Vijayaram and Vasudevan
EURASIP Journal on Advances in Signal Processing (2022) 2022:126
https://doi.org/10.1186/s13634-022-00965-1

EURASIP Journal on Advances in Signal Processing

RESEARCH **Open Access**

Wireless edge device intelligent task offloading in mobile edge computing using hyper-heuristics

B. Vijayaram* and V. Vasudevan

*Correspondence: vijayaramb@gmail.com

Kalasalingam Academy of Research and Education, Kalasalingam University, Krishnankoil, Srivilliputhur, Tamilnadu, India

Abstract

To overcome with the computation limitation of resource-constrained wireless IoT edge devices, providing an efficient task computation offloading and resource allocation in distributed mobile edge computing environment is consider as a challenging and promising solution. Hyper-heuristic in recent times is gaining popularity due to its general applicability of same solution to solve different types of problems. Hyper-heuristic is generally a heuristic method or framework which iteratively evaluates and chooses the best low-level heuristic, to solve different types of problems. In this paper, we try to solve wireless device task offloading in mobile edge computing, which is a non-convex and NP-Hard problem by using a proposed novel Hyper-Heuristic Framework using Stochastic Heuristic Selection (HHFSHS) using Contextual Multi-Armed Bandit (CMAB) with Epsilon-Decreasing strategy, considering two key Quality of Service (QoS) objectives computation time and energy consumption. These multiobjective criteria are modeled as single-objective optimization problem with the goal to minimize latency and energy consumption of wireless devices without losing the pareto optimality. Finally, evaluate its performance by comparing with other individual meta-heuristic algorithms.

Keywords: Mobile edge computing, Hyper-heuristics, Meta-heuristics, Task offloading, Optimization

1 Introduction

In the past decades, we have witnessed many revolutionary wireless mobile devices, wearables, IOT sensors, all these small resource constraint devices need an innovative way of doing effective computation outside its hardware resource and get the results back faster for decision making, with the introduction of 5G, this can be achieved by utilizing mobile edge computing infrastructure. With the adoption of 5G network, the expectation for faster network is inevitable to achieve fast speed, low end-to-end latency, and high reliability. Large-scale applications like high-definition video, virtual reality (VR), augmented reality (AR), IOT wireless sensor-based industrial application, UAV and autonomous vehicles will all eventually generate a large amount of data. This not only puts a strain on the system, but also backhaul. In order to provide better service to the end users for latency-sensitive computational operations, the core network 'Users' devices are placed closer to the network's edge. This will lessen the burden on the server while lowering the delay in the network and processing as computations are done at the mobile edge server placed at proximity to the edge devices.

To address the issues of low-processing capacity and restricted resources in wireless edge devices, the concept of Mobile edge computing (MEC) computation offloading has been presented by the industry. Computation offloading is the process of allocating computationally heavy jobs to a nearby Mobile Edge Server (MES) which has appropriate processing and computing resources. MES is then queried for the derived results. Figure 1 shows a typical Mobile edge computing environment where Wireless Devices (WD) {WD_1, WD_2,...,WD_n} connect to the nearest MES {MES_1, MES_2,...,MES_n}, all these MES are interconnected through fiber optics or high bandwidth wireless channels. Further all MES have fiber optics connection with Centralized Cloud Data Center (CCDC). WDs take stochastic intelligent decision to connect to least utilized nearby MES for task offloading and getting the results back either through the same MES or through the CCDC, it depends on the mobility of the WD. For example, if WD_1 which initially in range of MES_1 starts the offloading to MES1 and being in mobility, it comes to the range of MES3 and retrieves the results from MES3 due to MES collaboration with CCDC.

More research are carried out to intelligently offload the task to mobile edge servers using meta-heuristics, but that does not bring in some generality to the solution space and applicability of algorithm to wide variety of problems, this motivates to

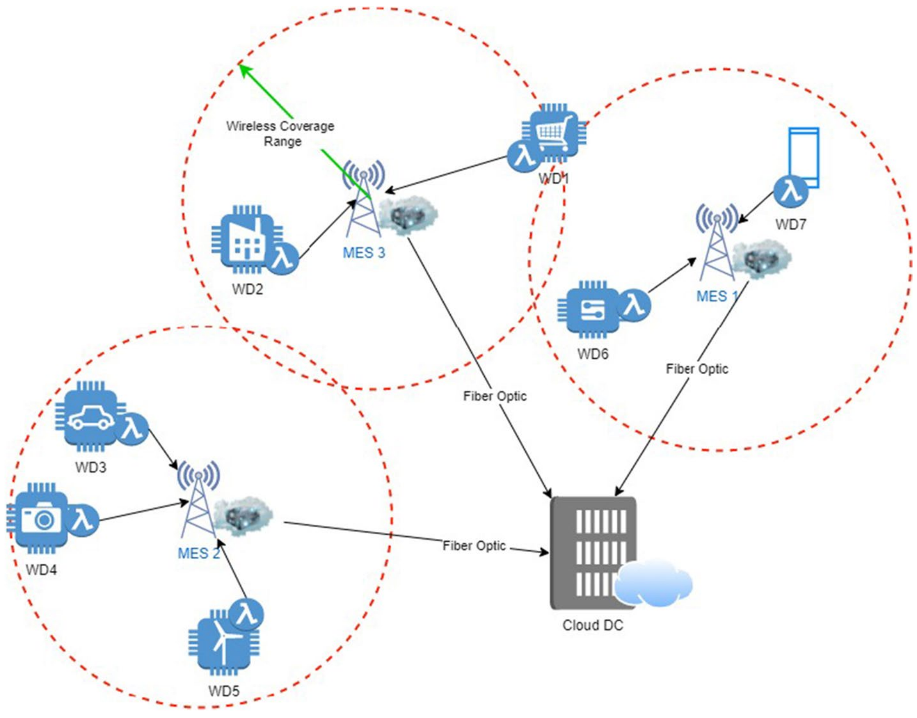


Fig. 1 Mobile edge computing environment

bring in hyper-heuristics [1] to achieve the generality for solving wide variety of problems from different domains. However, the major challenge is in generating heuristics or selecting heuristics automatically.

A hyper-heuristic is an interesting methodology of selecting or generating heuristics to solve different hard computational search problems in an automated way using the same solution in hand. According to the author Burke [2] in his book, he categorized hyper-heuristics into two broad categories as heuristic selection and heuristic generation (refer Fig. 2 – adapted from [2]), this is the first level in our first dimension (the nature of the search space). The second level in this dimension corresponds to the distinction between constructive and local search hyper-heuristics. This categorization deals with the nature of the low-level heuristics used in the hyper-heuristic framework. Construction and perturbation are the terms used to refer these low-level heuristics classes.

Almost all existing hyper-heuristics search contains two stages: heuristics selection and its move acceptance. Figure 3 (adapted from [2]) depicts the high-level element blocks and its interaction of hyper-heuristic using perturbative heuristic selection and Acceptance Criteria move acceptance.

2 Related works

Task computation offloading and resource allocation strategies are interesting problems to be solved in mobile edge computing and vehicular edge network. Several interesting research has been done in this field using Meta-Heuristic, Neural Networks and Fuzzy Logics. Task offloading can be thought as completely offload to MES, or partially offload only memory-intensive processing to the MES and remaining processing is done locally, or entire task is locally computed without offloading to the MES. Most of the task which depends on external data or needs aggregated data from different sources are offloaded to the MES, because of its nature to download lot of data from external sources and do data aggregation which is memory-intensive processing. It is reasonable to offload such tasks to MES completely.

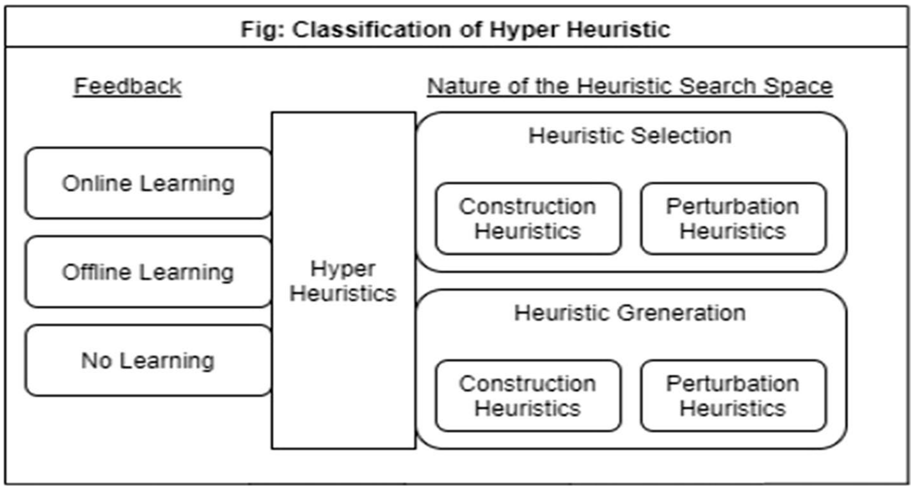


Fig. 2 Hyper-heuristic classification based on feedback and heuristic search space

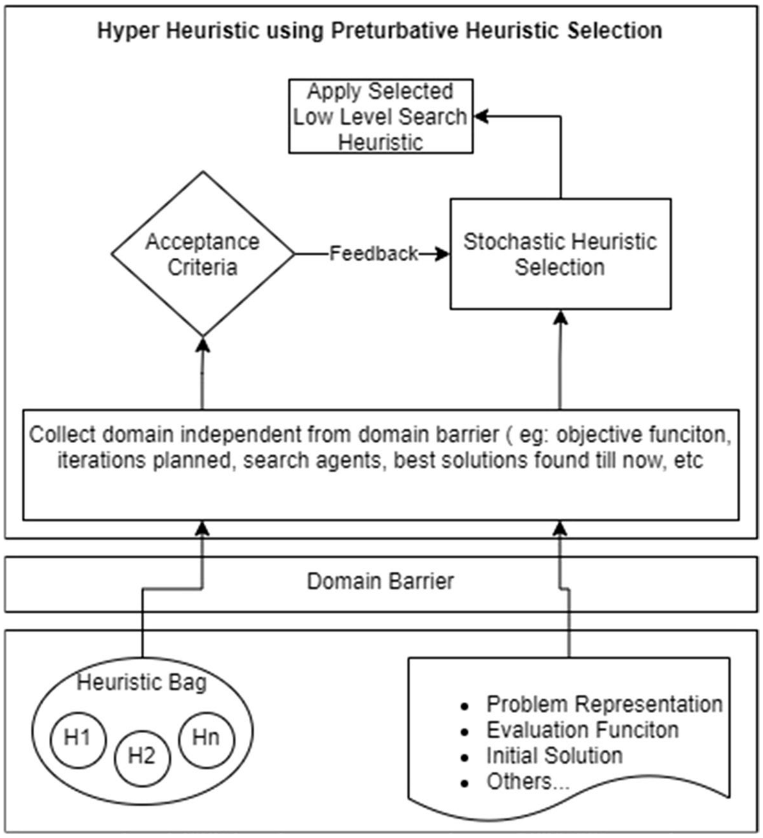

Hyper Heuristic using Preturbative Heuristic Selection
Apply Selected Low Level Search Heuristic
Acceptance Criteria
Feedback
Stochastic Heuristic Selection
Collect domain independent from domain barrier (eg: objective funciton, iterations planned, search agents, best solutions found till now, etc
Domain Barrier
Heuristic Bag
H1
H2
Hn
Problem Representation
Evaluation Funciton
Initial Solution
Others...

Fig. 3 Hyper-heuristic using perturbative heuristic selection

Mareli et al. [3] used bio-inspired meta-heuristic algorithm cuckoo search to optimize task offloading by tweaking the switching parameter. Miao et al. [4] proposed a new intelligent computation offloading-based MEC architecture in combination with artificial intelligence (AI) technology. Their methodology effectively reduces the total task delay with the increasing data and subtasks. Li and Wang et al. [5] solved multiobjective optimization problem by using particle swarm optimization (PSO) with energy constraint MES placement algorithm to arrive at the optimal solution. Huang et al. [6] used whale optimization algorithm (WOA) meta-heuristic to solve the multi-target problem targeting two criteria like task energy consumption and task processing time. Coronel et al. [7] used Meta-Heuristic algorithms with multiple objectives for placement optimization of wireless switches in Electrical Power distribution system. Zakaryia et al. [8] used Queuing network and evolutionary Genetic Algorithm to offload task effectively focusing on minimizing the task response time. Feng et al. [9] used hybrid algorithm GWO-WOA for solving task offloading problem of IOT devices in mobile edge computing environment while considering three optimization criteria. Khan et al. [10] in their paper proposed a task scheduling method based on a hybrid optimization algorithm is presented, which effectively schedules jobs with the least amount of waiting time. Anisetti et al. [11] in their paper proposed an energy-efficient task offloading and transmission power allocation scheme that reduces completion time and energy consumption.

You et al. [12] in their paper proposed a PSO algorithm for task offloading from a resource-constrained wireless edge devices to MES considering energy and low latency multiobjective criteria. Pham et al. [13] in their paper tried to solve resource allocation in wireless network using WOA. Li et al. [14] in their paper tried to fill the gap of task deadline constraints which other existing offloading algorithm failed to do so.

Zhuang et al. [15] in their paper proposed hyper-heuristic algorithm for fog computing to achieve QoS requirements. Alshareef et al. [16] in their paper used multiobjective hyper-heuristic approach to solve multiobjective software module clustering optimization problem by combining and controling three genetic and evolutionary algorithms namely multiobjective genetic algorithm (MOGA), non-dominated sorting genetic algorithm (NSGAII) and strength pareto evolutionary algorithm (SPEA2). Huang et all [17] in their paper explored meta-heuristics energy-efficient computation offloading (EE-CO) approach to minimize energy consumption focusing on delay and security constraints.

3 System method

In this paper, we will discuss about the case of Wireless Devices (WDs) offloading their tasks to nearby MES based on their computation data size. The collection of WDs is represented as $\mathrm{N} = \{1, 2, \ldots, \mathrm{n}\}$. Task computation time needed to finish the task is represented as $\mathrm{TC} = \{\mathrm{tc}_1, \mathrm{tc}_2, .., \mathrm{tc}_\mathrm{n}\}$, and task data size is represented as $\mathrm{TS} = \{\mathrm{ts}_1, \mathrm{ts}_2, .., \mathrm{ts}_\mathrm{n}\}$, where 'i' is a particular wireless device in the collection N. Each wireless device's task is considered as the combination of TC and TS, which can be represented as $task_i = \{tc_i, ts_i\}$. Network Access points or Wifi Routers is used for communication and data transfer between wireless devices and MES. For task computation, wireless device can completely do it locally or completely offload to MES or partially do locally and partially at edge server. This can be represented as a offloading decision set $\mathrm{Y} = \{\mathrm{y}_1, \mathrm{y}_2, .., \mathrm{y}_\mathrm{n}\}$, where y_i ranges between [1, 0], both inclusive. If $\mathrm{y}_\mathrm{i} = 1$, then WD_i completely offload its task to MES and if $\mathrm{y}_\mathrm{i} = 0$, then WD_i does the full task computation within itself. If $\mathrm{y}_\mathrm{i} > 0$ and $\mathrm{y}_\mathrm{i} < 1$, then WD_i offloads $\mathrm{y}_\mathrm{i} \times 100\%$ of tasks to the MES and remaining $(1 - \mathrm{y}_\mathrm{i}) \times 100\%$ of tasks done locally. For offloading decision on edge server, edge server's available processing capacity is considered, and the system method is a combined minimization of overall task processing time / completion time and amount of energy consumed for completing that task. We assert that by employing the hyper-heuristics approach to identify the optimal task offloading decision to do local computing or mobile edge computing or combination of both and thereby latency and energy can be greatly reduced.

3.1 Local computation model

Here we will formulate the model for task execution locally at wireless device. Let us consider td_i^{loc} as the local processing time or time delay and ec_i^{loc} as the energy consumed for processing that task locally. F_i^{loc} is denoted as the maximum available CPU cycles of WD_i.

Let f_i^{loc} represent the current available CPU cycles for the computation task at that moment of the WD_i and tc_i represent the task-required computation time, then the local task processing time or time delay td_i^{loc} to process the task locally by the $\mathrm{WD_i}$ is represented as:

$$td_i^{loc} = \frac{tc_i}{f_i^{loc}} \tag{1}$$

The energy consumption for local task processing is represented as:

$$ec_i^{loc} = C\left(f_i^{loc}\right)^2 tc_i \tag{2}$$

where C is the effective switched capacitance of the device based on its chip architecture.

3.2 Edge computation model

Here we will formulate the model for task execution remotely at MES in this section. Communication rate is considered based on the assumption that mobile devices are connected through the wireless channel. Let B represent the bandwidth of the wireless channel, and let's assume that the bandwidths for WDs are equally allocated for task offloading. Let θ_i is the bandwidth allocated to wireless channel for $\mathrm{WD_i}$. Based on Shannon formula (r_i-channel capacity in bits per sec), the $\mathrm{WD_i}$ communication rate (R_i) is represented as [18]:

$$R_i = r_i\theta_i = B\log\left(1 + \frac{tp_i cg_i}{BN_0}\right)\theta_i \tag{3}$$

where tp_i is transmission power of $\mathrm{WD_i}$ and cg_i is channel gain of $\mathrm{WD_i}$, and N_0 denotes the background channel noise. Now, the total task processing time delay has two parts, one is task transmission time and other is task processing time. $transT_i^o$ is the task transmission time, and it is calculated as:

$$transT_i^o = \frac{ts_i}{R_i} \tag{4}$$

Let F denote the entire available edge server computing resources and f_i^{edge} is the CPU cycles allocated to the $\mathrm{WD_i}$ to complete its task at the MES and $procT_i^{edge}$ represent the $\mathrm{WD_i}$ task computation time needed at the MES and it is calculated as:

$$procT_i^{edge} = \frac{tc_i}{f_i^{edge}} \tag{5}$$

The time required for sending back the result or response from MES can be neglected because the size of the output of the computed data is less. The total time for the $\mathrm{WD_i}$ to process the offloaded task completely using the MES is calculated as:

$$totalT_i^p = transT_i^o + procT_i^{edge} \tag{6}$$

The overall energy consumption ec_i^p is calculated as:

$$ec_i^p = transP_i^o transT_i^o + transP_i^e procT_i^{edge} \tag{7}$$

where $transP_i^o$ is transmission power required to upload the data from $\mathrm{WD_i}$ to the MES through the wireless channel, and $transP_i^e$ is the power required for the $\mathrm{WD_i}$ to wait for the result from MES.

3.3 Problem formulation

While formulating this problem, we considered N wireless devices with varying task computation workloads and tasks dependencies. The decision Y is made based on TC and TS. Here we consider both task processing time delay and task energy consumption. There are several ways to solve multiobjective problems, one way is to optimize the first objective function and try to optimize the second objective function while maintaining the first objective function value intact. The second way is to optimize both objective functions simultaneously. Since these two target constraints are a measure of different metrics, they must be normalized for calculation to avoid biasing.

So, the equation for calculating total time delay becomes:

$$TT = \sum_{i=1}^{n} \frac{\left[\left(1-y_i\right) totalT_i^l + y_i totalT_i^p\right] - T_{\min}}{T_{\max} - T_{\min}} \tag{8}$$

where $T_{\min}$ is the minimum task processing time delay in the set N, and $T_{\max}$ is the maximum task processing time delay in the set N.

And the equation for calculating total energy consumption becomes:

$$EC = \sum_{i=1}^{n} \frac{\left[\left(1-y_i\right) ec_i^l + y_i ec_i^p\right] - E_{\min}}{E_{\max} - E_{\min}} \tag{9}$$

where $E_{\min}$ is the minimum task energy consumption in the set N, and $E_{\max}$ is the maximum task energy consumption in the set N.

Combining both objectives, an improved mathematical formula is arrived to minimize the impact of dimensions and makes the formula controllable using different decision variables. Finally, the target minimization objective function is formulated as:

$$\begin{aligned} Z &= TT + \eta EC \\ &= \sum_{i=1}^{n} \frac{\left[\left(1-y_i\right) totalT_i^l + y_i totalT_i^p\right] - T_{\min}}{T_{\max} - T_{\min}} + \eta \sum_{i=1}^{n} \frac{\left[\left(1-y_i\right) ec_i^l + y_i ec_i^p\right] - E_{\min}}{E_{\max} - E_{\min}} \end{aligned} \tag{10}$$

The coefficient η is used as weight to adjust the optimization objective function results. Here we consider the total time latency target as baseline with coefficient value as 1. The coefficient η of the total energy consumption which ranges from 0.001 to 1 is adjusted based on careful pareto optimality study to get the required weighted normalization for the two targets.

Finally, this optimization problem can be solved using a single target minimization equation given as:

$$
\begin{aligned}
&\min_{y,\theta_i f_i^{edge}} Z\\
&s.t. 0 \le f_i^{edge} \le y_i F, \forall i \in N\\
&\sum_{i=1}^{n} f_i^{edge} \le F, \forall i \in N\\
&0 \le \theta_i \le y_i B, \forall i \in N\\
&\sum_{i=1}^{n} \theta_i \le B, \forall i \in N\\
&y_i \in [0,1], \forall i \in N
\end{aligned}
\tag{11}
$$

The aim of the problem is to minimize the objective function Z, considering the two target constraints namely low energy consumption and low-processing time without losing the pareto optimality.

4 Problem solutions

In this research, three meta-heuristics Grey Wolf Optimizer (GWO), Tabu Search (TS) and Cuckoo Search (CS) are modified and used in low-level hyper-heuristic selection in the proposed technique and tested, let discuss those meta-heuristics overview and their modifications for using it in the proposed algorithm.

4.1 Grey wolf optimizer

Grey wolf optimizer (GWO) is one of nature inspired meta-heuristic swarm intelligence algorithm. This algorithm is unique from other algorithms due to its methodology of adopting social hierarchy and hunting behavior of grey wolves. Seyedali Mirjali proposed GWO [19] in 2014 and proposed multiobjective (MOGWO) [20] in 2016. In recent times, GWO is used in many optimization research aspects. In fact, Xu et al. [21] proposed a fusioned Cuckoo Search with the Improved GWO algorithm to achieve better result. Grey wolves usually dwell in packs with some dominant social hierarchy as shown in Fig. 4 (adapted from [19]). These wolves are represented as 4 main groups namely alpha wolves (α), beta wolves (β), delta wolves (δ), and omega wolves (ω). Wolves which usually lead in prey hunting are called alpha wolves; wolves which supports

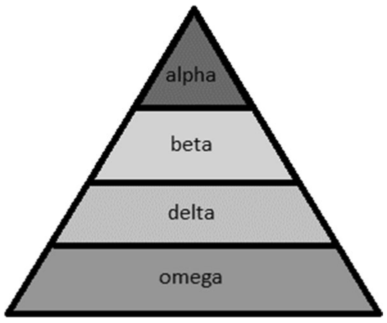


Fig. 4 Social Hierarchy of Grey Wolfs

helping alpha wolves are called beta wolves; wolves which helps in guarding the territory boundaries and does the whistle blowing job are called delta wolves; and wolves which are lazy and does not actively take part in hunting but only interested in eating the leftover food are called omega wolves, which is usually dominated by other top category wolves.

In GWO algorithm, the final optimized fit solution is represented as alpha, then the second less optimized fit solution is represented as beta and the third least optimized fit solution is represented as delta. All the left-over trivial solutions are represented as omega. GWO has 3 stages in the algorithm: encircling, hunting, or attacking, and searching. The positions of the wolves during the encircling stage, is updated by [19]:

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(i) - \vec{X}(i)\right| \tag{12}$$

where 'i' is the iteration index, $\overrightarrow{X}_p$ represent the position vector of prey, and $\overrightarrow{X}$ represent the position vector of wolves. $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors, calculated by the below equations:

$$\vec{A} = 2 \cdot \vec{a} \cdot \vec{r}_1 - \vec{a} \tag{13}$$

$$\vec{C} = 2 \cdot \overrightarrow{r_2} \tag{14}$$

Here the variable $\overrightarrow{a}$ is nonlinearly decreased for ¾ of the iteration and linearly decreased from 2 to 0 for remaining iterations. This is done to support better exploration and exploitation, respectively, and variables $\overrightarrow{r}_1$ and $\overrightarrow{r_2}$ are random absolute vectors in range [0, 1].

During the hunting stage, additional weight coefficient 0.01 is considered for alpha position as alpha wolfs are closer to the prey, this value can be tweaked based on the convergence behavior and final positions of the wolves are updated by equation [17]:

$$\overrightarrow{D_\alpha} = \left|\vec{C}_1 \cdot \overrightarrow{X_\alpha} - \vec{X}\right|, \overrightarrow{D_\beta} = \left|\vec{C}_2 \cdot \overrightarrow{X_\beta} - \vec{X}\right|, \overrightarrow{D_\delta} = \left|\overrightarrow{C_3} \cdot \overrightarrow{X_\delta} - \vec{X}\right| \tag{15}$$

$$\vec{X}_1 = \vec{X}_\alpha - \overrightarrow{A_1} \cdot \overrightarrow{D_\alpha}, \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} \cdot \overrightarrow{D_\beta}, \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} \cdot \overrightarrow{D_\delta} \tag{16}$$

$$\vec{X}(t+1) = \frac{\overrightarrow{(X_1 * 0.01)} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{17}$$

GWO has good exploitation and exploration ability, which helps in avoiding local minimum trap.

4.2 Tabu search

Tabu Search (TS) is one of meta-heuristic local search algorithm specialized in optimizing the heuristics model parameters. Some of the local search heuristic methods have the pitfall to stuck in local minima. TS helps to overcome this problem by enhancing the local search exploration phase by prohibiting already visited solutions also known as Tabu. TS does sometimes deterministically accept trivial solutions to avoid local minima convergence. The steps involved in TS algorithm are given below,

Algorithm: Step 1: Start with any random best acceptable solution, say bS = S_0.

Step 2: Generate neighboring random solutions N(bS) based on the current best solution bS. From N(bS), the solutions that are in the Tabu List are removed except for the solutions that fit the Aspiration Criteria. This solution will become the new N(bS).

$$bS' \in N(bS) = \left\{N(bS) - T(bS)\right\} + A(bS) \tag{18}$$

Step 3: Choose the best solution out of $N(bS)$ and label this new solution $bS^{'}$. If the solution $bS^{'}$ is better than the current best solution, update the current best solution. After, regardless of if $bS^{'}$ is better than bS, we update bS to be $bS^{'}$.

Step 4: Update the Tabu List $T(bS)$ by removing all moves that are expired past the Tabu Tenure and add the new move s' to the Tabu List. Additionally, update the set of solutions that fit the Aspiration Criteria $A(bS)$.

Step 5: Search stops if the termination criteria is met or else it will move onto the next iteration. Termination Criteria is used here is max number of iterations.

4.3 Cuckoo search

Cuckoo Search (CS) algorithm is one of bio-inspired meta-heuristic algorithm developed based on reproduction behavior of cuckoo birds [14]. Potential solutions are associated with cuckoo eggs in CS. Cuckoos birds usually lay their eggs in other's nests with the hope of their off springs being raised by other. On a random probability say 25%, when the host cuckoos discover those foreign eggs in their nests, some of the foreign eggs are thrown out of the nest or cuckoos will completely discard that entire nest. The CS algorithm consist of three basic rules as follows:

- Eggs are laid in random nests by cuckoo bird.
- Best nests which contain best quality eggs are selected and carried forward to next generation.
- Host cuckoo will identify a foreign egg with a probability pa ϵ [0,1] from a set of random nests. If foreign egg is found, the host cuckoo can either throw the foreign egg away or completely abandon the whole nest and build a new nest elsewhere.

During the iteration, based on the above three rules, the new position of cuckoo nests is updated by.

$$x_j(t+1) = x_j(t) + \alpha \oplus Levy(\lambda), i = 1, 2, \ldots, n \tag{19}$$

Here the product $\oplus$ representative entry-wise multiplication. $x_j(t+1)$ denotes new solutions for cuckoo 'i', $x_j(t)$ denotes the current solutions. The step size is controlled by $\alpha > 0$, Let's assume its value as 1. The levy-flight is provided by following Mantegna's algorithm.

In Mantegna's algorithm, the step length s is calculated by

$$s = \frac{u}{|v|^{1/\beta}} \tag{20}$$

where u and v values are arrived based on normal distributions. That is

$$u \sim N\left(0, \sigma_u^2\right), v \sim N\left(0, \sigma_v^2\right) \tag{21}$$

were,

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \sigma_v = 1 \tag{22}$$

This distribution obeys the expected Levy's distribution for $|s| >= |s_0|$ where s_0 is the smallest step. Its value can be carefully chosen between 0.01 and 1.

4.4 Proposed hyper-heuristic framework using stochastic heuristic selection (HHFSHS)

Based on the Heuristic framework depicted in Fig. 3, a novel stochastic heuristic selection based on online learning acceptance criteria feedback is proposed which uses couple of well-known meta-heuristics like GWO, CS, TS as part of the low-level heuristic search with certain careful parameter tweaks to improve exploration and exploitation behaviors. The reason behind choosing these three meta-heuristics is based on the performance and behavior to tackle local minima trap and achieve optimum convergence in most of the problem space. Let's discuss about those modification on meta-heuristics below.

The GWO algorithm updates the wolve position just by averaging out the alpha, beta and delta positions during each iteration using Eq. (17), this may lead to local minimum trap or slow convergence as alpha position progress slows down due average calculation. This is clearly seen in the convergence comparison in Fig. 5. To mitigate this, the equation is modified to add a fixed weight to alpha wolve position to emphasis the importance of alpha wolves leading the group. The encircling and attacking phases of iteration is called as exploration and exploitation phases, respectively, instead of having the linearly decreasing value, this algorithm is modified to use nonlinear function for ¾ of the max iteration and linear function for ¼ of the max iteration to support exploration and exploitation stochastically.

The TS algorithm is used to store all previous search best positions and avoids the search agents to search again in previously searched position, thereby improving the performance.

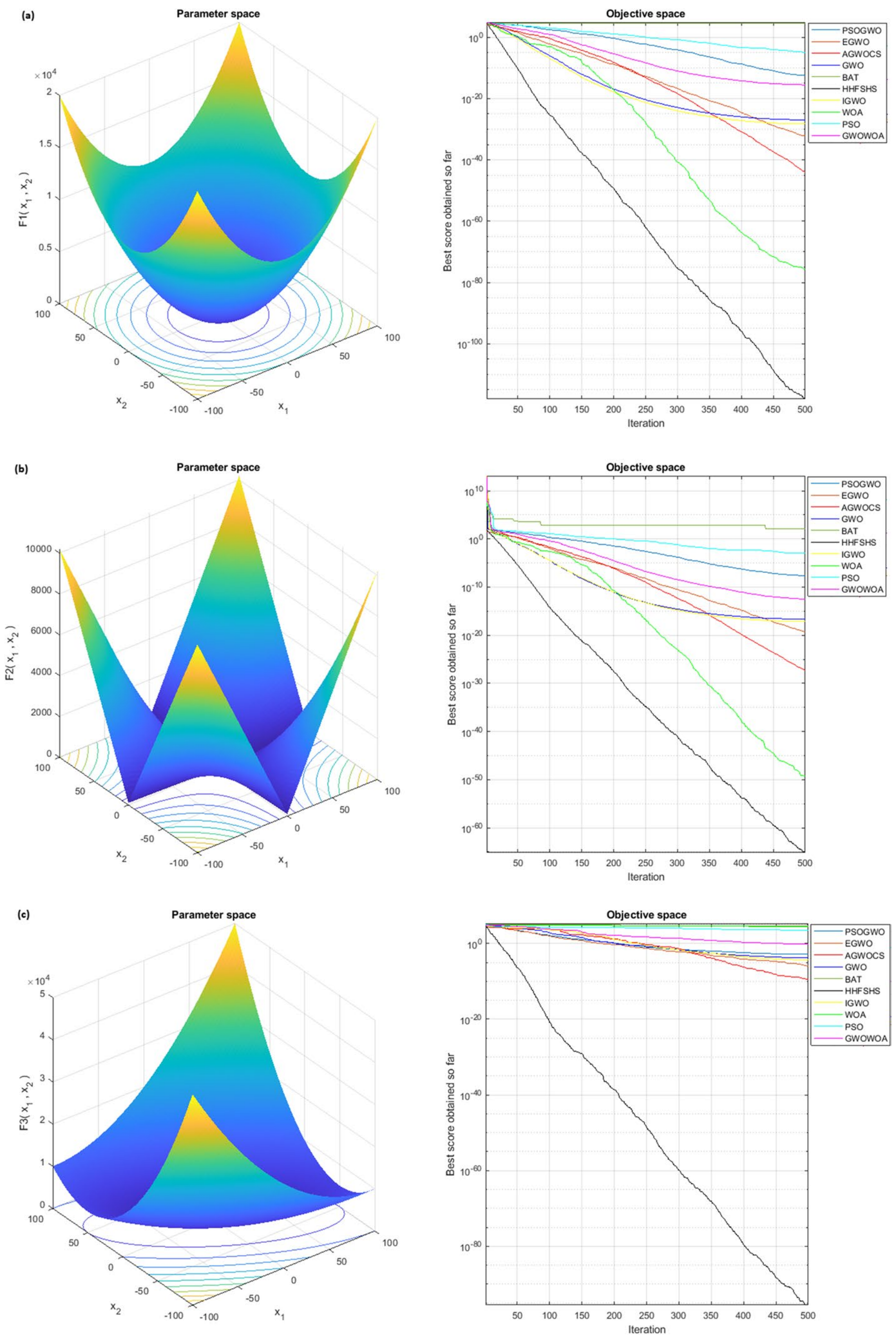


Fig. 5 HHFSHS algorithm Convergence comparison with other meta-heuristics

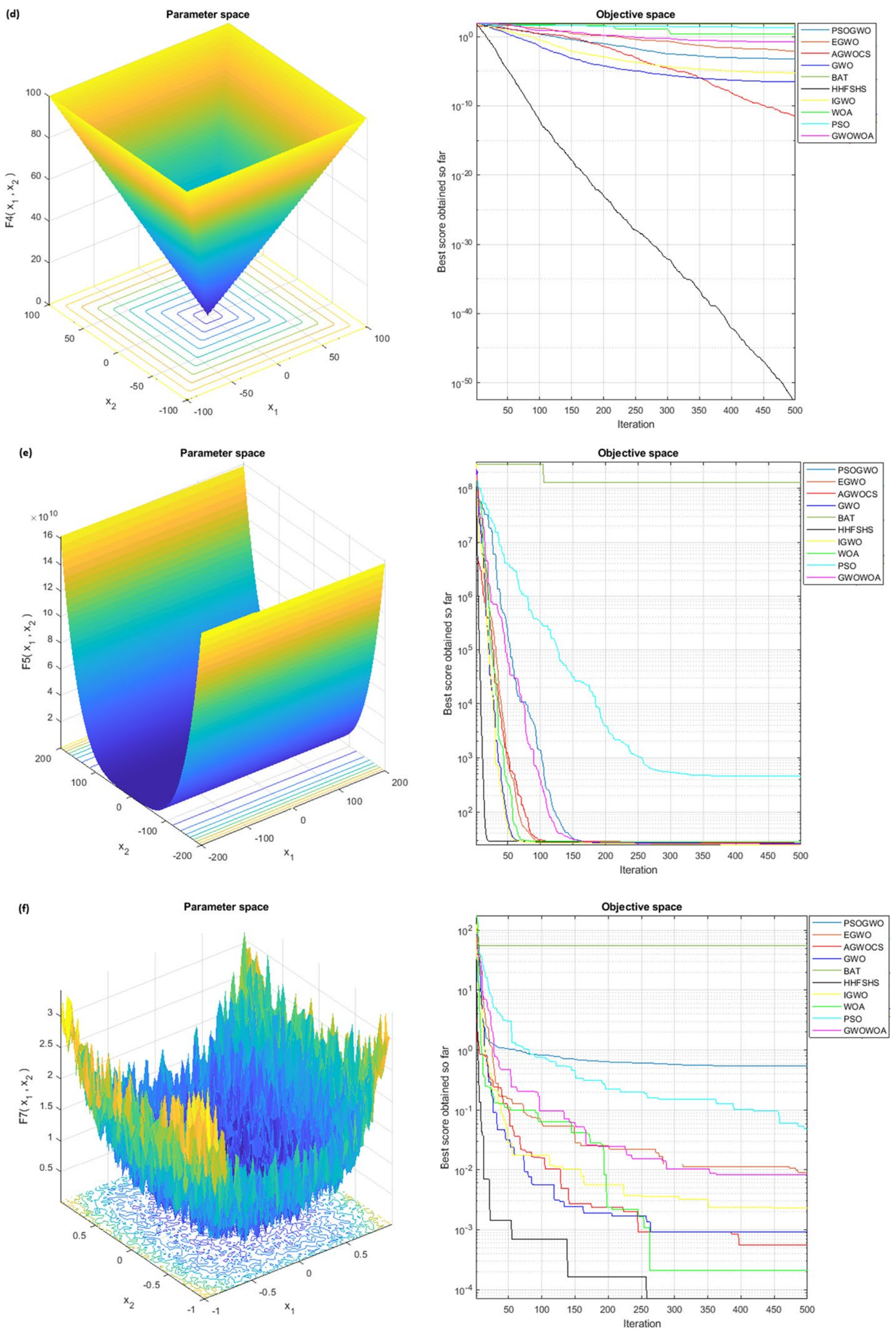
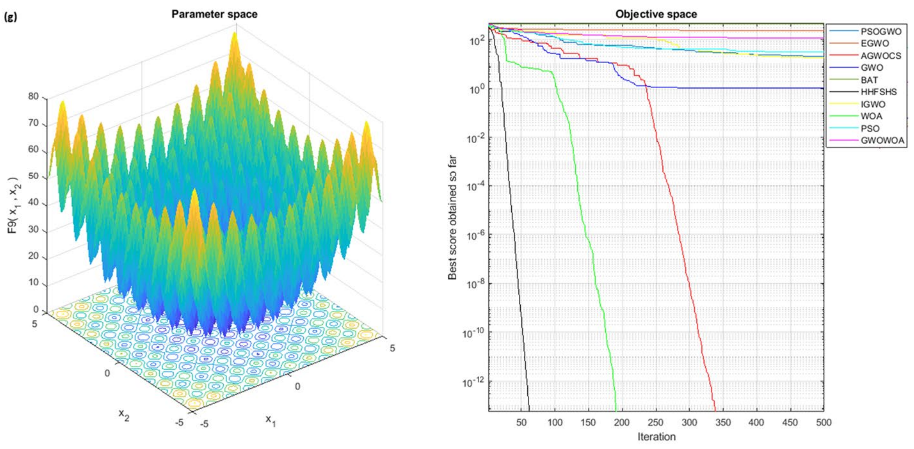
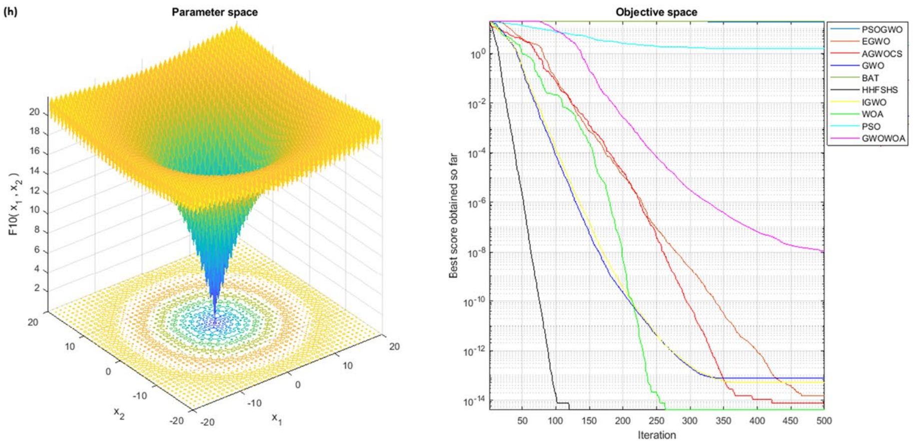


Fig. 5 continued

(g)



(h)



(i)

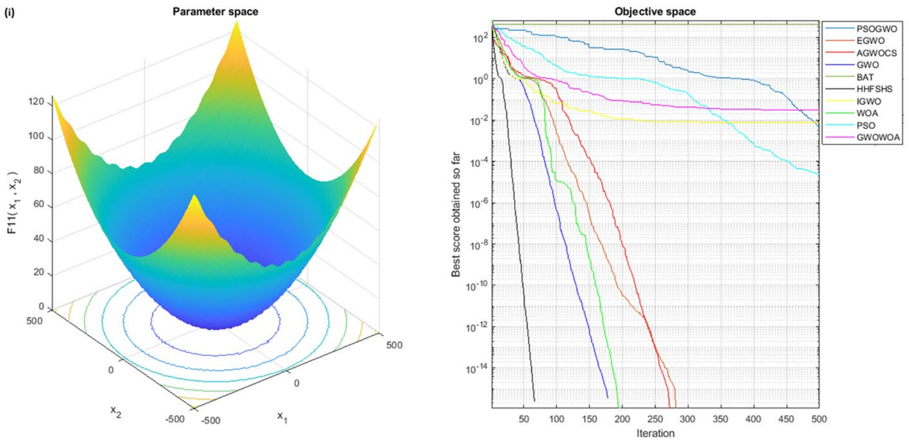


Fig. 5 continued

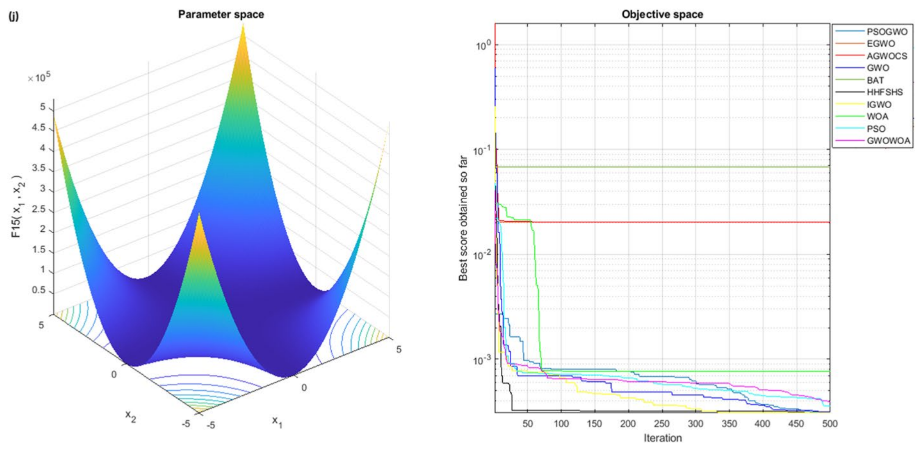


Fig. 5 continued

The CS algorithm uses cuckoo random walk and levy-flights to update its nest position using a fixed probability index say 0.25. Random walk is usually walking straight for a while and take 90 degrees turn and continue walking, this will bring high randomness which will support in local minima avoidance and boost better exploration capability.

Using these modified meta-heuristic algorithms in low-level heuristic search as part of the hyper-heuristic framework, a novel hyper-heuristic based on perturbation low-level heuristic selection using Contextual Multi-Armed Bandit (CMAB) Epsilon-Decreasing strategy is formulated with move acceptance criteria formulated as depicted in Eq. (23), where 0 represent criteria not satisfied or loss and 1 represent criteria satisfied or high profit. Epsilon-Decreasing strategy is used to favor exploration initially and gradually favor exploitation later by starting with higher $\in$ value and decrease over time. Consideration is given such that the rate of decrease shouldn't be too quick.

During the algorithm iteration, last best convergence is compared with n last convergences, if there is improvement in the best convergence value, then the chosen low-level heuristic from heuristic Bag is continued for further iterations, in case there is no improvement in the best convergence value, then a stochastic heuristic selection is picked using CMAB with Epsilon-Decreasing strategy with an additional penalization on iteration count for which there is no improvement and continued for further iteration. This process is repeated till the end of the iteration.

$$\text{Acceptance Criteria (AC)} = \{0, 1\}, \text{s.t}, C_l - C_{l-n} < C_t \tag{23}$$

where C_l is last best convergence, C_{l-n} is last 'n' convergences, C_t is convergence tolerance ($1e^{-5}$).

Based on the above hyper-heuristic framework and formulation (refer Fig. 3), a novel Hyper-Heuristic Framework using Stochastic Heuristic Selection (HHFSHS) is proposed as below algorithm.

Algorithm: Pseudocode of proposed HHFSHS algorithm

```
        Initialize the Heuristic Bag HB_i (i=1, 2,..m) which contain couple of local search heuristics
(GWO, WOA, CS,TS)
        Initialize the search space population SPi (i = 1, 2, ..., n)
        Max_Iter = 500
        Convergence Iteration Count (Ci) = 5
        Convergence Tolerance (Ct) = 1e-5
        Initialize the Acceptance Criteria (AC) using equation (23)

        Hc = Randomly pick 1 Heuristic from HB_i; using CMAB Epsilon-Decreasing strategy

        Calculate the fitness value for all search agents in the population SPi
        Select Xα (Best Solution) from solutions according to the fitness values
        ConvergenceGraph.Add(Xα);

        t=1
        while t < Max_Iter do

            SPi = Update the search agents positions based on Current Heuristic (Hc)
            Calculate the fitness value of each search agent in the population SPi
            Select Xα (Best Solution) from solutions according to the fitness values

            if (AC is not met or no significant convergence)
                Hc = Randomly pick 1 Heuristic from HB_i; using CMAB Epsilon-Decreasing
                    strategy
                t = t- Ci; //penalize the iteration
                Remove 'Ci' number of recent no significance convergence solutions from
                        ConvergenceGraph // Discard no significant solution
            else
                ConvergenceGraph.Add(Xα);
            end
            t = t + 1
        end while
Return Xα
```

Table 1 Uni-model test functions

S. No	Function	Dim	Range	$f_{\min}$
1	$f_1(x) = \sum_{a-1}^{n} x_1^2$	30	[− 100,100]	0
2	$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[− 10,10]	0
3	$F(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_i^2 \right)^2$	30	[− 100,100]	0
4	$F_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	30	[− 100,100]	0
5	$F_5(x) = \sum_{i=1}^{n} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right]$	30	[− 30,30]	0
6	$F_6(x) = \sum_{i=1}^{n} (x_i + 0.5)^2$	30	[− 100,100]	0
7	$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[− 1.28,1.28]	0

Table 2 Multi-model test functions

S. No	Function	Dim	Range	$f_{\min}$	
1	$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	[− 500,500]	− 418.9829 × Dim	
2	$F_9(x) = -\sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	[− 5.12,5.12]	0	
3	$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[− 32,32]	0	
4	$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[− 600,600]	0	
5	$F_{12}(x) = \left\lvert \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} \right.$ $+ \left\lvert + y_i = 1 + \frac{x_i + 1}{4} u(x_i, a, k, m) = \left\{ \begin{array}{c} (x_i - a)^m x_i \\ x_i \\ (-x_i - a)^m x_i \end{array} \right. \right.$		30	[− 50,50]	0
6	$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(2\pi x_i + 1)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[− 50,50]	0	
7	$F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	[− 65,65]	1	
8	$F_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_i(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	[− 5,5]	0.00030	
9	$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	2	[− 5,5]	− 1.0316	

5 Experimental analysis and results

5.1 Simulation setup

Bio Inspired Heuristic algorithms like PSOGWO, EGWO, Augmented GWOCS (AGWOCS), GWO, BAT, Improved GWO (IGWO), WOA, PSO, GWOWOA were compared with HHFSHS and investigated in MATLAB version R2021b. The test environment was Dell laptop with the following specifications: RAM of 10 GB, CPU is Intel® Core™ i5-2540 M CPU @ 2.60 GHz and 64-bit windows 10 Pro operating system.

5.2 Test functions

The unimodal and multi-model benchmark test functions used to validate the performance of each Optimization algorithm are tabulated in Tables 1 and 2.

These optimization test functions complexity quality is defined by the number of peaks encountered in the function landscape. These peaks can negatively impact the optimization process when the optimization algorithm gets stuck in between the peaks. Couple of test function results are shown below for discussion on performance of the proposed algorithm.

5.3 Results and discussion

Performance evaluation scenarios is setup considering couple of MES in the wireless access area and couple of WDs (N = 30) are distributed around the MES coverage region. Each WD, with its own computation task, task's data size and task's required CPU cycles are randomly generated, specifically $ts_i \sim N$ (0,20) MB and $tc_i \sim N$ (500, 100) cycles/bit. The total available CPU cycles of the mobile edge servers is F = 30 GHz, and the allocated CPU cycles of the WD_i is set to {0.5,0.6, …1.0} GHz randomly. The transmission power $transP_i^o$ is set as 100 mW, and the power required to wait for the result $transP_i^e$ is set as 10 mW. With these setting, simulation experiment is done to evaluate the proposed algorithm. The aim of the proposed algorithm is to get faster and better convergence and thereby reducing the overall processing time and energy consumption in offloading tasks. As the multiobjective problem (MOP) is normalized and devised as a single objective minimization problem, we will consider the performance from the perspectives of convergence and stability. Also, couple of standard meta-heuristic methods also included as comparisons. In our experiment, nine other meta-heuristics are evaluated and compared with HHFSHH

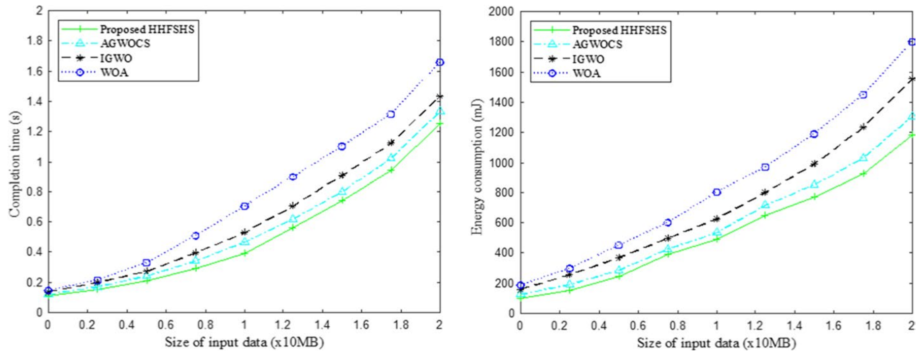


Fig. 6 Result comparison

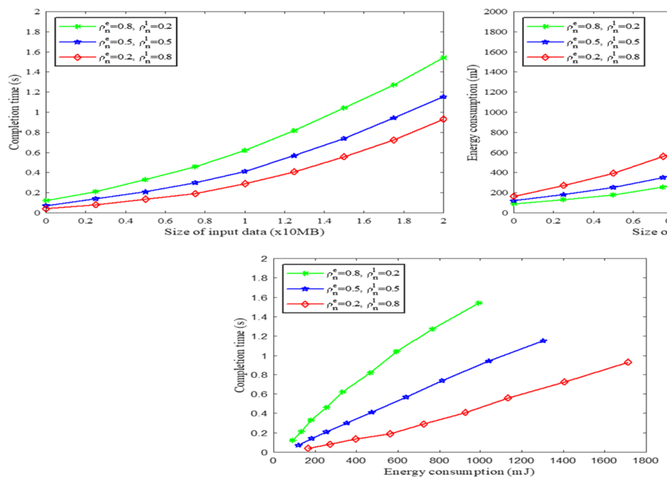


Fig. 7 Performance of HHFSHS algorithm with relative percentage tasks

Table 3 Statistical analysis on latency and energy consumption

Method	Latency (s)			Energy consumption (mJ)		
	Mean	SD	SE of Mean (SEM)	Mean	SD	SE of Mean (SEM)
HHFSHS	0.386	0.392	0.131	417.629	368.285	122.761
AGWOCS	0.431	0.414	0.138	476.381	401.383	133.794
IGWO	0.488	0.446	0.138	576.840	468.111	156.037
WOA	0.584	0.522	0.174	692.115	543.136	181.045

technique with 3 low-level heuristics GWO, CS, TS tweaked and used in heuristic bag of our proposed heuristic framework. There is no restriction on choice of low-level heuristics for using in this framework. Some of the key parameters chosen for evaluation are.

- Optimization target function Z. Algorithm is effective if it can arrive at the lowest minimum value of Z function.
- Overall task processing time. Computation task offloading in mobile edge environment is delay sensitive and offloading decisions should be taken quickly, otherwise it will fail in its purpose.
- Overall task energy consumption, as wireless devices typically IoT devices have less power resource, so it must be effectively used.
- Stability of results in multiple iterations with the same inputs. Meta-heuristic algorithms have uncertainty due to the facts of its techniques to arrive at the global minimum, which sometimes stuck at the local optima. The results so obtained is also affected by this kind of uncertainty. However, these uncertainty in the results should be minimized as low as possible. The algorithm result may vary

for each iteration with same input, as the stochastic selection of low-level heuristic is based on the CMAB Epsilon-Decreasing strategy and move acceptance using acceptance criteria evaluation

- Finally, the convergence curves of different heuristics methodology are investigated. The values of the Z function obtained considering four different characteristics of applications like Argument Reality, HealthCare, Compute Intensive and Infotainment, with 3 different workload distributions.

From the result (refer Fig. 5), specifically in Fig. 5a-d, it is evident that none of the heuristic converged properly to global minima, but the proposed HHFSHS heuristic has managed to avoid local minima and converged faster to global minima with lesser processing time. It also seen in most of the multi-model test functions the proposed algorithm HHFSHS has converged well compared to other heuristics due to the fact the hyper-heuristic scholastically selects the lower-level search heuristic according to the CMAB Epsilon-Decreasing strategy and move acceptance using acceptance criteria evaluation. HHFSHS is performed well in terms of convergence in uncertain problem space due to its adaptable low-level heuristic selection dynamically based on the CMAB Epsilon-Decreasing strategy. AGWOCS a hybrid meta-heuristic algorithm relatively performed well close to the proposed algorithm HHFSHS and performed well than IGWO and WOA as it overcomes the disadvantage of GWO local minima trap behavior with the advantage of CS to improve global search and avoid local minima. It also can be derived from result that the lowest function values of the different heuristics are in the order HHFSHS < AGWOCS < IGWO < WOA.

Based on the simulation result (refer Fig. 6), it is evidence that the completion time of offloading task increases with the size of the data increases and energy consumption increases with the size of the data increases, as more data need to be transferred from wireless device to edge server and get the processed data back which obviously increase the waiting time thereby consuming more energy in the wireless device. Our method HHFSHS is the suboptimal completion time and is suboptimal in energy consumption when the wireless device count is 30. Other algorithms like AGWOCS and IGWO and WOA relatively performed well when compared to HHFSHS, while other heuristics like PSO and BAT algorithms has not performed well so not considered for comparison here. From the result, it is evident that the task processing completion time and energy consumption is less for HHFSHS, due to its faster convergence to find the optimal solution in problem space compared to other Heuristics.

Experiment is done considering four different characteristics of applications like Argument Reality, HealthCare, Compute Intensive and Infotainment, with 3 different workload distribution (i) 80% task offloaded to edge and 20% task in local computation; (ii) 50% task offloaded to edge and 50% in local computation; and finally, (iii) 20% task offloaded to edge and 80% task in local computation. Results based on this setup is shown in Fig. 7. It is very clear that as more percentage of tasks are computed locally, it takes less processing time as there is no involvement of data transfer in the network to MES and no wait time. It also very clear that as more percentage of tasks are computed locally, it takes more energy, and it takes less energy when it offloads major percentage of task to edge server and get the job done.

Based on Statistical Analysis (ref Table 3), it is evidence that proposed HHFSHS has low latency and energy consumption compared other meta-heuristics. Standard Error of Mean (SEM) is less for HHFSHS compared to other methods, which means it has smoother and consistent behavior in task offloading for different population samples. All these experimental result analyses provide evidence that hyper-heuristic-based technique has overcome the disadvantage of individual heuristic by adaptively switching to better lower-level heuristic and adapting generically for different problem domains. By having a better convergence, the proposed HHFSHS algorithm able to performance better with reduced energy consumption and faster processing compared to other individual meta-heuristics algorithms.

6 Conclusion and future work

In this work, we analyzed a Hyper-Heuristic Framework using Stochastic Heuristic Selection (HHFSHS) for computation task offloading model with the goal to minimize the latency and energy consumption optimization in MEC. Then, the formulated model is normalized to aid in improving the model even for multi-dimensions. The goal of the formulated model is to arrive at the minimum value. The proposed HHFSHS algorithm has been applied to solve the optimization problem. The experiment shows better results of HHFSHS approach compared to other heuristics algorithms. However, the algorithm proposed still can have better feedback for selection of low-level heuristics. Since different lower-level heuristic are selected stochastically using CMAB Epsilon-Decreasing strategy, the results may vary based on the Epsilon value selection, Epsilon decreasing rate and acceptance criteria resulting in slightly varying result for each run.

Future work will be based on the proposed algorithm going to experiment with other Heuristic local search algorithms with online feedback mechanism using Deep Reinforcement Learning (DRL) and evaluate in Vehicular Edge Computing consider mobility as an additional parameter, since mobility is one of the key features affecting the task offloading and resource allocation in Vehicular Edge Network.

Abbreviations

MEC	Mobile edge computing
MES	Mobile edge server
WD	Wireless device
DRL	Deep reinforcement learning
QoS	Quality of service
AR	Augment reality
VR	Virtual reality
GWO	Grey wolf optimizer
CS	Cuckoo search
TS	Tabu search
WOA	Whale optimization algorithm
AGWOCS	Augmented whale optimization cuckoo search
IGWO	Improved grey wolf optimizer
AC	Acceptance criteria
HHFSHS	Hyper-heuristic framework using stochastic heuristic selection
CMAB	Contextual multi-armed bandit

Acknowledgements
The authors appreciate the support from guide and professors at Kalasalingam Academy of Research and Education, Kalasalingam University, Krishnan Koil, TN, India.

Author contributions
Conceptualization: B. Vijayaram; Formal Analysis: B. Vijayaram; Methodology: B. Vijayaram; Investigation: V. Vasudevan; Supervision: V. Vasudevan. All authors read and approved the final manuscript.

Authors' Information
Mr. B. Vijayaram—Correspondence Author (Primary Author) is a Research Scholar in Kalasalingam Academy of Research and Education, Krishnan koil, Tamilnadu, India. Has more than 15 years of product-based industry experience in security, industrial automation and medical radiology domain.
Sr. Prof Dr V. Vasudevan—Secondary Author is working as Registrar in Kalasalingam Academy of Research and Education, Krishnan koil, Tamilnadu, India. Has a Maths PhD, headed MCA dept of Kalasalingam University from 1997 to 2003. Then headed the IT dept for over ten years. During the same period, was the chief superintend of university exams for 6 years, Dean hostels for four years, Dean admissions & dean placements for three years from 2011 to 2014. Then currently working as a Registrar from 2013. So for 25 students completed PhD under my guidance and has the credit 67 international publications. I have 25 years of experience in teaching and research experience.

Funding
Not applicable.

Availability of data and materials
Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Consent for Publication
No individual human details, images or videos are used during the current study.

Competing Interests
The authors declare that they have no competing interests.

Received: 12 August 2022 Accepted: 16 December 2022
Published online: 30 December 2022

References

1. E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, S. Schulenburg, Hyper-heuristics: an emerging direction in modern search technology, in *Handbook of metaheuristics*. ed. by F. Glover, G.A. Kochenberger (Springer, Boston, 2003), pp.457–474
2. E. Burke, M.R. Hyde, G. Kendall, G. Ochoa, E. Özcan, A classification of hyper-heuristic approaches, pp. 449–468 (2010)
3. M. Mareli, B. Twala, An adaptive Cuckoo search algorithm for optimisation. Appl. Comput. Inform. **14**, 107–115 (2018)
4. Y. Miao, G. Wu, M. Li, A. Ghoneim, M. Al-Rakhami, M.S. Hossain, Intelligent task prediction and computation offloading based on mobile-edge cloud computing. Futur. Gener. Comput. Syst. **102**, 925–931 (2020)
5. Y. Li and S. Wang, An energy-aware edge server placement algorithm in mobile edge computing, San Francisco, CA, USA, (2018)
6. M. Huang, Q. Zhai, Y. Chen, S. Feng, F. Shu, Multi-objective whale optimization algorithm for computation offloading optimization in mobile edge computing. Sensors **21** (2021)
7. E. Coronel, B. Baran, P. Gardel, Optimal placement of remote controlled switches in electric power distribution systems with a meta-heuristic approach. IEEE Latin Am. Trans. **20**(4), 590–598 (2022)
8. S.A. Zakaryia, S.A. Ahmed, M.K. Hussein, Evolutionary offloading in an edge environment. Egypt. Inf. J. **22**, 257–267 (2021)
9. S. Feng, Y. Chen, Q. Zhai, M. Huang, F. Shu, Optimizing computation offloading strategy in mobile edge computing based on swarm intelligence algorithms. EURASIP J. Adv. Signal Process. **2021**, 36 (2021)
10. M.S.A. Khan, R. Santhosh, Task scheduling in cloud computing using hybrid optimization algorithm. Soft Comput. (2021)
11. M. Anisetti, X. Gu, L. Jin, N. Zhao, G. Zhang, Energy-efficient computation offloading and transmit power allocation scheme for mobile edge computing. Mob. Inf. Syst. **2019**, 3613250 (2019)
12. Q. You, B. Tang, Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things. J. Cloud Comput. **10**, 41 (2021)
13. Q.-V. Pham, S. Mirjalili, N. Kumar, M. Alazab, W.-J. Hwang, Whale optimization algorithm with applications to resource allocation in wireless networks. IEEE Trans. Veh. Technol. **69**, 4285–4297 (2020)
14. Z. Li, V. Chang, J. Ge, L. Pan, H. Hu, B. Huang, Energy-aware task offloading with deadline constraint in mobile edge computing. EURASIP J. Wirel. Commun. Netw. **2021**, 56 (2021)
15. Y. Zhuang, H. Zhou, A Hyper-Heuristic resource allocation algorithm for fog computing. In *Proceedings of the 2020 the 4th International Conference on Innovation in Artificial Intelligence,* (2020)
16. H. Alshareef and M. Maashi, Application of Multi-Objective Hyper-Heuristics to Solve The Multi-Objective Software Module Clustering Problem. Appl. Sci. **12** (2022).
17. X. Huang, Y. Yang and X. Wu, A Meta-Heuristic Computation Offloading Strategy for IoT Applications in an Edge-Cloud Framework, in *Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control*, New York (2019).

18. X. Deng, Z. Sun, D. Li, J. Luo, S. Wan, User-centric computation offloading for edge computing. IEEE Int. Things J. **8**, 12559–12568 (2021)
19. S.M. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
20. S. Mirjalili, S. Saremi, S.M. Mirjalili, L.S. Coelho, Multi-objective grey wolf optimizer. Exp. Syst. Appl. **47**, 106–119 (2016)
21. H. Xu, X. Liu, J. Su, An improved grey wolf optimizer algorithm integrated with Cuckoo Search, in *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)* (2017)