# ARGENT: Energy-Aware Scheduling in Edge Computing Using Energy Valley Optimizer

Mehrab Toghani[1], Sareh Maleki[1], Abolfazl Younesi[1], Sepideh Safari[2], Shaahin Hessabi[1]
[1]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[2]School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
Emails: {mehrab.toghani79, sareh.maleki78, abolfazl.yunesi, hessabi}@sharif.edu, {Sepideh.safari}@ipm.ir

***Abstract*—This paper presents a novel approach for energy-aware task scheduling in heterogeneous edge computing environments. The Energy Valley Optimizer (EVO) is exploited for the optimal allocation of tasks to edge devices with limited resources. The primary objective of this algorithm is to minimize energy consumption and task completion time while achieving optimal resource utilization. The EVO updates task positions in the search space using our proposed alpha, beta, and gamma reduction processes, avoiding local optima. Our evaluations on both small and large scales with real-world datasets indicate that our approach reduces energy consumption by an average of 14.42%, reaching up to 20.96% savings, and decreases task completion time by an average of 15.98%, with a peak improvement of 24.42%, in comparison to state-of-the-art methods. Additionally, EVO shows a 5.28% improvement in success ratio, which means it is robust in meeting task deadlines.**

***Keywords*—Scheduling, Edge Computing, Energy Valley Optimizer, Energy Consumption, Multi-Objective Optimization**

## I. Introduction

Edge computing has emerged as a critical paradigm in modern distributed computing systems, enabling data processing close to the source of generation, which in turn reduces latency and bandwidth consumption [1]. With the increase in IoT devices and applications with large computational data, effective task allocation and scheduling in edge environments have become major challenges. One of the most significant issues in edge computing is the high energy consumption associated with executing tasks on resource-constrained edge devices [2][3]. This challenge necessitates the development of energy-efficient task scheduling strategies to optimize and sustain system performance.

Task scheduling in edge computing environments is a complex, multi-objective problem that involves determining the optimal distribution of tasks among heterogeneous edge nodes while minimizing energy consumption and latency, and maximizing resource utilization. Traditional scheduling approaches often fail to adequately meet these conflicting objectives, particularly in dynamic and uncertain environments. As a result, metaheuristic algorithms have gained importance due to their ability to find near-optimal solutions to complex scheduling problems within reasonable time frames [3][4]. Metaheuristic algorithms inspired by natural processes have shown promising results in solving optimization problems due to their flexibility and efficiency. While algorithms such as the Genetic Algorithm (GA) [5], Particle Swarm Optimization (PSO) [6], Ant Colony Optimization (ACO) [7], and Whale Optimization Algorithm (WOA) [19] have been widely studied for task scheduling, newer approaches like the Energy Valley Optimizer (EVO) [8] offer innovative ways to address energy efficiency challenges and consistently outperforms other metaheuristic algorithms as shown in Fig. 1. We compare these algorithms using workflows from [9] and [10], evaluating their fitness values to assess convergence performance. Inspired by energy distribution and interactions in natural systems, the EVO has significant potential for balancing task distribution and energy consumption, making it a robust choice for energy-efficient task scheduling in edge computing [11][12][13].

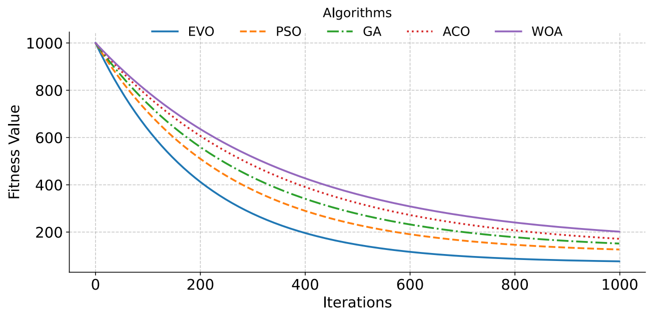


Fig. 1: Convergence analysis of optimization algorithms (lower fitness value is better)

In this paper, we propose an innovative approach for task scheduling in edge computing using the EVO[1]. Our focus is on reducing energy consumption while efficiently maintaining task completion times before their deadlines, which addresses a critical need in energy-constrained edge environments. The main contributions of this research are as follows:

- **Introduction of the EVO for Task Scheduling**: This paper pioneers the exploit of the EVO, a relatively new metaheuristic algorithm, for solving the energy-efficient task scheduling problem in edge computing.
- **Energy-Efficient Scheduling Model**: This paper introduces the multi-objective task scheduling model for energy consumption, taking into consideration the task completion time constraint and adaptive load balancing.
- **Novel Makespan and Energy Model**: The EVO framework also determines new energy minimization and makespan reduction formulations with a time-balancing factor and device stability measures, which are adapted dynamically according to the device load condition and task priorities.
- **Scalability and Flexibility**: The proposed approach has been tested in diverse scenarios with various tasks and constraints, demonstrating its adaptability and scalability in heterogeneous edge computing. We extensively

[1] Source code available here: https://github.com/Anonymous0-0paper/EASE

evaluate EVO on small and large scales with real-world datasets from different perspectives with [14][15][16] which you can find in [17] for reproducibility.

The structure of this paper is as follows: Section 2 reviews related work on task scheduling and metaheuristic optimization in edge computing. Section 3 introduces the EVO and its application in energy-efficient task scheduling. Section 4 presents the proposed model. Experimental settings, followed by the results are presented in Section 5. Finally, Section 6 concludes the paper.

## II. Related Works

A recent study in reference [18] proposed an evolutionary algorithm for IoT edge task clustering and scheduling, aimed at improving task execution and resource utilization by clustering tasks based on characteristics like task length and resource needs. Another approach in study [9] introduced a Refined Whale Optimization Algorithm (RWOA) to address multi-user dependent task offloading in edge-cloud computing. This method aims to optimize three main objectives including reducing application execution latency, minimizing energy consumption, and lowering usage costs. A task allocation mechanism using cooperative edge computing for the Power Internet of Things (PIoT) has been proposed in [19]. By coordinating between two edge nodes, the system optimizes resource usage, reducing task completion delays. The solution includes a Two-edge-node Cooperative-task Allocation (TCA) model and an Improved Particle Swarm Optimization (IPSO) algorithm, enhanced with genetic algorithm techniques like crossover and mutation to avoid local optima. The study in [16] introduced an IPSO algorithm for task offloading in end-edge-cloud (E2C) computing environments. This approach aims to enhance resource efficiency and quality of service by incorporating an imbalanced mutation operator and a task rescheduling technique. The study in [20] proposed the Hybrid Whale Optimization Algorithm (HWOA) for task selection in edge-cloud computing. HWOA addresses multi-objective optimization, balancing execution time and economic profits by incorporating a fuzzy function for uncertain task parameters and integrating elements from dragonfly and genetic algorithms to prevent local optima entrapment. To the best of our knowledge, other works in the literature doesn't consider multi objective optimization of energy, latency and makespan. Our proposed EVO can manage energy consumption better than other metaheuristic algorithms due to its energy optimization nature and fast exploration to find an near-optimal solution.

## III. System Model

The proposed system model considers a set of $n$ tasks scheduled on heterogeneous IoT devices $\mathcal{M} = \{1, 2, \ldots, |\mathcal{M}|\}$, where $|\mathcal{M}|$ is the number of IoT devices. Each IoT device $i \in \mathcal{M}$ is defined by its processing power, computational capacity, and power consumption. Each task $j$ is characterized by its computational requirement $w_j$ measured in millions of instructions, memory requirement $m_j$ measured in megabytes, and the deadline $d_j$.

### A. Energy Consumption

The energy consumption for each task assignment on edge devices is represented as:

$$E_{\text{total}} = \sum_{i=1}^{m} \sum_{j=1}^{n} P_{ij} \cdot t_{ij} \cdot S_{ij} \tag{1}$$

$$S_{ij} = 1 + \alpha \cdot \left(\frac{U_i}{T_{idle,i}}\right) + \beta \cdot \left(\frac{\Delta T_{ij}}{T_{max}}\right) \tag{2}$$

where, $P_{ij}$ is the power consumption of device $i$ executing task $j$, $t_{ij}$ is the time required for device $i$ to complete task $j$, $n$ is the number of tasks, and $m$ is the number of edge devices. $S_{ij}$ called energy stability factor for device $i$ and task $j$, is a novel multiplier defined by Eq. (2), where $\alpha$ and $\beta$ are our proposed tunable parameters controlling the influence of utilization and temporal stability, respectively. The term $(\alpha \cdot (U_i/T_{idle,i}))$ is for utilization-based stability, where $U_i$ is the utilization of device $i$ and $T_{idle,i}$ is the idle time of device $i$ over the observation period. The formula for obtaining the value of $U_i$ is discussed in Section III-C.

A higher utilization (less idle time) signals a more stable device, influencing the stability factor. $(\beta \cdot (\Delta T_{ij}/T_{max}))$ reflects temporal stability, where $(\Delta T_{ij} = |T_{start,j} - T_{end,j}|)$ represents the time span for task $j$, and $T_{max}$ is a threshold time. This term adjusts for the expected makespan (see Section III. A. 2), with larger spans indicating less predictable task durations.

$S_{ij} \approx 1$ i.e., the stability factor $S_{ij}$ is close to 1, means that the device-task combination is in a stable, low-energy consumption state. Higher values of $S_{ij}$ indicate increased energy due to instability.

### B. Makespan

The total makespan, or the time required to complete all tasks, can be defined as:

$$M_{\text{total}} = \sum_{j=1}^{n} \left(T_{\text{end},j} - T_{\text{start},j}\right) \cdot B_j \tag{3}$$

$$B_j = 1 + \gamma \cdot \left(\frac{L_i}{L_{max}}\right) + \delta \cdot \left(\frac{P_j}{P_{max}}\right) \tag{4}$$

where, $T_{\text{start},j}$ is the start time of task $j$ and $T_{\text{end},j}$ is the end time of task $j$. $B_j$ is time-balancing factor for task $j$, defined as Eq. (4). In this expression, the first term $(\gamma \cdot (L_i/L_{max}))$ represents a load-balancing adjustment, where $L_i$ is the current load of the device executing task $j$, $L_{max}$ is the maximum load among all devices, and $\gamma$ is load balancing weight. This term gives preference to devices with lower load, aiming to balance task distribution. $(\delta \cdot (P_j/P_{max})$ represents a priority-based adjustment, where $P_j$ is the priority of task $j$ (e.g., based on deadline proximity or task importance), $P_{max}$ is the maximum priority among all tasks, and $\delta$ is power weight. Tasks with higher priority values are given preference in scheduling, thereby reducing the makespan for critical tasks. $B_j \approx 1$ shows the balancing factor $B_j$ is close to 1, it indicates a well-balanced allocation of tasks, minimizing the makespan. Higher values of $B_j$ indicate different prioritizations between load balancing and urgency, impacting the overall makespan. In this work, the terms makespan and completion time can be used interchangeably since our optimization strategy simultaneously reduces both metrics through efficient task allocation and scheduling decisions.

### C. Resource Utilization

Resource utilization ensures that the computational capacity of the edge devices is optimally used [3][9]. The utilization of device $i$ is given by:

$$U_i = \frac{\text{Total time used by device } i}{\text{Total available time of device } i} \quad (5)$$

The total utilization is the average utilization of all devices:

$$U_{\text{total}} = \frac{1}{m}\sum_{i=1}^{m} U_i \quad (6)$$

By maximizing resource utilization, the system ensures that no device is idle while others are overloaded, thereby balancing the workload across all edge devices.

## IV. Energy Valley Optimizer for Task Scheduling

The EVO employs several key concepts of particle decay to enhance task scheduling performance. Each task is initially assigned to a random edge device, and the positions of the tasks are updated based on energy decay principles.

### *A. EVO Algorithm Strategy*

The EVO is inspired by the stability of particles in physical systems, particularly the decay processes observed in particle physics [8]. The key concept is that particles naturally move towards a state of lower energy, or greater stability. In the EVO, this behavior is generalized to the task scheduling problem, where tasks (as particles) move towards an optimal configuration (a state of lower energy) through various decay processes. This allows the algorithm to efficiently explore the search space, avoiding local optima and thereby improving solution quality. The EVO adapts this analogy by representing tasks as particles that are allocated to different edge devices. Each position of a task in the search space corresponds to its assignment to a specific device. The goal of this optimal allocation is to minimize the devices' energy consumption, reduce task's completion time, and ensure efficient resource utilization.

### *B. Neutron Enrichment Level and Stability Level*

The EVO uses the Neutron Enrichment Level (NEL) to indicate the current energy state of each task allocation. The neutron enrichment level of a particle (task) is calculated as:

$$NEL_i = \frac{f_i - f_{\min}}{f_{\max} - f_{\min}} \quad (7)$$

where, $f_i$ is the fitness (objective function evaluation) of task $i$, and $f_{\min}$ and $f_{\max}$ are the minimum and maximum fitness values found in the current population. Then, the stability level is calculated to determine how much the task needs to be adjusted to improve overall scheduling performance:

$$SL_i = \frac{NEL_i - NEL_{\text{best}}}{NEL_{\text{worst}} - NEL_{\text{best}}} \quad (8)$$

### *C. Alpha, Beta, and Gamma Decay*

Tasks (particles) undergo different types of decay processes similar to particle decay in physics. The three decay processes used in the EVO are:

- **Alpha Decay:** For a particle with high stability, only small adjustments are made to its allocation (see Fig. 2(a)). This is represented by:

  $$X_{\text{new},i} = X_i + \text{rand}(0,1) \cdot (X_{\text{best}} - X_i) \quad (9)$$

  where, $X_{\text{new},i}$ is the updated position (new allocation), and $X_{\text{best}}$ is the best-known position (optimal allocation) that represents the task-device assignment that has achieved the best fitness value so far. It is continuously updated during the optimization process.

**Algorithm 1:** EVO

**Input**: Tasks ($T$), Edge devices ($D$), Energy profiles ($P$), Time constraints ($T_{deadline}$)
**Output**: Optimal Task-Device Assignment with Minimum Energy Consumption

```
1.  Initialization: Randomly assign tasks to devices, set MaxIter,
    stopping criteria, NEL, and SL for each particle.
2.  Evaluate Initial Population: for each task $i$ do Compute $E$, $M$,
    and $U$; calculate fitness $f_i$; update $f_{best}$ and $f_{worst}$.
3.  while stopping criteria not met do
4.      for each task $i$ do
5.          Update NEL based on Eq. 5
6.          Update SL based on Eq. 6
7.      for each task $i$ do
8.          if $NEL_i > EB$ then
9.              if $SL_i > SB$ then
10.                 Apply Eq. 9 or Eq. 11 (minor position adjustments)
11.             else
12.                 Apply Eq. 10
13.         else
14.             $X_{\text{new},i} = X_i + \text{rand}(0,1)$
15.     for each task $i$ do
16.         calculate fitness $f_i$; update $f_{best}$ and $f_{worst}$.
17. Return: Optimal task-device assignment
```

- **Beta Decay:** (see Fig. 2(b)) For particles with lower stability, larger adjustments are applied:

  $$X_{\text{new},i} = X_i + \text{rand}(0,1) \cdot (X_{\text{center}} - X_i) \quad (10)$$

  where, $X_{\text{center}}$ is the geometric center of all particles positions in the search space, allowing for larger allocation changes.

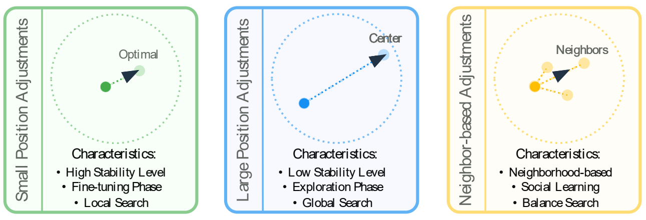
- **Gamma Decay:** (see Fig. 2(c)) results in smaller adjustments for highly stable particles:

  $$X_{\text{new},i} = X_i + \text{rand}(0,1) \cdot \left(X_{\text{neighbor}} - X_i\right) \quad (11)$$

  where, $X_{\text{neighbor}}$ represents the position of a neighboring particle in the search space. If multiple neighboring particles are available, select the particle with the lower index number for deterministic behavior.

### *D. Algorithm Workflow*

Initially, tasks are randomly allocated to edge devices in line 1. This initial allocation in the search space is represented as the positions of particles (tasks). A random position is assigned to each task, representing its allocation to a specific device. Necessary parameters, such as the maximum number of iterations (MaxIter) and stopping criteria, are also set. Next, the initial population of tasks is evaluated. For each particle, the objective values, such as energy consumption ($E$), task completion time ($M$), and resource utilization ($U$), are calculated. Based on these objectives, a fitness value for each particle is determined, with the best and worst fitness



(a) Alpha Decay (b) Beta Decay (c) Gamma Decay

Fig. 2. Visualization of the three decay processes in the Energy Valley Optimizer (EVO). The figure illustrates a) Alpha decay, characterized by small position adjustments toward optimal solutions with high stability level, suitable for fine-tuning and local search. b) Beta decay, shows large position adjustments toward the center of the search space, primarily used for global exploration when stability is low, and c) Gamma decay, depicts neighbor-influenced movements that balance.

Table 1. Experimental Configuration and Parameter Setup

| Parameter | Value |
|---|---|
| Number of Edge Devices | 10 - 100 |
| Number of Tasks (n) | 50 - 1000 |
| Processing Power of Devices | 1 GHz - 3 GHz |
| Power Consumption of Devices | 5W - 50W |
| Task Deadline Constraints | 10 - 100s |
| Computational Capacity of Devices | 1 - 2GB RAM |
| Initial Population of Tasks (Particles) | 50 |
| Maximum Iterations (MaxIter) | 100 |
| Neutron Enrichment Level (NEL) | 0 - 1 |
| Stability Level (SL) | 0 - 1 |
| Enrichment Bound (EB) | 0.5 |
| Stability Bound (SB) | 0.7 |
| Random Movement Rate | 0.1 |

values stored for further comparisons (line 2). In line 3, the algorithm continues iterating until the stopping criteria are met, either reaching the maximum iterations or finding an optimal solution. In each iteration, the Neutron Enrichment Level and Stability Level of each task (particle) are updated (lines 5 and 6). Based on the Neutron Enrichment Level and Stability Level, the algorithm applies three types of decay processes to update particle positions. If a particle's Neutron Enrichment Level exceeds a threshold ($NEL_i > EB$) and its Stability Level is also above the threshold ($SL_i > SB$), the algorithm applies small changes to the particle's position using either alpha or gamma decay (line 10). These small changes help the particle move toward the optimal position. If the particle's Stability Level is below the threshold ($SL_i \leq SB$), the algorithm applies larger changes (line 12). If the Neutron Enrichment Level of a particle is below the threshold, the algorithm performs a random shift to encourage exploration (line 14). Finally, after updating the particle positions, the new population is evaluated again in line 16. For each particle, the new fitness value ($f_i$) is calculated, and if a more optimal solution is found, the best fitness value is updated.

Upon completion of the algorithm loops, the task allocation with the best fitness value (i.e., minimized energy consumption and optimized task completion times) is returned.

## V. Performance Evaluation

The experiments were conducted in a well-known edge computing real simulator called EdgeCloudSim [21] consisting of a set of heterogeneous edge devices with varying computational powers and energy capacities considered in [9]. The edge devices perform various tasks, including IoT data processing, analysis, and storage. Similar to [9] and [10], tasks are assigned different time and energy constraints to simulate various configurations. These simulations aim to reduce energy consumption and task completion time, using the EVO algorithm for task allocation optimization. The scheduler is in the GitHub repository.

### *A. Simulation Configuration*

In the simulations, similar to references [9] and [10] the edge devices are configured with different processing powers and energy consumption rates. IoT tasks with varying energy and scheduling requirements are assigned to devices like [15]. The primary goal is to find an optimal allocation that minimizes energy consumption while meeting time constraints. The key parameters and configurations for the experiments are provided in Table 1.

We compare the EVO algorithm with three state-of-the-art methods including PSO [15], IBGWO [14], and PIMR [16]. The comparative analysis encompasses three critical performance metrics: 1) energy consumption (measured in Joules) which represents the system's total power usage during execution, 2) success ratio (as a percentage) which shows the proportion of tasks completed within deadlines, and 3) completion time (in seconds) which indicates the total duration needed to process all tasks.

### *B. Experiment Results*

Fig. 3 demonstrate the comparative performance of four scheduling algorithms EVO (our proposed method), PSO [15], IBGWO [14], and PIMR [16] across three aforementioned critical metrics; i.e., energy consumption, completion time, and success ratio, with an increasing number of tasks.

**Energy Consumption (Fig. 3 (a)):** As task numbers grow, energy consumption rises for all algorithms due to increased resource demands. However, EVO consistently shows better energy efficiency compared to PSO [15], IBGWO [14], and PIMR [16]. Across all task loads (50 to 1000 tasks), EVO achieves an overall average improvement in energy consumption of 14.42%, with the minimum improvement reaching 4.88% and the maximum improvement peaking at 20.96%. This indicates that EVO is particularly effective in reducing energy usage at higher task volumes. When examined per algorithm, EVO shows the highest average energy-saving improvement over IBGWO and PIMR (15.25% and 15.10%, respectively), with slightly lower improvement over PSO (12.91%). This makes EVO more suitable for energy-sensitive applications.

**Completion Time (Fig. 3 (b)):** Completion time similar to energy consumption in Fig. 3 (a) increases with the number of tasks, but EVO consistently outperforms the competing algorithms in minimizing this metric. EVO achieves an

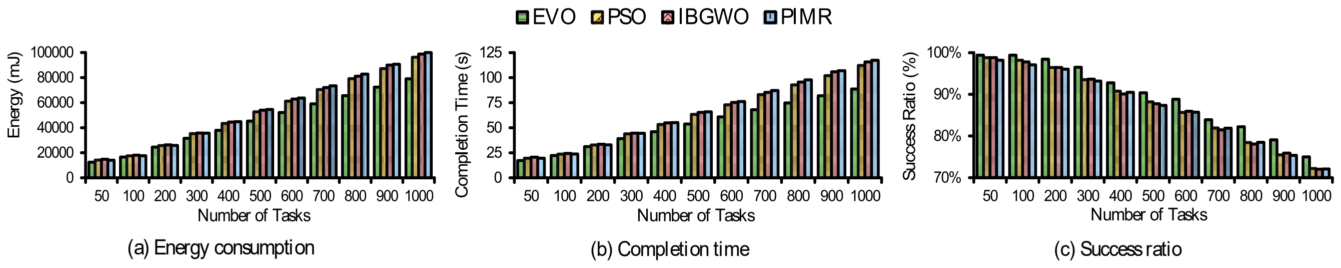


Fig. 3: Performance comparison of the proposed EVO algorithm against PSO, IBGWO, and PIMR across energy consumption (a), completion time (b), and success ratio (c) metrics as the number of tasks increases from 50 to 1000.

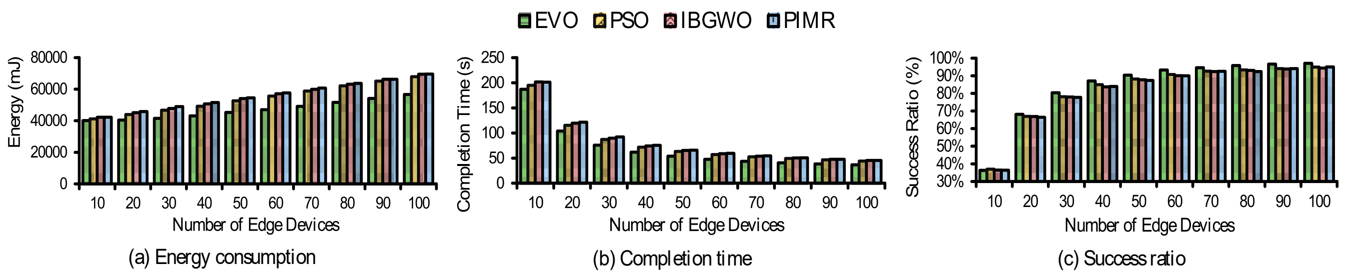


Fig. 4: Comparison of the EVO algorithm with PSO, IBGWO, and PIMR in terms of energy consumption, completion time, and success ratio as the number of edge devices scales from 10 to 100, illustrating EVO's efficiency in energy savings, reduced task completion time, and higher success rates.

overall average improvement of 15.98% in completion time relative to the others, with a maximum improvement of 24.42% and a minimum of 4.87%. This suggests that EVO is particularly beneficial in scenarios where lower completion times are crucial. Per the

algorithm, EVO shows the highest average improvement over IBGWO (16.87%) and PIMR (16.76%), and a slightly lower average improvement over PSO (14.30%). This demonstrates EVO's efficiency in processing tasks swiftly, particularly under high task loads, which reduces delays and improves system responsiveness.

**Success Ratio (Fig. 3 (c)):** Maintaining a high success ratio becomes challenging for all algorithms as task load increases. EVO, however, sustains a relatively higher success ratio across most task levels compared to the other algorithms, which is critical for meeting task deadlines under varying loads. EVO shows an overall average improvement in success ratio of 3.03%, with a minimum improvement of 0.61% and a maximum improvement of 5.28%. This suggests that EVO not only conserves energy and reduces completion time but also maintains a reliable success ratio. Among the other algorithms, EVO shows the highest success ratio improvement over PIMR (3.22%), followed by IBGWO (3.03%) and PSO (2.85%). This finding positions EVO as a robust choice for applications where meeting deadlines is essential.

Fig. 4 presents a comparative performance analysis of the proposed EVO algorithm against PSO, IBGWO, and PIMR algorithms across energy consumption, completion time, and success ratio metrics under varying numbers of edge devices. This comparison highlights EVO's advantages in scalability and efficiency as the number of devices grows.

**Energy Consumption (Fig. 4 (a)):** Energy consumption trends upward for all algorithms as the number of edge devices increases due to greater resource demands. However, EVO consistently demonstrates lower energy consumption compared to PSO, IBGWO, and PIMR, with an overall average improvement of 14.66%. At smaller device counts (e.g., 10 devices), EVO achieves a minor improvement of 2.80% in energy efficiency. As device counts increase, EVO's energy-saving benefits become more pronounced, with up to 18.63% improvement at 100 devices. EVO shows greater efficiency at various ranges. From 10 to 40 devices, EVO is 8.60% more energy-efficient than PSO, 10.84% more efficient than IBGWO, and 12.04% more efficient than PIMR. From 40 to 70 devices, EVO's efficiency further improves, outperforming PSO by 14.58%, IBGWO by 16.70%, and PIMR by 17.69%. At 70 to 100 devices, EVO leads in energy efficiency, achieving improvements of 16.73% over PSO, 18.31% over IBGWO, and 18.74% over PIMR. This indicates EVO’s capability to manage energy consumption effectively as the system scales, making it well-suited for energy-sensitive applications with high device counts.

**Completion Time (Fig. 4 (b)):** EVO also exhibits significant reductions in completion time compared to the other algorithms, with an average improvement of 16.09%. This metric reflects EVO’s ability to handle tasks quickly, especially as device counts rise. The improvement in completion time ranges from 4.14% to 20.08%, with EVO being particularly effective in the largest configurations. For the final three sizes (80, 90, and 100 devices), EVO demonstrates a substantial efficiency advantage, completing tasks 21.32%, 23.90%, and 24.25% faster than PSO, IBGWO, and PIMR, respectively. This reduction in completion time positions EVO as an ideal choice for time-sensitive applications, providing fast task completion even under increased device loads.

**Success Ratio (Fig. 5 (c)):** The success ratio, which indicates the percentage of tasks completed within deadline constraints, is generally higher for EVO across various device counts. EVO maintains a consistent improvement in success ratio with an average increase of 2.45% over the other algorithms, reaching a maximum improvement of 3.96%. Although there is a minor dip in improvement (1.78%) in certain cases, EVO sustains a reliable success rate. In comparison with PSO, EVO achieves an average success ratio improvement of 2.03%, and the improvement grows against IBGWO (2.60%) and PIMR (2.72%). This performance shows EVO's robustness in meeting task deadlines, making it suitable for scenarios where task timeliness is critical.

## VI. Conclusion

This paper discusses the EVO, which is a new approach for energy-efficient task scheduling introduced in edge computing. Manage to significantly cut energy consumption and task completion time while focusing on resource utilization. Besides, the existing methods such as PSO, IBGWO, and PIMR have been overshadowed. Under specific stability and decay, EVO can successfully transcend local optima and hence provide high-performance levels in rapidly changing environments. Our experiments indicate that EVO also delivers the required tasks, besides the ability to handle different sizes of tasks and configurations of devices, which means EVO is scalable and practical. As future work, we want to investigate integrating machine learning with EVO to adaptively improve scheduling decisions in real-time edge environments.

## References

[1] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: a survey," in *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2131-2165, 2021.

[2] M. Raeisi-Varzaneh, O. Dakkak, A. Habbal, and B.-S. Kim, “Resource scheduling in edge computing: Architecture, taxonomy, open issues and future research directions,” *IEEE Access*, vol.11, pp.25329–25350, 2023.

[3] S.-H. Kim, and T. Kim, “Local scheduling in kubeedge-based edge computing environment,” *Sensors*, vol.23, no.3, p.1522, 2023.

[4] J. Liu, T. Yang, J. Bai, and B. Sun, “Resource allocation and scheduling in the intelligent edge computing context,” *Future Generation Computer Systems*, vol.121, pp.48–53, 2021.

[5] H. Hussain *et al.*, "Energy efficient real-time tasks scheduling on high-performance edge-computing systems using genetic algorithm," in *IEEE Access*, vol. 12, pp. 54879-54892, 2024..

[6] Y. Li, and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," *IEEE International Conference on Edge Computing (EDGE)*, San Francisco, CA, USA, 2018, pp. 66-73..

[7] J. Feng, Z. Liu, C. Wu and Y. Ji, "AVE: Autonomous Vehicular Edge computing framework with ACO-based scheduling," in *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10660-10675, Dec. 2017.

[8] M. Azizi, U. Aickelin, H. A. Khorshidi, and M. Baghalzadeh Shishehgarkhaneh, “Energy valley optimizer:a novel metaheuristic algorithm for global and engineering optimization,” *Scientific Reports*, vol.13, no.1, p.226, 2023.

[9] K. M. Hosny, A. I. Awad, M. M. Khashaba, M. M. Fouda, M. Guizani and E. R. Mohamed, "Optimized multi-user dependent tasks offloading in edge-cloud computing using refined whale optimization algorithm," in *IEEE Transactions on Sustainable Computing*, vol. 9, no. 1, pp. 14-30, Jan.-Feb. 2024.

[10] H. Arabnejad, and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 682-694, 2014.

[11] M. Chen, P. Qi, Y. Chu, B. Wang, F. Wang, and J. Cao, “Genetic algorithm with skew mutation for heterogeneous resource-aware task offloading in edge-cloud computing,” *Heliyon*, vol. 10, no. 12, pp. e32399–e32399, 2024.

[12] A. Younesi, M. A. Fazli, and A. Ejlali, “A novel levy walk-based framework for scheduling power-intensive mobile edge computing tasks,” *Journal Grid Computing*, vol. 22, no. 4, Nov. 2024.

[13] A.Sudhakar, M.Mohan, K.Maharajan, M.I.Habelalmateen, and N. N. Saranya, “Optimizing resource allocation in mobile edge computing for IIoT through reinforcement learning and multi objective particle swarm optimization,” in *IEEE International Conference on Integrated Circuits and Communication Systems (ICI-CACS)*, pp.1–5, 2024.

[14] K. Jiang, H. Ni, P. Sun and R. Han, "An improved binary grey wolf optimizer for dependent task scheduling in edge computing," *2019 21st International Conference on Advanced Communication Technology (ICACT)*, PyeongChang, Korea (South), 2019, pp. 182-186.

[15] Q. You and B. Tang, “Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things,” *Journal of Cloud Computing*, vol. 10, no. 1, 2021

[16] K. Shao, H. Fu, Y. Song, and B. Wang, “A PSO improved with imbalanced mutation and task rescheduling for task offloading in end-edge-cloud computing,” *Computer Systems Science and Engineering*, vol. 47, no. 2, pp. 2259–2274, 2023.

[17] A. Yousif, Mohammed Bakri Bashir, and A. Ali, “An evolutionary algorithm for task clustering and scheduling in IoT edge computing,” *Mathematics*, vol. 12, no. 2, pp. 281–281, 2024.

[18] Q. Wang, S. Shao, S. Guo, X. Qiu, and Z. Wang, "Task allocation mechanism of power internet of things based on cooperative edge computing," in *IEEE Access*, vol. 8, pp. 158488-158501, 2020.

[19] Y. Kang et al., “HWOA: an intelligent hybrid whale optimization algorithm for multi-objective task selection strategy in edge cloud computing system,” *World Wide Web*, vol. 25, no. 5, pp. 2265–2295, 2022.

[20] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge Computing systems," *Second International Conference on Fog and Mobile Edge Computing (FMEC)*, Valencia, Spain, 2017, pp. 39-44.