Iraqi Journal of Science, 2023, Vol. 64, No. 8, pp: 4187-4197
DOI: 10.24996/ijs.2023.64.8.38

ISSN: 0067-2904

Intelligent Task Scheduling Using Bat and Harmony Optimization

Wijdan Abd Almahdi [1*], Hussein Attia Lafta[1], Yossra Hussain Ali[2]
[1]*Department of Computer Science, College of Science for Women, University of Babylon, Iraq*
[2]*Department of Computer Sciences, University of Technology, Baghadad, Iraq*

Received: 30/5/2022 Accepted: 27/10/2022 Published: 30/8/2023

Abstract

Cloud computing describes computer services provided through the internet and includes a wide range of virtualization resources. Because cloud computing is made up of a sizable number of heterogeneous autonomous systems with an adaptable computational architecture, it has been widely adopted by many businesses. The scheduling and management of resource utilization, however, have become more difficult as a result of cloud computing. Task scheduling is crucial, and this procedure must schedule tasks on the virtual machine while using the least amount of time possible. Utilizing an effective scheduling strategy enhances and expedites cloud computing services. Optimization techniques are used to resolve cloud scheduling problems. The purpose of this research is to address tasks distribution within the system to improve system performance overall and reduce task execution time. Two well-known optimization algorithms (the Bat, and Harmony search algorithms) were used in this approach as well as a combination approach Bat Algorithm Harmony Search (BAHS) that integrates the two. When compared to the other algorithms used for task scheduling, the (BAHS) method was chosen because it is flexible and produces effective results. Tests were run on a dataset that was created randomly. The suggested algorithm results were compared to the other popular algorithms in the field. The results show that the suggested swarm-based scheduling techniques can produce more accurate results than those of the competing algorithms in terms of the makespan , mean, and standard division.

Keywords: Cloud Computing, Task Scheduling, Harmony search algorithm, Hybrid Optimization Algorithm, Bat Algorithm.

جدولة المهام الذكية باستخدام تحسين الخفافيش والانسجام

وجدان عبد المهدي* , حسين عطية لفته , يسرى حسين علي
[1]قسم علوم الحاسوب ، كلية العلوم للبنات ، جامعة بابل ، العراق
[2]قسم علوم الحاسب الآلي الجامعة التكنولوجية، بغداد، العراق

الخلاصة

تصف الحوسبة السحابية خدمات الكمبيوتر المقدمة عبر الإنترنت وتتضمن مجموعة واسعة من موارد المحاكاة الافتراضية. نظرًا لأن الحوسبة السحابية تتكون من عدد كبير من الأنظمة المستقلة غير المتجانسة مع بنية حسابية قابلة للتكيف ، فقد تم اعتمادها على نطاق واسع من قبل العديد من الشركات. ومع ذلك ، أصبحت

*Email: wijdanalshammari1988@gmail.com

جدولة وإدارة استخدام الموارد أكثر صعوبة نتيجة للحوسبة السحابية. تعد جدولة المهام أمرًا بالغ الأهمية ، ويجب أن يقوم هذا الإجراء بجدولة المهام على الجهاز الظاهري أثناء استخدام أقل قدر ممكن من الوقت. يؤدي استخدام استراتيجية جدولة فعالة إلى تحسين خدمات الحوسبة السحابية وتسريعها. تُستخدم تقنيات التحسين لحل مشكلات جدولة السحابة. الغرض من هذا البحث هو معالجة توزيع المهام داخل النظام لتحسين أداء النظام بشكل عام وتقليل وقت تنفيذ المهام. تم استخدام خوارزميات تحسين مشهورة (خوارزميات البحث Bat و (Harmony في هذا النهج بالإضافة إلى نهج تركيبي لوغاريتم Bat Algorithm Harmony Search (BAHS) الذي يدمج الاثنين. عند مقارنتها بالخوارزميات الأخرى المستخدمة لجدولة المهام ، تم اختيار طريقة (BAHS) لأنها مرنة وتنتج نتائج فعالة. تم إجراء الاختبارات على مجموعة بيانات تم إنشاؤها عشوائيًا. تمت مقارنة نتائج الخوارزمية المقترحة مع الخوارزميات الشائعة الأخرى في هذا المجال. تظهر النتائج أن تقنيات الجدولة المقترحة القائمة على السرب يمكن أن تنتج نتائج أكثر دقة من تلك الخاصة بالخوارزميات المنافسة من حيث Makespan ، والمتوسط ، والتقسيم القياسي.

1. Introduction

A cloud computing environment is a collaborative information technology environment that is meant to measure and distribute scalable information technology resources for efficient and environmentally responsible usage. While offering dynamic stability and scalability, the goal of cloud computing is to enable computation over a broad range of resources [1]. As a kind of public-sector cluster and utility computing, it is a hybrid of the two. Some analysts believe that this endeavor has the potential to make computing a utility in the near future.

Many apps can be found on the Internet, where big data servers are used to host web-based applications. Users can access a vast amount of processing and storage resources on-demand via subscription-based services provided by the platform. Virtualized data centers, on the other hand, are represented by the clouds. In the cloud, virtual machines can be used to provide scalable, on-demand, and pay-per-use resources. Cloud computing is available over public and private networks as a new paradigm for deploying resources.

The National Academy of Science and Technology (NAST) defines Cloud computing as "on-demand, network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that may be quickly given and released with no administrative effort or service provider interaction".

According to the National Institute of Standards and Technology's (NIST) definition of cloud computing [2], "on-demand self-service, wide network access, resource pooling, quick elasticity, and quantified service" are the five key aspects of cloud computing. In cloud computing, there are three major cloud service models: software as a service (SaaS), infrastructure as a service (IaaS), and platform as a service (PaaS) that provide internet-based resources such as real computers, virtual machines, and virtual storage to customers [3][4]. An application's runtime environment is provided by the PaaS. It offers the essential development and deployment tools for the building of mobile applications.

The SaaS paradigm makes it feasible to provide software programs to end-users as a service. The term "hosted software" refers to applications that run on a server and are accessed through the Internet. Accordingly, it is possible for the software to be installed in one of three ways: on-premises, in a combination of on-premises, and the cloud. All software on a server that can be accessed over the Internet fall under this category, which is known as a public cloud service.

The public cloud is a deployment model that provides a system and two services that are open to the public on a large scale. This solution may not be as secure as other options since it is open. Private clouds, on the other hand, are systems and services that are owned by a single organization. This kind of cloud provides more security due to the anonymity it provides. There is a community cloud that provides systems and services to a number of organizations. It is only possible for the company that owns the cloud to have access to the information stored inside [5]. Using both public and private clouds together is known as "hybrid cloud computing". A hybrid cloud can provide some data deployment options to businesses in some cases.

The remaining sections of this research are organized as follows: in Section 2, related works in task scheduling algorithms and a brief introduction are included , followed by a description of the scheduling problem in cloud computing in Section 3. Section 4 presents the Task Scheduling Problem Formulation and Sections 5,6,7 present the Bat Optimization Algorithm, Harmony Search Algorithm, and The Proposed Hybrid BAHS Algorithm, respectively. Section 8 describes the experimental results and discussion. Finally, a conclusion is provided in Section 9.

2. Related Work

Before delving into the BAHS technique, which will be explained in detail in the next section, this section will highlight relevant works on task scheduling algorithms. Table 1 provides an overview of the associated works of the task scheduling algorithms, as well as their benefits and downsides.

A generalized mixed scheduling method was presented by Choudhury et al., [6] as a task scheduling technique for cloud computing's dual-objective workflow scheduling. This approach was devised to reduce the amount of time and money needed to perform a job. It is also possible that this approach will not work in more complex projects. Navimipour [7] provides a cloud computing work scheduling technique built on the ABC algorithm, a well- known algorithm in the computer science community. This method considers not only the time spent doing the work but also the time spent waiting and the number of jobs that were omitted as a result. However, there is no evidence to support the efficacy of this approach.
Since its inception, Tawfeek et al., [8] have advocated the ACO-based cloud work scheduling technique. To reduce the amount of time it takes to accomplish a sequence of tasks in a cloud environment. This approach outperforms both the FCFS and RR algorithms in terms of performance. As a result, it will take longer to come up with suitable answers than other methods, which is a drawback of this strategy.

Hamad and Amara [9] proposed utilizing GA-based task scheduling to determine job and performance. The use of this method has reduced the time and costs connected with it while also increasing the efficiency of resources. However, the accuracy with which the parameters are collected has an influence on the GA algorithm's performance.

[10] Raghavan and coworkers, using Bayesian reasoning, designed a technique for scheduling jobs that reduced manufacturing costs. High-dimensional fields, on the other hand, are a struggle for BA [11-13]. Consequently, this approach does not work well when trying to schedule tasks with a lot of variables.

Polepally and Chatrapati [14] utilized the Distributed Algorithm to schedule jobs for load balancing based on a distributed algorithm (DA). There was a significant drop in task

transmission when compared to previous methodologies. Besides, the DA approach has a high level of computing complexity due to the large number of distance computations it conducts.

Table 1: Related Works.

Author	Algorithm	Merits	Drawbacks
Choudhary et al. [6]	Gravitational search algorithm	Reduced time-to-makespan Support simple	
Navimipour [7]	Artificial bee colony	Execution time and waiting time have both been reduced.	Less performance in high dimension
Tawfeek et al. [8]	Ant colony optimization	Reduced makespan and improved load-balance	Slow convergence
Hamad and Amara [9]	Genetic algorithm	Reduced makespan, cost	Relies on the precisely defined Parameters
Raghavan et al. [10]	Bat algorithm	Overall cost reduction	Not suitable in high dimension
Polepally and Chatrapati[14]	Dragonfly algorithm	Reduced migrated task time complexity	High

3. Definition and Formulation of Problem

The allocation or mapping of a job to a processing unit is essential for task scheduling in cloud computing. Consider: for n tasks [T1, ..., Tn], and m processing machines [M1,..... Mm] available to compute those tasks that must fulfill condition n>m, or the number of tasks must be more than the number of processing or virtual machines accessible (VMs). Schedule the supplied tasks on VMs to minimize the time it takes and increase VM usage.

The implemented and utilized optimization objective functions for N tasks is When compared to the search space, the global minimum occupies a minimal area according to Easom function [12]. To zoom out, just reverse the function, and this can be defined as:

$$f_{Easo}(x_1, x_2) = -\cos(x_1) * \cos(x_2) * \exp\left(-((x_1 - \pi)^2 + (x_2 - \pi)^2)\right) \quad (1)$$

The test region is limited to the area between

$$-100 \leq x_i \leq 100, \quad i = 1{:}2 \quad (2)$$

And the global minimum is

$$f(x_1, x_2) = -1 \quad (3)$$

obtainable for

$$(x_1, x_2) = (\pi, \pi) \quad (4)$$

From the standpoint of the overall cloud computing system, the faster a job is completed, the more efficient the system is. This means the system's throughput is greater and resource utilization is higher. It indicates that the cloud computing system uses less energy to perform a workflow application request to some extent. As a result, a more effective task scheduling approach can minimize the overall cloud computing system's completion time.

4. Task Scheduling Problem Formulation

Tasks in the cloud are planned with the goal of improving different quality-of-service metrics. The challenge of task scheduling is to determine how many jobs can be allocated to a given number of virtual machines. The following assumptions are considered while modeling this issue:

1- All jobs provided are separate from each other.
2- If you have many virtual machines, you cannot assign the same job to multiple virtual machines at the same time.
3- The virtual machines are varied in terms of processing power and makespan.
4- The VMM enables resource consumption to be optimized with the least amount of time spent on it.

5. Bat Optimization Algorithm

Inspired by bats' social behavior and the phenomenon of echolocation, which is used to detect distance, this search technique is a new metaheuristic swarm intelligence optimization strategy for global numerical optimization. It was created with the intention of being used in global numerical optimization. To estimate or idealize bat echolocation, the criteria, which are simplified in the [13] approach, can be applied.

All bats utilize echolocation to hone their sense of location, and they seem to intuitively "know" where they are at any given time. To detect prey, bats use a variety of foraging techniques, including flying at random, flying at a fixed frequency at a specific place, and using a changeable wavelength and loudness A0. They could modify their produced pulses' amplitude r [0, 1], as well as their wavelength and frequency r [0, 1], depending on how near they are to their target. It is believed that, although the loudness can fluctuate, it ranges from a small constant (positive) A_{min} to a large A0. As a result of this, each bat is defined in terms of its position (x) and velocity (v), as well as its frequency (f), loudness (A), and emission pulse rate (r). It is possible to compute new position (solutions) (x_i^t) and velocities (v_i^t) using the method for

$$f_i = f_{min} + (f_{max} - f_{min}) \beta, \quad (5)[3]$$

$$v_i^t = v_i^{t-1} + (x_i^t + x_*) f_i \quad (6)[3]$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (7)[3]$$

Where $\beta \in [0, 1]$ is a constant vector and [0, 1] is a random vector generated from a uniform distribution. A global best position (solution) is established based on a comparison of the best positions (solutions) among all the n bats worldwide. Implementation frequently assigns the frequency f values between 0 and 100, depending on the size of the domain in which the topic of concern is handled. Initially, each bat is given a random frequency taken from the range [f_{min}, f_{max}] in an evenly distributed manner. The random walk technique is used to produce a new solution for each bat once the best solution has been picked from the best solutions previously identified.

$$x_{new} = x_{old} + A^t, \quad (8)[3]$$

Here, the scaling factor is $\varepsilon \in [-1, 1]$. The average loudness of all the bats at time step t is $A^t = \langle A_i^t \rangle$ .fi, which controls how fast particles move and updates the velocities and positions of bats, which is very similar to how the standard PSO works [15]. This is because fi controls how far and quickly they move. BA is, to some degree, a mix of the conventional PSO and the more intense local search governed by the loudness and pulse rate of the sound. Another thing to keep in mind is that the loudness A_i and rate ri of pulse emission alter as the iterations progress, as seen in (9).

$$A_i^{t+1} = \alpha A_i^t ,$$

$$r_i^t = r_i^0 [1 - \exp(-\gamma t)] \quad (9)[3]$$

where α and γ are two constants that are always the same. Basically, it is very similar to the "cooling factor" in the simulated annealing (SA) [16]. For this project, we set $\alpha = \gamma = 0.9$.

Algrthim#1: Bat Algorithm (BA)
Input: Set of parameters, T: Task, V: Virtual Machines, v_{max}: maximum values, v_{min}: minimum values, FL: Fuzzy Linear Algorithms, fi: pulse frequency, ri: pulse rates, Ai: loudness Output: gbest Step 1: Set the number of bats required along with their parameters, including frequency, position, velocity, pulse emission rate, and loudness. Step 2: Assign initial values to the parameters mentioned above along with the minimum frequency and maximum frequency. Step 3: Repeat Step 3 to Step 10 till the maximum number of iterations is not reached. Step 4: Compute new solutions by modifying the parameters. Step 5: If any randomly generated pulse rate is higher than the pulse generated by existing bats, execute Step 6; otherwise, execute Step 7. Step 6: Select the best solution among all and try to compute another local solution around the same to avoid the trap of the optimal local solution. Step 7: Compute any other solution using a random search. Step 8: If the loudness of a randomly generated sound is lower than that of any other bat, and the frequency is also lower than the frequency of the best bat, then follow Step 9. Step 9: Accept that solution and keep on increasing the value of pulse emission rate and decreasing the value of loudness. Step 10: Among all the bats, select the best bat and save the result in gbest.

6-Harmony Search Algorithm

The HS algorithm was created by Geem and colleagues in 2001[17] based on natural musical performance processes that occur when a performer attempts to enhance their state of harmony, like what occurs during improvised jazz. Jazz improvisation strives to produce musically pleasing harmony, like the optimization process, which seeks a global solution (a perfect state) based on an objective function. The aesthetic quality of each musical instrument is influenced by its pitch, just as the objective function value is determined by the values assigned to each design variable [17]. The HS algorithm has outperformed other more traditional optimization methods in a variety of applications, including water distribution and games [17,18]. The advantages of HS can be summarized as follows:

i. When utilizing the HS method, it is not necessary to set the beginning values for the decision variables.

ii. Since the HS method uses random stochastic searches, the derivative data is also unnecessary.

iii. After reviewing all the previous vectors, the HS algorithm develops a new vector. These characteristics boost the HS algorithm's flexibility and give better outcomes.

iv. HS excels in its ability to quickly identify the solution space regions that deliver the best results.

Algrthim#2: Harmony Search Algorithm (HSA)

Input: Set of parameters, T: Task, V: Virtual Machines, v_{max}: maximum values, v_{min}: minimum values, hmcr: harmony memory acceptance rate, bw: bandwidth, par: pitch adjustment rate
Output: sbest
Step 1: Load random solutions into Harmony Memory (HM).
Step 2: From HM, create a new harmony.
Step3: If the new harmony is superior to the poorest harmony in HM, include it in HM.
Step 4: Return to Step 2 if the termination requirements are not met.
Otherwise, print the best result(sbest).

7. The Proposed Hybrid BAHS Algorithm

To examine the efficiency of the recommended (BAHS) method in decreasing the makespan, the presented hybrid optimization was used. It consists of implementing both the Bat and the harmony algorithms. In this algorithm (BAHS), when the Bat algorithm's termination condition is reached, the Bat algorithm serves as the first step in distributing work to VMs and acquiring a new population. The output of the Bat algorithm (vmax, vmin) is fed as the input to the harmony search algorithm to get better optimization results. Time is measured after executing the harmony search algorithm. Algorithm (#3) illustrates the introduced hybrid optimization algorithm steps.

Algrthim#3: Hybrid Optimization Algorithm (BAHS)

Input: Set of parameters, T: Task, V: Virtual Machines, v_{max}: maximum values, v_{min}: minimum values, FL: Fuzzy Linear Algorithms, fi: pulse frequency, ri: pulse rates, Ai: loudness, hmcr: harmony memory acceptance rate, bw: bandwidth, par: pitch adjustment rate
Output: sbest
Step 1: Set the number of bats required along with their parameters, including frequency, position, velocity, pulse emission rate, and loudness.
Step 2: Assign initial values to the parameters mentioned above along with the minimum frequency and maximum frequency.
Step 3: Repeat Step 3 to Step 10 till the maximum number of iterations is not reached.
Step 4: Compute new solutions by modifying the parameters.
Step 5: If any randomly generated pulse rate is higher than the pulse generated by existing bats, execute Step 6; otherwise, execute Step 7.
Step 6: Select the best solution among all and try to compute another local solution around the same to avoid the trap of the optimal local solution.
Step 7: Compute any other solution using a random search.
Step 8: If the loudness of a randomly generated sound is lower than that of any other bat, and the frequency is also lower than the frequency of the best bat, then follow Step 9.
Step 9: Accept that solution and keep on increasing the value of pulse emission rate and decreasing the value of loudness.
Step 10: Among all the bats, select the best bat and save the result in gbest.
Step 11: Using gbest as an input parameter, compute the maximum and minimum values of vmax and vmin.
Step 12: Load random solutions into Harmony Memory (HM).
Step 13: From HM, create a new harmony.
Step14: If the new harmony is superior to the poorest harmony in HM, include it in HM.
Step 15: Return to Step 13 if the termination requirements are not met.
Otherwise, print the best result(sbest).

8. Experiential Results

This section introduces the performance evaluation of the proposed workflow scheduling algorithms for Cloud computing. To verify the effectiveness of the offered methods, the BA, HS, and proposed combined BAHS algorithms were compared with other metaheuristics algorithms including GSA, PSO, ABC, and DA. These comparison algorithms have been used to solve tasks scheduling issues and exhibit high performance in several areas. Five separate datasets of 30, 50, 100, 200, and 300 tasks each randomly created were used in the experiments. 20 VMs were employed while scheduling these task data sets .

Before using the suggested methods, a few parameters that (stated in Table 2 needed to be initialized), Since meta-heuristic algorithms are within the category of stochastic optimization techniques, they require at least 10 separate runs to provide significant statistical results. The average outcomes of each algorithm are shown in this research after each strategy has been tested in 20 separate runs. The measurements of the best results for all techniques in the most recent iteration are shown in Tables 3, 4,5,6, and 7. They include mean value (mean) and standard deviation (SD).The bolded values represent the best results. According to the obtained results, the Bat Algorithm Harmony Search (BAHS) algorithm consistently outperforms other algorithms in terms of accuracy, stability, and efficiency.

Table 2: The Initial Parameters of the Attained Approaches.

Number of particles	**20**
VM number	20
W_{max}	0.9
W_{min}	0.4
Min -position	0
Max- position	Number of VM-1
V_{min}	-(Number of VM/5)
V_{max}	Number of VM/5
⟦iter⟧_max	1000
Stopping critre	⟦iter⟧_max

Table 3: Makespan performance comparison for all methods with 30 Tasks, Successful outcomes, are specified in bold

proposed Algorithm	**Mean**	**SD**
HS	155.033	**10.600**
BA	461.764	**39.747**
BAHS	**154.164**	**11.579**
GSA1	1782.899	**94.090**
ABC1	1657.308	**56.511**
DA1	1824.568	**130.417**
Linear-PSO1	1640.456	**76.775**
Sigmod-PSO1	1651.659	**99.025**
Chaotic-PSO1	1619.406	**72.086**
Simulated-PSO1	1601.072	**76.630**
Logarithm-PSO1	1588.758	**98.88**

Table 4: Makespan performance comparison with 50 Tasks for all methods, Successful outcomes are highlighted in bold

proposed Algorithm	Mean	SD
HS	151.134	**9.227**
BA	657.552	**38.161**
BAHS	**153.091**	**11.091**
GSA1	2871.138	**150.251**
ABC1	2679.831	**127.000**
DA1	2926.428	**249.679**
Linear-PSO1	2619.736	**149.784**
Sigmod-PSO1	2615.608	**149.807**
Chaotic-PSO1	2595.078	**136.330**
Simulated-PSO1	2615.686	**129.989**
Logarithm-PSO1	2561.115	**111.579**

Table 5: Makespan performance comparison with 100 Tasks for all methods, Successful outcomes are highlighted in bold.

proposed Algorithm	Mean	SD
HS	153.999	**10.428**
BA	1371.218	**116.843**
BAHS	**152.493**	**10.122**
GSA1	5688.609	**201.596**
ABC1	5560.941	**286.251**
DA1	5805.924	**427.296**
Linear-PSO1	4897.139	**218.028**
Sigmod-PSO1	5108.829	**111.947**
Chaotic-PSO1	4907.81	**175.892**
Simulated-PSO1	5019.134	**273.51**
Logarithm-PSO1	4838.359	**198.567**

Table 6: Makespan performance comparison with 200 Tasks for all methods, Successful .outcomes are highlighted in bold

proposed Algorithm	Mean	SD
HS	**157.401**	**12.201**
BA	2633.489	**194.23**
BAHS	158.779	**10.724**
GSA1	11,702.281	**346.546**
ABC1	12,214.631	**710.296**
DA1	11,010.559	**868.830**
Linear-PSO1	9647.562	**412.172**
Sigmod-PSO1	9823.410	**308.805**
Chaotic-PSO1	9536.721	**311.809**
Simulated-PSO1	9790.057	**314.911**
Logarithm-PSO1	9463.451	**243.235**

Table 7: Makespan performance comparison with 300 Tasks for all methods, Successful .outcomes are highlighted in bold

proposed Algorithm	Mean	SD
HS	158.601	**12.172**
BA	4036.088	**192.581**
BAHS	**148.400**	**9.102**
GSA1	18,307.598	**584.043**
ABC1	19,421.584	**1072.2**
DA1	16,892.254	**1488.877**
Linear-PSO1	14,244.216	**862.247**
Sigmod-PSO1	14,738.096	**442.266**
Chaotic-PSO1	14,352.904	**391.344**
Simulated-PSO1	14,762.614	**741.026**
Logarithm-PSO1	14,185.25	**349.48**

9. Discussion

The Bat Algorithm Harmony Search (BAHS) algorithm is predicted to be more likely to stray from the local optimum. Overall, the findings show that the Bat Algorithm Harmony Search (BAHS) algorithm outperforms the comparator algorithms in terms of makespan . When a healthy balance between the global search and local search phases throughout the search process is maintained, global search is favored over local search in the early stages while local search is favored in the latter stages. Additionally, in high dimension scenarios, the Bat Algorithm Harmony Search (BAHS) algorithm outperforms the comparative algorithms in terms of SD findings, indicating that the suggested techniques can more reliably deliver superior scheduling solutions. The Bat Algorithm Harmony Search (BAHS) algorithm performs best overall for Cloud computing work scheduling

10. Conclusion and Future Work

The bat technique and the Harmony search algorithm can be used to create a hybrid scheduling algorithm, as shown in this study. The hybrid approach initially implements the Bat algorithm to a randomly generated dataset of tasks. Afterward, the Harmony search algorithm is applied to the results to provide the best scheduling outcomes. The results obtained from the hybrid algorithm are compared to those obtained by implementing the bat and Harmony alone, using a different number of tasks. The results illustrate the success of the hybrid approach using 300 tasks and give mean of makespan of 148.400 , standard deviation is equal to 9.102 compared to other algorithms. When it comes to task scheduling in Cloud computing, the hybrid method provides an improved performance, as well as a reduction in mean of makespan as well as the standard division.

For future work, the proposed scheduler (BAHS) can be used to schedule processing applications in the cloud with numerous optimization targets, such as load balancing and cost. Some tests can also be conducted and compared to other algorithms. Moreover, the provided methods can be adapted to other systems, such as fog computing.

References

[1] Jaber, S., Ali, Y., & Ibrahim, N. " An Automated Task Scheduling Model Using a Multi-objective Improved Cuckoo Optimization Algorithm". *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 1, pp. 295-304, 2022.

[2] M. F. Younis, "ESJF Algorithm to Improve Cloud Environment", I*raqi J. Sci*., pp. 4171– 4180, 2021.

[3] Wang, Gaige, and Lihong Guo. "A novel hybrid bat algorithm with harmony search for global numerical optimization." *Journal of Applied Mathematics*, 2013.

[4] T. Sharma and V. K. Banga, "Efficient and enhanced algorithm in cloud computing," *Int. J. Soft Comput. Eng*. ISSN, vol. 2231, p. 2307, 2013.

[5] X. Meng, Y. Liu, X. Gao, and H. Zhang, "A new bio-inspired algorithm: chicken swarm optimization," in *international conference in swarm intelligence*, pp. 86–94, 2014.

[6] A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, "A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing," *Futur. Gener. Comput. Syst*., vol. 83, pp. 14–26, 2018.

[7] N. J. Navimipour, "Task scheduling in the cloud environments based on an artificial bee colony algorithm," in *International Conference on Image Processing*, pp. 38–44, 2015.

[8] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," in *2013 8th international conference on computer engineering & systems* (ICCES), pp. 64–69, 2013.

[9] S. A. Hamad and F. A. Omara, "Genetic-based task scheduling algorithm in cloud computing environment," *Int. J. Adv. Comput. Sci. Appl*., vol. 7, no. 4, pp. 550–556, 2016.

[10] S. Raghavan, P. Sarwesh, C. Marimuthu, and K. Chandrasekaran, "Bat algorithm for scheduling workflow applications in cloud," in 2015 *International Conference on Electronic Design, Computer Networks & Automated Verification* (EDCAV), pp. 139–144, 2015.

[11] A. R. Jordehi, "Chaotic bat swarm optimisation (CBSO)," *Appl. Soft Comput*., vol. 26, pp. 523–530, 2015.

[12] P. Kumar, B. J. Sowmya, A. Kanavalli, N. J. Roy, S. L. Karthik, and P. Goyal, "Improved Whale Optimization Algorithm for Clustering," *NVEO-NATURAL VOLATILES Essent. OILS Journal| NVEO*, pp. 13135–13144, 2021.

[13] X.-S. Yang and L. Press, "*Nature-inspired metaheuristic algorithms second edition." Luniver press UK*, 2010.

[14] V. Polepally and K. Shahu Chatrapati, "Dragonfly optimization and constraint measure-based load balancing in cloud computing," *Cluster Comput*., vol. 22, no. 1, pp. 1099–1111, 2019.

[15] E. Baccarelli, P. G. V. Naranjo, M. Shojafar, and M. Scarpiniti, "Q*: Energy and delay-efficient dynamic queue management in TCP/IP virtualized data centers," *Comput. Commun*., vol. 102, pp. 89–106, 2017.

[16] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science* (80-.)., vol. 220, no. 4598, pp. 671–680, 1983.

[17] X.-S. Yang, "Harmony search as a metaheuristic algorithm," in *Music-inspired harmony search algorithm*, *Springer*, pp. 1–14, 2009.

[18] Z. W. Geem, "Harmony search algorithm for solving sudoku," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pp. 371–378, 2007.