چهارمین کنفرانس بین‌المللی
دانشجویان و مهندسان
برق و انرژی‌های پاک
fourth International Conference for
Students and Engineers of
Electrical and Clean Energy

AO-HHO-LevyFligth Algorithm for Task Scheduling in Cloud

Najmeh Dehghani [1], Farshad Abdi[2], Mohammad Sadeghzadeh[3]

1 Master of Science in Computer Engineering- Software, Islamic Azad University, Tehran Science and Research Branch (Fars), Iran, Dehghani.najmeh@gmail.com.
2 Faculty of Mechanics, Electrical Power and Computer ،Islamic Azad University, Science and Research Branch, Tehran, Iran, MSc student in System Telecommunications, farshad73abdi@gmail.com.
3 Computer Engineering Department , Faculty of Engineering , Bu Ali Sina University, *m.sadeghzadeh@eng.basu.ac.ir*

ABSTRACT

Today, cloud computing has the power to respond to the different users requests. In cloud computing, the problem of user requests scheduling or Task Scheduling (TS) is a very efficient technique that, if done optimally, increases system performance. Therefore, in this article, a Meta-Heuristic (MH) TS algorithm is proposed using the hybrid of Aquila Optimizer (AO), Harris-Hawks-Optimization (HHO) and LevyFligth named AO-HHO-LevyFligth. In AO-HHO-LevyFligth, the combination of HHO exploration phase with AO exploitation phase as well as LevyFligth is used to obtain the best timing. Comparison between AO-HHO-LevyFligth and HHO methods, Particle Swarm Optimization (PSO), AO and Firefly Algorithm (FA) according to criteria (throughput and makespan), shows the high efficiency of AO-HHO-LevyFligth in solving the problem is the TS.

Keywords: Harris-Hawks, Aquila Optimizer, Task scheduling and cloud computing

1. *INTRODUCTION*

Today, cloud computing is very efficient for processing and storing the tasks and requests of various user applications, as well as responding to these applications that these applications can be from the users' smart devices [1-4]. In other words, cloud resources make it easier for users to access these resources [4]. Task Scheduling (TS) is used to manage resources and tasks in the cloud and fog environment, which assigns the set of user tasks to the most suitable resources in the cloud to be executed [5]. It has been proven that the TS problem in all computing environments (fog and cloud) is a type of NP-hard [6].

Todays, Optimization algorithms such as Meta-Heuristic (MH) are very widely used to solve various problems[7-13]. In TS and Internet of Things (IoT) fields:

Boviri et al. [14] have used the combination of Q learning algorithm and neural network to schedule IoT tasks on cloud resources. Qobaei-Arani et al. [15] used MFO to schedule tasks in fog and reduce task performance time and increase Quality of Service (QoSs). Abualigah et al. [16] have used the Gray Wolf Optimizer (GWO) to task allocation in the cloud with the aim of reducing time. The authors [17] used Genetic Algorithm (GA) for TS in order to optimally allocate resources to tasks with the aim of increasing customer satisfaction and reducing preparation time.

The mentioned articles in the literature that have used MH algorithms, some are weak in the exploration phase and some are weak in the exploitation phase, which makes the algorithm unable to find the most optimal solutions. Therefore, in this article, we combined two algorithms, AO and HHO, which in them, the LévyFlight method is used in the exploration and exploitation phases, the AO exploration phase is combined with the HHO exploitation phase, and the obtained algorithm is called AO-HHO-LevyFligth to solve the TS problem in the cloud with the aim reduce the time to perform the tasks we use. That is, the Objective function in the AO-HHO-LevyFligth algorithm is equal to the minimization of the task makespan time. In the comparison part, the AO-HHO-LevyFligth algorithm has been compared with the HHO, Firefly Algorithm (FA), AO and Particle Swarm Optimization (PSO) algorithms. It has been proven that the proposed method performs better than the compared algorithms in terms of task makespan time.

Continue the article:

Part 2: system model and problem formulation, part 3: prerequisites, part 4: proposed method, part 5: evaluations and experimental results and part 6: conclusion.

2. System model and problem formulation

According to the architecture in this article, user requests are sent as tasks on cloud resources and are scheduled and executed to be processed and stored [1]. Cloud servers consist of resources that have high storage and computing capabilities for storing and processing the tasks of the respective users. Therefore, Task Scheduler schedules tasks on resources. In this article, we use the word node, which represents the server.

Assumptions:

1- The task set with n tasks is T = {Task1, Task2, Task3, . . . , Taskn} .
2-Feature of each task includes task length, deadline and memory requirements.
3-The cloud consists of m nodes as $Node_{cloud}$ = {node_cloud1, node_cloud2, node_cloud3, . . . , node_cloudm}.
4-For m nodes and n tasks, the Expected Computation Time (ECT) is represented with ECT matrix of size n×m. Scheduler uses this matrix to make TS in the nodes. $ect_{i,j}$ as:

$$ect_{i,j}=\frac{Task\_length_i}{node.Pow_j} \quad (1)$$

Where, $ect_{i,j}$ represented the expected execution time for one task(i-th task) on one node(j-th node), $Task\_length_i$ represented the task length (i-th task), and $node.Pow_j$ represented the task processing speed (j-th node). In this article the main objective is reduces makespan time and [12, 13] the makespan time as:

$$M\_time= \max_{j \in 1,2,\dots,n2} \sum_{i=1}^{n1} ect_{i,j} \quad (2)$$

3. Prerequisites

In this section, AO, LevyFligth and HHO algorithms are explained.

3.1. Aquila Optimizer
In AO [18] the Aquila provides 4 different hunting methods for each prey as:

Step 1: Explore extensively in the search space as:

$$AA(ti+1)=AA_{best}(ti)\times(1-\frac{ti}{Ti})+(AA_m(ti)- AA_{best}(ti)\times rand) \quad (3)$$

$$AA_m(ti)=\frac{1}{N}\sum_{k=1}^{N} AA_i(ti) \quad (4)$$

$AA_m(ti)$ is the mean position in the current iteration for search agents, $AA_{best}(ti)$ is the best position in so far, N is the size of population, rand ∈[0, 1], and ti is the current iteration and Ti is the maximum iteration number.

Step 2: Narrow exploration as:

$$AA(ti+1)=AA_{best}(ti)\times LF1(Di)+ AA_R(ti)+(y1-x1)\times rand \quad (5)$$

Where, Di is the size of dimension, $A_R(ti)$ is the random position, and Levy Flight function shows with LF1 (Di):

$$LF1(Di)=s\times\frac{u1\times\sigma}{|v1|^{\frac{1}{\beta}}1} \quad (6)$$

$$\sigma=\left[\frac{r1(1+\beta1)\times\sin(\frac{\pi\beta1}{2})}{r1(\frac{1+\beta1}{2})\times\beta1\times2^{(\frac{\beta1-1}{2})}}\right] \quad (7)$$

Where s1, u1 and v1 ∈ [0, 1], β = 0.01, y1 and x1 as:

$$\begin{cases} x1 = r1 \times \sin(\theta 1) \\ y1 = r1 \times \cos(\theta 1) \\ r1 = r_1 + 0.00565 \times Di_1 \\ \theta 1 = -w1 \times Di_1 + \frac{3\times\pi}{2} \end{cases} \quad (8)$$

Where, Di_1 is the integers and $Di_1 \in [1, Di]$, and $r_1 \in [1, 20]$.

Stage 3: Extensive exploitation as:

$$AA(ti+1)=(AA_{best}(ti)- AA_m(ti))\times\alpha\text{-rand}+((UB1-LB1)\times rand+LB1)\times\delta \quad (9)$$

Where UB1 is the upper bound and LB1 is lower bound of the search space, that equal to the lowest and highest virtual machine index, respectively. α and δ are equal to 0.1.

Stage 4: Limited exploitation as:

$$\begin{cases} AA(ti+1) = QF1 \times AA_{best}(ti) - (Gi_1 \times AA(ti)) \times rand - Gi_2 \times LF1(Di) + rand \times Gi_1 \\ QF1(ti) = ti^{\left(\frac{2\times rand-1}{(1-T)^2}\right)} \\ Gi_1 = 2 \times rand - 1 \\ Gi_2 = 2 \times \left(1 - \frac{ti}{Ti}\right) \end{cases} \quad (10)$$

Where AA(ti) is the current position and QF1(ti) is the quality function for balancing in search. $Gi_1 \in [-1,1]$. Gi_2 decreases linearly from 2 to 0.

3.1. **Harris Hawks Optimizer**

This article is done using AO and HHO [19] and Objective function minimization. Each Hawk is shown as a vector $\overrightarrow{H_1}$. If the HHO algorithm alone applies to the virtual machines vector in the TS, the initial location of the search agents is determined randomly. Then, the value of the Objective function is calculated for all agents, and the search agent that the obtained Objective function for it is less than other Hawks, is taken as a metric, and its location is determined as the virtual machines vector. In HHO, the search agents are waiting to identify the bait based on the 2 methodologys.

$$\overrightarrow{H(t\imath+1)} = \begin{cases} \overrightarrow{H_{rand}(t\imath)} - r_1 \left|\overrightarrow{H_{rand}(t\imath)} - 2r_2\overrightarrow{H(t\imath)}\right|, & q \geq 0.5 \\ (H_{rabbit}(ti) - H_m(ti)) - r_3\left(LB1 + r_4(UB1 - LB1)\right), & q < 0.5 \end{cases} \quad (11)$$

Where, $\overrightarrow{H(t\imath+1)}$ is the vector of the next location of the search agents in the iteration t^{th}, $H_{rabbit}(ti)$ is the location of the bait in the iteration t^{th}, $\overrightarrow{H(t\imath)}$ is the current location vector of the search agents in the iteration t^{th}, $\{r_1, r_2, r_3, r_4$ and $q\} \in$ rand [0, 1] updated in each iteration. $\overrightarrow{H_{rand}(t\imath)}$ is the location vector of search agents that is randomly selected.

Therefore, to perform a soft blockade, it is assumed that the search agents can evaluate their next move according to Eq. (12).

$$H1 = H_{rabbit}(ti) - E|JH_{rabbit}(ti) - \overrightarrow{H(t\imath)}| \quad (12)$$

The escape energy of baits:

$$E = 2E_0(1 - ti/T) \quad (13)$$

Where, E_0 indicate the primary escape energy, and T is the maximum repetitions. For each iteration $E_0 \in (-1, 1)$.

That is:

$$H2 = H1 + S \times LF(Di) \quad (14)$$

Size of S is 1×Di, that is a random vector, and LF is the received Levy Flight function.

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \quad (15)$$

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}}\right)^{\frac{1}{\beta}} \quad (16)$$

Where {u, v}∈rand[0, 1]. B=1.5. So:

$$\overrightarrow{H(t_1+1)} = \begin{cases} H1, & \text{if} F(H1) < F(H(ti)) \\ H2, & \text{if} F(H2) < F(H(ti)) \end{cases} \quad (17)$$

Then, the algorithm's output is the optimal virtual machine vector.

3.2. **LévyFlight**

LévyFlight is a statistical description of motion. There are many natural and artificial phenomena that are based on LévyFlight statistical description [20]. The flight behavior of many animals has shown typical features of randomness to search for food from one place to another. The selection of the direction to change location relies on the LévyFlight statistical model [21]. Even light can be related to LévyFlight [20, 21]. In other words, LévyFlight is used for optimization and optimal search and improves the results. In this article, we further improve the hybrid AO-HHO-LévyFlight in the exploration and exploitation by LevyFlight.

4. Proposed scheduling with AO-HHO-LévyFlight

In this section the TS in cloud using the combination of HHO and AO is described. In the TS using AO-HHO-LévyFlight, the solutions updated in the exploitation with AO or HHO operators. But in the exploration, only AO operators are used.

The start of the optimization in AO_HHO-Lévyflight is to generate N agents randomly in the cloud to perform the TS problem and convert them into integers. Then, the Objective function is calculated for all members of the population in order to evaluate the quality of each member using Eq. (2). Then, according to the minimization criterion of the Objective function, the member of the population with the lowest value of the Objective function is selected as the best solution. Then the position of other members of the population are updated according to the solution with the lowest value of the Objective function.

The updating of the population members in the AO_HHO-Lévyflight exploration phase is done using Eq. (5), which is the same as the AO exploration phase. But in the exploitation phase of Proposed-method, updating of each member of the population is done according to the competition between the exploitation phases of AO and HHO according to the Ph_i for population members as:

$$Ph_i = \frac{\text{Objective\_function}_i}{\sum_{i=1}^{N} \text{Objective\_function}_i} \quad (18)$$

Therefore, the each member of the population Ph_i is updated by Eq.(18). Updating the population members continues until the stop is reached. The member of the population position_i is updated in the exploitation phase as.

$$\text{position}_i = \begin{cases} \text{operators of AO} & Ph_i > \text{thresh} \\ \text{operators of HHO} & \text{otherwise} \end{cases} \quad (19)$$

5. Evaluation metrics and experimental results

In this article, all the simulations have been done by MATLAB R2017a. 5 hosts and 23 resources (Virtual Machine (VMs)) are set for the cloud system. Table 1 shows the features of hosts and VMs.

Table 1. Features of hosts and VMs

	Specification	*Amount*
Client	*Clients Count*	*[61, 110]*
Host	*Hosts Number*	*5*
	Capacity of CPU	*[100, 4000]*

	Capacity of Storage	*1 TB*
	Bandwidth of Network	*10 Gb/s*
	Size of RAM	*6 GB*
VM	*VMs Number*	*23*
	Capacity of CPU	*[100, 4000]*
	Capacity of Storage	*20 GB*
	Size of RAM	*1 GB*
	Bandwidth of Network	*1 Gb/s*

Two datasets NASA Ames iPSC/860 and HPC2N [22] evaluated. The AO_HHO-LévyFlight scheduling algorithm is compared with AO, FA [23], PSO [24], and HHO. 30 runs are considered for each algorithm and the average of these 30 runs is considered as the output.

5.1. **Evaluation metrics**

The evaluation metric that used in this article, includes throughput time, Objective function and makespan.

a.Objective function

In minimization problems, the optimal solutions correspond to the lowest value of the objective function, which in this article is the makespan time (Eq.(2)).

b. Makespan time

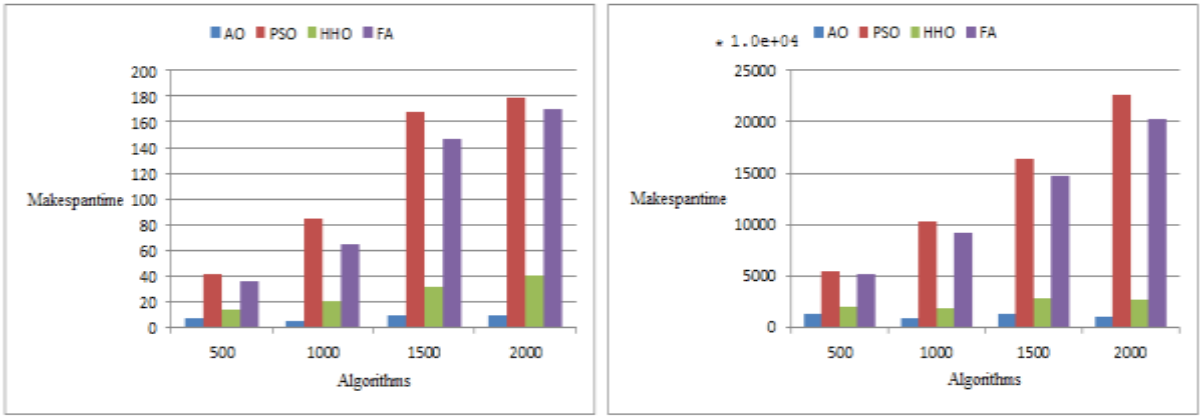
5.2. **Comparison phase**

In this section, the AO_HHO-LévyFlight is evaluated using objective function, makespan time (second or sec), then compared with PSO, AO, FA and HHO. Table 2 shows the results of makespan time for datasets.

Table 2. Results of Makespan time

	NASA Ipsc				HPC2N			
Dataset	Tasks				Tasks			
Algorithm	500	1000	1500	2000	500	1000	1500	2000
Proposed-method	41.52	83.65	128.5149	158.53	2558.59	9457.63	16050.41	24458.32
AO	47.61	89.38	138.29	168.41	3835.49	10375.79	17266.33	25460.43
PSO	81.62	169.32	297.56	338.25	7945.46	19789.69	32459.72	47141.17
HHO	53.75	105.32	160.52	199.79	4574.52	11248.79	18831.39	27080.25
FA	75.85	149.81	276.48	329.12	7763.73	18677.25	30728.65	44751.51

According to Table 2, Proposed-method is better than other for makespan time, which is the same value of the objective function, for both datasets, and it is lower for Proposed-method than other methods. According to Table 2, AO and Proposed-method are stronger than HHO. But FA and PSO have a different structure, which makes them slow in achieving the good solution.



(a) (b)

Fig. 1. difference value for makespan time.

Fig.1 shows the makespan difference for Proposed-method compared to the compared methods. According to Fig.1, for the number of different tasks, the makespan difference value is the highest for PSO, and then it is higher than that of FA. The makespan difference value for HHO is less than all algorithms. The amount of makespan difference is as:

$$Diff\_makespan = makespan_{AO\_HHO} - makespan_{i} \quad (20)$$

In which, $makespan_{AO\_HHO}$ is the makespan of Proposed-method and $makespan_{i}$ is the makespan of other methods.

6. Conclusions

In this article, the optimization of the TS problem in the cloud environment according to the QoS requirements was investigated using the combination of HHO, AO and LévyFlight, which is called Proposed-method. Proposed-method was used to improve the HHO discovery phase, which is implemented in the exploitation phase of one of the AO or HHO algorithms based on a probability and a threshold value. Also, we further improve the hybrid AO-HHO-LévyFlight in the exploration and exploitation by LevyFlight. In Proposed-method, the minimization of the objective function, which is makespan, was used for optimization in optimizer algorithms.

References

[1]A. Zanella, et al., “Internet of things for smart cities,” IEEE Internet of Things Journal, vol. 1, pp. 22-32, 2014.
[2] L. Atzori, et al., “The internet of things: A survey,” Computer networks, vol. 54, pp. 2787-2805, 2010.
[3] M. Masdari and A. Khoshnevis, “A survey and classification of the workload forecasting methods in cloud computing,” Cluster Computing, pp. 1-26, 2019.
[4] A. Shakarami, et al., “Data replication schemes in cloud computing: a survey,” Cluster Computing, pp. 1-35, 2021.
[5] Z.-G. Chen, et al., “Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach,” IEEE Trans. Cybern,vol. 49 (8), pp. 2912–2926, 2018.
[6] Y. Wang, et al., “Power Scheduling Optimization Method of Wind-Hydrogen Integrated Energy System Based on the Improved AUKF Algorithm,” Mathematics, vol. 10(22), 4207, 2022.
[7] T. Salehnia and A. Fathi, “Fault tolerance in LWT-SVD based image watermarking systems using three module redundancy technique,” Expert Systems with Applications, vol. 179, 115058, 2021.
[8] S. Raziani, et al., “Selecting of the best features for the knn classification method by Harris Hawk algorithm,” Conference: 8th International Conference on New Solutions in Engineering, Information Science and Technology of the Century aheadAt, 2021.
[9] T. Salehnia, T. et al. (2021). Multilevel image thresholding using GOA, WOA and MFO for image segmentation. in Proceedings of the 8th International Conference on New Strategies in Engineering. Information Science and Technology in the Next Century.
[10] Y. Cheng et al., Vehicular Fog Resource Allocation Approach for VANETs Based on Deep Adaptive Reinforcement Learning Combined With Heuristic Information, IEEE Access (Volume: 12), pp: 139056 - 139075, 2024, doi: 10.1109/ACCESS.2024.3455168.
[11] T. Salehnia, (2023). A multi-level thresholding image segmentation method using hybrid Arithmetic Optimization and Harris Hawks Optimizer algorithms. Expert Systems with Applications.
[12] T. Salehnia, et al. (2023). An optimal task scheduling method in IoT-Fog-Cloud network using multi-objective moth-flame algorithm. Multimedia Tools and Applications, doi: https://doi.org/10.1007/s11042-023-16971-w.
[13] T. Salehnia, et al. (2023). SDN-based optimal task scheduling method in Fog-IoT network using combination of AO and WOA. Handbook of Whale Optimization Algorithm, doi: https://doi.org/10.1016/B978-0-32-395365-8.00014-2.
[14] H.R. Boveiri, et al., “An efficient swarm-intelligence approach for task scheduling in cloud-based internet of things applications,” J. Ambient Intell. Humanized Comput. vol. 10 (9), pp. 3469–3479, 2019.

[15] M. Ghobaei-Arani, et al., “An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing,” Trans. Emerg. Telecommun. Technol, vol. 31 (2), e3770, 2020.
[16] L. Abualigah, et al., “TS-GWO: IoT tasks scheduling in cloud computing using Grey Wolf optimizer,” in: Swarm Intelligence for Cloud Computing, Chapman and Hall/CRC, pp. 127–152, 2020.
[17] D. Zeng, et al., “Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system,” IEEE Trans. Comput, vol. 65 (12) , pp. 3702–3712, 2016.
[18] L. Abualigah et al., "Aquila Optimizer: A novel meta-heuristic optimization algorithm," Computers & Industrial Engineering, vol. 157, p. 107250, Jul. 2021. doi: 10.1016/j.cie.2021.107250.
[19] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," Future generation computer systems, vol. 97, pp. 849-872, 2019, doi: https://doi.org/10.1016/j.future.2019.02.028.
[20] Kamaruzaman., A. F., et al., “Lévy flight algorithm for optimization problems—a literature review,” Applied Mechanics and Materials, vol. 421, pp. 496–501, 2013.
[21] Brown, C.T., et al., “Lévy flights in dobe Ju/'hoansi foraging patterns,” Human Ecology, vol. 35, no. 1, pp. 129–138, 2007.
[22] Parallel workloads archive, http://www.cse.huji.ac.il/labs/parallel/ workload/logs.html. (Accessed July 2020), 2020.
[23] X. Sh. Yang, “Multiobjective firefly algorithm for continuous optimization”, Engineering Computation, vol. 29, pp. 175-184, 2013.
[24] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of ICNN'95-international conference on neural networks, vol. 4: IEEE, pp. 1942-1948, 1995.