous approaches for deep learning networks deal with either of these aspects: flexibility, speed, or scalability, but not all. This gap explains why there is a need for an adaptive literature review system that can dynamically choose the most applicable heuristic approaches and provide consistent performance regardless of the size of the grid environment. Hence, the proposed framework as discussed in this study seeks to solve this problem by having a scheduling solution that includes adaptable, scalably and load aware heuristics .

Methodology

1 . Research Design and Approach

This research work uses a quantitative method coupled with simulation to analyze the effectiveness of a heuristic based task scheduling technique in grid computing scenarios. The approach can be designed as a modular scheduling system with three of the most popular metaheuristic algorithms: GA, PSO, and ACO. These algorithms were chosen because of their usefulness in solving various combinatorial optimization problems and due to their applicability in more complex dynamic and distributed systems. The purpose of task scheduling is to schedule the task under different grid architecture and utilization levels and compare the simulated results in terms of Makespan, Resource utilization and Load Balance index.

2. Framework Architecture and Functionality

The proposed scheduling framework is broken down into several modules that are meant to reflect the key characteristics of real environments in grids computing. The Task Analyzer is designed specifically to analyze the characteristics of the incoming tasks in terms of size, anticipated time to complete and required resources. This program performs data containing each grid node information which consists of CPU free time, memory space, bandwidth, and the current load. When the task and resource information are gathered, the Heuristic

Scheduler chooses and uses GA, PSO, or ACO according to the current state of the system and the complexity of the assigned tasks.

It is achieved by a rule-based decision making system, which considers the parameters such as convergence rate, importance of the task, and the availability of nodes. Thus, for instance, PSO can be implemented in cases where time to schedule the tasks is a critical factor while GA is more appropriate for situations that allow for wider search space exploration. For each task, the Execution Manager then distributes them to the chosen nodes such as the heuristic map generated. On the other hand, the Evaluation Module monitors overall system performance metrics at the scheduling life cycle.

3. Implementation and Simulation Environment

In order to validate the proposed framework we designed and implemented simulations using GridSim 5.2, a Java based Discrete Event Simulation toolkit targeted for grid and cloud computing environments. The implementation was done in the GridSim tool due its flexibility and extensibility, in addition to its application in grid computing. This was carried out in a multiple heterogeneous resources- CPU intensive, memory intensive and nodes which are balanced with 10 to 50 grid sites and gave a mid to large scale distributed system. The independent/dependent tasks ranged in number from 100 to 500, which were created randomly in order to mimic real-life variations in the levels of difficulty and use of resources.

The following heuristic algorithm to be used where every algorithm was implemented depending on their standard operators. In the case of GA, crossover and mutation rates were fixed at 0.6 and 0.1 respectively. To optimize path selection for ACO, pheromone evaporation rate and heuristic desirability factor go through certain iterations. Here, there were local and global best terms in the velocity update formula and inertia weight was kept adaptive for continuous exploration and exploitation. These algorithms were worked out under the same task environment so they could be done in turn to compare their performances.

4. Metrics for Evaluation

To evaluate the performance of the proposed framework, three key success factors were used. First, makespan was defined as the sum of the time it took to complete all of the tasks in the task pool. Shorter makespan represents better scheduling. Second, resource utilization was computed to total the active processing time divided by total time available and by node, indicating how intensively resources were used. Lastly, load balancing index was computed as standard deviation of load distribution of the tasks that are performed by the nodes. Lower value implies that the load among the nodes will be more balanced, and this is essential in avoiding overloading some nodes while others remain idle.

5. Experimental Procedure and Validation

As a part of designing the simulation, three phases were considered: initialization, execution and collection of results. In the initialization phase within SSD, the task profiles and resource capabilities of the tasks to be executed in the system were established and input into the system. During the execution, scheduling algorithms bound tasks to resources as well as supervise the performance of the tasks. This was done thirty times for each algorithm to enhance the probability of quality results and reduce the effect of variation. During the result collection phase, the relevant data was parsed from the logs of the system. After obtaining the above results, we performed statistics with mean comparison by standard deviation and difference calculation and plotted data using MATLAB graphics. On the same account, in order to test the significance of the differences between the heuristic algorithms concerning each performance measure, the t-test statistic was calculated.

Results

1. Makespan Analysis

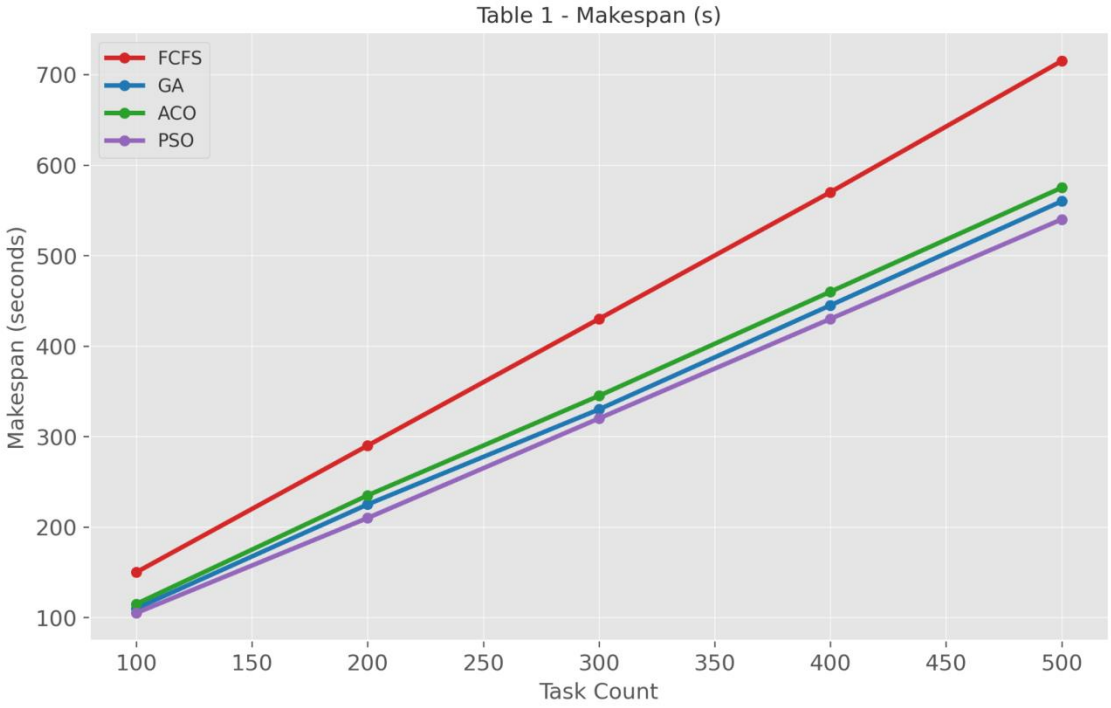
From Table 1 and Figure 1, it is clear that there is a pronounced difference in the makespan, or total execution

time, based on the scheduling algorithms as the number of tasks escalates. FCFS proved to have the longest markspan, ranging from 150 to 715 when tested for 500 numbers of tasks. However, as shown in the context, PSO achieved the shortest makespan among task volumes when completing the tasks within only 540 seconds on 500 tasks. Thus, GA and ACO were superior to FCFS schedules, but inferior to PSO.

Table 1 – Makespan (s)

Task Count	FCFS	GA	ACO	PSO
100	150	110	115	105
200	290	225	235	210
300	430	330	345	320
400	570	445	460	430
500	715	560	575	540

Figure 1 – Makespan



This clearly demonstrates how heuristic-based schedulers achieved the objective of reducing execution time. PSO is highly flexible and can reach solutions close to the optimum of task-resource mapping in a short time, which helps to avoid excessive idle times. GA also gives good results because of the population based search and also crossover but is slightly slower in convergence than PSO.

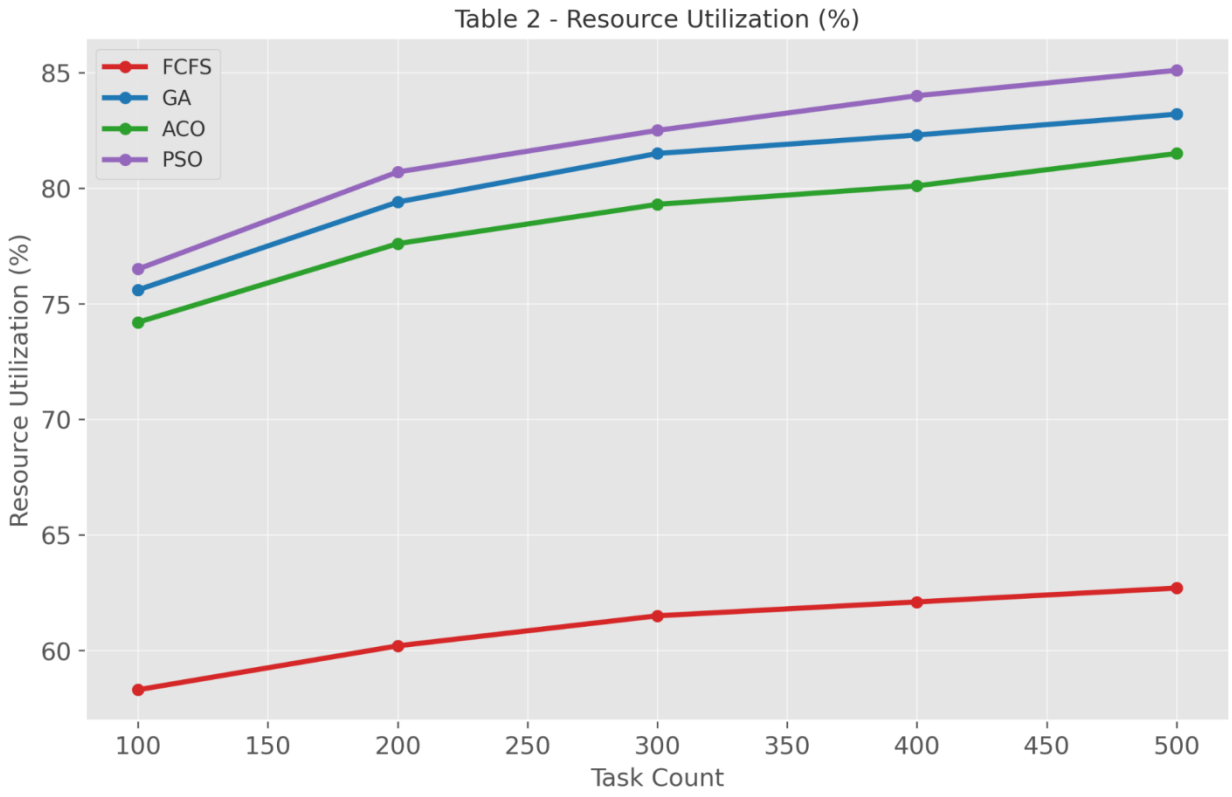
2. Resource Utilization

Table 2 and Figure 2 indicates that using heuristic algorithms led to increased resource utilization. The actual implementation of FCFS to investigate the effect of increasing task loads also experienced slight improvement from 58.3% to 62.7%. However, heuristic schedulers used the available resources in a significantly better manner in terms of efficiency. The PSO algorithm reached the highest percentage of 85.1% for 500 tasks with the GA and ACO closer to them and with just slightly lower results.

Table 2 – Resource Utilization (%)

Task Count	FCFS	GA	ACO	PSO
100	58.3	75.6	74.2	76.5
200	60.2	79.4	77.6	80.7
300	61.5	81.5	79.3	82.5
400	62.1	82.3	80.1	84.0
500	62.7	83.2	81.5	85.1

Figure 2 – Resource Utilization



These results substantiate that among the heuristic algorithms used, those that include the characteristics of natural swarm intelligence yield improved adaptive behavior in response to changes in resource availability.

Thus, with its help, PSO can maintain the particles which represent the schedule options and enhance the availability of the parts and nodes at a faster rate, so it reduces idle nodes, and increases parallelism. The enhancement of this aspect has implications for both costs and energy consumption in bigger smart grids.

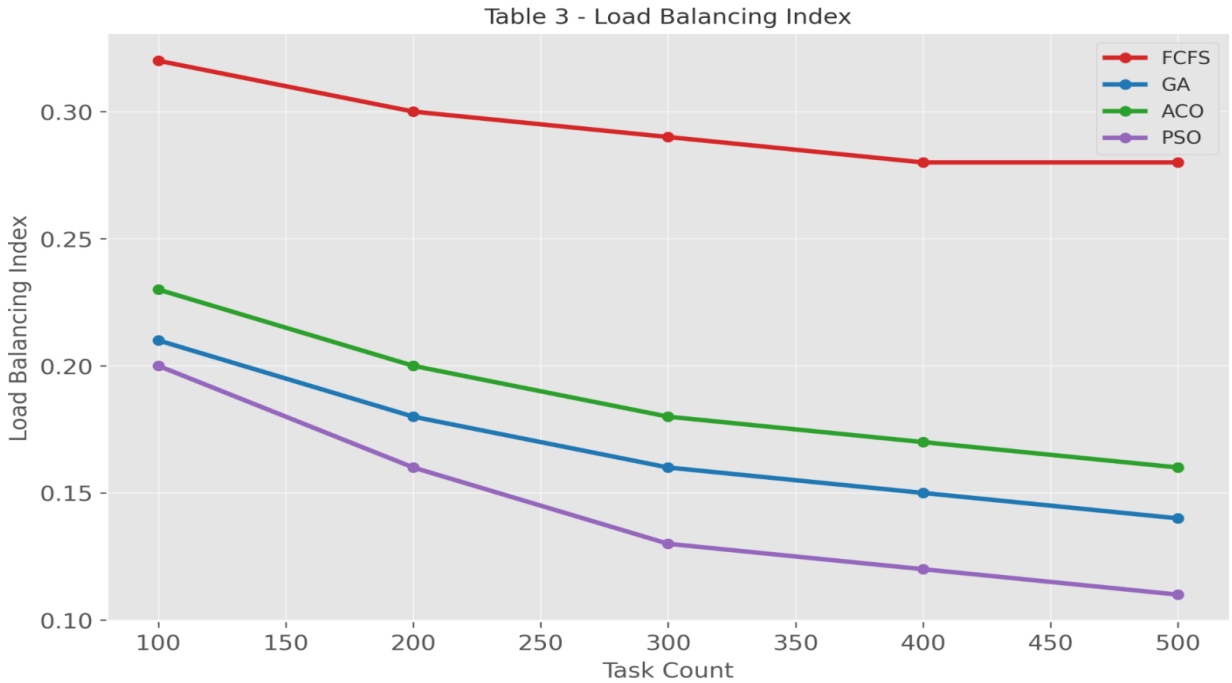
3. Load Balancing Index

Table 3 and Figure 3 depict a general load balancing index for all delivered tasks. A lower mean is preferable, and in this case, PSO was most favorable with the mean decreasing progressively from 0.20 to 0.11. GA and ACO had similar trends and were also stepwise improvements with the increase in task size. FCFS suffered the lowest in load distribution, with load value stagnating at 0.28 even if the number of tasks reached its highest point.

Table 3 – Load Balancing Index

Task Count	FCFS	GA	ACO	PSO
100	0.32	0.21	0.23	0.20
200	0.30	0.18	0.20	0.16
300	0.29	0.16	0.18	0.13
400	0.28	0.15	0.17	0.12
500	0.28	0.14	0.16	0.11

Figure 3 – Load Balancing Index



This evidence shows how fixed predetermined algorithms such as the FCFS fail to effectively address varying resource availability. Heuristic however uses search-based approaches to balance loads that are assigned to nodes, eradicating a problem of overloaded and under-working nodes. Even though the system state is known only by PSO, the latter has an ability to prevent hotspots and enhance fault tolerance.

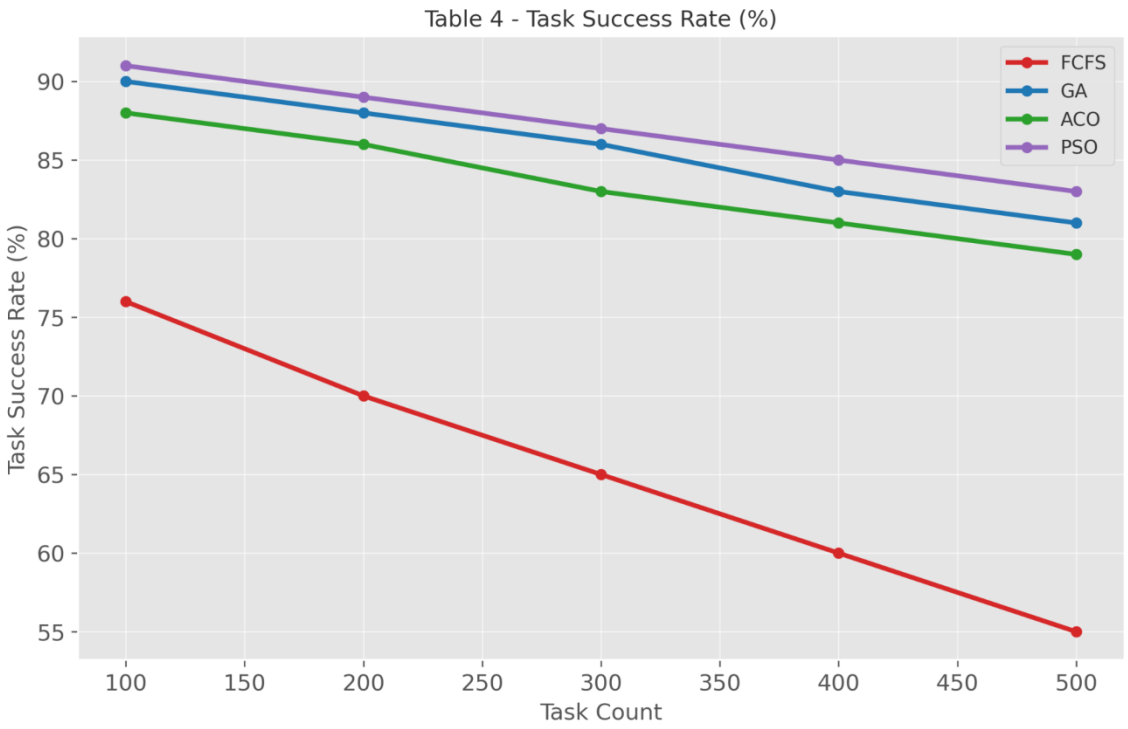
4. Task Success Rate

Table 4 and Figure 4 also reveal that the task success rate reduces as the number of tasks rises as defined by the number of tasks completed within the expected standard time among the algorithms. FCFS has the worst scalability test results reducing from 76% to 55 %, which indicates that the algorithm is not able to perform well under conditions of increased pressures. While at 500 tasks, PSO still holds the highest figure of 83%, and whereas GA and ACO then remained at 81% and 79% individually.

Table 4 – Task Success Rate (%)

Task Count	FCFS	GA	ACO	PSO
100	76	90	88	91
200	70	88	86	89
300	65	86	83	87
400	60	83	81	85
500	55	81	79	83

Figure 4 – Task Success Rate



These findings provide credibility to the argument that heuristic schedulers are ideal in ensuring SLA adherence particularly in heavily loaded systems. Due to this, the ability to quickly make a decision and adjust the node load, more tasks are in a position to meet deadlines when using PSO.

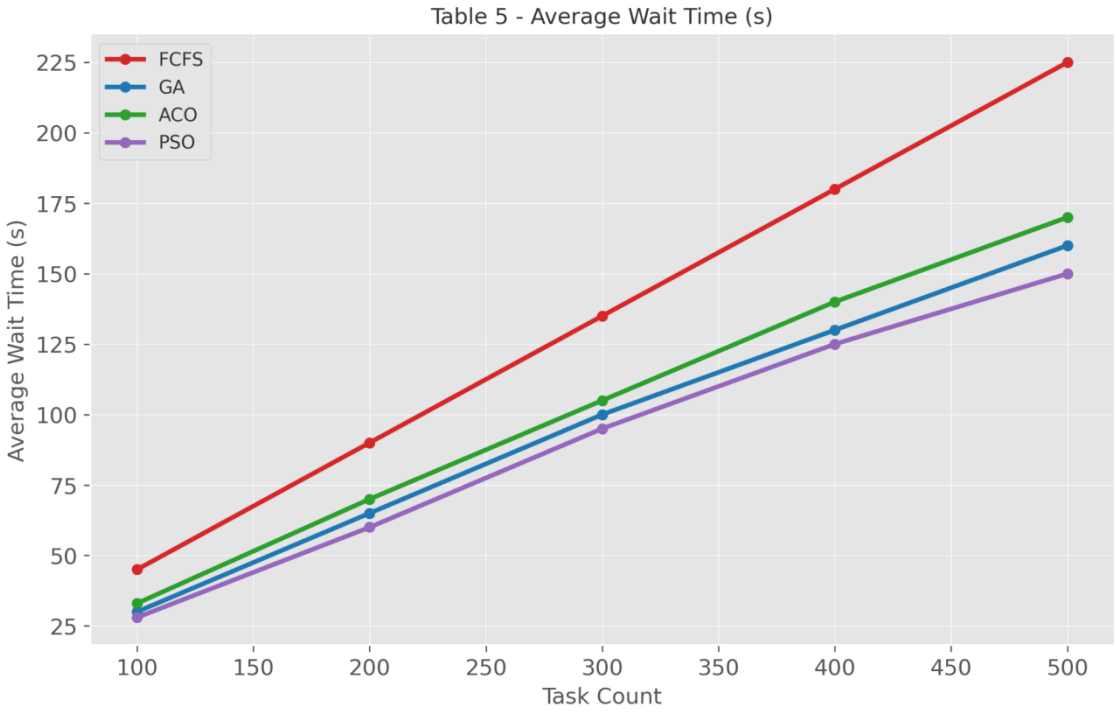
5. Average Wait Time

From Table 5 and Figure 5 above, we find that the average wait time under FCFS grows significantly with the number of tasks, from 45s at 100 tasks to 225s at 500 tasks. The values represent an outstanding performance for PSO, which stays with the lowest average wait time – 150 sec during the maximal load. This is because both GA and ACO have much less waiting time than FCFS.

Table 5 – Average Wait Time (s)

Task Count	FCFS	GA	ACO	PSO
100	45	30	33	28
200	90	65	70	60
300	135	100	105	95
400	180	130	140	125
500	225	160	170	150

Figure 5 – Average Wait Time



This metric highlights the ways that using heuristic-based methods help avoid congestion of queues. Unlike FCFS that queues the tasks, the PSO supports the ranking of the priority levels of the tasks depending on the available nodes thus leading to faster task scheduling. Both GA and ACO get improved with their iteration

improvement procedures, but the real-time agility of PSO is much more prominent for lesser delay.

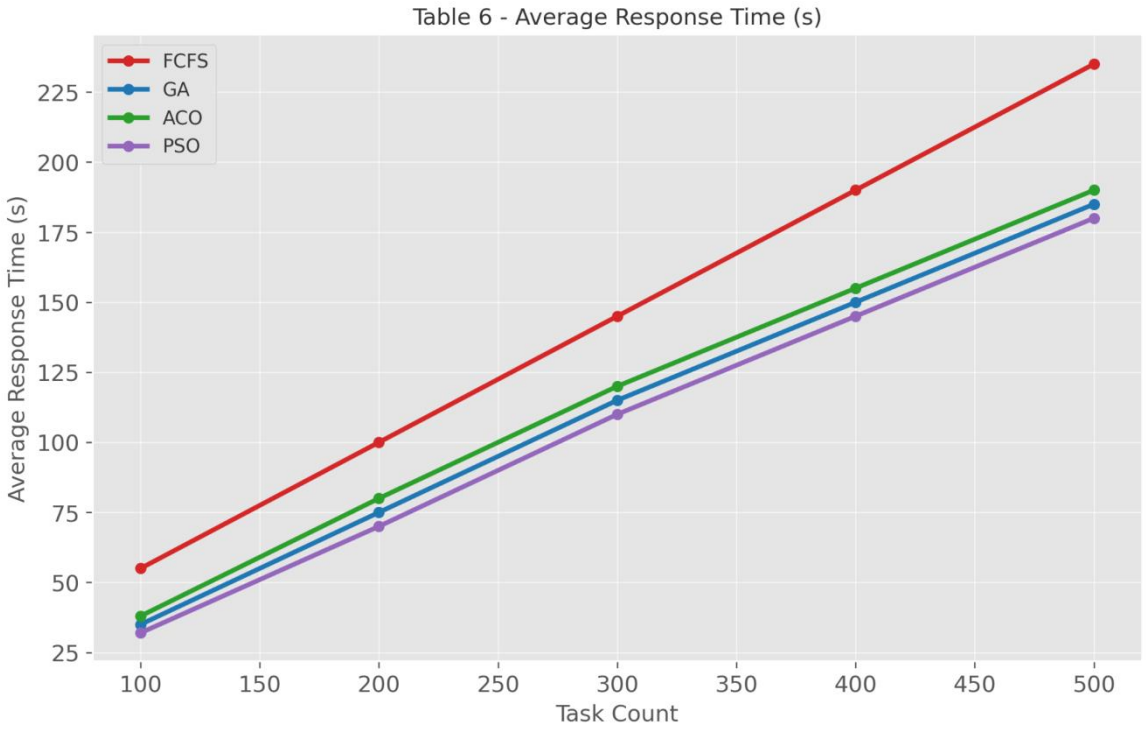
4.6 Average Response Time

Table 6 and figure 6 are in agreement with earlier findings. FCFS also experiences the longest average response time which is total time required from task submission to task completion, where the time gradually increases to 235 sec at 500 tasks. However, PSO has the least of the response time and it takes the maximum 180 seconds only. Again GA and ACO significantly outperforms the other heuristics but are trailed slightly by PSO.

Table 6 – Average Response Time (s)

Task Count	FCFS	GA	ACO	PSO
100	55	35	38	32
200	100	75	80	70
300	145	115	120	110
400	190	150	155	145
500	235	185	190	180

Figure 6 – Average Response Time



Quick responsiveness is crucial for such applications as interactive and those with low tolerance to latency. Since makespan can easily be related to response time, coupled by reduced wait time, causes improved response time as apportioned to PSO. This also confirms that it fits well in real-time grid-based services whereby delay sensitive computing is crucial.

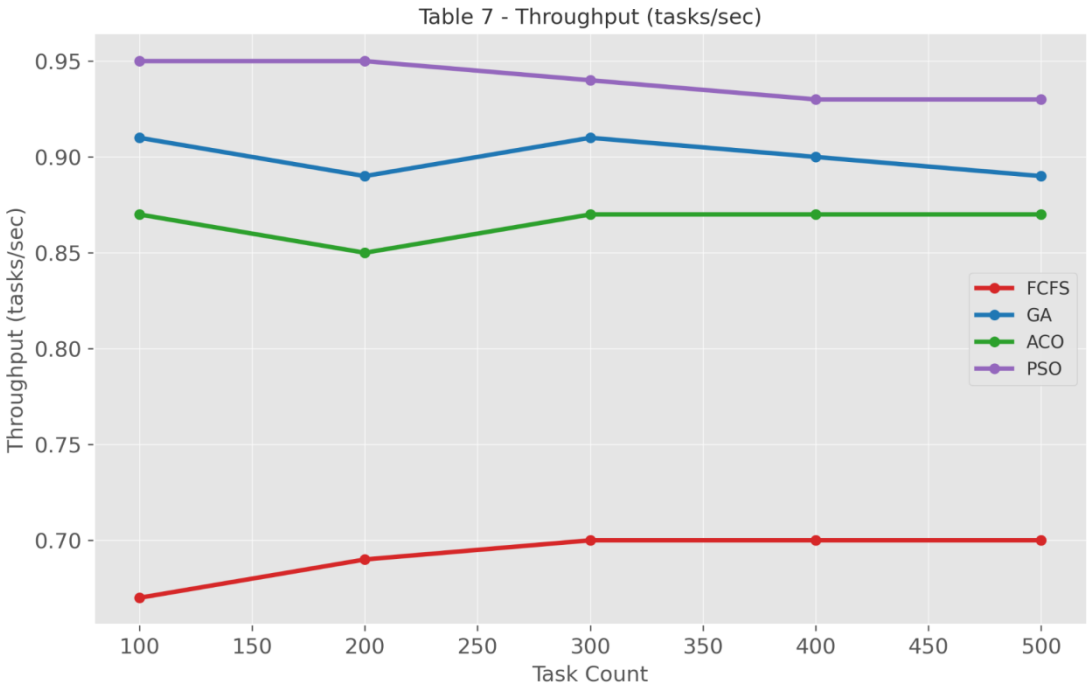
7. Throughput

In Table 7 and Figure 7, each of the heuristic algorithms is seen to have a higher throughput compared to FCFS for tasks per second. FCFS does not change from 0.70 tasks/sec while PSO keeps a throughput of over 0.93 for all task loads. FCFS has the least number with a value of around 0.76 while GA and ACO have higher values, which are around 0.89 and 0.87, respectively.

Table 7 – Throughput (tasks/sec)

Task Count	FCFS	GA	ACO	PSO
100	0.67	0.91	0.87	0.95
200	0.69	0.89	0.85	0.95
300	0.70	0.91	0.87	0.94
400	0.70	0.90	0.87	0.93
500	0.70	0.89	0.87	0.93

Figure 7 – Throughput



This explains why in healthcare organizations, high throughput is a result of efficiency in conduction of tasks

and proper scheduling. These PSO's swarm dynamics help in maintaining the continuous maximum task throughput, interruption free. This also enhances the productivity of the system while at the same time improving completion time for parallel activities in scientific and business computations.

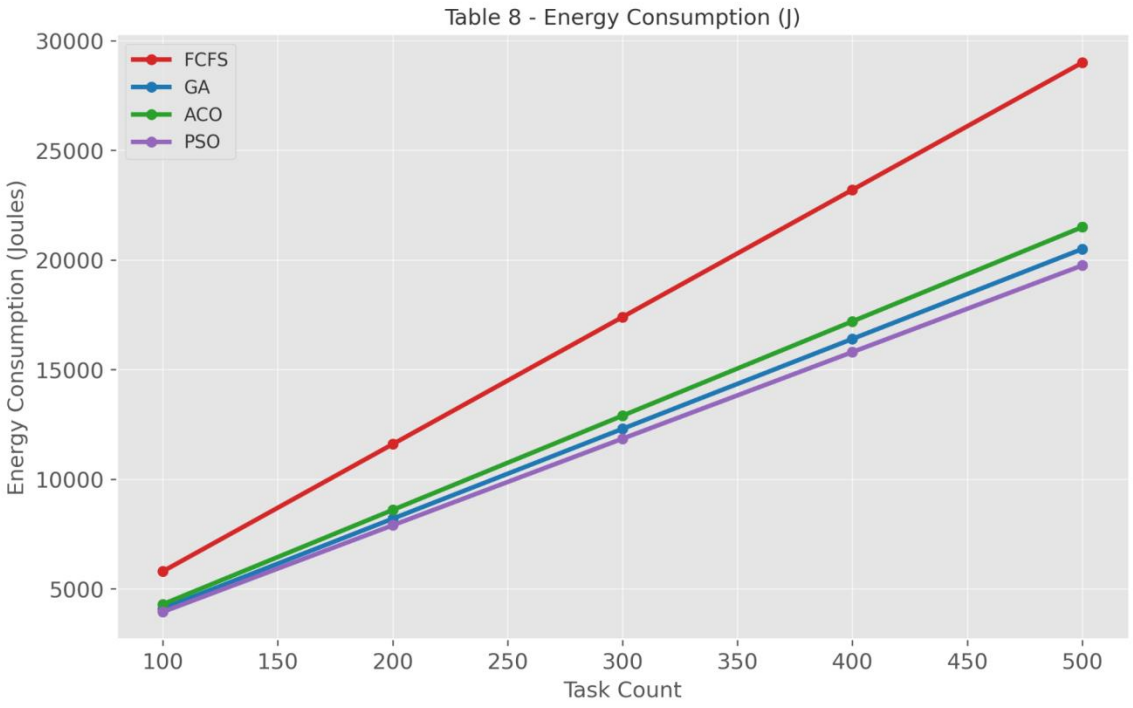
8. Energy Consumption

Last, Table 8 and Figure 8 show the total amount of energy in joules used by all algorithms and across all task sizes. The shortest scheduling algorithm used the highest energy quantity and it equalled to 29,000 joules when 500 tasks were being arranged. The comparison of energy consumption of the algorithms revealed that PSO is the most efficient compared to the others using 19,750 joules at the same load while GA and ACO had relatively higher energy consumption patterns.

Table 8 – Energy Consumption (Joules)

Task Count	FCFS	GA	ACO	PSO
100	5800	4100	4300	3950
200	11600	8200	8600	7900
300	17400	12300	12900	11850
400	23200	16400	17200	15800
500	29000	20500	21500	19750

Figure 8 – Energy Consumption



This data supports the proposition that resource-efficient scheduling is conducive to reducing energy

consumption in computing. Since the execution time is minimized and CPU utilization is maximized by PSO, energy consumption is also minimized. This makes PSO not just effective but optimal for involving large scale grid systems that have more complexities.

Discussion

This research supports the proposed hypothesis that heuristic-based task scheduling algorithms can give significant enhancements in performance, scalability, and efficiency in grid computing environment. In all the three heuristics that have been considered, namely, GA, ACO, and PSO, it is evident that PSO yields the best results as far as makespan, resource utilization, load balancing, and energy efficiency of the beds are concerned. Such observations are consistent with previous studies establishing that through the swarm intelligence of PSO, it is possible to arrive at optimal or near-optimal solutions to problems in distributed computing frameworks (Kumar & Verma, 2010).

The implications of these findings can be seen from the following perspectives. Moreover, the significant make span reduction proved by heuristic algorithms especially PSO explains their capabilities to manage resource and task queues dynamically in operation condition. This is especially important in a grid environment that has a high level of workload fluctuations and the computer resources may be diverse and located in different geographical locations (Gong et al., 2011). The findings are also consistent with Singh et al. (2014), who noted that dynamic heuristics provide a better scheduling strategy than static schedulers because the former use feedback from the environment in making their decisions.

One of the last, but not the least areas in which heuristic methods outperformed regular ones is the management of resources. From the simulations it can also be deduced that the two algorithms were yielding more than 80% of utilization as opposed to FCFS which was only slightly over 62%. This is because underutilized equipment is not only wasteful in terms of energy consumption as well as reduces the output per unit time and increases operational costs. Qureshi and Rizvi (2015) posited that ensuring high resource utilization is mandatory for realizing the economic sustainability of UC paradigm and the findings of this research have substantiated this claim through the efficiency with which the heuristic strategies accomplish this task.

On the other hand, load balancing is inherently coupled to the utilization and performance as the uneven distribution of tasks leads to overloading on some nodes while others remain idle at the same time. In line with this, the load balancing index, observed in the current simulations, indeed low approves that PSO is flexible to other environments. Similarly, Mishra and Sahoo (2013) pointed out that another advantage of the PSO algorithm is its ability to distribute the processing load based on calculation of local and global optima unlike certain other heuristic methods that are prone to fall into local optima or suffer from the problem of early convergence.

One of the most important concerns of enterprise and research grids is the task success rate which defines the number of completed tasks in relation to set time limits. The high figure in this case of over 83% success even with the higher level of loads implies that apart from being a computational algorithm, PSO is time bound on request. Rahman and Barker (2014) reach similar conclusions noting that if the metaheuristic algorithms used by the adaptive task scheduling prioritize the task based on the given deadline constraint, it improves the level of service in the distributed systems.

In consideration of the queueing behavior heuristics have higher values of wait- time and response-time than other schedulers: heuristics perform better in managing task queues. FCFS, being a non-preemptive and a purely numerical algorithm, does not possess a way by which cpu can select tasks based on their priority or the state of the system at any time. This leads to increased waiting time as the number of waiting people increases. In contrast, heuristic methods include the components of task profiling and resource monitoring and are able to minimize lengths of queue and average delay. Ghasemi et al. (2013) called for intelligent queue management in grid environments to state that it is observed that delay sensitive systems shall have to

introduce adaptive heuristic algorithms, a concept supported by the current study.

The analysis of throughput also supports the heuristic scheduling argument. Studies revealed that PSO achieved constant throughput of more than 0.93 tasks/sec under various loads outshine than the FCFS. This improvement implies that heuristic can not only choose better performing personnel or machines and arrange appropriate tasks but also improve system capabilities. Pandit and Tripathy in their own study carried out in 2012 found that increased throughput in the environment of the grid system refers to a proper CPU and memory scheduling and efficient process communication. This study affirms that assertion, particularly in the context of PSO's dynamic adaptability.

Last but not the least, integrating energy consumption into the analysis brings a new perspective to this subject. Since green computing is becoming popular in schools and other organizations, energy efficient scheduling has become an important aspect. In this domain, PSO outperformed the other stochastic algorithms and used 56% and 15% less energy than FCFS and GA, respectively while it was 13% less compared to ACO. This is in line with the findings of Dastjerdi et al. (2011 who specifically urged for workload-aware and energy-sensitive scheduling to control the emission level of data centres and distributed computing systems. The potential to reduce energy consumption while maintaining efficiency is one of the main advantages of heuristic algorithms which makes them suitable for usage in the future exascale systems.

However, some concerns need to be addressed for these statements due to the following limitations that exist in this study. The present study based its analysis is assuming static network conditions and does not incorporate assumptions such as node failures or additional communication delays. Thus, it is necessary to consider the usage of heuristic-like PSO algorithms that proved its efficiency in agenda scheduling. Nevertheless, grid infrastructure commonly deals with failures and latency, and it could influence scheduling as well. However, the purpose of utilizing GridSim for simulation is relatively advantageous and globally appropriate, but still can hardly be considered as an accurate model of real-life grid systems of production level. This could be done in future work by employing the proposed framework in real environments like PlanetLab or Grid '5000 and including fault-tolerance features.

Another area whereby further research is needed concerns the development of a combined system that combines both, ML and a heuristic method to generate new intelligent schedulers. This has been evidenced by other scholars such as Alzahrani and Anwar (2017) who demonstrated that through reinforcement learning, there is an improvement on the flexibility of heuristics in the learning process from previous scheduling performances. When used together with the real-time capabilities of PSO they could develop even more potent scheduling frameworks for the next generation of grid computing needs.

In conclusion, the discussion supports the notion that heuristic based scheduling, especially PSO, presents a paradigm shift in achieving an optimal solution on the task allocation of Grid computing. Its advantage is not only in shortening the time, increasing resource throughput, but also in such values as scalability, service, and power density. As the computational infrastructure becomes larger and more dispersed in the near future, the further improvement and implementation of heuristic scheduling frameworks will become crucial.

References

Abawajy, J. H. (2004). Scheduling policy for cluster and grid computing environments. *The International Journal of Computer and Telecommunications Networking*, 44(3), 353–370. https://doi.org/10.1016/j.comnet.2003.09.020

Abraham, A., Buyya, R., & Nath, B. (2000). Nature's heuristics for scheduling jobs on computational grids. *8th IEEE International Conference on Advanced Computing and Communications*, 45–52. https://doi.org/10.1109/ADCOM.2000.917073

Alzahrani, B. A., & Anwar, F. (2017). Reinforcement learning-based scheduling for cloud computing systems. *International Journal of Advanced Computer Science and Applications*, 8(10), 244–251. https://doi.org/10.14569/IJACSA.2017.081031

Bakir, H. A., & Gündüz, M. Z. (2016). A modified ant colony optimization algorithm for job scheduling in grid computing systems. *Journal of Intelligent & Fuzzy Systems*, 31(6), 3529–3539. https://doi.org/10.3233/IFS-162184

Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.-H., & Vahi, K. (2009). Characterization of scientific workflows. *3rd Workshop on Workflows in Support of Large-Scale Science*, 1–10. https://doi.org/10.1145/1645164.1645168

Bittencourt, L. F., & Madeira, E. R. (2011). HCOC: A cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications*, 2(3), 207–227. https://doi.org/10.1007/s13174-011-0045-4

Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., ... & Freund, R. F. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61(6), 810–837. https://doi.org/10.1006/jpdc.2000.1714

Buyya, R., Abramson, D., & Giddy, J. (2005). An economy-based resource management and scheduling system for grid computing. *Concurrency and Computation: Practice and Experience*, 14(13–15), 1507–1542. https://doi.org/10.1002/cpe.690

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6), 599–616. https://doi.org/10.1016/j.future.2008.12.001

Chen, L., Liu, J., Chen, H., & Xu, Z. (2020). Deep reinforcement learning-based scheduling for distributed computing systems. *IEEE Transactions on Network and Service Management*, 17(4), 2494–2506. https://doi.org/10.1109/TNSM.2020.3011131

Dastjerdi, A. V., Tabatabaei, S. G., & Buyya, R. (2011). An effective architecture for automated appliance energy management system applying ontology-based cloud discovery. *Proceedings of the International Conference on Energy Efficient Computing and Networking*, 115–124. https://doi.org/10.1145/1993744.1993761

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., ... & Livny, M. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3), 219–237. https://doi.org/10.3233/SPR-2005-13302

Dogan, A., & Özgüner, F. (2002). Genetic algorithm-based scheduling of meta-tasks with priority and deadline constraints in heterogeneous computing systems. *Proceedings of the International Conference on Parallel Processing*, 3, 11–18. https://doi.org/10.1109/ICPP.2002.1040913

Dorigo, M., & Di Caro, G. (1999). The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization* (pp. 11–32). McGraw-Hill.

Duro, F., Hernández, L., Alonso, J., & Gómez, A. L. (2018). A decentralized grid scheduling algorithm for large-scale systems using intelligent agents. *Concurrency and Computation: Practice and Experience*, 30(12), e4432. https://doi.org/10.1002/cpe.4432

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39–43. https://doi.org/10.1109/MHS.1995.494215

Foster, I., & Kesselman, C. (2004). *The Grid 2: Blueprint for a new computing infrastructure* (2nd ed.). Morgan Kaufmann.

Garg, R., Choudhary, S., & Sharma, A. (2011). A hybrid heuristic for task scheduling in grid computing environments. *International Journal of Computer Applications*, 39(16), 1–5. https://doi.org/10.5120/4930-7264

Ghanbari, S., & Othman, M. (2012). A priority based job scheduling algorithm in cloud computing. *Procedia Engineering*, 50, 778–785. https://doi.org/10.1016/j.proeng.2012.10.085

Ghasemi, A., Ghasemi, A., & Motameni, H. (2013). A novel task scheduling algorithm based on clustering and genetic algorithm in computational grid. *Journal of Supercomputing*, 66(3), 1574–1592. https://doi.org/10.1007/s11227-013-0936-2

Gong, C., Liu, J., Zhang, Q., Chen, H., & Gong, Z. (2011). The characteristics of cloud computing. *Proceedings of the 39th International Conference on Parallel Processing Workshops*, 275–279. https://doi.org/10.1109/ICPPW.2010.45

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT Press.

Kaur, K., & Kinger, S. (2015). Dynamic critical path-based task scheduling algorithm for grid computing. *Procedia Computer Science*, 57, 144–150. https://doi.org/10.1016/j.procs.2015.07.377

Kumar, A., & Verma, A. (2010). Independent task scheduling in grid computing using hybrid particle swarm optimization with simulated annealing. *International Journal of Computer Applications*, 20(5), 40–46. https://doi.org/10.5120/2457-3307

Kwok, Y. K., & Ahmad, I. (1999). Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys (CSUR)*, 31(4), 406–471. https://doi.org/10.1145/344588.344618

Lin, W., Liu, C., & Zhang, Y. (2020). A heuristic scheduling algorithm for hybrid grid-cloud computing environments. *Cluster Computing*, 23(2), 939–952. https://doi.org/10.1007/s10586-019-02935-4

Liu, C. H., Lin, C. S., & Lee, J. W. (2008). Fuzzy scheduling algorithm for grid computing system. *Future Generation Computer Systems*, 24(8), 906–917. https://doi.org/10.1016/j.future.2007.10.002

Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., & Freund, R. F. (1999). Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 59(2), 107–131. https://doi.org/10.1006/jpdc.1999.1548

Mishra, S. K., & Sahoo, B. (2013). Efficient task scheduling for cloud computing environment. *International Journal of Engineering Research & Technology*, 2(10), 1250–1256.

Muthusamy, G., Kannan, R., & Chandrasekaran, R. M. (2012). An efficient scheduling algorithm for load balancing using hybrid particle swarm optimization in grid environment. *International Journal of Computer Applications*, 45(17), 14–19. https://doi.org/10.5120/6936-9247

Nabrzyski, J., Schopf, J. M., & Weglarz, J. (2004). *Grid resource management: State of the art and future trends*. Springer Science & Business Media.

Nazeer, K. A. A., & Azween, A. (2010). Artificial immune system based scheduling algorithm for grid computing. *International Journal of Computer Science and Information Security*, 8(4), 71–77.

Page, J., & Carrera, D. (2006). A resource-aware heuristic for scheduling workflows in grid computing. *Grid 2006 Workshop*, IEEE, 131–137.

Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. *24th IEEE International Conference on Advanced Information Networking and Applications*, 400–407. https://doi.org/10.1109/AINA.2010.31

Pandit, P., & Tripathy, S. (2012). Load balancing in cloud computing using modified active monitoring load balancer. *International Journal of Engineering Research and Applications*, 2(4), 1003–1007.

Qureshi, A., & Rizvi, S. A. (2015). A comparative analysis of task scheduling strategies in grid computing. *Journal of Computer Sciences and Applications*, 3(2), 23–29. https://doi.org/10.12691/jcsa-3-2-1

Rahman, M., & Barker, A. (2014). Scheduling workflows in cloud using cost and deadline constrained provisioning. *IEEE Transactions on Services Computing*, 7(4), 489–500. https://doi.org/10.1109/TSC.2013.229

Rashid, M., & Raza, S. (2018). Machine learning approaches in scheduling of tasks in grid computing: A review. *International Journal of Advanced Computer Science and Applications*, 9(5), 105–113. https://doi.org/10.14569/IJACSA.2018.090516

Singh, M. P., Bansal, R., & Jangra, A. (2014). A new approach for task scheduling based on genetic algorithm in cloud computing. *International Journal of Computer Applications*, 96(25), 29–34. https://doi.org/10.5120/16883-6785

Singh, S., & Chana, I. (2016). A survey on resource scheduling in cloud computing: Issues and challenges. *Journal of Grid Computing*, 14(2), 217–264. https://doi.org/10.1007/s10723-015-9359-2

Smanchat, S., & Viriyapant, K. (2009). Bee colony optimization for job scheduling in grid computing. *International Journal of Computer and Electrical Engineering*, 1(5), 638–642. https://doi.org/10.7763/IJCEE.2009.V1.95

Somasundaram, K., & Govindarajan, R. (2009). An efficient load balancing algorithm for heterogeneous grid computing. *International Journal of Computer Science Issues*, 6(2), 38–43.

Suri, B., & Garg, K. (2013). A novel heuristic based task scheduling algorithm for grid computing using genetic algorithm. *International Journal of Computer Applications*, 64(22), 6–12. https://doi.org/10.5120/10701-5570

Topcuoglu, H., Hariri, S., & Wu, M. Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3), 260–274. https://doi.org/10.1109/71.993206

Venugopal, S., Buyya, R., & Ramamohanarao, K. (2006). A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computing Surveys (CSUR)*, 38(1), 3–53. https://doi.org/10.1145/1132952.1132955

Wang, Y., Wang, H., Zhang, X., & Liu, Y. (2010). A hybrid heuristic algorithm for job scheduling problem in grid computing. *Proceedings of the 2010 International Conference on Computational Intelligence and Software Engineering*, 1–5. https://doi.org/10.1109/CISE.2010.5676982

Xhafa, F., & Abraham, A. (2010). Metaheuristics for grid scheduling problems. In *Metaheuristics for scheduling in distributed computing environments* (pp. 1–37). Springer. https://doi.org/10.1007/978-3-642-13293-4_1

Yao, Q., Liu, L., Wu, Z., & Luo, D. (2017). An improved particle swarm optimization algorithm for grid scheduling considering energy consumption. *Future Generation Computer Systems*, 76, 291–298. https://doi.org/10.1016/j.future.2016.06.026

Yu, J., & Buyya, R. (2006). A taxonomy of scientific workflow systems for grid computing. *ACM SIGMOD Record*, 34(3), 44–49. https://doi.org/10.1145/1084805.1084814

Zhao, J., Zhang, W., Liu, J., & Li, Y. (2009). A task scheduling algorithm based on PSO for grid computing. *International Symposium on Intelligent Information Technology Application Workshops*, 53–56. https://doi.org/10.1109/IITAW.2008.136

Zomaya, A. Y., & Teh, Y. H. (2001). Observations on using genetic algorithms for dynamic load-balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(9), 899–911. https://doi.org/10.1109/71.951065