Multi-Objective Optimization-Oriented Resource Allocation in the Fog Environment: A New Hybrid Approach

Sonti Harika, Koneru Lakshmaiah Education Foundation, India*
B. Chaitanya Krishna, Koneru Lakshmaiah Education Foundation, India

ABSTRACT

Fog computing is a decentralized computer system where data, processing, storage, as well as applications are located anywhere between the cloud and data source. Fog computing takes the cloud closer to users, decreasing the latency and allowing the deployment of new delay-sensitive appliances. An important feature of a fog-cloud network is the process of decision making on assigning the resources to execute the tasks of application. This paper aims to propose a resource allocation strategy for fog computing that determines the effective process under the consideration of the objectives that includes the constraints like credibility score, concurrency, price affordability, and task time computation. Moreover, the credibility score is determined based on the execution efficiency, service response rate, access reliability, and reboot rate. Thereby, the optimal allocation of resources is handled by a new hybrid monarch-dragon algorithm (HM-DA) that hybridizes the concept of dragonfly algorithm (DA) and monarch butterfly optimization (MBO) algorithm.

KEYWORDS

Allocation Strategy, Credibility Score, Fog Computing, HM-DA Optimization, Reboot Rate

1. INTRODUCTION

Fog computing (Ghobaei-Arani, et al., 2019; Bellendorf and Mann, 2020; Alemneh, et al., 2020; Li, 2020; Shen, et al., 2020; Das, et al., 2020) is a promising approach that brings cloud computing, storage, and networking services closer to the end-user. This technology benefiting many industries such as manufacturing, e-health, education, oil and gas, smart cities, smart homes, and smart grids. It processes data as well as real-time applications at the edge to reduce latency, however, it is higher in cloud computing (Abdulkareem, 2019; Zheng, et al., 2019; Kim, et al., 2020; Khayer, et al., 2020) since it requires high bandwidth and remote processing. Even cloud and fog offer similar resources and services, fog computing is characterized by low latency with wider dispersed and geographically distributed nodes.

In a fog computing environment, resource selection strategy becomes a complex task due to the heterogeneous resources, dynamic nature of resources, and other environmental requirements (Nguyen, et al., 2019) as well. Moreover, it is an extremely challenging task when user expectations are contradictory and application requirements are expanded. The various fog resource allocation and scheduling solutions can only be more practicable if they are tested using an appropriate task
model. So, building an appropriate task model in a fog computing environment has been a major concern and needs to be addressed. Resource allocation and selection in a fog environment fall under the class of Multi-Criteria Decision Making (MCDM) problems (Chen, et al., 2019; Jie, et al., 2019; Li, et al., 2019; Abedin, et al., 2019). In several fog applications and storing data services, the complicated work can be partitioned into several cooperative subtasks with complicated relations and communications, such as interdependencies, concurrency, conflicts, and substitution, among others. Furthermore, price cost and execution time are important to these subtasks. In addition, fog computing infrastructure requires extensive research to solve many challenges such as QoS, efficient resource utilization, deadlock, task scheduling, and dynamic resource management (Hu and Li, 2009; Chen, et al., 2019). In contrast, optimization methods are good for minimizing the total execution time of a task and also solve the other existing challenges.

A new resource scheduling algorithm based on optimized fuzzy clustering is introduced to separate the resources, by which the scale of the resource gets reduced (Nguyen, et al., 2019; Mohammad, et al., 2018; Qayyum, et al., 2015; Mukherjee, 2019). Nowadays, renowned meta-heuristics algorithms such as Particle Swarm Optimization (PSO), Dragonfly Algorithm (DA), Monarch Butterfly Optimization (MBO), etc (Beno, et al., 2014; Zhang and Li, 2018; Gu, et al., 2018; Gao, et al., 2020) are exploited for resource allocation strategy in Fog computing. The existing algorithms have some drawbacks like the PSO algorithm has a low convergence rate Furthermore, the DA method is easy to implement, and it needs few parameters for tuning, and also it suits for different applications. Also, the MBO algorithm has simple and easy operations and this method produces significant results when compared to standard existing algorithms such as ABC, ACO, BBO, and DE respectively (Wang, et al., 2019).

So, the researchers proposed our method with DA and MBO models. The main contribution of the presented methodology is listed below:

- Introduces an optimization-based resource allocation strategy for fog computing that determines under diverse parameters like credibility score, concurrency, price affordability, and task time computation.
- Introduces a new Hybrid Monarch-Dragon Algorithm (HM-DA) that hybrids the concepts of both DA and MBO to improving the convergence rate.
- To overcome the drawbacks of DA such as minimum internal memory and slow convergence, the researchers proposed the model with hybridized DA and MBO algorithms.

In this paper, section II describes the review on resource allocation of fog computing. The proposed optimization-based resource allocation strategy in the fog environment is represented in section III. Section IV portrays the dataset creation in fog computing. Section V depicts the optimal resource allocation by the HM-DA algorithm. Section VI indicates the fitness computation of the proposed algorithm. The result and discussion are analyzed experimentally in section VII. Finally, the conclusion to this research work is presented in section VIII.

2. LITERATURE REVIEW

2.1 Related Works

In 2020, (Naha, et al., 2020) have proposed a resource allocation and provisioning algorithm by resource ranking in a hierarchical and hybrid fashion. The proposed algorithm was evaluated in a simulation environment by extending the CloudSim toolkit in a realistic Fog environment. This method reduces the overall delay and cost and further has decreased processing time. Still, a plan is needed for designing a complicated simulation environment with big data IoT applications.

In 2019, (Mishra, et al., 2019) have stated that the traditional MCDM methods were not suitable for the fog computing environment hence they are scalable, dynamic, and distributed. Therefore, a
new MCDM model was modeled, which obtained an optimal ranking of alternatives in both scalable and dynamic environments. An Adaptive MCDM (A-MCDM) achieves improved precision and decreased Mean Absolute Error (MAE) measure. However, determining the virtual machine kind that is needed for satisfying application-level requirements is needed.

In 2019, (Goudarzi, et al., 2019) have introduced a hierarchical technique with fog-driven resource allocation and dynamic distributed clustering. Moreover, a policy-aware resource allocation method was proposed that considered both inter and intra-cluster interferences. Finally, the experimental outcomes have proved the betterment of the proposed work with respect to increased throughput, user satisfaction, interference, and fairness when compared to other existing models. The future aim is on adding the virtualization tactic for using resources more effectively.

In 2019, (Sood and Singh, 2019) have established a new fog layer in optical elements that utilized the resources of the optical network. However, the proposed method used the Optical Line Terminals (OLTs), Optical Network Units (ONUs), and Passive Optical Network (PON) for delivering the cloud-based services more effectively with low latency. Social Network Analysis (SNA) based deadlock manager was proposed via Free Resource Fog (FRF) for detecting the deadlock from all running tasks. SNA offers improved resource availability with low latency, cost-effectiveness, and energy effectiveness. But, this method needs the maximum utilization of resources.

In 2019, (Wen, et al., 2019) have conducted collaborative research on fog computing and downlink Non-Orthogonal Multiple Access (NOMA) Hierarchical Networks (HiNets). In this work, a two-stage Stackelberg game model was proposed that considered Macro Remote Radio Head (MRRH) as a leader and Small Remote Radio Heads (SRRHs) as followers. Moreover, a bilateral user-sub channel matching scheme based on large Equivalent Channel Gain (ECG) priority was used in the fog computing process.

In 2019, (Li, et al., 2019a) have deployed a framework consisting of 3 parallel algorithms such as offloading, buffering, and resource allocation that increased the balance of resource allocation, task completion ratio, and throughput. The task offloading was decided based on task urgencies that considered estimated execution time, deadline, and transmission delay. The core drawbacks are: it only focuses on laxity time and has reduced system probability.

In 2019, (Li, et al., 2019b) have examined the issues in heterogeneous resource allocation and task scheduling for multiple devices in the wireless IoT networks. Moreover, the proposed model could transfer more data with minimal latency and limited resources. The deployment of NOMA in IoT networks enabled multiple IoT devices to transmit data in the same Fog computing Nodes (FN) at the same time, frequency, and code domain. Improved Genetic Algorithm (IGA) achieves better system throughput with improved outage probability. But needs consideration in effective computation offloading strategy and further investigation is needed on that.

In 2018, (Jia, et al., 2018) have investigated the computing resource allocation issues in three-layer fog computing networks. The proposed double-matching strategy has an extension of the deferred acceptance algorithm from two-side matching to three-side matching. The double-matching problem can be extended to three-layer networks, which is the future concern of this methodology.

Table I depicts the features and challenges of conventional resource allocation models in fog computing.

2.2 Objectives

This research is motivated to overcome the disadvantages such as complicated simulation environment (Naha, et al., 2020), maximum utilization of resources (Goudarzi, et al., 2019), system probability reduction (Li, et al., 2019), Computation offloading problems (Li, et al., 2019b) and so on.

Thus, the researchers consider the objectives such as credibility score, concurrency, price affordability, and task time computation. Furthermore, the researchers proposed a novel HM-DA model to handle the optimal allocation of resources. In addition, the proposed model produces similar outcomes such as improved resource availability (Sood and Singh, 2019), achieve stable results (Jia,
et al., 2018), better scalability (Goudarzi, et al., 2019), enhanced performance of communication systems (Wen, et al., 2019) and so on. Moreover, our model is very well suited for tourism service applications.

### 2.3 Fog Resources Vs. Cloud Resources

Fog is really a smart network that offloads clouds to allow more data collection, storage, and processing. The concepts of cloud vs. fog computing are very equivalent to one another. Still, there exist differentiations among both cloud and fog computing on certain parameters. The comparative study among fog and cloud computing is discussed below:

Ø Centralized cloud computing consists of a huge data centre that may be located across the world, a thousand miles away from consumer devices. Fog architecture is decentralized and consisting of millions of tiny nodes closer to the consumer devices.

Ø Fog serves as a mediator among the hardware and data centers that are closer to end-users. In cloud computing, data processing takes place in remote data centers.

<table>
<thead>
<tr>
<th>Author [citations]</th>
<th>Adopted scheme</th>
<th>Features</th>
<th>Challenges</th>
</tr>
</thead>
</table>
| Naha et al. (Naha, et al., 2020) | Resource Allocation and Provisioning Algorithms | • Reduces the overall delay and cost  
• Decreases processing time | • A complicated simulation environment is not considered |
| Mishra et al. (Mishra, et al., 2019) | A-MCDM | • Improved precision  
• Less error | • The virtual machine is not determined for satisfying application-level requirements |
| Goudarzi et al. (Goudarzi, et al., 2019) | Distributed dynamic clustering method | • Better scalability  
• Avoids the severe inter-cluster interference | • Requires virtualization tactic for using resources more effectively |
| Sood and Singh (Sood and Singh, 2019) | SNA | • Offers improved resource availability  
• Offers low latency, cost-effective, and energy effective | • Maximum utilization of resources are necessary |
| Wen et al. (Wen, et al., 2019) | Pricing-based Distributed Power Algorithm | • Significantly lower interference  
• Enhanced performance of communication systems | • The total interference includes the cross-tier interference as well |
| Li et al. (Li, et al., 2019) | Virtual Queueing Model | • Avoids the task starvation that causes long delays  
• Achieves better tradeoff among throughput and high task completion ratio optimization | • Only focuses on laxity time  
• Reduce the system probability |
| Li et al. (Li, et al., 2019b) | IGA | • Achieves better system throughput  
• Improved outage probability | • No consideration in effective computation offloading strategy  
• Computation offloading problems as well needs to be investigated |
| Jia et al. (Jia, et al., 2018) | Double-matching strategy | • High-cost efficiency  
• Achieves stable results | • The double-matching problem can be extended to further three-layer networks |
Ø Cloud is more powerful than fog with respect to computing capabilities and storage capacity.
Ø Fog performs short-term edge analysis due to its instant responsiveness, whereas the cloud performs long-term deep analysis due to its slower responsiveness.
Ø Fog provides low latency while the cloud provides high latency.
Ø A cloud system collapses without an internet connection. Fog computing uses various protocols and standards as the risk of failure is low.

3. PROPOSED OPTIMIZATION BASED RESOURCE ALLOCATION STRATEGY IN FOG ENVIRONMENT

The proposed method aims to introduce a new resource allocation strategy for fog computing, by which the fog user can select the satisfying resources from a set of pre-allocated resources. The resource allocation in the cloud to fog computing is carried out by designing a non-linear functionality that considers the constraints or parameters like credibility scores, price affordability, concurrency, and task time computation. The credibility score is determined based on the evaluation of execution efficiency, service response rate, access reliability, and Reboot rate. Accordingly, in this work, the optimal allocation of resources is carried out by a new hybrid optimization algorithm termed HM-DA. Fig. 1 shows the working strategy of the proposed resource allocation model in the Fog environment.

4. DATASET CREATION: FOG COMPUTING

A fog application could be decomposed into the composition of tasks executed on fog resources. Accordingly, the task of the user can be disintegrated into several sub-tasks. The time and price costs of each sub-task on each resource may vary. Consequently, it is essential to schedule tasks on resources reasonably to satisfy the user’s rapid response requirements with minimal user costs.

Tourism service is one of the prospective appliances that provide a variety of services with high mobility, wide geographical distribution, and low latency. In this research work, services offered by multiple service providers are decomposed, and it includes travel, transportation, weather, hotel, ticketing, and other services.
Fig. 2 described the implementation scenario of the tourism service appliances. According to the tourism sequences, the fog and cloud resources are allocated by the users. For example, user 1 holds the transitions $t_{11}, t_{2}, t_3, t_4, t_{51}$ and user 2 hold the transitions like $t_{12}, t_2, t_3, t_{51}$.

Table II illustrates the notations of transitions for tourism service application cases. For each task, Millions of Instructions per Second (MIPS) will be randomly generated for a dataset that lies between 200 and 300. Similarly, for each fog resource, 100 MIPS/CPU will be generated randomly and for each cloud resource, 100 MIPS/CPU will be randomly generated.

<table>
<thead>
<tr>
<th>No</th>
<th>Transition</th>
<th>Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$t_{11}$</td>
<td>Air Ticket Bk</td>
<td>booking an air ticket</td>
</tr>
<tr>
<td>2</td>
<td>$t_{12}$</td>
<td>Train Ticket Bk</td>
<td>booking a train ticket</td>
</tr>
<tr>
<td>3</td>
<td>$t_{13}$</td>
<td>Ship Ticket Bk</td>
<td>booking a ship ticket</td>
</tr>
<tr>
<td>4</td>
<td>$t_{14}$</td>
<td>Bus Ticket Bk</td>
<td>booking a bus ticket</td>
</tr>
<tr>
<td>5</td>
<td>$t_5$</td>
<td>Hotel Room Res</td>
<td>reserving a hotel room</td>
</tr>
<tr>
<td>6</td>
<td>$t_3$</td>
<td>Auto Rental</td>
<td>hiring an auto</td>
</tr>
<tr>
<td>7</td>
<td>$t_4$</td>
<td>Weather Inq</td>
<td>inquiring the weather</td>
</tr>
<tr>
<td>8</td>
<td>$t_{51}$</td>
<td>Tour Res (Indoor and outdoor)</td>
<td>reserving a tourism item for both indoor and outdoor</td>
</tr>
<tr>
<td>9</td>
<td>$t_{52}$</td>
<td>Game Ticket Res</td>
<td>reserving a game ticket</td>
</tr>
<tr>
<td>10</td>
<td>$t_{51}; t_7$</td>
<td>-</td>
<td>auxiliary transitions for loop services</td>
</tr>
</tbody>
</table>
5. OPTIMAL RESOURCE ALLOCATION BY HM-DA ALGORITHM

5.1 Solution Encoding

Fig. 3 illustrates the input solution to the proposed algorithm for optimal allocation. The tasks are arranged as the user order (i.e.), for generation 1 → user 1, generation 2 → user 2, generation 3 → user 4 (if user 3 is unavailable). Here, \( r_j \) indicates the user’s tasks in which \( j = 1, 2, \ldots, N_{\text{tasks}} \). Here, \( r_j \).

For example, if \( N_{\text{fog}} = N_{\text{cloud}} = 10 \) (for simplicity), then, \( r_j \in \{0, 1, 2, \ldots, 9, 10, \ldots, 19\} \) in which 0 to 9 indicates the fog resources and 10 to 19 indicates the cloud resources.

Thus, the user tasks for encoding can be formulated as given in Eq. (1), (2) and (3).

\[
\begin{align*}
    r_j &\in \{0, 1, 2, \ldots, N_{\text{fog}}, N_{\text{fog}+1}, \ldots, N_{\text{fog}} + N - 1_{\text{cloud}}\} \\
    N_{\text{fog}} &= \| R_{\text{fog}} \| \\
    N_{\text{cloud}} &= \| R_{\text{cloud}} \|
\end{align*}
\tag{1, 2, 3}
\]

Where \( N_{\text{fog}} = \| R_{\text{fog}} \| \) indicates the number of fog resources and \( N_{\text{cloud}} = \| R_{\text{cloud}} \| \) represents the number of cloud resources.

If the same user submits more than one task (concurrent service), it selects different resources for the tasks.
6. FITNESS COMPUTATION OF PROPOSED ALGORITHM

6.1 Credibility Scores

The credibility score evaluation of resource \( R_j \) received from a single user \( user_i \) is the basic credibility evaluation \( CB_{reij} \) of resource \( R_j \). When \( user_i \) interacts with \( R_j \), the \( CB_{reij} \) can be expressed as in Eq.(4), where \( \omega_k \in [0,1] \), \( \sum_{k=1}^{4} \omega_k = 1 \), \( \omega_k \) can be assigned based on the different preference of users with response time \( S_T \), execution time \( E_T \), reboot rate \( F_T \), and access reliability \( A_T \). Here, \( \omega_1 \), \( \omega_2 \), \( \omega_3 \) and \( \omega_4 \) denotes the weight factors that are assigned with a value of 0.25.

\[
CB_{reij} = \frac{\omega_1 (S_T) + \omega_2 (E_T) + \omega_3 (1 - F_T) + \omega_4 (A_T)}{4}
\] (4)

6.2 Price Affordability

The user price and machine price are defined. The user's affordable price will be high. Then only the resource will be allocated to the particular user. But if the machine price is more than user price means the resource will be allotted to some other else so the penalty will be assumed. For optimal resource allocation, the price affordability penalty will be low. If the user price is higher than the machine price, a penalty of 0 will be assigned. On the other hand, if the machine price is higher than the user price, a penalty of 1 will be assigned. The price affordability \( P \) is determined in Eq. (5). Here, \( B \) indicates the user price and \( K \) denotes the machine price.

\[
P = \begin{cases} 
0; & \text{if } B > K \\
1; & \text{else} 
\end{cases}
\] (5)

6.3 Concurrency

If more number of resources allocated to the same user at the defined time \( t \), then the resources are executed simultaneously. This is termed as concurrency. The penalty of this concurrent resources will be estimated for optimal resource allocation. The concurrency value must be low. In particular instances, if the same resources are allocated to the same user for multiple tasks at a time \( t \), then penalty 1 is added otherwise, penalty 0 is added. The concurrency \( Q \) is defined as in Eq. (6).

\[
Q = \begin{cases} 
1; & \text{if same resources allocated to same user for multiple tasks at time } t \\
0; & \text{else} 
\end{cases}
\] (6)

If high number of resources allotted to same user means the penalty will be high. So the resources has to be evenly distributed to reduce the penalty.

6.4 Task Time Computation

Time computation is defined as the maximum time to complete a task for every user. The time cost \( G \) is determined in Eq. (7), where, \( c(i) \) indicates the task completion time.
\[ G = \sum_{i}^{N} c(\text{user}_i) \]  

(7)

The \( c(\text{user}_i) \) and \( T_i \) is expressed as in Eq. (8) and Eq. (9), where, \( \sum_{j=1}^{i-1} T(\text{user}_j) \) indicates the waiting time of the resources shared by previous users.

\[ c(\text{user}_i) = \left( \sum_{j=1}^{i-1} T(\text{user}_j) \right) + T_i \]  

(8)

\[ T_i = \max_k \left( T_k(\text{user}_i) \right) \]  

(9)

Where \( T_k(\text{user}_i) \) indicates the completion time of \( k^{th} \) task. On combining Eq. (8) and Eq. (9), the researchers get Eq. (10).

\[ C(\text{user}_i) = \max_k \left[ \sum_{j=1}^{i-1} T_k(\text{user}_j) + T_k^i \right] \]  

(10)

The following example shows the task time computation of users. Resource 1 is shared by user 1 and user 2. For user 2’s task 1, the time can be computed using Eq. (11).

\[ \text{task}_1\text{time} = \text{task}_1\text{time}(\text{user}1) + \text{task}_1\text{time}(\text{user}2) \]  

(11)

User 2’s task completion time can be computed as per Eq.(12). Here, both \( \text{task}_1\text{time} \) and \( \text{task}_2\text{time} \) can be run in parallel. Eq. (13) indicates the time of task 3.

\[ \text{task}_2\text{time} = \max(\text{task}_1\text{time}, \text{task}_2\text{time}) \]  

(12)

\[ \text{task}_3\text{time} = \text{task}_2\text{time} + \text{task}_3\text{time} \]  

(13)

Hence, after the completion of task 2 on resource 2, task 3 can be executed as given in Eq. (14), (15), and (16), where \( \{T_{11} + T_{12}\} \) indicates the task 1 time, \( T_{11} \) denotes the user 1’s task 1 and \( T_{12} \) represents the user 2’s task 1.

\[ \text{task}_3\text{time} = \{T_{11} + T_{12}\}, \max\{T_{12}, T_{22}\}, \{T_{22} + T_{32}\} \]  

(14)
task\textsubscript{time} = \max \{T_{11} + T_{12}, T_{22}, T_{22} + T_{32}\} \quad (15)

Therefore, \( T_{22} < T_{22} + T_{32} \).

\[ \text{task}\textsubscript{time} = \max \{T_{11} + T_{12}, T_{22} + T_{32}\} \quad (16) \]

Thus the task time can be calculated as the sum of response task time and the tasks submitted to the same resources as shown in Eq. (17). Here, \( X \) indicates the task time.

\[ X = \sum \left( \text{response task time} \& \text{tasks submitted to same resources} \right) \quad (17) \]

The overall fitness function can be expressed as in Eq. (18) and hence \( Y \) indicates the fitness function. \( \omega_5, \omega_6, \omega_7 \) and \( \omega_8 \) be a weighting factors.

\[ Y = \frac{\min \left[ \omega_5 \times \left( 1 - CB_{v_i} \right) + \omega_6 \times P + \omega_7 \times \left( Q \right) + \omega_8 \times \left( X \right) \right]}{4} \quad (18) \]

### 6.5 HM-DA Optimization Algorithm

Although the existing DA model results in precise measurements; it also limits with few drawbacks such as minimum internal memory and slow convergence. Therefore, to overcome the drawbacks of the existing DA, the concept of MBO is merged with it to introduce a new HM-DA algorithm. Two significant stages are included: (i) Exploration and (ii) Exploitation Eq. (19) demonstrated the computed modelling for separation. Where, \( H_i \) denotes the \( l^{th} \) nearer individual position, \( H \) represents the current individual position, and \( U \) specifies the count of nearby individuals.

\[ D_i = -\sum_{i=1}^{U} \left( H - H_i \right) \quad (19) \]

The alignment formula is as per Eq. (20). Here, \( Z_i \) indicates the velocity of \( l^{th} \) nearby individual. In addition, the cohesion formula is specified in Eq. (21). Attraction to food is estimated as per Eq. (22), where \( H^+ \) points out food source position.

\[ J_i = \sum_{i=1}^{U} \frac{Z_i}{U} \quad (20) \]
Distraction to the enemy is defined in Eq. (23), where the enemy position is represented by $H^-$. Here, two vectors namely step ($\Delta H$) and position ($H$) are computed to update the dragonfly’s position as specified below.

$$V_i = H^- + H$$

The step vector is described as per Eq. (24), which shows the moving direction of the dragonfly. Where, $q$ points out the separation weight, $D_i$ refers to the separation of $i^{th}$ individual, $O$ indicates the $i^{th}$ individual cohesion, $g$ points out cohesion weight, $J_i$ and $W_i$ denotes the alignment and food resources of $i^{th}$ individual, $a$ refers to the alignment weight, $f$ signifies the food factor, $b$ corresponds enemy factor, $h$ symbolizes the inertia weight, $Vn_i$ symbolizes enemy’s position of $i^{th}$ individual and $it$ refers the iteration counter.

$$\Delta H(it + 1) = \left(qD_i + aJ_i + gO_i + fW_i + bVn_i\right) + h\Delta H(it)$$

After calculating the step vector, the position vectors are defined as per Eq. (25).

$$H(it + 1) = H(it) + \Delta H(it + 1)$$

The procedure of the proposed HM-DA logic is as follows: As per the proposed HM-DA, the population size $N_{pop}$ is assigned as 10. Consequently, the first 5 solutions are updated using the DA algorithm, i.e. using Eq. (24), whereas, the remaining 5 solutions are updated using the butterfly adjusting operator of the MBO algorithm as shown in Eq. (26).

$$H^{i+1}_{k,j} = H^{i+1}_{k,j} + \alpha \times \left(dx_k - 0.5\right)$$

In Eq. (26), $H^{i+1}_{k,j}$ indicates the $k^{th}$ element of $H_{j}$ which presents the position of the monarch butterfly $j$. $\alpha$ refers to the weighting factor and $dx$ is the walk step of the monarch butterfly $j$.

The pseudo-code for HM-DA methodology is manifested in Algorithm 1.
7. RESULTS AND DISCUSSION

7.1 Simulation Procedure

The proposed HM-DA-based resource allocation model was executed using MATLAB and the results were observed. The implementation was carried out in 5 instances for 6 scenarios. Accordingly, the betterment of the proposed HM-DA model was evaluated by comparing it over the traditional models like DA (Mirjalili1, 2015), MBO (Wang, et al., 2019), Priced Time Petri Nets (PTPN) (Ni, et al., 2017), A-MCDM (Mishra, et al., 2019) and Preference Ranking Organization Method For Enrichment Evaluations (PROMETHEE) (Mishra, et al., 2019). Here, the fitness was computed for different iterations that ranges from 0 to 100. Moreover, the allocation of fog and cloud resources by proposed and conventional models are shown for varied scenarios.

For scenario 1, an equal weightage of 0.25 is assigned for $w_5, w_6, w_7$ and $w_8$, respectively. For scenario 2, the $w_5$ were assigned with the value of 0.4 and $w_6, w_7$ and $w_8$ are assigned with the value

<table>
<thead>
<tr>
<th>Table 3. Algorithm 1: proposed HM-DA method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong>: Dragonflies population, step vector, butterfly adjusting operator</td>
</tr>
<tr>
<td><strong>Output</strong>: Optimal solution for resource allocation</td>
</tr>
<tr>
<td>Initialize the dragonflies population $\left( N_{\text{pop}} \right)$</td>
</tr>
<tr>
<td>Initialize step vectors $\Delta H_i$</td>
</tr>
<tr>
<td>while the end condition is not satisfied</td>
</tr>
<tr>
<td>Calculate the objective values of all dragonflies</td>
</tr>
<tr>
<td>Update the food source and enemy</td>
</tr>
<tr>
<td>Update $q, a, g, f$ and $b$</td>
</tr>
<tr>
<td>Calculate $D, J, O, W$ and $V$ using Eq. (19) to (23)</td>
</tr>
<tr>
<td>Update neighboring radius</td>
</tr>
<tr>
<td>if a dragonfly has at least one neighboring dragonfly</td>
</tr>
<tr>
<td>For first 5 population, i.e. for $\left( N_{\text{pop}} \right) = 1, 2, 3, 4, 5$</td>
</tr>
<tr>
<td>Update velocity vector using Eq. (24)</td>
</tr>
<tr>
<td>Update solutions of position vector using Eq.(25)</td>
</tr>
<tr>
<td>For remaining 5 population, i.e. for $\left( N_{\text{pop}} \right) = 6, 7, 8, 9, 10$</td>
</tr>
<tr>
<td>Update solutions using butterfly migration operator of MBO algorithm as per Eq. (26)</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>Check and correct the new positions based on the boundaries of variables</td>
</tr>
<tr>
<td>end while</td>
</tr>
</tbody>
</table>
of 0.2. For scenario 3, the \( \omega_6 \) were assigned with the value of 0.4 and \( \omega_5, \omega_7 \) and \( \omega_8 \) are assigned with the value of 0.2. For scenario 4, the \( \omega_7 \) was assigned with the value of 0.4 and \( \omega_5, \omega_6 \) and \( \omega_8 \) was assigned with the value of 0.2. For scenario 5, the \( \omega_8 \) was assigned with the value of 0.4 and hence the \( \omega_5, \omega_6 \) and \( \omega_7 \) are assigned with the values of 0.1. For scenario 6, an equal weightage of 1 was assigned for \( \omega_5, \omega_6, \omega_7 \) and \( \omega_8 \), correspondingly.

### 7.2 Computation metrics of Fog and Cloud resources

The computation of both fog and cloud resources is summarized in Table IV.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Fog resources</th>
<th>Cloud resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response time</td>
<td>BW</td>
<td>10% of BW</td>
</tr>
<tr>
<td></td>
<td>Memory</td>
<td>10% of Memory</td>
</tr>
<tr>
<td>Execution time</td>
<td>CPU</td>
<td>CPU</td>
</tr>
<tr>
<td></td>
<td>Memory</td>
<td>10% of Memory</td>
</tr>
<tr>
<td></td>
<td>EBS</td>
<td>10% of EBS</td>
</tr>
<tr>
<td>Reboot rate</td>
<td>Instant storage</td>
<td>Instant storage ( \times 10 )</td>
</tr>
<tr>
<td>Access reliability</td>
<td>BW</td>
<td>10% of BW</td>
</tr>
</tbody>
</table>

#### 7.2.1 Response Time

The response time is used to calculate the response rate of a resource for a user’s task through the duration of the response time of the fog resource that offers the service. The response time \( S_T \) can be expressed as in Eq. (27).

\[
S_T = \left( \frac{Bw}{\max(Bw)} + \frac{M}{\max(M)} \right)^{\frac{1}{2}}
\]

Where, \( Bw \) denotes the bandwidth and \( M \) indicates the memory.

#### 7.2.2 Execution Time

Execution efficiency characterizes the speed of a resource executing a user’s tasks from the view of execution time. The execution time \( E_T \) is represented by using Eq. (28).

\[
E_T = \left( \frac{cpu}{\max(cpu)} + \frac{M}{\max(M)} + \frac{EBS}{\max(EBS)} \right)^{\frac{1}{3}}
\]
7.2.3 Reboot Rate

Reboot rate characterizes the stability of a resource. If the number of reboots of the user’s task is low, the stability of the resource will be more. Else, the stability will be worse. The reboot rate $F_T$ is determined in Eq. (29), where, $I_s$ denotes the instant storage.

$$F_T = \left(1 - \frac{I_s}{\max(I_s)}\right)$$  \hspace{1cm} (29)

7.2.4 Access Reliability

Reliability describes whether the provided service is successful or not. Access reliability is defined as the ratio of the bandwidth value to the maximum number of bandwidth for the user during a period of time. The access reliability $A_T$ is determined in Eq. (30).

$$A_T = \frac{Bw}{\max(Bw)}$$  \hspace{1cm} (30)

7.2.5 User Price Affordability

During simulation, 20 users are generated. Table V demonstrates the user’s requirement of price cost $c$ to complete a task in unit standard time.

7.2.6 Machine Price Affordability

Table VI indicates the price cost $c$ per unit standard time to perform the tasks on each machine. During the process of resource mapping to the user resource directory, the user is randomly selected as per the sequence of user grouping and the machine that fulfills the user’s request.

<table>
<thead>
<tr>
<th>User ID</th>
<th>$U_1$</th>
<th>$U_2$</th>
<th>$U_3$</th>
<th>$U_4$</th>
<th>$U_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>1.421</td>
<td>1.010</td>
<td>1.217</td>
<td>1.331</td>
<td>1.194</td>
</tr>
<tr>
<td>User ID</td>
<td>$U_6$</td>
<td>$U_7$</td>
<td>$U_8$</td>
<td>$U_9$</td>
<td>$U_{10}$</td>
</tr>
<tr>
<td>$c$</td>
<td>1.399</td>
<td>1.044</td>
<td>1.248</td>
<td>1.252</td>
<td>1.201</td>
</tr>
<tr>
<td>User ID</td>
<td>$U_{11}$</td>
<td>$U_{12}$</td>
<td>$U_{13}$</td>
<td>$U_{14}$</td>
<td>$U_{15}$</td>
</tr>
<tr>
<td>$c$</td>
<td>1.098</td>
<td>1.270</td>
<td>1.373</td>
<td>1.297</td>
<td>1.144</td>
</tr>
<tr>
<td>User ID</td>
<td>$U_{16}$</td>
<td>$U_{17}$</td>
<td>$U_{18}$</td>
<td>$U_{19}$</td>
<td>$U_{20}$</td>
</tr>
<tr>
<td>$c$</td>
<td>1.239</td>
<td>1.141</td>
<td>1.21</td>
<td>1.073</td>
<td>1.199</td>
</tr>
</tbody>
</table>
7.3 Performance Analysis

The performance of the proposed HM-DA-based resource allocation model over the conventional methods with respect to fitness is described in this section. Here, Fig. 4 illustrated the performance analysis of the proposed HM-DA method over the traditional models at 5 instances for scenario 1. As the number of iterations increases, the proposed algorithm could converge to the least fitness, which is the assurance of precise resource allocation. Eq. (22) gets satisfied by the proposed algorithm, where the conventional algorithms show poor performance in this aspect.

Particularly, in Fig 4(a) for instance 1 the proposed HM-DA method attains minimal fitness at 80th at scenario 1. In Fig. 4(b), the adopted HM-DA method for instance 2 is 9.52% and 47.6% superior to the traditional models like DA and MBO, correspondingly at 100th iteration with minimized fitness. Fig. 5 illustrated the performance analysis of the proposed HM-DA method over the traditional models at 5 instances for scenario 2. Almost for all the instants, the proposed algorithm declares better performance in satisfying the defined fitness of optimal resource allocation.

Table 6. Price cost \(c\) unit standard time which task executes on the machine \(c\)

<table>
<thead>
<tr>
<th>Machine</th>
<th>(M_1)</th>
<th>(M_2)</th>
<th>(M_3)</th>
<th>(M_4)</th>
<th>(M_5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c)</td>
<td>1.086</td>
<td>1.2886</td>
<td>1.367</td>
<td>1.16427</td>
<td>1.353</td>
</tr>
<tr>
<td>Machine</td>
<td>(M_6)</td>
<td>(M_7)</td>
<td>(M_8)</td>
<td>(M_9)</td>
<td>(M_{10})</td>
</tr>
<tr>
<td>(c)</td>
<td>1.223</td>
<td>1.170</td>
<td>1.350</td>
<td>1.430</td>
<td>1.057</td>
</tr>
</tbody>
</table>

Figure 4. Performance analysis of proposed HM-DA method over the traditional models for (a) instance 1 (b) instance 2 (c) instance 3 (d) instance 4 € instance 5 for scenario 1
Fig. 6 demonstrates the performance analysis of the proposed HM-DA method over the traditional models at 5 instances for scenario 3. As the number of iterations increases, the fitness value is found to be minimal, thus ensuring the superiority of the proposed HM-DA method. Particularly, in Fig 6(a) the proposed HM-DA method for instance 1 is 22.72% and 56.41% superior to the traditional schemes such as DA and MBO, respectively at the 80th iteration. In Fig. 6(b), the adopted HM-DA method at the 60th iteration is 26.31% and 78.94% superior to the traditional models including DA and MBO, correspondingly for instance 2. Thus the improvement was obtained in the proposed HM-DA method. Fig 7 indicates the performance analysis of the proposed HM-DA method over the traditional models at 5 instances for scenario 4.
Fig. 8 illustrated the performance analysis of the proposed HM-DA method over the traditional models at 5 instances for scenario 5. Particularly, for some instance in Fig 8(a) the conventional model gets closer to the proposed algorithm for some iterations; however, the proposed algorithm shows its better performance with reduced fitness at the 100th iteration. Fig. 9 illustrated the performance analysis of the proposed HM-DA method over the traditional models of scenario 6 for various instances. However, in Fig 9(a) the proposed HM-DA method at the 40th iteration is 21.21% and 55.17% superior to the traditional schemes such as DA and MBO, respectively for instance 1. This ensures the improvement of the proposed HM-DA method in optimal resource allocation.
7.4 Analysis of Credibility Score and Objective Functions

The analysis of credibility score and objective functions of the proposed HM-DA method over the conventional methods like DA, MBO, PTPN, A-MCDM, and PROMETHEE for various scenarios are described in this section. Here, Fig. 10 demonstrates the analysis of credibility scores with respect to access reliability, execution rate, reboot rate, and response rate. As defined, the proposed HM-DA has shown high credibility score with respect to all the defined objectives. It is observed that the reboot rate should be lower for high credibility score, which is proved by the proposed algorithm. In Fig. 10 (a), the access reliability of the proposed HM-DA method at scenario 1 is superior to the traditional schemes such as DA, MBO, PTPN, A-MCDM, and PROMETHEE, respectively for the credibility score. However, in Fig. 10(d), the response rate of the proposed HM-DA method at scenario 4 attains the value of 0.19 whereas, the existing method holds the value of 0.17, 0.16, 0.165, 0.13, and 0.13 for DA, MBO, PTPN, A-MCDM, and PROMETHEE, respectively.

Fig. 11 demonstrates the analysis of an objective function with respect to concurrency, credibility scores, price affordability, and time cost for various scenarios. From the graphs, the credibility score is higher for all scenarios; thereby the concurrency value was lower for both proposed and traditional models for all scenarios. In Fig. 11 (a), the time cost of the proposed HM-DA method attains the value of 0.02 that is superior to the traditional schemes such as DA, MBO, PTPN, A-MCDM, and PROMETHEE that attained the values of 0.03, 0.05, 0.05, 0.13 and 0.13, respectively. Thus the proposed model shows the betterment in optimal resource allocation over other conventional models.

7.5 Makespan Analysis

The performance analysis of the time cost of the proposed HM-DA method over the conventional methods is described in this section (Fig 12). In fact, “makespan denotes the total time taken by the resources to complete the executing of all tasks”. On noticing the graphs, the time is taken (makespan) by the proposed model for all scenarios is lower when compared over the existing schemes. Hence, less makespan proves the efficiency of the proposed model with better robustness. At scenario 6,
the proposed HM-DA is better than other traditional models like DA, MBO, PTPN, A-MCDM, and PROMETHEE, respectively.

Figure 10. Analysis on credibility scores for proposed HM-DA method over the traditional models (a) scenario 1 (b) scenario 2 (c) scenario 3 (d) scenario 4 (e) scenario 5 (f) scenario 6
7.6 Task list

Table VII demonstrated the task list for five instances for twenty users. The tasks are assigned at varied instances by the user. The numbers (1 to 10) in the Table VI denotes the tasks, i.e., presented in Table II. User 1 allocates task 2 (Train Ticket Bk) and 5 (Hotel Room Res) for instance 1, task 6 (Auto Rental) and 8 (Tour Res) for instance 2, task 9 (Game Ticket Res) and 10 (auxiliary transitions for loop services) for instance 3. Similarly, other users define their tasks for five instances.

Table VIII shows the resource allocation of the proposed model in fog resource and cloud resource for scenario 1. In the proposed method, user 1 allocates their six different tasks in machines no 8, 9, and 4 for fog instances 1 and 2, and the remaining tasks are allocated in machines no 3, 7, 1, 8, and 9 for cloud instances 2, 3, and 5.
Figure 12. Performance analysis of the proposed method over the traditional models with respect to time cost

Table 7. Task list for various instances

<table>
<thead>
<tr>
<th>User</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
</tr>
</tbody>
</table>
This work proposes a fog computing resource allocation technique that identifies the most efficient process under a variety of conditions. As a result, the proposed HM-DA algorithm, which combines the concepts of DA and MBO, was used to handle the optimal resource allocation. The objective function was used to allocate resources, which incorporated constraints such as credibility score, concurrency, price affordability, and job time computation. Furthermore, the credibility score was calculated using execution efficiency, service response rate, access reliability, and the Reboot rate. According to the performance analysis, the proposed HM-DA method was superior to traditional schemes such as DA and MBO at the 80th iteration in scenario 1. Furthermore, the proposed HM-DA approach had a time cost of 0.02 that was better than traditional methods like DA, MBO, PTPN, A-MCDM, and PROMETHEE, which had time costs of 0.03, 0.05, 0.05, 0.13, and 0.13, respectively. Thus the betterment of the proposed HM-DA method was obtained.

### 8. CONCLUSION

Table 8. Resource Allocation for proposed HM-DA Fog Resource and Cloud Resource

<table>
<thead>
<tr>
<th>User</th>
<th>Fog Instances</th>
<th>Cloud Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1  2  3  4  5</td>
<td>1  2  3  4  5</td>
</tr>
<tr>
<td>1</td>
<td>10  9</td>
<td>6  10  10  10</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>6  10  10  10</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>6  8  10</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>6  0  10  10</td>
</tr>
<tr>
<td>5</td>
<td>10  10</td>
<td>6  10  10</td>
</tr>
<tr>
<td>6</td>
<td>10  10  10</td>
<td>10  10</td>
</tr>
<tr>
<td>7</td>
<td>10  10</td>
<td>10  6  10  10</td>
</tr>
<tr>
<td>8</td>
<td>10  9  10  10</td>
<td>10  6</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>10  10  10  10</td>
</tr>
<tr>
<td>10</td>
<td>10  10</td>
<td>6  10</td>
</tr>
<tr>
<td>11</td>
<td>10  10</td>
<td>6  10</td>
</tr>
<tr>
<td>12</td>
<td>10  9  10  10</td>
<td>10  0  10  10  10</td>
</tr>
<tr>
<td>13</td>
<td>10  10</td>
<td>9  10  10</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>10  10  10</td>
</tr>
<tr>
<td>15</td>
<td>10  10</td>
<td>10  6  10</td>
</tr>
<tr>
<td>16</td>
<td>10  10  10</td>
<td>10  0  10  10  10</td>
</tr>
<tr>
<td>17</td>
<td>10</td>
<td>10  10  10</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>10  6  10  10</td>
</tr>
<tr>
<td>19</td>
<td>10</td>
<td>6  8  1  10</td>
</tr>
<tr>
<td>20</td>
<td>10  10</td>
<td>10  6  10</td>
</tr>
</tbody>
</table>

The publisher has waived the Open Access Processing fee for this article.
REFERENCES


Sonti Harika is Research Scholar with 3 years of research experience in the Department of Computer Science and Engineering at Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP. Her research interests include Cloud Computing, Fog Computing. She is author of more than 4 papers in refereed journals and international conferences.

B. Chaitanya Krishna is working as an Associate Professor with 10 years of experience in the Department of Computer Science and Engineering at Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP. His research interests include Software Engineering, Soft Computing, Artificial Intelligence and Machine Learning. He is author of more than 15 papers in refereed journals and international conferences.