MODELLING OF HORSE HERD OPTIMIZATION BASED MULTI OBJECTIVE TASK SCHEDULING APPROACH IN CLOUD COMPUTING ENVIRONMENT

by

Faten K. KARIM[a], Sara GHORASHI[a*], Salem ALKHALAF[b], Anis BEN ISHAK[c], and Sameer ALSHETEWI[d]

[a] Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia
[b] Department of Information Technology, College of Computer, Qassim University, Buraydah, Saudi Arabia
[c] Department of Quantitative Methods, Higher Institute of Management, University of Tunis, Tunis, Tunisia
[d] Department of General Information Technology, Excellence Services Directorate, The Executive Affairs, Ministry of Defense, Riyadh, Saudi Arabia

Original scientific paper
https://doi.org/10.2298/TSCI2502583K

Cloud computing, which offers scalable and flexible resources, faces a key challenge in task scheduling, directly impacting system performance and user satisfaction. Effective scheduling is crucial for optimizing resource use and reducing makespan. The NP-completeness of the task scheduling problem complicates achieving optimal outcomes. Scheduling applications is critical in cloud computing due to the need to map future tasks to resources in real time. Many existing methods focus on makespan and resource consumption but overlook factors like energy usage and migration time, which affect web services. This study proposes a horse herd optimization-based multi-objective task scheduling approach (HHO-MOTSA) to address these gaps. The HHO-MOTSA aims to minimize makespan, energy usage, and cost by modelling the social behaviors of horses, including grazing, hierarchy, sociability, and defense mechanisms. A fitness function helps evaluate solutions, where a low value indicates minimized energy, makespan, and cost. Performance tests using CloudSim show that HHO-MOTSA outperforms other methods in effective task scheduling.

Key words: *task scheduling, cloud computing, horse herd optimization, makespan, energy utilization, fitness function*

Introduction

Cloud computing (CC) is a shared technology delivering on-demand services via the network, offering users flexible billing and virtually unlimited resources [1]. Task scheduling (TS) involves mapping tasks to available resources in a cloud environment, adhering to users' QoS constraints like cost and makespan [2]. Workflow applications, often related to fields such as biology and astronomy, have moved to the cloud, where providers use schedulers for resource management [3]. Given the diversity of CC users, mapping heterogeneous requests to virtual resources is complex [4]. Effective scheduling is crucial to improving cloud functional-

* Corresponding author, e-mail: saabdelghani@pnu.edu.sa

ity, minimizing wait time, and optimizing task execution [5]. Traditional scheduling methods do not directly apply to the distributed nature of CC, necessitating customized algorithms that improve performance, optimize resources, and reduce latency [6]. These advances are essential for meeting the growing demand for cloud services and improving distributed computing [7]. Poor scheduling can degrade QoS, increase energy consumption and makespan, and lead to SLA violations [8]. Metaheuristic approaches like PSO, GA, and ACO have been applied to TS, but there's room for improvement in scheduling strategies, as the problem remains NP-hard [9]. Efficient schedulers must prioritize tasks based on factors like electricity costs and make informed VM assignments [10]. This study proposes a HHO-MOTSA for CC, aiming to minimize makespan, energy use, and cost. The HHO-MOTSA uses the social behavior of horses and a fitness function (FF) evaluate solutions, with performance analysis conducted using CloudSim.

Literature works

Mokni *et al.* [11] proposed an algorithm combining sequencing, scheduling, and partitioning to optimize user-provider conflicts, using MAS for complexity reduction and fuzzy logic for ambiguity. Abualigah and Diabat [12] introduced a hybrid antlion optimizer with differential evolution minimize makespan and optimize resources. Kruekaew and Kimpan [13], a multiobjective algorithm using ABC and Q-learning improves resource use and VM throughput. Mohammadzadeh and Masdari [14] developed a hybrid optimizer combining seagull and grasshopper algorithms for TS. Mangalampalli *et al.* [15] applied cat swarm optimization for task and VM prioritization. Emara *et al*. [16], a modified GA optimizes resource management via a matrix structure. Yu *et al.* [17] proposed a bat algorithm to improve load balancing. Finally, Malti *et al*. [18] presented a pollination-based TS method using TOPSIS and Pareto techniques for solution quality improvement.

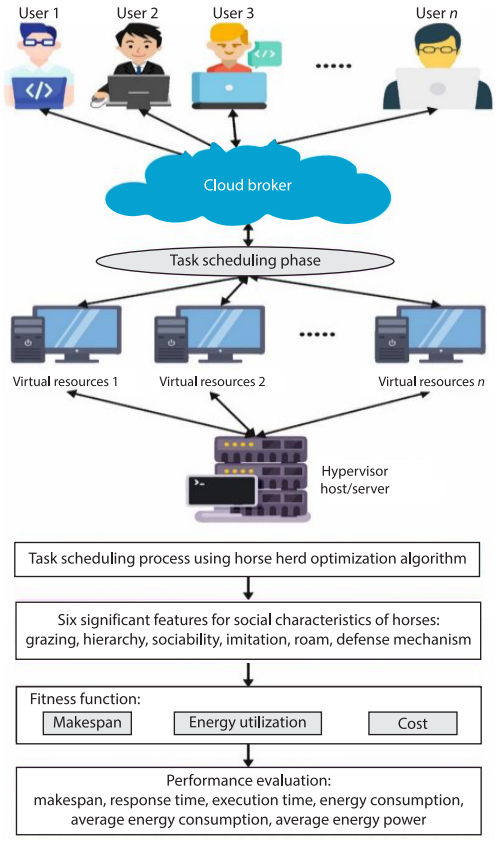

Figure 1. Workflow of HHO-MOTSA algorithm

The proposed model

This manuscript details the HHO-MOTSA approach for multi-objective task scheduling in cloud computing, aiming to reduce makespan, energy use, and cost. Figure 1 shows the HHO-MOTSA algorithm's workflow.

System model

The cloud provides physical machines (PM) through an interface. Task requests are managed by the request manager, while the resource observer tracks PM memory, CPU, and storage. The TS module schedules tasks to minimize the FF by collecting data from both managers to allocate resources to PM.

Application model and resource model

A group of independent tasks represented as $T = (t_1, t_2, t_3, ..., t_m)$. The arrival time of this task is neglected as they arrive dynamically [19]. A collection of parameters $t_i = t_a; t_s; t_d; t_f$ represents the task t_i given by the user. The task length or size, arrival time, finishing time, and deadline of task t_i are denoted as t_s, t_a, t_d, and t_f, correspondingly. Consider t_s as the starting time of task t_i on PM P_k. Likewise, T_r the ready time of PM P_k at the data centre. We let T_{exe} be the execution time of task t_i on PM P_j because of its heterogeneity with respect to CPU processing abilities of PM, then the task size ratio, T_s, to CPU capability of PM, P_c, is T_{exe}:

$$T_{\text{exe}} = \frac{T_s}{P_c} \tag{1}$$

where t_f is the implies the finishing task time t_i on PM P_j and it is readily calculated:

$$T_f = T_s + T_{\text{exe}} \tag{2}$$

where x_{ij} is the task-to-PM mapping from the cloud infrastructure. The x_{ij} take the value of "1" if task t_i is allocated to PM P_j, or else "0". Then, the finishing time is used to confirm if the timing constraint of the task is met, viz., if the task is completed within the given time:

$$x_{ij} = \begin{cases} 0; \text{ if } t_f > t_d \\ 1 \text{ or } 0; \text{if } t_f \le t_d \end{cases} \tag{3}$$

The Cloud data center (CDC) has a limitless set of PM $PM = PM_1; PM_2;..., PM_m$, which provides the physical infrastructure to create virtualized resources for satisfying the requirements of the user. An active PM set $PM^a = \{PM^a_1; PM^a_2;..., PM^a_m\} \subseteq PM_i$ is con sidered as $PM_i = (B_i, C_i, M_i)$ where M_i shows the memory capacity, which is computed, B_i denoted discrete pair of frequency and voltage of PM_i, and C_i indicates the CPU ability.

Problem definition

Consider that there is n number of tasks as $T_n = (T_1, T_2, T_3,...,T_n)$ and m refers to the number of PM $PM = (PM_1, PM_2, PM_3,..., PM_m)$ in the present cloud infrastructure. A PM consists of memory, CPU, and bandwidth for data transmission. Every task should be executed in a separate PM instance. The main purpose is to find the solution that allocates each task to the PM so that the energy utilization, ϵ, makespan, χ, and the cost, ζ, of each application $F(x)$ is minimalized, and the bandwidth, CPU resource, and memory requirement is met and it is expressed as follows. In eq. (4), the group of possible solutions is S, x is a solution, the makespan function is $\chi(x)$, the image of χ in the multiobjective space is $F(x)$, $\epsilon(x)$ energy function, $\zeta(x)$ cost function, the bandwidth use function is $g_i(x)$, the memory input is $\Gamma(x, i)$, x, the input bandwidth limit is $Y(x, i)$, the input CPU consumption is $\Psi(x, i)$, and the CPU usage function is $h(x)$:

$$\begin{aligned} \underset{x}{\text{Minimize}}\, F(x) &= \left[\chi(x), \zeta(x), \epsilon(x)\right] \\ h_i(x) &\le \Psi(x,i), \ i \in (1,m) \\ g_i(x) &\le Y(x,i), \ i \in (1,m) \\ q_i(x) &\le \Gamma(x,i), \ i \in (1,m), \ x \in S \end{aligned} \tag{4}$$

Design of horse herd optimization algorithm

The HHO is inspired by the behaviour of horses at varying ages, categorizing horse activity into hierarchy, grazing, imitation, socializing, roaming, and defensive mechanisms

[20]. According to eq. (1), HHO simulates the movement of a horse in all the iterations. The $X_m^{Iter,AGE}$ is the location of m^{th} horse, *Iter* specifies the existing iteration, and *AGE* and $\vec{V}_m^{Iter,AGE}$ are the ranges of age and velocity of the present horse. The average lifespan of a horse is 25-30 years, and they show different behavioral tendencies as they age. The lives of Horse are divided into four groups:

– δ: horses at 0-5 ages,
– γ: horses at 5-10 ages.
– β: horses at 10-15 ages, and
– α: horses ages 15 and older

$$X_m^{Iter,AGE} = \vec{V}_m^{Iter,AGE} + X_m^{(Iter-1),AGE}, \quad AGE = \alpha, \beta, \gamma, \delta \tag{5}$$

The HHO ranks horses by their best replies, selecting the top 10% and dividing the rest into 20%, 30%, and 40%. These divisions help compute the velocity vector, with the formula defining this vector for horses of different age groups throughout the cycles:

$$\begin{gathered} \vec{V}_m^{Iter,\alpha} = \vec{G}_m^{Iter,\alpha} + \vec{D}_m^{Iter,\alpha} \\ \vec{V}_m^{Iter,\beta} = \vec{G}_m^{Iter,\beta} + \vec{H}_m^{Iter,\beta} + \vec{S}_m^{Iter,\beta} + \vec{D}_m^{Iter,\beta} \\ \vec{V}_m^{Iter,\gamma} = \vec{G}_m^{Iter,\gamma} + \vec{H}_m^{Iter,\gamma} + \vec{S}_m^{Iter,\gamma} + \vec{I}_m^{Iter,\gamma} + \vec{D}_m^{Iter,\gamma} + \vec{R}_m^{Iter,\gamma} \\ \vec{V}_m^{Iter,\delta} = \vec{G}_m^{Iter,\delta} + \vec{I}_m^{Iter,\delta} + \vec{R}_m^{Iter,\delta} \end{gathered} \tag{6}$$

Horses, as grazing animals, graze throughout their lives. Each horse grazes within a defined area, modeled by the algorithm with a *g* coefficient, and their grazing behaviors are:

$$\vec{G}_m^{Iter,AGE} = g_{Iter}\left(\vec{u} + \rho\vec{l}\right) + \left[X_m^{(Iter-1)}\right], \quad AGE = \alpha, \beta, \gamma, \delta \tag{7}$$

$$g_m^{Iter,AGE} = g_m^{Iter-1} \times \omega_g \tag{8}$$

where the parameter $\vec{G}$ refers to the motion of i^{th} horse, representing the tendency of i^{th} horse to graze. In all the cycles, these components linearly decrease with *g*. The *u* and *l* imply the up and low grazing space boundaries correspondingly. The *g* represents the grazing area, while *h* models hierarchical behavior in horses, where they follow the leader, often the most experienced. Horses adhere to hierarchical rules particularly during middle age, and this can be mathematically expressed:

$$\vec{H}_m^{Iter,AGE} = h_m^{Iter,AGE}\left[X_*^{(Iter-1)} - X_m^{(Iter-1)}\right], \quad AGE = \alpha, \beta, \gamma \tag{9}$$

$$h_m^{Iter,AGE} = h_m^{(Iter-1),AGE} \times \omega_h \tag{10}$$

where $\vec{H}_m^{Iter,AGE}$, and $X_*^{(Iter-1)}$ are the impact of the fittest position of horses on the velocity and the optimum place of horses, correspondingly. Herd behavior offers protection from predators, enhancing survival and safety. Despite their social nature, horses may occasionally conflict due to individual traits. Horses aged 5-15 typically prefer to stay with the herd:

$$\vec{S}_m^{Iter,AGE} = s_m^{Iter,AGE}\left[\left(\frac{1}{N}\sum_{j=1}^{N} X_j^{(Iter-1)}\right) - X_m^{(Iter-1)}\right], \quad AGE = \beta, \gamma \tag{11}$$

$$s_m^{Iter,AGE} = s_m^{(Iter-1),AGE} \times \omega_s \tag{12}$$

Now $\vec{S}_m^{Iter,AGE}$ and $s^{Iter,AGE}$ are social movement vectors of i^{th} horses and its direction the herd at *Iter* iteration, correspondingly. With the ω_s element, the directionwards the herd decreases at all the iterations. The N refers to the overall amount of horses in the group. Horses copy each other and choose bad and positive routines and behaviors, like finding the best grazing position. Juvenile horses copy others, and these behaviors persist until they mature. Likewise, these behaviours are inspired by HHO and represented as i, which is given:

$$\vec{I}_m^{Iter,AGE} = i_m^{Iter,AGE}\left[\left(\frac{1}{pN}\sum_{j=1}^{pN}\hat{X}_j^{(Iter-1)}\right) - X^{(Iter-1)}\right], \quad AGE = \gamma \tag{13}$$

$$i_m^{Iter,AGE} = i_m^{(Iter-1),AGE} \times \omega_i \tag{14}$$

Let $\vec{I}_m^{Iter,AGE}$ be the motion vector of i^{th} horses from the top horse's direction with $\hat{X}$ location. The pN refers to the amount of horses positioned in the optimum location. The p is ten percent of the overall amount of horses, and ω_i shows the reducing factor for each cycle for j_{Iter}. In response to danger, horses exhibit a fight-or-flight reaction, primarily opting to run. They compete for water and food to avoid risky environments and competition, similar to wolves. The defence mechanism of Horse acts from the HHO by running out of people who reveal incorrect behaviour and the d factor defines it. These behaviors are represented as a negative co-efficient to retain away from incorrect posture:

$$\vec{D}_m^{Iter,AGE} = -d_m^{Iter,AGE}\left[\left(\frac{1}{qN}\sum_{j=1}^{pN}\breve{X}\right) - X^{(Iter-1)}\right], \quad AGE = \alpha, \beta, \gamma \tag{15}$$

$$d_m^{Iter,AGE} = d_m^{(Iter-1),AGE} \times \omega_d \tag{16}$$

where $\vec{D}_m^{Iter,AGE}$ indicates the i^{th} escape factor of a horse from the average horse with the worst location. The qN indicates the overall amount of horses having the worst location. The ω_d shows the reduction factor for each d_{Iter} and q is suggested to be considered twenty percent of the overall amount of horses. Figure 2 depicts the flowchart of HHO.

The sixth behavior in HHO is wandering. Horses, especially when young, explore new areas out of curiosity, which decreases with age. The r is a random movement:

$$\vec{R}_m^{Iter,AGE} = r_m^{Iter,AGE}\, pX^{(Iter-1)}, \quad AGE = \gamma, \delta \tag{17}$$

$$r_m^{Iter,AGE} = r_m^{(Iter-1),AGE} \times \omega_r \tag{18}$$

where $\vec{R}$ refers to the random velocity vector of i^{th} horses to local search. The ω_d denotes the reduction factor of $r_m^{Iter,AGE}$ for each iteration. The velocities of δ, γ, β, and α horses are calculated:

$$\begin{aligned}\vec{V}_m^{Iter,\delta} = &\left[g_m^{(Iter-1),\delta}\omega_g\left(\breve{u} + \rho\breve{l}\right) + \left[X_m^{(Iter-1)}\right]\right] + \\ &+\left[i_m^{(Iter-1),\delta}\omega_i\left[\left(\frac{1}{pN}\sum_{j=1}^{pN}\hat{X}_j^{(Iter-1)}\right) - X^{(Iter-1)}\right]\right] + \left[r_m^{(Iter-1),\delta}\omega_r\, pX^{(Iter-1)}\right]\end{aligned} \tag{19}$$

$$\vec{V}_m^{Iter,\gamma} = \left[g_m^{(Iter-1),\gamma} \omega_g \left(\breve{u} + \rho \breve{l} \right) + \left[X_m^{(Iter-1)} \right] \right] + \left[h_m^{(Iter-1),\gamma} \omega_h \left[X_*^{(Iter-1)} - X_m^{(Iter-1)} \right] \right] +$$
$$+ \left[s_m^{(Iter-1),\gamma} \omega_s \left[\left(\frac{1}{N} \sum_{j=1}^{N} X_j^{(Iter-1)} \right) - X_m^{(Iter-1)} \right] \right] + \left[i_m^{(Iter-1),\gamma} \omega_i \left[\left(\frac{1}{pN} \sum_{j=1}^{pN} \hat{X}_j^{(Iter-1)} \right) - X^{(Iter-1)} \right] \right] - \quad (20)$$
$$- \left[d_m^{(Iter-1),\gamma} \omega_d \left[\left(\frac{1}{qN} \sum_{j=1}^{pN} \breve{X}_j^{(Iter-1)} \right) - X^{(Iter-1)} \right] \right] + \left[r_m^{(Iter-1),\gamma} \omega_r pX^{(Iter-1)} \right]$$

$$\vec{V}_m^{Iter,\beta} = \left[g_m^{(Iter-1),\beta} \omega_g \left(\breve{u} + \rho \breve{l} \right) + \left[X_m^{(Iter-1)} \right] \right] + \left[h_m^{(Iter-1),\beta} \omega_h \left[X_*^{(Iter-1)} - X_m^{(Iter-1)} \right] \right] +$$
$$+ \left[s_m^{(Iter-1),\beta} \omega_s \left[\left(\frac{1}{N} \sum_{j=1}^{N} X_j^{(Iter-1)} \right) - X_m^{(Iter-1)} \right] \right] - \left[d_m^{(Iter-1),\beta} \omega_d \left[\left(\frac{1}{qN} \sum_{j=1}^{pN} \breve{X} \right) - X^{(Iter-1)} \right] \right] \quad (21)$$

$$\vec{V}_m^{Iter,\alpha} = \left[g_m^{(Iter-1),\alpha} \omega_g \left(\breve{u} + \rho \breve{l} \right) + \left[X_m^{(Iter-1)} \right] \right] - \left[d_m^{(Iter-1),\alpha} \omega_d \left[\left(\frac{1}{qN} \sum_{j=1}^{pN} \breve{X}_j^{(Iter-1)} \right) - X^{(Iter-1)} \right] \right] \quad (22)$$

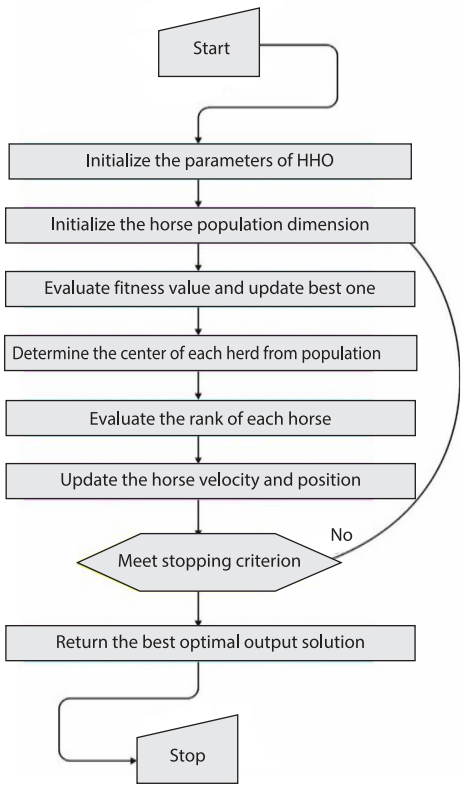

Figure 2. Flowchart of HHO

Deriving fitness function using the HHO-MOTSA technique

Fitness values for each individual are calculated as per eq. (23), with lower values indicating reduced energy consumption, makespan, and cost:

$$\text{Fit} = \min\left(\gamma_1 \times \epsilon + \gamma_2 \times \chi + \gamma_3 \times \zeta \right) \quad (23)$$

where the weightage parameters γ_1,γ_2, and γ_3 provide weight to various parameters *i.e.*, energy consumption, ϵ, makespan, χ, and cost, ζ. This performance with lower fitness values is known as the optimum result. Using these functions, other performances can be upgraded to find the optimum result. The fitness value evaluates the efficiency of possible solutions considering the prey location. The objective is to define the optimum task allocation the PM, such that the energy, cost, and makespan are reduced. Each parameter applied in FF is evaluated as: eq. (30) computes the energy consumption, ϵ, eq. (31) computes makespan, χ, and eq. (34) computes the cost, ζ. Moreover, the fitness of each individual solution is defined using:

Energy consumption, ϵ: In CDC, the energy expended by various elements is storage systems, servers, cooling, equipment electric, and network components. The PM is the primary energy con-

sumer and is largely affected by the CPU, disk storage, and RAM. The energy consumption of i^{th} PM, represented as E_i, is classified into 2 parts: static power, P_s, and dynamic power, P_d. The amount of dynamic and static powers produces the overall quantity of power usage:

$$P_t = P_s + P_d \tag{24}$$

The P_d aspects of PM are impacted by the frequency of CPU, f_i^d, and the square of supply voltage, $(V_i^t)^2$. The α_i denotes the relationships between dynamic power and these variables are described by proportionality constant. The dynamic power of PM P_d is the product of this constant with the CPU frequency f_i^d and $(V_i^t)^2$ is directly proportional to P_d:

$$P_d = \alpha_i (V_i^t)^2 f_i^d = \alpha_i (f_i^d)^3 \tag{25}$$

where β_i is the power ratio for idle PM and $f_{\max}$ and $P_{\max}$ are the maximal frequency and power utilization by host PM, correspondingly. The α_i is the proportionality coefficient for host PM_i:

$$\alpha_i = \frac{(1-\beta_i) P_{\max}}{(f_{\max})^3} \tag{26}$$

$$P_d = (1-\beta_i) P_{\max} (f_{\max})^3 (f_i^d)^3 \tag{27}$$

$$P_i = P_s + P_d = \beta_i P_{\max} S_i^t + \frac{(1-\beta_i) * P_{\max}}{(f_{\max})^3} (f_i^d)^3 \tag{28}$$

The parameter S_i^t is the execution status of PM, at time, t, where the value of binary variable 1 if PM_i, is currently active or else not. The quantity of energy consumed by PM_i from the starting time, s_t, to the finishing time, f_t, is estimated:

$$E_T = \int_{s_t}^{f_t} \left[\beta_i P_{\max} S_i^r + \frac{(1-\beta_i) P_{\max}}{(f_{\max})^3} (f_i^d)^3 \right] \mathrm{d}t \tag{29}$$

where E_T is the energy consumed by a single PM and is defined:

$$\epsilon = \sum_{i=1}^{m} P_{Ti} \tag{30}$$

where the amount of PM is denoted as m and the overall power consumed can be represented as ϵ.

Makespan, χ: Makespan is the time elapsed from the task initiation the task finishing and it is mathematically modelled:

$$\chi = \sum_{i=1}^{n} \max T_{fi} \tag{31}$$

where the overall amount of tasks is n and the finishing time of i^{th} task is T_{fi}.

Cost model, ζ: The CC tasks and resources have varying costs and CPU requirements. This study models resource costs by classifying them into memory and CPU to align user budgets with these variations:

$$Ct_c = v \times \eta + \rho \tag{32}$$

where ρ is the cost involved with CPU transmission, Ct_c – the CPU cost, v – the base price when the PM has been employed by the limited consumption, and η – the duration of the task t_i uses PM φ:

$$Ct_M = v \times \phi + \varpi \tag{33}$$

Likewise, Ct_M refers to memory cost, the base price for 1 GB of memory can denoted as v, and the time task t_i implement in PM is ϕ. The cost for the memory communication time is

ϖ. The cost function is attained using the equations where C_{CPU} is CPU cost and C_{mem} is memory cost:

$$C_{\text{CPU}} = \sum_{i=1}^{m} C_{ct}^{i} \tag{34}$$

$$C_{\text{mem}} = \sum_{j=1}^{m} M_{ct}^{j} \tag{35}$$

$$\zeta = C_{\text{CPU}} + C_{\text{mem}} \tag{36}$$

Experimental validation

This section compares the HHO-MOTSA technique's scheduling performance with various tasks [21]. Table 1 and fig. 3 show that HHO-MOTSA consistently has the lowest makespan (MSP) compared to CCS, ICSA, CSRSA, and EERS-CEPO, with MSP increasing as task numbers rise. For 100 tasks, HHO-MOTSA achieves 126 ms MSP, while the others range from 265-521 ms. For 600 tasks, HHO-MOTSA's MSP is 1621 ms, compared to 1862-2605 ms for the other methods.

Table 1. The MSP analysis of the HHO-MOTSA system with recent systems under various tasks

Makespan [ms]					
Number of tasks	CCS	ICSA	CSRSA	EERS-CEPO	HHO-MOTSA
50	250	243	289	237	202
100	467	358	521	265	126
150	746	630	917	460	285
200	970	809	1032	599	355
250	1149	979	1118	800	606
300	1343	1157	1195	978	754
350	1537	1366	1311	1103	959
400	1714	1559	1459	1266	1092
450	1932	1737	1652	1404	1171
500	2141	1940	1768	1544	1377
550	2381	2179	1993	1746	1512
600	2605	2289	2126	1862	1621

Table 2 and fig. 4 show that HHO-MOTSA achieves the lowest response time (RT) compared to CSRSA, CCS, and ICSA. The RT values increase with more iterations.

With 100 iterations, HHO-MOTSA has an RT of 792 ms, while CCS, ICSA, CSRSA, and EERS-CEPO have RT of 1663 ms, 1617 ms, 1111 ms, and 960 ms, respectively. With 250 iterations, HHO-MOTSA's RT is 519 ms, compared to 852 ms, 831 ms, 794 ms, and 660 ms for the other methods.

Table 3 and fig. 5 show that HHO-MOTSA achieves the lowest execution time (EXT) compared to CCS, ICSA, CSRSA, and EERS-CEPO. For 100 tasks, HHO-MOTSA has an EXT of 2000 ms, while the others range from 2210-3634 ms. For 600 tasks, HHO-MOTSA's EXT is 5171 ms, compared to 5406-6551 ms for the other methods. The EXT values increase with more tasks.

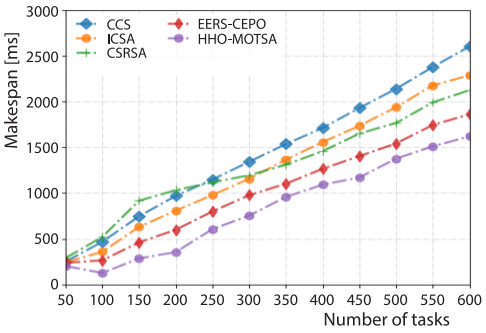

Figure 3. The MSP analysis of HHO-MOTSA technique under various tasks

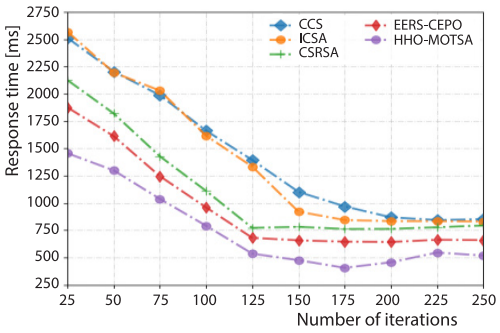

Figure 4. The RT analysis of the HHO-MOTSA technique under various iterations

Table 2. The RT analysis of the HHO-MOTSA technique with recent systems under various iterations

Response time [ms]					
Number of iterations	CCS	ICSA	CSRSA	EERS-CEPO	HHO-MOTSA
25	2513	2570	2126	1876	1456
50	2204	2196	1821	1613	1299
75	1988	2030	1428	1246	1038
100	1663	1617	1111	960	792
125	1394	1333	772	681	536
150	1102	921	782	658	476
175	969	844	762	646	406
200	869	834	763	643	457
225	844	834	778	663	546
250	852	831	794	660	519

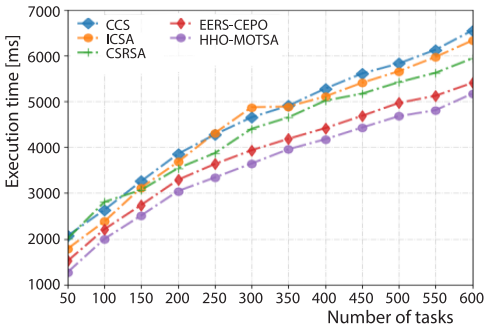

Figure 5. The EXT analysis of HHO-MOTSA technique under various tasks

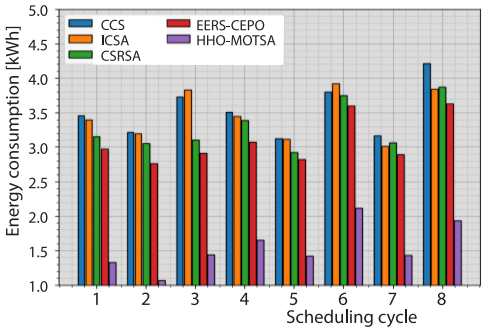

Figure 6. The ECON analysis of HHO-MOTSA technique under scheduling cycles

Table 3. The EXT analysis of the HHO-MOTSA technique with recent systems under various tasks

Execution time [ms]					
Number of tasks	CCS	ICSA	CSRSA	EERS-CEPO	HHO-MOTSA
50	2068	1785	2005	1518	1272
100	2634	2381	2807	2210	2000
150	3260	3120	3070	2741	2511
200	3855	3681	3543	3290	3044
250	4278	4309	3872	3635	3340
300	4656	4875	4405	3935	3646
350	4907	4890	4653	4185	3957
400	5280	5109	5017	4418	4175
450	5610	5408	5171	4686	4431
500	5829	5660	5421	4968	4679
550	6130	5972	5626	5125	4810
600	6551	6332	5940	5406	5171

Table 4 and fig. 6 show that HHO-MOTSA has the lowest energy consumption (ECON) compared to CCS, ICSA, CSRSA, and EERS-CEPO. With 1 scheduling cycle, HHO-MOTSA consumes 1.33 kWh, while the others range from 2.97-3.46 kWh. With eight cycles, HHO-MOTSA's ECON is 1.93 kWh, compared to 3.63-4.21 kWh for the other methods. The ECON values increase with more scheduling cycles. Table 5 and fig. 7 show HHO-MOTSA with the lowest average energy consumption (AECON) compared to CCS, ICSA, and CSRSA. AECON increases with task count. With 50 tasks, HHO-MOTSA has an AECON of 0.85 kJ, while CCS, ICSA, CSRSA, and EERS-CEPO have AECON of 2.68 kJ, 2.45 kJ, 2.46 kJ, and 2.32 kJ, respectively. With 600 tasks, HHO-MOTSA's AECON is 7.21 kJ, compared to 8.52-10.78 kJ for the others. Table 6 and fig. 8 show that HHO-MOTSA has the lowest average execution power (AEXP) compared to CCS, ICSA, CSRSA, and EERS-CEPO. With 100 tasks, HHO-MOTSA's AEXP is 1162 W, while the others range from 1434-1741 W. With 600 tasks, HHO-MOTSA's AEXP is 4016 W, compared to 4377-6175 W for the others. AEXP increases with task count.

Table 4. The ECON outcome of HHO-MOTSA technique with recent systems under scheduling cycles

Energy consumption [KWh]					
Scheduling Cycle	CCS	ICSA	CSRSA	EERS-CEPO	HHO-MOTSA
1	3.46	3.40	3.15	2.97	1.33
2	3.22	3.20	3.05	2.76	1.07
3	3.73	3.83	3.10	2.91	1.44
4	3.51	3.45	3.39	3.07	1.65
5	3.12	3.11	2.92	2.82	1.42
6	3.80	3.92	3.75	3.60	2.12
7	3.16	3.01	3.06	2.89	1.43
8	4.21	3.84	3.87	3.63	1.93

Table 5. The AECON analysis of HHO-MOTSA technique with recent systems under various tasks

Average energy consumption [kJ]					
Number of tasks	CCS	ICSA	CSRSA	EERS-CEPO	HHO-MOTSA
50	2.68	2.45	2.46	2.32	0.85
100	3.41	3.18	3.08	2.80	1.46
150	3.84	3.47	3.49	2.95	1.45
200	4.40	4.06	4.01	3.42	2.05
250	5.07	4.83	4.65	4.33	3.11
300	5.74	5.56	5.18	4.91	3.49
350	6.72	6.55	5.90	5.24	3.96
400	7.56	7.16	6.47	5.77	4.43
450	8.40	8.13	7.12	6.69	5.36
500	9.24	9.21	7.55	7.34	5.94
550	9.64	9.54	8.78	7.94	6.66
600	10.78	10.05	9.31	8.52	7.21

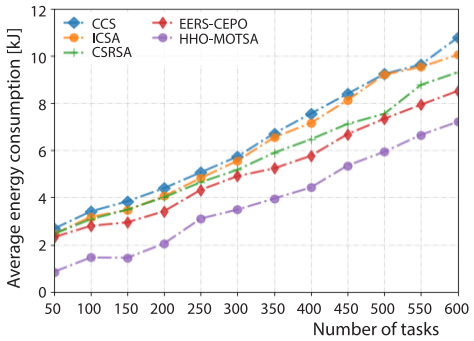

Figure 7. The AECON analysis of HHO-MOTSA technique under various tasks

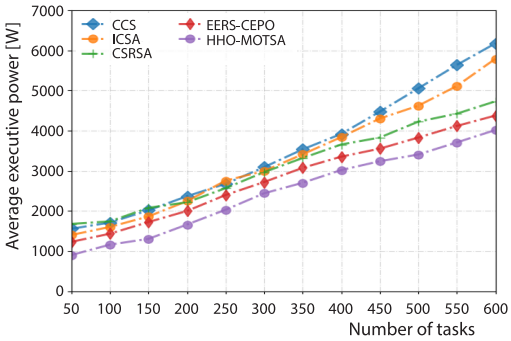

Figure 8. The AEXP analysis of the HHO-MOTSA technique under various tasks

Table 6. Analysis of HHO-MOTSA AEXP *vs.* recent systems across tasks

Average executive power [W]					
Number of tasks	CCS	ICSA	CSRSA	EERS-CEPO	HHO-MOTSA
50	1557	1403	1678	1232	903
100	1712	1603	1741	1434	1162
150	2018	1865	2079	1725	1306
200	2368	2245	2217	2001	1662
250	2676	2755	2568	2398	2031
300	3092	2997	2968	2721	2441
350	3535	3412	3319	3075	2702
400	3919	3840	3656	3353	3020
450	4473	4302	3829	3552	3239
500	5055	4622	4225	3829	3407
550	5638	5113	4424	4118	3708
600	6175	5791	4731	4377	4016

Thus, the HHO-MOTSA technique can be applied for enhanced scheduling performance in the CC environment.

Conclusion

In this manuscript, we focus on the designs and development of HHO-MOTSA in the CC platform. The drive of the HHO-MOTSA algorithm is to accomplish multi-objective TS in the CC to decrease the makespan, energy utilization, and cost. In addition, the HHO-MOTSA technique mainly relies on the social characteristics of horses at distinct ages by the use of six significant features. Moreover, the HHO-MOTSA technique designs an FF to estimate every individual, and a low fitness value represents that the minimization of energy utilization, makespan, and cost can be accomplished. The performance analysis of the HHO-MOTSA technique takes place using the CloudSim tool. The widespread simulation outcomes implied the effective TS solution of the HHO-MOTSA approach with other compared methods.

Acknowledgment

Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2025R 300), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

References

[1] Pirozmand, P., *et al*., Multi-Objective Hybrid Genetic Algorithm for Task Scheduling Problem in Cloud Computing, *Neural Comput. Appl*., *33* (2021), 19, pp. 13075-13088
[2] Khorsand, R., Ramezanpour, M., An Energy-Efficient Taskscheduling Algorithm Based on a Multi-Criteria Decision-Making Method in Cloud Computing, *Int. J. Commun. Syst*., 33 (2020), 9, e4379
[3] Prasanna Kumar, K. R., Kousalya, K., Amelioration of Task Scheduling in Cloud Computing Using Crow Search Algorithm, *Neural Comput. Appl*., *32* (2020), 10, pp. 5901-5907
[4] Agarwal, M., Srivastava, G. M. S., Opposition-Based Learning Inspired Particle Swarm Optimization (OPSO) Scheme for Task Scheduling Problem in Cloud Computing, *J. Ambient Intell. Humaniz. Comput*., *12* (2021), 10, pp. 9855-9875
[5] Panwar, N., *et al*., The TOPSIS-PSO Inspired Non-Preemptive Tasks Scheduling Algorithm in Cloud Environment, *Clust. Comput*., *22* (2019), 4, pp. 1379-1396
[6] Shen, Y., *et al.*, Adaptive Task Scheduling Strategy in Cloud: When Energy Consumption Meets Performance Guarantee, *World Wide Web*, *20* (2017), 2, pp. 155-173
[7] Panda, S. K., Jana, P. K., An Energy-Efficient Task Scheduling Algorithm for Heterogeneous Cloud Computing Systems, *Clust. Comput*., *22* (2019), 2, pp. 509-527
[8] Krishnadoss, P., Jacob, P., The OCSA: Task Scheduling Algorithm in Cloud Computing Environment, *Int. J. Intell. Eng. Syst*., *11* (2018), 3, pp. 271-279
[9] Fanian, F., *et al*., A New Task Scheduling Algorithm Using Firefly and Simulated Annealing Algorithms in Cloud Computing, *Int. J. Adv. Comput. Sci. Appl*., *9* (2018), 2, pp. 194-202
[10] Sanaj, M. S., Joe Prathap, P. M., Nature Inspired Chaotic Squirrel Search Algorithm (CSSA) for Multi Objective Task Scheduling in an IAAS Cloud Computing Atmosphere, *Eng. Sci. Technol.*, *23* (2020), 4, pp. 891-902
[11] Mokni, M., *et al*., Multi-Objective Fuzzy Approach to Scheduling and Offloading Workflow Tasks in Fog-Cloud Computing, *Simulation Modelling Practice and Theory*, *123* (2023), 102687
[12] Abualigah, L. Diabat, A., A Novel Hybrid Antlion Optimization Algorithm for Multi-Objective Task Scheduling Problems in Cloud Computing Environments, *Cluster Computing*, *24* (2021), Mar., pp. 205-223
[13] Kruekaew, B., Kimpan, W., Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm with Reinforcement Learning, *IEEE* Access, *10* (2022), Feb., pp. 17803-17818
[14] Mohammadzadeh, A., Masdari, M., Scientific workflow scheduling in multi-cloud computing using a hybrid multi-objective optimization algorithm, *Journal of Ambient Intelligence and Humanized Computing*, *14* (2023), 4, pp. 3509-3529

[15] Mangalampalli, S., *et al*., Multi Objective Task Scheduling in Cloud Computing Using Cat Swarm Optimization Algorithm, *Arabian Journal for Science and Engineering*, *47* (2022), 2, pp. 1821-1830
[16] Emara, F. A., *et al*., Genetic-Based Multi-objective Task Scheduling Algorithm in Cloud Computing Environment, *International Journal of Intelligent Engineering & System*s, *14* (2021), 5, pp. 571-582
[17] Yu, D., *et al*., Multi-Objective Task Scheduling Optimization Based on Improved Bat Algorithm in Cloud Computing Environment, *International Journal of Advanced Computer Science and Applications*, *14* (2023), 6, pp. 1091-1100
[18] Malti, A. N., *et al*., Multi-Objective Task Scheduling in Cloud Computing, Concurrency and Computation, *Practice and Experience*, *34* (2022), 25, e7252
[19] Behera, I., Sobhanayak, S., Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach. Journal of Parallel and Distributed Computing, *183* (2024), 104766
[20] Arasteh, B., *et al*., A Modified Horse Herd Optimization Algorithm and Its Application in the Program Source Code Clustering, in: *Complexity*, Wiley, N.Y., USA, 2023
[21] Mansour, R. F*., et al*., Design of Cultural Emperor Penguin Optimizer for Energy-Efficient Resource Scheduling in Green Cloud Computing Environment, *Cluster Computing*, *26* (2023), 1, pp. 575-586

Paper submitted: July 17, 2024
Paper revised: November 13, 2024
Paper accepted: November 28, 2024