

Automatika

Journal for Control, Measurement, Electronics, Computing and Communications



ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/taut20

Effective task scheduling based on interactive autodidactic school algorithm for cloud computing

G. Senthilkumar, B. Suvarnamukhi, S. Lekashri & M. Mohammed Thaha

To cite this article: G. Senthilkumar, B. Suvarnamukhi, S. Lekashri & M. Mohammed Thaha (2024) Effective task scheduling based on interactive autodidactic school algorithm for cloud computing, *Automatika*, 65:1, 159-166, DOI: [10.1080/00051144.2023.2288484](https://doi.org/10.1080/00051144.2023.2288484)

To link to this article: <https://doi.org/10.1080/00051144.2023.2288484>



© 2023 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.



Published online: 10 Dec 2023.



Submit your article to this journal [↗](#)



Article views: 654



View related articles [↗](#)



View Crossmark data [↗](#)



Citing articles: 1 View citing articles [↗](#)



Effective task scheduling based on interactive autodidactic school algorithm for cloud computing

G. Senthilkumar^a, B. Suvarnamukhi^b, S. Lekashri^c and M. Mohammed Thaha^d

^aDepartment of Computer Science and Engineering, Panimalar Engineering College, Chennai, India; ^bDepartment of CSE, Neil Gogte Institute of Technology, Hyderabad, India; ^cDepartment of ECE, Kings Engineering College, Sriperumbudur, India; ^dDepartment of Computer Science and Engineering, B.S. Abdur Rahman Crescent Institute of Science and Technology, Vandalur, India

ABSTRACT

The topic of load balanced task scheduling has emerged as a prominent and intricate area of study within the realm of Cloud computing. Swarm intelligence-based meta-heuristic algorithms are commonly considered more suitable for the purposes of Cloud scheduling and load balancing. These algorithms employ a combination of local and global search strategies in order to ascertain the ideal location. To achieve an optimal mapping strategy for task allocation to resources, it is imperative to find a suitable equilibrium between local and global search techniques, since this approach has demonstrated significant efficacy. This research introduces a new approach to task scheduling using the Autodidactic Interactive School Optimization Algorithm (IASOA). The objective of this method is to decrease the time required for job execution while also enhancing throughput. The assessment of the suggested methodology has been executed, and a comparative analysis has been performed with five established algorithms in relation to makespan and throughput. The tests were subsequently extended to encompass a comparative analysis of the suggested methodology alongside four other established meta-heuristic scheduling methodologies. The study of the simulated experimentation reveals that the proposed approach yielded noteworthy advantages in makespan and throughput, with improvements of up to 10% and 60% respectively.

ARTICLE HISTORY

Received 4 September 2023
Accepted 14 November 2023

KEYWORDS

Task scheduling; Makespan; cloud computing; optimization algorithm



1. Introduction

Cloud computing is an emerging approach within the information technology (IT) sector that facilitates the delivery and management of hardware and software resources via the Internet. The virtualization technology is the primary catalyst for the cloud computing paradigm, since it enables the resources within the cloud system to exist in a virtualized state. Within the cloud platform, users have the ability to rent various computer resources, including storage, memory, CPU, apps, and development platforms, through network connectivity.

The scheduling of tasks and resources is a critical and complex undertaking within the realm of cloud computing. When end-users submit tasks, also known as user requests, to the cloud system, these requests undergo processing by a scheduling algorithm. Subsequently, the requests are allotted to the virtual machines (VMs) that are currently available. The objective of task scheduling is to optimize the utilization of resources and improve job execution. Task scheduling is a problem in multi-objective optimization that falls within the category of issues classified as non-deterministic polynomial (NP) hard. Approximation methods are commonly employed for tackling problems of this nature.

In the context of task scheduling in cloud computing, approximation methods, such as metaheuristics, have the ability to identify an optimal solution or a solution that closely approximates the optimum within a reasonable timeframe. These approaches consider multiple performance parameters, including completion time, cost, resource utilization, and other factors that collectively impact the overall quality of service (QoS) experienced by end-users. In recent times, the research community has shown significant interest in bio-inspired metaheuristics owing to its notable resilience and effectiveness. Swarm intelligence, a category of population-based techniques, stands as a significant representation within the realm of nature-inspired algorithms.

These methods replicate the collective ordered behavior of a group of organisms in nature, without the presence of any centralized coordination component. Swarm algorithms exhibit distinct characteristics of randomization, and their search process is facilitated through the utilization of two mechanisms, namely exploitation (intensification) and exploration (diversification). During the exploration phase, the algorithm conducts a comprehensive search of the whole search space, while in the exploitation phase, it focuses on

CONTACT G. Senthilkumar  gsenthilkumar.cse23@gmail.com  Department of Computer Science and Engineering, Panimalar Engineering College, Poonamallee, Chennai-600123

doing localized searches in the vicinity of the existing optimal solutions.

Numerous swarm algorithms are readily accessible and expounded upon in contemporary computer science literature. The benchmark functions [1–3] have been effectively validated, and numerous implementations for actual issues that yield exceptional outcomes are readily accessible. Various swarm algorithms are available in the existing literature [4–7]. Swarm algorithms have been applied in various domains, including classification and feature selection [8,9], and localization [10]. Furthermore, the literature analysis reveals that there are several implementations of swarm algorithms in the realm of cloud computing [11–16]. The objective of the research outlined in this publication is to enhance a specific case of multi-objective task scheduling in the context of cloud computing by implementing the IAS algorithm.

The subsequent sections of the paper are structured in the following manner. Section 2 provides an overview of the relevant literature. The work that is being proposed is outlined in section 3. The analysis of the performance of the proposed work is presented in section 4. Section 5 encompasses the presentation of the final remarks.

2. Review of literature

The authors of [17] propose a hybrid load-balancing approach that combines the methods found in Teaching Learning Based Optimization (TLBO) with the Grey Wolves Optimization (GWO) method. In order to accomplish effective load balancing, the suggested methodology combines the beneficial aspects of the Grey Wolf Optimization (GWO) and the Teaching-Learning-Based Optimization (TLBO) algorithms. This takes into account the time factor as well as the corresponding cost factor. In addition to this, the use of this method results in a significant reduction in the amount of time spent waiting for items in the task queue. On the other hand, there is not enough care given to throughput.

In the framework of Cloud Computing, the researchers referred to in reference [18] have developed and put into practice a strategy for the scheduling of work that takes into mind any deadlines that may be present. The genetic algorithm (GA) was used by the scheduling system to maximize the amount of time spent executing tasks and the cost of resources. This was accomplished by taking into account the degree to which virtual machine (VM) performance and acquisition latency varied. However, when confronted with broad and complicated issues, GA have difficulties in terms of their ability to scale. The current predicament presents a difficult obstacle. The ability to foresee and prepare for potential outcomes in the future is referred to as the “look-ahead” notion.

The Large-scale Genetic Algorithm (LAGA) is a variation of the GA that has been recommended by [19] as a suitable strategy for solving the issues given by large-scale distributed systems, such as Cloud and Grid computing environments. The LAGA is widely regarded as a suitable solution for the scheduling of compute-intensive operations and assuring stability throughout run-time. This is because LAGA takes into account the environment in which it is operating. The methodology that has been suggested involves determining task orders by taking into account the amount of time it takes resources in each generation to finish their work. In the following stage, known as the mutation step, the resource with the highest success rate is chosen to be mutated. The reliability and reduction in job failure rate are both included in the objectives of this strategy’s scheduling. This method, on the other hand, does not take into account the scheduling criteria of makespan and throughput.

The authors have presented a Node duplication-based Genetic Algorithm (NGA), which is a GA that has been created specifically for multi-processor heterogeneous systems [20]. This NGA is a genetic algorithm that was introduced by the authors. The evaluation and optimization of communication delay time and application completion time in relation to assigned resources is the primary focus of the National Grid Architecture (NGA), which was developed by the National Science Foundation (NSF). The fitness function of this method is subjected to a process of evolution that takes place in two stages. The tasks’ appropriateness for the lawful order is taken into consideration in the process of scheduling those tasks, which may entail reporting to the system on the status of all other tasks. In this instance, the legal system is responsible for organizing a collection of independent responsibilities to be carried out by a single processing unit. At this stage, the National Geospatial-Intelligence Agency (NGA) determines whether or not the processor is suitable for use by determining whether or not it is able to complete the task in the least amount of time possible. The Next Generation Algorithm (NGA) has inherited from its forerunner, the genetic algorithm, the difficulties associated with scaling up in order to solve large and complex problems.

An examination of the similarities and differences between the GA and the Particle Swarm Optimization (PSO) approaches was carried out in an earlier research project that the author carried out [21]. In order to conduct this research, multiple different test scenarios were utilized. The empirical evidence reveals that the GA is outperformed by the PSO method in the majority of the test cases when comparing the two algorithms in terms of solution quality and the amount of computational efficiency achieved. In the setting of distributed systems, it is argued that the performance of the PSO algorithm is superior to that of

the GA. This conclusion is based on their experimental findings.

Gravitational Emulation Local Search (GELS) and PSO are both incorporated into the task scheduling technique that was provided by the author in reference [22]. This method's goal is to shorten the makespan while simultaneously increasing the likelihood that all operations will be finished on time. On the other hand, there is no concern given to throughput. An investigation on the job scheduling methods used in the cloud was carried out by the authors of reference [23] utilizing the PSO method. The authors have organized the previously conducted research on PSO-based optimization into categories according to the number of different objectives that needed to be optimized. Both single-objective and multiple-objective formulations are included in the classification of PSO algorithms. The author has come to the conclusion that there is a requirement for more focus and improvement in order to achieve balanced scheduling and meet Quality of Service (QoS) criteria, such as makespan, throughput, and resource utilization. There has been a lack of attention paid to the consideration and investigation of inertia weight.

The authors of [24] proposed several meta-heuristic algorithms, and these algorithms were founded on the ideas of IRRO (Iterative Random Restart Optimization) and CSO (Cuckoo Search Optimization). The suggested method incorporates the useful aspects of both CSO and IRRO algorithms, which makes it possible to take a unified approach to the global and local search processes. The proposed method also includes the implementation of a dynamic scheduling framework that goes by the name of the IRRO-CSO based Dynamic Scheduling Framework (ICDSF). We have made use of a variety of evaluation criteria, including response time, premature convergence, makespan, and throughput. On the other hand, the application of ARUR is not considered to be a valid criterion for assessment.

The authors present a novel method that they refer to as RTPSO-B in [25]. This method combines the Rang and Tune based PSO technology with the Bat algorithm method. The goal of the RTPSO-B method, which is a modified version of PSO algorithm, is to improve the efficiency with which jobs are scheduled in systems that make use of cloud computing. By incorporating the data localization strategy, this method offers a solution to the problem of the existing PSO's excessive inertia weight. The employment of a low inertia weight makes the local search mechanism easier to operate, whereas the utilization of a high inertia weight makes the global search process easier to carry out. The PSO technique has been incorporated into the Bat algorithm in order to improve the algorithm's overall performance in terms of optimization. Utilization of Cloud resources, makespan, and cost are typically included

as primary components of the evaluation parameters for scheduling algorithms. On the other hand, throughput is rarely taken into consideration during the review process.

The Integer-PSO algorithm [26] is a variation of the PSO method that was developed specifically for the purpose of scheduling work in environments that make use of cloud computing. The application of Integer-PSO is appropriate for the optimization of task scheduling in the Cloud, encompassing both single and multiple objectives. This is because Integer-PSO can take into account more than one objective at a time. The Integer-PSO is a solution that has been developed for the problem of bi-objective optimization. The goals of this research include determining how long it takes to complete a task, how much time and money are spent on calculation and administration, and measuring both of these factors. However, the Integer-PSO method does not take into consideration throughput as a main objective for task scheduling. Instead, it depends on a pre-determined value for the inertia weight to determine which tasks should be completed first.

In the research that they conducted, the authors cited in reference [27] presented a method for load balancing that they dubbed Hyper-heuristic. This method makes use of a honey bee and an improvement detecting operator. The methodology that has been provided has the capability to distribute the workload of cloud-based tasks over several virtual machines in such a way as to reduce the amount of time that is necessary to finish all of the chores.

The authors have proposed a Task-oriented Load Balancing method that they name TBSLB-PSO in their work [28]. This method makes use of a technique known as PSO. The TBSLB-PSO algorithm improves load balancing by using task migration to move jobs from virtual machines (VMs) that are overloaded to those that are underloaded. These VMs are known as "underloaded." The proposed scheduling method intends to reduce the workload associated with load balancing by streamlining the process of work migration without calling for the shutdown of any virtual machines (VMs) that are already operating at capacity.

Motivated by these existing works, this article intends to present an effective task scheduling algorithm based on IAS algorithm.

3. Proposed task scheduling algorithm

The primary objective of this study is to optimize the trade-off between power consumption and performance to the greatest extent. The algorithms used in TS can be classified into two categories: deterministic and non-deterministic. Deterministic algorithms consistently produce the same outcome for a given input, while non-deterministic algorithms yield varied outcomes dependent on the decisions made during their

execution. The proposed work relies on IAS algorithm and the background of IAS algorithm is presented as follows.

3.1. IAS Algorithm

The IAS can be classified as a metaheuristic algorithm that leverages student interactions within an autodidactic classroom setting to yield a globally optimum solution. The autodidactic educational institution is founded upon the principle of independent acquisition of knowledge and does not involve the presence of an instructor. The IAS, as described in the literature [21], is a population-based algorithm that comprises exceptional individuals and the rest of the student community. The students' knowledge is assessed through their achieved marks, which in turn can be used to identify their academic performance. Nevertheless, there is an ongoing battle within the student community to attain the position of lead student. Therefore, this algorithm is founded upon the principles of self-learning, discourse, and competition in order to achieve a dominant position. The representation of the student generation and their academic success is depicted as follows.

$$\begin{aligned} st_i &= L_{bd} + k_s(0, 1) * U_{bd} - L_{bd} \\ MK_i &= f(st_i) \\ \text{while}(s = 1 : tl_{student}); \\ f(Ld_s) &= \text{Min}\{M_k\} \end{aligned}$$

Here, st_i stands for the i^{th} student, L_{bd} and U_{bd} denote lower and upper bounds of the variables, $k_s(0, 1)$ is the random number that lies in between 0 and 1. $tl_{student}$ is the total number of students, MK_i is the respective mark of i^{th} student and Ld_s is the leading student.

The algorithm is structured around three distinct phases, namely individual training, communal training, and the new student challenge. During the individual training session, the student in a leadership role engages in discussions and imparts knowledge to the rest of the student community. The collective training session encompasses a preliminary conversation regarding the most recent session with the student community, as well as a troubleshooting session with the lead students. Through this approach, the pupils' knowledge is augmented. The new student is presented with a challenge upon entering the classroom. The encounter between the new student and the student lead presents a potential challenge for the latter in terms of sustaining their leadership position. The aforementioned process persists till the specified termination criterion is satisfied.

3.1.1. Individual training session

The training session is conducted by the student leader in collaboration with the other students. At the outset,

the students are randomly assigned, ensuring that each student is paired with a minimum of two other pupils. This discourse enhances the student's knowledge, with the extent of knowledge acquisition contingent upon the student's level of skill. The representation of this session is as follows.

$$\begin{aligned} &\text{Choose a student } st_j, i \neq j \\ FS_s^* &= FS_s + k_s(1, 2) * (Ld_s - bc_s * FS_s) \\ FS_t^* &= FS_t + k_t(1, 2) * (Ld_s - bc_t * FS_t) \\ &\text{while}(s = 1 \text{ to } tl_{student}); \\ &\text{Accept } FS_s^* \text{ and } FS_t^* \text{ when better marks} \\ &\text{are achieved by } FS_s \text{ and } FS_t \end{aligned}$$

Here, FS_s^* and FS_t^* are the first and second fellow students and the proficiency competence is indicated by bc_s, bc_t . $k_s(1, 2)$ and $k_t(1, 2)$ are the vectors in random between 1 and 2. The competences are randomly assigned with value 1 or 2.

3.1.2. Collective training session

Upon the conclusion of the individual training session, every student is afforded the opportunity to review the preceding session, engaging in interactive discussions with fellow group members. During this collaborative session, the student leader engages in discussions on various issues with their peers. The collective session is denoted by the subsequent information.

$$\begin{aligned} ClCp_{st} &= (ClCp_s * FS_s + ClCp_t * FS_t) / \\ &(ClCp_s + ClCp_t) \\ FS_s^* &= FS_s + k_s(1, 2) * (Ld_s - ClCp_s * ClCp_{st}) \\ FS_t^* &= FS_t + k_t(1, 2) * (Ld_s - ClCp_t * ClCp_{st}) \\ &\text{while}(s = 1 \text{ to } tl_{student}); \end{aligned}$$

$ClCp_{st}$ represents the collective capability of the student group, which is computed on the basis of student competency weight. k_s, k_t are the random vectors and the collective proficiency competencies are chosen as either 1 or 2.

3.1.3. New student challenge

This phase enhances the level of exploration, allowing for the replacement of the student lead whenever a superior student is identified. The representation of this can be illustrated as follows.

$$\begin{aligned} New_s &= L_{bd} + k * (U_{bd} - L_{bd}) \\ AF_1 &= \text{round}(k(0, 1)) \\ AF_2 &= 1 - AF_1 \\ Ld_s^* &= AF_1 * Ld_s + AF_2 * New_s \\ &\text{Score more marks than } Ld_s \end{aligned}$$

3.2. Proposed task scheduling

In this context, New_s refers to the newly enrolled student. The initial and subsequent modifying factors are denoted as AF_1 and AF_2 , respectively. The variable $k(0, 1)$ represents a random vector ranging from 0 to 1. Ld_s^* has emerged as the top-performing student in the class. Therefore, the functionality of the IAS algorithm is elucidated.

In the proposed algorithm, each student indicates a unique schedule. Each student represents n subtasks, starting from 0, 1, 2, . . . , $n - 1$. Let $TK(h)$ indicate a group of subtasks with height h . Initially, the subtasks are assigned to low power consuming processor. However, when a subtask is about to meet the deadline, then they are assigned to high performance processor to get them done faster. Hence, the major objective of the work is to reduce makespan and power consumption. When a subtask meets its deadline, then a penalty is issued, which is given by.

$$Fit = Max(|\mu, 100|) \quad (1)$$

In the above equation, μ is the time missed by the subtask with respect to the deadline. Suppose, when the subtask is accomplished before its deadline, then the time is added to the points.

$$Fit = Points + \mu \quad (2)$$

The subtasks with greater fitness value is chosen for execution and its probability is given by

$$p_i = \frac{fit(i)}{\sum_{j=1}^n fit(j)} \quad (3)$$

Hence, this approach takes care for not violating the precedent constraints and minimizes the makespan and power consumption. The following section analyses the performance of the proposed work.

4. Performance analysis

The following part provides an overview of the computational environment utilized for conducting the simulation. In order to conduct tests and assess the efficacy of the suggested scheduling strategy, a simulation environment was employed. This is due to the fact that within the simulation environment, researchers have the flexibility to utilize a wide range of resources, including diverse quantities and characteristics, in order to conduct their studies. Furthermore, tests can be conducted multiple times without limitations on time or cost of execution. The experimental configuration for conducting tests includes a personal computer (PC) with a central processing unit (CPU) of Intel core i5 T8500 3.0 GHz, a memory of 20.00 GB, and a hard drive of 2 TB. The experiments are implemented using the Java-based Eclipse IDE 3.0 and the Cloudsim

Table 1. Simulation parameters.

Parameter	Value
Simulation tool	Cloudsim
Processor	Intel core i5-8500 3.00 GHz
Total machines	20
Total tasks	8000

3.0.3 simulation tool. Table 1 provides a comprehensive overview of the experimental setup employed for the purpose of conducting the experiments. The performance of the proposed work (IAS) is compared against the existing PSO [28], WO [12], INT-PSO [26], Hybrid [25], GWO [18].

In order to give statistically meaningful and realistic evaluations, meta-heuristic-based algorithms need to go through multiple iterations. This is because stochastic optimization approaches fall into this category. In this particular investigation, every tactic is implemented five times for each and every instance of the HCSP dataset. The averages of the results are then computed and presented for the purpose of comparison. The makespan is the evaluation parameter that is of the biggest importance and serves as the last requirement for cloud users. Figure X is a graphical representation of the findings that regard to the duration of time that is required to finish a collection of activities, which is known as the makespan. These findings pertain to both the IAS as well as existing state-of-the-art methodologies. i_hilo, c_hilo, c_lohi, and i_lohi are all cases that are included in the HCSP benchmark dataset. These examples were taken into consideration for this analysis.

The Hybrid [25] and GWO [18] techniques have shown improved performance when dealing with c_hilo and i_hilo cases from the HCSP dataset. On the other hand, they have shown significantly less success when addressing c_lohi and i_lohi scenarios. The PSO [28] method has shown that it performs significantly better on c_hilo and i_hilo instances of the HCSP dataset when compared to its performance on c_lohi and i_lohi examples of the same dataset. The reason for this is that the task sizes in the HCSP instances c_hilo and i_hilo are significantly less when compared to the task sizes in the instances c_lohi and i_lohi. In addition, the linearly declining mode of the GWO [18] approach has enabled it to demonstrate consistent performance across all instances of the HCSP dataset.

The IAS method demonstrates that it is possible to shorten the makespan for each and every instance of the HCSP dataset. It achieves improvements ranging from 1% to 7% on the i_hilo instances, 2% to 11% on the c_hilo instances, 1% to 5% on the i_lohi instances, and 1% to 4% on the c_lohi instances of the HCSP benchmark dataset when compared to the state-of-the-art PSO [28], WO [12], INT-PSO [26], Hybrid [25], GWO [18]. This illustrates that the strategy that was proposed does an effective job of maintaining a more optimal

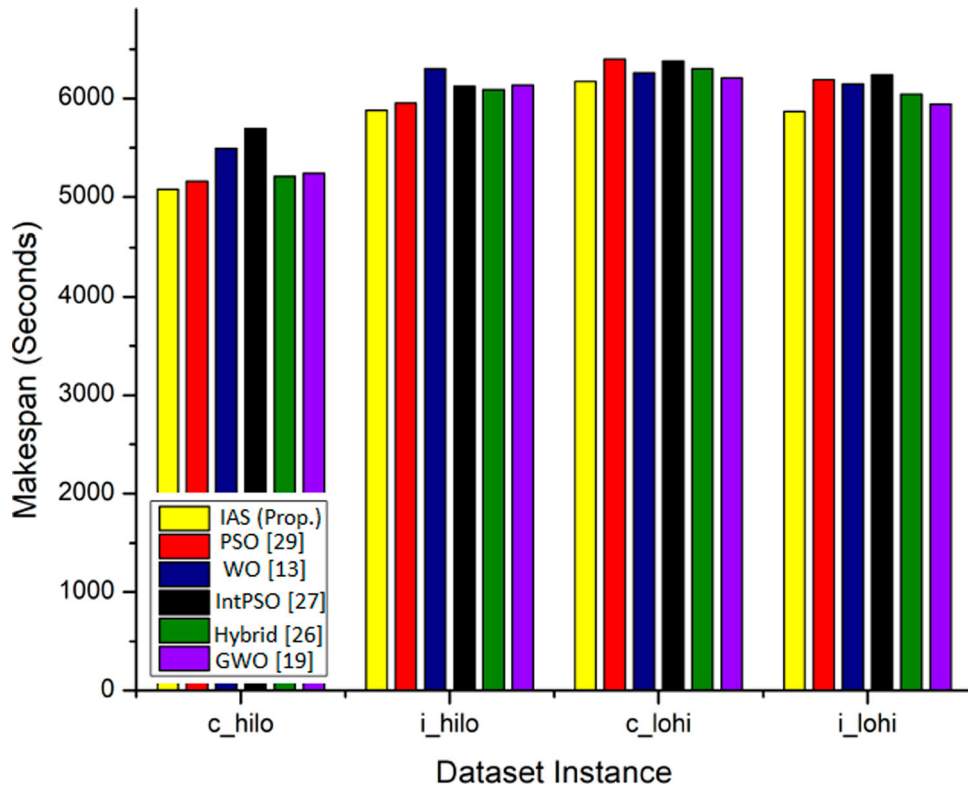


Figure 1. Makespan analysis.

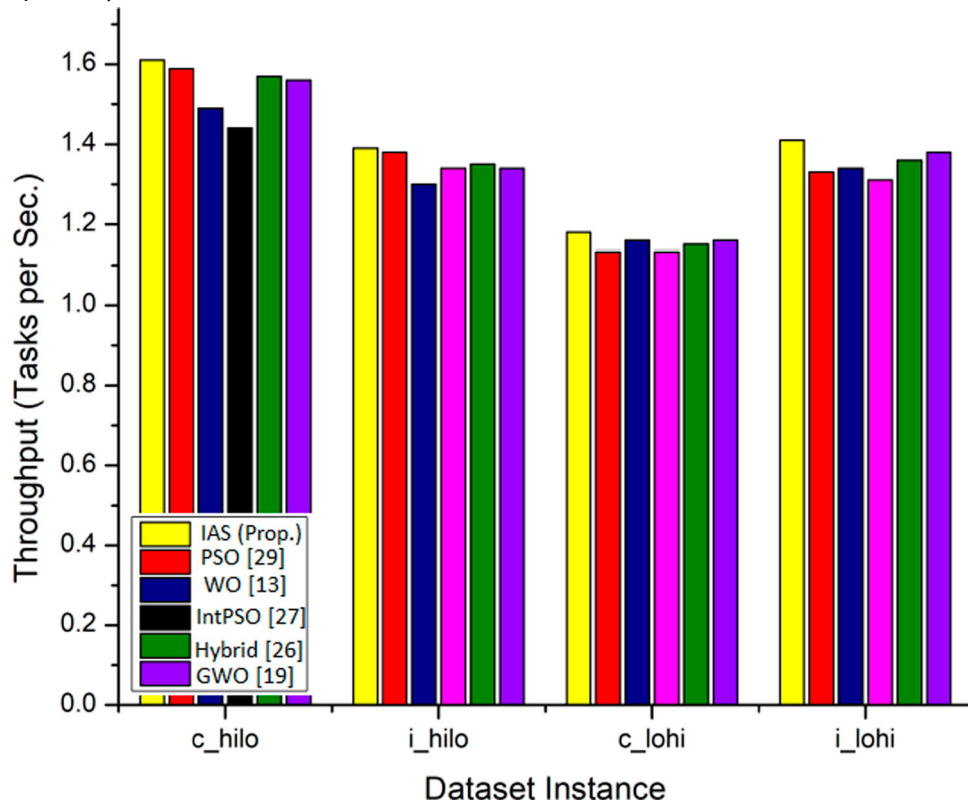


Figure 2. Throughput analysis.

equilibrium between the local and global search processes (Figure 1).

Throughput is another important statistic for evaluating the performance of the IAS. Figure 2 illustrates the throughput outcomes obtained from the implementation of the IAS strategy, as well as its comparison with other contemporary approaches. The comparison

is conducted specifically for the c_hilo, i_hilo, c_lohi, and i_lohi instances of the HCSP dataset. Similarly, the throughput data exhibit comparable patterns across all of the compared techniques, mirroring the findings reported in the makespan results.

According to the findings presented in Figure 2, it can be observed that the suggested scheduler has

demonstrated improved throughput performance in the range of 1–7% for the execution of *c_hilo*, 2–12% for *i_hilo*, 1–6% for *c_lohi*, and 1–4% for the execution of *i_lohi* instances of the HCSP dataset. This improvement is in comparison to the performance produced by IAS with PSO [28], WO [12], INT-PSO [26], Hybrid [25], GWO [18].

5. Conclusion

The topic of task scheduling in the cloud computing model poses a significant challenge, as it directly impacts system performance. In this study, a proposed task scheduler algorithm, hybridized BA (BAAEQRL), is introduced as a potential solution for addressing the specific issue at hand. The cloud system model employed in experimental settings embodies a multi-objective optimization conundrum. The allocation of a high-performance processor is prioritized when a sub-task is approaching its deadline. This concept efficiently addresses the issues of makespan and power consumption. The efficacy of the suggested study is evaluated by comparing its performance with that of several existing works, with a focus on the metrics of makespan and power usage. The performance of the suggested work appears to be adequate, exhibiting improved outcomes. Accurate decision-making in the majority of Internet of Things (IoT) applications necessitates real-time responses. In the forthcoming endeavor, the objective is to enhance the efficiency of response time and implement scheduling techniques in order to deliver prompt or nearly prompt responses for applications that are sensitive to delays.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

- [1] Karaboga D, Akay B. A modified artificial bee colony (ABC) algorithm for constrained optimization problems. *Appl Soft Comput.* 2011;11(3):3021–3031. doi:10.1016/j.asoc.2010.12.001
- [2] Bacanin N, Tuba M. Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators. *Stud Inf Control.* 2012;21(2):137–146. doi:10.24846/v21i2y201203
- [3] Tuba M, Nebojsa M. Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems. *Neurocomputing.* 2014;143:197–207. doi:10.1016/j.neucom.2014.06.006
- [4] Cheng L, Wu X-h, Wang Y. Artificial flora (AF) optimization algorithm. *Appl Sci.* 2018;8:329. doi:10.3390/app8030329
- [5] Bezdán T, Tuba E, Strumberger I, et al. Automatically designing convolutional neural network architecture with artificial flora algorithm. In: Tuba M, Akashe S, Joshi A, editors. *ICT Systems and Sustainability Vol. 1077.* Singapore: Springer; 2020. p. 371–378.
- [6] Bacanin N, Bezdán T, Tuba E, et al. Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics. *Algorithms.* 2020;13(3):67. doi:10.3390/a13030067
- [7] Bacanin N, Bezdán T, Tuba E, et al. Monarch butterfly optimization based convolutional neural network design. *Mathematics.* 2020;8(6):936. doi:10.3390/math8060936
- [8] Tuba E, Strumberger I, Bezdán T, et al. Classification and feature selection method for medical datasets by brain storm optimization algorithm and support vector machine. *Procedia Comput Sci.* 2019;162:307–315. 7th International Conference on Information Technology and Quantitative Management (ITQM 631 2019): Information technology and quantitative management based on Artificial Intelligence. doi:10.1016/j.procs.2019.11.289
- [9] Zivkovic M, Bacanin N, Tuba E, et al. Wireless sensor networks life time optimization based on the improved firefly algorithm). 2020 International Wireless Communications and Mobile Computing (IWCMC); 2020. p. 1176–1181.
- [10] Strumberger I, Minovic M, Tuba M, et al. Performance of elephant herding optimization and tree growth algorithm adapted for node localization in wireless sensor networks. *Sensors.* 2019;19(11):2515. doi:10.3390/s19112515
- [11] Kalra M, Singh S. A review of metaheuristic scheduling techniques in cloud computing. *Egyptian Informatics Journal.* 2015;16(3):275–295. doi:10.1016/j.eij.2015.07.001
- [12] Sreenu K, Sreelatha M. W-scheduler: whale optimization for task scheduling in cloud computing. *Cluster Comput.* 2019. doi:10.1007/s10586-017-1055-5
- [13] Bacanin N, Tuba E, Bezdán T, et al. Artificial flora optimization algorithm for task scheduling in cloud computing environment. In: Yin H, Camacho D, Tino P, editors. *Intelligent Data Engineering and Automated Learning – IDEAL 2019.* IDEAL 2019. Lecture Notes in Computer Science Vol. 11871. Cham: Springer; 2019. p. 437–445. https://doi.org/10.1007/978-3-030-33607-3_47
- [14] Strumberger I, Tuba M, Bacanin N, et al. Cloudlet scheduling by hybridized monarch butterfly optimization algorithm. *J Sens Actuator Netw.* 2019;8(3):44, doi:10.3390/jsan8030044
- [15] Strumberger I, Tuba E, Bacanin N, et al. Dynamic tree growth algorithm for load scheduling in cloud environments. 2019 IEEE Congress on Evolutionary Computation (CEC); 2019. p. 65–72. doi:10.1109/CEC.2019.8790014
- [16] Bacanin N, Bezdán T, Tuba E, et al. Task scheduling in cloud computing environment by grey wolf optimizer. 2019 27th Telecommunications Forum (TELFOR). IEEE; 2019. p. 1–4.
- [17] Mousavi S, Mosavi A, Varkonyi-Koczy AR. A load balancing algorithm for resource allocation in cloud computing. In: Luca D, Sirghi L, Costin C, editors. *Recent Advances in Technology Research and Education. INTER-ACADEMIA 2017.* Advances in Intelligent Systems and Computing Vol. 660. Cham: Springer; 2017. p. 289–296. https://doi.org/10.1007/978-3-319-67459-9_36
- [18] Meena J, Kumar M, Vardhan M. Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint. *IEEE Access.* 2016;4:5065–5082. doi:10.1109/ACCESS.2016.2593903

- [19] Wang X, Yeo CS, Buyya R, et al. Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm. *Future Gener Comput Syst.* 2011;27:1124–1134. doi:10.1016/j.future.2011.03.008
- [20] Singh J, Singh H. Efficient Tasks scheduling for heterogeneous multiprocessor using genetic algorithm with node duplication. *Indian J Comput Sci Eng.* 2011;2:402–410.
- [21] Zhang L, Chen Y, Sun R, et al. A task scheduling algorithm based on PSO for grid computing. *Int J Comput Intell Res.* 2008;4:37–43.
- [22] Pooranian Z, Shojafar M, Tavoli R, et al. Hybrid meta-heuristic algorithm for job scheduling on computational grids. *Informatica.* 2013;37:157–164.
- [23] Alkayal ES, Jennings NR, Abulkhair MF. Survey of task scheduling in cloud computing based on particle swarm optimization. In *Proceedings of the 2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, Ras Al Khaimah, United Arab Emirates, 21–23 November; 2017. p. 1–6.
- [24] Torabi S, Safi-Esfahani F. A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing. *J. Supercomput.* 2018;74:2581–2626. doi:10.1007/s11227-018-2291-z
- [25] Valarmathi R, Sheela T. Ranging and tuning based particle swarm optimization with bat algorithm for task scheduling in cloud computing. *Clust Comput.* 2019;22:11975–11988. doi:10.1007/s10586-017-1534-8
- [26] Beegom AA, Rajasree MS. Integer-PSO: A discrete PSO algorithm for task scheduling in cloud computing systems. *Evol Intell* 2019;12:227–239. doi:10.1007/s12065-019-00216-7
- [27] Gupta A, Bhadauria HS, Singh A. Load balancing based hyper heuristic algorithm for cloud task scheduling. *J Ambient Intell Humaniz Comput.* 2021;12:5845–5852. doi:10.1007/s12652-020-02127-3
- [28] Ramezani F, Lu J, Hussain FK. Task-based, system, load, balancing, in, cloud, computing, using, particle, swarm optimization. *Int J Parallel Program.* 2014;42:739–754. doi:10.1007/s10766-013-0275-4