of CPU utilisation and power consumption. The work uses a deep reinforcement learning model for training and model prediction. The proposed model is compared with a greedy algorithm using power consumption and average waiting time as performance parameters.

Work has also been carried out in job scheduling. For example, Ibrahim Attiya et al. presented a hybrid job scheduling approach in cloud computing using a modified Harris Hawk Optimisation and simulated annealing algorithm [45]. This work aims to minimise the makespan and improve the convergence speed. Both standard and synthetic workloads were employed to analyze the performance of the this work.

Authors in [46] proposed a multi-objective task scheduling technique, based on Gaussian Cloud Whale Optimisation Algorithm (GCWOAS2) in cloud computing. A three-layer scheduling model was presented in this work. The goal is to reduce the operating cost of the system by minimising task completion time by effectively utilising virtual machine resources and maintaining the load balancing of each virtual machine. To develop the best scheduling scheme in the GCWOAS2 approach, an opposition-based learning mechanism is initially employed to establish the scheduling strategy. Then, to dynamically widen the search range, an adaptive mobility factor is provided. To improve the unpredictability of the search, a Whale Optimisation technique based on the Gaussian cloud model is presented.

To summarise, prior research shows that the meta-heuristic approaches listed above can identify an appropriate solution for VM scheduling in cloud computing. However, experiments were carried out using randomly generated data in some works and most of the work focused on two to three objectives without taking into account load balancing, SLA violation and execution time concurrently. The proposed work in this article focuses on multi-objective VM placement along with load balancing which was not addressed in the existing approaches.

Motivation

The motivation of this work is to develop a new meta-heuristic algorithm to achieve better performance in the field of cloud computing. Where existing work as shown in the literature work uses traditional algorithms, this work proposes the Harris Hawk Optimisation (HHO) model to improve the performance of the cloud environment in terms of power consumption and utilisation of the system. The existing models are being compared with our proposed model to study the performance.

Problem formulation

The cloud data centre in this work consists of N VM and K PMs. The resource requirements of VMs are CPU and RAM. The requirements of CPU and memory of VM_i are represented as VM_{cpu_i}. and VM_{ram_i} respectively. The CPU and memory capacity of PMj are represented as PM_{cpu_j} and PM_{ram_j} respectively. Table 2 depicts the terminologies used in this work.

Each PM has enough capacity in this cloud data centre to allocate a set of VMs. Let $r = (r_{pm}, pm \in PM)$ denote the VM placement approach satisfying the resource allocation policy is feasible i.e. resources allocated to every VM are fewer than the overall capacity of the PM as represented in Eq 1.

$$\sum_{pm:pm\in PM} W_{pm} \cdot r_{pm} \leq 0, \tag{1}$$

Where W_{pm} represents the se'ver's willingness to offer resources or performance weight. Considering the proposed VMP method, let γ_{pm} be the fairness among the association of PMs. Once $\gamma_{pm} = 1$, the utility function of the pm is represented as $UT_{pm}(r_{pm}(t)) = W_{pm} log\, r_{pm}(t)$.

Table 2. Key terminologies.

Terminologies	Description
$VM = \{vm1, vm2, \ldots vmn\}$	Set of VMs
$PM = \{pm1, pm2, \ldots pmk\}$	Set of PMs
$VM = (pm)$	Set of VMs hosted by a PM $j \in PM$
$r_{pm}(t) = L_{pm}(t)$	VM resource needs to be aggregated at a PM
$r_{vp}(t) = L_{vp}(t)$	The VM resource demands placed on PM
$Capacity_{pm}$	PM capacity (e.g., CPU power, memory)

https://doi.org/10.1371/journal.pone.0289156.t002

Next, maximising the cumulative utilities of all PMs in the data centre is expressed as:

$$\max \sum_{pm \in PM} W_{pm}\ log\ r_{pm}(t) \tag{2}$$

Virtual machine placement problem statement

Let $L_{vp}(t)$ symbolise the load of *VM i* which is hosted on physical machine pm and $L_{pm}(t)$ denote the aggregate load of PM, the following condition (3) must be satisfied:

$$L_{pm}(t) = \sum\nolimits_{i:i \in VM(PM)} L_{vp}(t) \tag{3}$$

Here $L_{vp}(t)$ is the VM's load requirements as the d dimensional vector, where d = 2 when CPU and memory are considered, $L(t)$ is given by

$$L(t) = \left(VM_{cpu_i}, VM_{ram_i}\right) \tag{4}$$

Further, $Capacity_{pm}$ is defined as the available server capacity on PM $j \in PM$ regarding its CPU and RAM. The following formula must hold true to confirm that the overall load on any PM is not more than its capacity.

$$\sum\nolimits_{vm:vm \in VM(PM)} L_{vp}(t) \leq\ Capacity_{\ pm} \tag{5}$$

Typically, optimal placement of virtual machines on servers and turning off other servers leads to maximisation of utilisation and minimising server power consumption. To reflect this, in our analysis, the following equation is utilized:

$$(Y_1) : \max \sum_{pm:pm \in PM} UT_{pm}\left(L_{pm}(t)\right) \tag{6}$$

Subject to

$$\sum_{vm:vm \in VM(pm)} L_{vp}(t) = L_{pm}(t), \forall\ pm \in PM, \tag{7}$$

$$\sum_{vm:vm \in VM(pm)} L_{vp}(t) \leq\ Capacity_{\ pm}, \forall\ pm \in PM \tag{8}$$

$$Over\ L_{vp}(t) \geq 0, vm \in VM, pm \in PM$$

Based on the constraints below, a single PM can host a set of VMs:

$$\sum_{i}^{n} VM_{cpu_i} \leq PM_{cpu_j}, \forall\ vm_i \in VM,\ and\ pm_j \in PM \tag{9}$$

$$\sum_{i}^{n} VM_{ram_i} \leq PM_{ram_j}, \forall\ vm_i \in VM,\ and\ pm_j \in PM \tag{10}$$

The above equation ensures that the total resources used by a group of VMs should not surpass the CPU and memory capacities of PM.

When only CPU and RAM are considered, the PM resource utilisation problem (Y_1) will be equivalent to:

$$\left(Y_1'\right) : \max \sum_{pm:pm\in PM} UT_{pm}\left(PM_{cpu_j}(t) \times PM_{ram\ j}(t)\right) \tag{11}$$

Subject to

$$\sum_{vm:vm\in VM(PM)} PM_{cpu_j}(t) \leq\ Capacity\ {}^{cpu}_{pm}, \forall\ pm \in PM \tag{12}$$

$$\sum_{vm:vm\in VM(PM)} PM_{ram\ j}(t) \leq\ Capacity\ {}^{ram}_{pm}, \forall\ pm \in PM \tag{13}$$

$$Over\ L_{vp}(t) \geq 0, vm \in VM, pm \in PM$$

To facilitate the subsequent derivation of the formula, let $r_{pm}(t) = L_{pm}(t)$ To maximise the data ce'tre's' overall aggregate utilities and find the best solution, a Lagrange function is defined as:

$$\begin{aligned} &LR\left(r_{vp}, r_{pm}, \gamma, \beta\right) \\ &= \sum_{pm:pm\in PM} \left\{ UT_{pm}\left(r_{pm}(t)\right)\ +\ \gamma_{pm}\left(\sum_{vm:vm\in VM} r_{vp}(t)\ -\ r_{pm}(t)\right)\right\} \\ &\quad +\ \sum_{pm:pm\in PM} \beta_{pm}\left(\ Capacity\ _{pm}\ -\ \sum_{vm:vm\in VM} r_{vp}(t)\ -\ \varepsilon^2_{pm}\right) \end{aligned} \tag{14}$$

Where $\gamma = (\gamma_{pm}, pm \in PM)$ and $\beta = (\beta_{pm}, pm \in PM)$ are Lagrange multiplier vectors, $\varepsilon^2 = \left(\varepsilon^2_{pm}, pm \in PM\right)$ is the relaxation factor vector. Let γ_{vm} denote the load requirement of the virtual machine vm. Let β_{pm} be the available capacity of the physical machine pm. Let the resource occupied by all VMs on physical machine pm be expressed as $\sum_{vm:vm\in VM(pm)} r_{vp}(t)$ and $\sum_{vm:vm\in VM(pm)} r_{vp}(t)\ \cdot\ \varepsilon^2_{pm} \geq 0$ represents the enduring resources present on the physical machine pm.

According to Eq (7), we obtain:

$$\begin{aligned} & LR\left(r_{vp}, r_{pm}, \gamma, \beta\right) \\ & = \sum_{pm:pm \in PM} \left\{ UT_{pm}\left(r_{pm}(t)\right) - \gamma_{pm} r_{pm}(t) \right\} \\ & \quad + \sum_{pm:pm \in PM} \sum_{vm:vm \in VM(PM)} r_{vp}(t)\left(\gamma_{pm} - \beta_{pm}\right) \\ & \quad + \sum_{pm:pm \in PM} \beta_{pm}\left(Capacity_{\ pm} - \varepsilon_{pm}^2 \right) \end{aligned} \tag{15}$$

Where $Capacity_{\ pm} - \varepsilon_{pm}^2$ represents the occupied resource of physical machine pm. To increase the PM utilisation and minimise server energy consumption, optimal packing is performed to place virtual machines then $\beta_{pm}\left(Capacity_{\ pm} - \varepsilon_{pm}^2 \right)$ can be considered as the gains of PM from packing.

To save energy any PMs that are not' in use should be turned off. The power consumption of active PMs is formulated as follows:

$$\sum_{j=1}^{k} PC\left(PM_{cpu_j}\right) = \begin{cases} PM_{idle} + (PM_{max} - PM_{idle}) \times PM_{cpu_j}, \text{ if } PM_{cpu_j} > 0 \\ 0, \ otherwise \end{cases} \tag{16}$$

where P_{idle} is the idle-state power of PM_j , P_{Max} is the maximum power of PM_j , and P_{cpuj} is the percentage value $\in$ [0, 1] that denotes the CPU utilisation.

Harris Hawk Optimisation for virtual machine placement

Heidari and Mirjalili et al. developed the Harris Hawks Optimisation Technique (HHO), a novel optimisation algorithm [47]. The algorithm mimics the natural behaviour and hunting strategy of Harris Hawks known as surprise pounce. The hawks collaborate to attack from many directions to startle the victim. Harris Hawks reveal a variety of pursuit methods dependent on the nature of the schemes and the victim's' evasive patterns. Exploration and exploitation tactics are proposed by the conventional HHO algorithm which is driven by exploring prey, surprise pounce, and Harris Haw's' particular attacking approach. The Harris Hawks are the candidate solutions and the targeted prey in each phase is the best candidate solution (nearly the optimal one). The exploration phase, transition from exploration to exploitation phase, and exploitation phase are the three phases of the HHO algorithm and are described below.

i) Exploration phase. All Harris Hawks are considered solutions during this phase denoted as a solution matrix H; is a $2 \times N$ matrix, where N is the number of VMs. Fig 2 depicts 4 feasible solutions where four hawk agents' solutions assign 5 VMs to 3 PMs. The four solutions are represented by the set $H = H_1, H_2, H_3, and\ H_4$. According to the number of VMs, all solutions are reviewed and ranked in ascending order of PMs that are utilised in this solution [12]. The order of sorting is H_3, H_1, H_4, and H_2. The best solution is H3 because it has the minimum number of PMs, which consumes less power.

The solution matrix of a single hawk H is defined in Eq 17. If there are *N* VMs to be allocated to *J* PMs, the value of the component of H of the matrix represents the PM index to host