scientific reports

Check for updates

OPEN

Energy efficient task scheduling for heterogeneous multicore processors in edge computing

Yanchun Liu[1]✉, Hongxue Qu[1], Shuang Chen[1] & Xuejun Feng[2]

Edge computing faces challenges in energy-efficient task scheduling for heterogeneous multicore processors (HMPs). Existing solutions focus on reactive workload adaptation and energy prediction but fail to effectively integrate dynamic voltage and frequency scaling (DVFS). This paper proposes a novel algorithm integrating task prioritization, core-aware mapping, and predictive DVFS. Our approach outperforms state-of-the-art methods, reducing energy consumption by 20.9% while maintaining a low 2.4% deadline miss rate. Experiments on real HMP platforms demonstrate the algorithm's scalability and adaptability to varying workloads. This work advances energy-efficient edge computing, balancing performance and power constraints.

Keywords Edge computing, Heterogeneous multicore processors, Energy-efficient task scheduling, DVFS, Task mapping, Task priority assignment, Performance optimization.

Edge computing has emerged as a promising paradigm to address the challenges of latency, bandwidth, and privacy in Internet of Things (IoT) applications[1]. By processing data closer to the source, edge computing reduces the reliance on cloud infrastructure and enables real-time decision making[2]. Heterogeneous multicore processors (HMPs) have become increasingly prevalent in edge computing due to their ability to provide high performance and energy efficiency[3]. HMPs combine multiple types of cores, such as high-performance cores and energy-efficient cores, on a single chip, allowing for flexible task allocation based on workload requirements[4].

Energy consumption is a critical concern in edge computing, as edge devices often operate on limited battery power or constrained energy budgets[5]. Efficient task scheduling algorithms play a crucial role in optimizing energy consumption while meeting the performance demands of applications[6]. However, designing energy-efficient task scheduling algorithms for HMPs in edge computing presents several challenges. First, the heterogeneity of cores in HMPs introduces complexities in task allocation and resource management[7]. Second, the dynamic and unpredictable nature of edge workloads requires adaptive and responsive scheduling strategies[8]. Third, the diverse requirements of edge applications, such as real-time constraints and data dependencies, further complicate the scheduling process[1].

Existing research on energy-efficient task scheduling for HMPs has primarily focused on cloud computing environments[3,4]. These approaches often rely on static workload characteristics and offline profiling, which may not be suitable for the dynamic nature of edge computing[2]. Moreover, the majority of existing algorithms consider a single objective, such as minimizing energy consumption or maximizing performance, without addressing the trade-offs between multiple conflicting objectives[6].

While existing research has made significant strides in energy-efficient task scheduling for HMPs, several limitations persist. Current approaches often focus on optimizing individual aspects of the scheduling problem, such as task prioritization[9], energy-aware scheduling[10], or thermal management[11]. However, these methods fail to capture the complex interplay between task characteristics, core heterogeneity, and dynamic power management in edge computing environments.

Our work addresses these limitations through a novel integrated approach that simultaneously considers task priorities, core performance characteristics, and dynamic voltage and frequency scaling (DVFS). This holistic method enables more efficient utilization of heterogeneous resources while adapting to the dynamic nature of edge computing workloads. Recent studies[12,13] have begun to explore the potential of integrated approaches, but the specific combination of techniques proposed in this paper remains largely unexplored.

By leveraging the synergies between task prioritization, performance-aware core mapping, and predictive DVFS optimization, our algorithm achieves superior energy efficiency without compromising on performance

[1]Department of Computer and Software Engineering, Shandong College of Electronic Technology, Jinan 250200, Shandong, China. [2]Inspur Communication Information Systems Co., Ltd., Jinan 250101, Shandong, China. ✉email: liuyanchun@sdcet.edu.cn

or deadline adherence. This innovative integration represents a significant advancement in the field of energy-efficient task scheduling for edge computing, paving the way for more sustainable and efficient edge computing systems.

This study aims to address the challenges of energy-efficient task scheduling in heterogeneous multicore processors for edge computing. Our primary research objectives are to develop an integrated task scheduling algorithm that considers task priorities, core heterogeneity, and dynamic voltage and frequency scaling; minimize energy consumption while meeting deadlines and performance requirements; achieve superior energy efficiency compared to existing algorithms; ensure scalability for large-scale deployments; and validate the algorithm's effectiveness through comprehensive experiments. By achieving these objectives, we aim to contribute to the development of more energy-efficient and sustainable edge computing systems.

To address these challenges, we propose a novel energy-efficient task scheduling algorithm for HMPs in edge computing. Our algorithm leverages machine learning techniques to dynamically learn the characteristics of edge workloads and adapt the scheduling decisions accordingly. By considering both the heterogeneity of cores and the real-time requirements of tasks, our algorithm aims to achieve a balance between energy efficiency and performance. The main contributions of this paper are as follows:

1. We present a comprehensive analysis of the challenges and opportunities for energy-efficient task scheduling in edge computing using HMPs.
2. We propose a novel task scheduling algorithm that combines machine learning techniques with energy-aware heuristics to dynamically optimize task allocation and resource management.
3. We evaluate the performance of our algorithm through extensive simulations and real-world experiments, demonstrating its effectiveness in reducing energy consumption while meeting the performance requirements of edge applications.

The remainder of this paper is organized as follows. Section II provides an overview of related work on energy-efficient task scheduling for HMPs. Section III describes the system model and problem formulation. Section IV presents our proposed task scheduling algorithm in detail. Section V discusses the experimental setup and evaluation results. Finally, Section VI concludes the paper and outlines future research directions.

Related work

Heterogeneous multicore processors in edge computing

Heterogeneous multicore processors (HMPs) have gained significant attention in edge computing due to their ability to provide high performance and energy efficiency[14]. Unlike traditional homogeneous multicore processors, HMPs integrate multiple types of cores with different performance and power characteristics on a single chip[15]. This heterogeneity allows for the efficient execution of diverse workloads by mapping tasks to the most suitable cores based on their requirements[16].

The architecture of HMPs in edge computing typically consists of a combination of high-performance cores (HPCs) and energy-efficient cores (EECs)[17]. HPCs are designed to deliver high computational performance and are suitable for compute-intensive tasks that require fast execution. On the other hand, EECs are optimized for low power consumption and are ideal for less demanding tasks or tasks that can tolerate lower performance[18]. By intelligently distributing tasks across these different types of cores, HMPs can achieve a balance between performance and energy efficiency.

In edge computing environments, HMPs find extensive application in various domains, such as autonomous vehicles, smart cities, and industrial automation[19]. For example, in autonomous vehicles, HMPs can be used to process sensor data, perform real-time object detection and tracking, and make critical decisions based on the processed information. The heterogeneous architecture of HMPs enables the efficient execution of these diverse tasks, ranging from low-level sensor data processing to high-level decision making.

Another prominent application of HMPs in edge computing is in the realm of Internet of Things (IoT)[20]. Edge devices, such as gateways and smart sensors, often employ HMPs to handle the processing and analysis of data generated by IoT devices. The heterogeneity of HMPs allows for the efficient execution of tasks with different requirements, such as data preprocessing, feature extraction, and machine learning inference[21]. By performing these tasks at the edge, HMPs enable real-time decision making and reduce the latency and bandwidth requirements of IoT applications.

The use of HMPs in edge computing also enables energy-efficient execution of multimedia applications[22]. Edge devices equipped with HMPs can efficiently process and analyze multimedia data, such as video and audio streams, in real-time. The heterogeneous architecture of HMPs allows for the efficient distribution of tasks across different types of cores, optimizing performance and energy consumption. This is particularly important in scenarios where edge devices operate on limited battery power or have stringent energy constraints.

However, the heterogeneity of HMPs also introduces challenges in task scheduling and resource management[23]. Efficient scheduling algorithms are required to map tasks to the appropriate cores based on their characteristics and requirements. Moreover, the dynamic nature of edge computing workloads necessitates adaptive and responsive scheduling strategies that can adapt to changing system conditions and workload patterns[24]. Addressing these challenges is crucial for fully leveraging the benefits of HMPs in edge computing environments.

Task scheduling algorithms

Task scheduling plays a crucial role in optimizing the performance and energy efficiency of heterogeneous multicore processors (HMPs) in edge computing. Numerous task scheduling algorithms have been proposed in

the literature, each with its own strengths and limitations[25]. In this section, we provide an overview of existing task scheduling algorithms and analyze their applicability and limitations in the context of HMPs.

One of the fundamental approaches to task scheduling is the list scheduling algorithm[26]. List scheduling maintains a priority list of tasks and assigns them to available cores based on their priorities. The priority of a task can be determined by various factors, such as its execution time, deadline, or resource requirements. While list scheduling is simple and effective, it may not fully exploit the heterogeneity of HMPs, as it does not consider the specific characteristics of different core types[27].

To address the limitations of list scheduling, researchers have proposed heterogeneity-aware scheduling algorithms. These algorithms take into account the different performance and power characteristics of cores in HMPs and aim to match tasks to the most suitable cores[28]. For example, the Heterogeneous Earliest Finish Time (HEFT) algorithm[9] prioritizes tasks based on their earliest finish time and assigns them to the core that minimizes the overall execution time. The HEFT algorithm has been widely adopted and has shown good performance in HMP environments.

Another class of scheduling algorithms focuses on energy-aware scheduling, which aims to minimize the energy consumption of HMPs while meeting performance requirements[29]. Energy-aware scheduling algorithms often employ dynamic voltage and frequency scaling (DVFS) techniques to adjust the operating frequency and voltage of cores based on the workload demands. For instance, the Energy-Aware Scheduling (EAS) algorithm[10] uses DVFS to reduce the energy consumption of idle cores and balances the workload across different core types to maximize energy efficiency.

In addition to performance and energy considerations, some scheduling algorithms also take into account the thermal constraints of HMPs[30]. Thermal-aware scheduling algorithms aim to prevent the formation of hotspots and ensure the reliable operation of HMPs by considering the temperature of cores during task allocation. The Thermal-Aware Scheduling (TAS) algorithm[11] uses a thermal model to predict the temperature of cores and adjusts the task allocation accordingly to maintain a balanced thermal distribution.

Machine learning techniques have also been explored for task scheduling in HMPs[31]. Machine learning-based scheduling algorithms can learn from historical data and adapt to the dynamic characteristics of workloads and system conditions. For example, the Reinforcement Learning-based Scheduling (RLS) algorithm[32] uses reinforcement learning to learn the optimal scheduling policy based on the feedback received from the system. The RLS algorithm has shown promising results in adapting to varying workload patterns and improving the overall system performance.

Despite the advancements in task scheduling algorithms for HMPs, several challenges remain. One of the main challenges is the trade-off between performance and energy efficiency[33]. Scheduling algorithms that prioritize performance may lead to higher energy consumption, while energy-aware algorithms may compromise on performance. Balancing these conflicting objectives is a complex problem that requires careful consideration of the specific requirements of edge computing applications.

Another challenge is the scalability of scheduling algorithms in the presence of a large number of tasks and cores[34]. As the number of tasks and cores increases, the complexity of scheduling decisions grows exponentially. Efficient and scalable scheduling algorithms are necessary to handle the increasing scale of edge computing systems.

Moreover, the dynamic and unpredictable nature of edge computing workloads poses challenges for task scheduling[35]. Edge devices often operate in environments with varying workload patterns and resource availability. Scheduling algorithms must be able to adapt to these dynamic conditions and make real-time decisions based on the current system state.

To address these challenges, there is a need for novel task scheduling algorithms that can effectively leverage the heterogeneity of HMPs, balance performance and energy efficiency, and adapt to the dynamic nature of edge computing workloads. In the following sections, we propose a novel energy-efficient task scheduling algorithm that aims to address these challenges and improve the overall system performance and energy efficiency.

The energy consumption of a task i executed on core j can be modeled as:

$$E_{i,j} = P_{i,j} \times t_{i,j} \tag{1}$$

where $P_{i,j}$ represents the power consumption of task i on core j, and $t_{i,j}$ represents the execution time of task i on core j.

The objective of energy-efficient task scheduling can be formulated as:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} E_{i,j} \times x_{i,j} \tag{2}$$

subject to:

$$\sum {}_{j=1}^{m} x_{i,j} = 1, \forall\, i \in \ 1, 2, \ldots, n$$

$$\sum {}_{i=1}^{n} x_{i,j} \leq \ 1, \forall\, j \in \ 1, 2, \ldots, m$$

where n is the number of tasks, m is the number of cores, and $x_{i,j}$ is a binary variable indicating whether task i is assigned to core j.

The objective function (2) aims to minimize the total energy consumption of executing all tasks on the available cores. The first constraint ensures that each task is assigned to exactly one core, while the second constraint ensures that each core executes at most one task at a time.

Energy-saving techniques

Energy consumption is a critical concern in edge computing, as edge devices often operate on limited battery power or have stringent energy constraints[36]. To address this challenge, various energy-saving techniques have been proposed for processors, aiming to reduce their power consumption while maintaining acceptable performance levels. In this section, we summarize the existing energy-saving techniques for processors, with a focus on dynamic voltage and frequency scaling (DVFS) and core sleep states.

Dynamic voltage and frequency scaling (DVFS) is a widely adopted technique for reducing the energy consumption of processors[37]. DVFS allows the processor to dynamically adjust its operating voltage and frequency based on the workload demands. By lowering the voltage and frequency during periods of low utilization, DVFS can significantly reduce the power consumption of the processor. When the workload intensity increases, the voltage and frequency can be scaled up to meet the performance requirements. DVFS has been extensively studied and implemented in various processor architectures, including heterogeneous multicore processors (HMPs)[38].

Core sleep states, also known as power gating, are another effective technique for reducing the energy consumption of processors[39]. In modern processors, each core can be individually put into a low-power sleep state when it is not actively executing tasks. During the sleep state, the core is essentially turned off, consuming minimal power. When a task is ready to be executed on the core, it is woken up from the sleep state, and the power consumption returns to the normal operating level. Core sleep states can be particularly effective in HMPs, where the heterogeneity of cores allows for the selective sleeping of cores based on the workload characteristics[40].

The combination of DVFS and core sleep states can lead to significant energy savings in processors. By dynamically adjusting the voltage and frequency of active cores and putting idle cores into sleep states, the overall power consumption of the processor can be minimized. However, the effectiveness of these techniques depends on the characteristics of the workload and the specific architecture of the processor. In HMPs, the heterogeneity of cores adds an additional layer of complexity to the energy management strategies[41].

To fully leverage the potential of DVFS and core sleep states in HMPs, intelligent energy management policies are required. These policies should take into account the performance requirements of tasks, the energy consumption characteristics of different core types, and the dynamic nature of the workload. Machine learning techniques, such as reinforcement learning, have shown promise in developing adaptive energy management policies that can learn from the system behavior and make optimal decisions in real-time[42].

In addition to DVFS and core sleep states, other energy-saving techniques have been explored for processors. These include power-aware task mapping, where tasks are assigned to cores based on their power consumption characteristics[43], and energy-aware scheduling algorithms, which consider the energy consumption of tasks during the scheduling process[44]. These techniques can be used in conjunction with DVFS and core sleep states to further optimize the energy efficiency of processors.

While energy-saving techniques have made significant advancements, there are still challenges to be addressed. One challenge is the trade-off between energy savings and performance. Aggressive power management techniques may lead to performance degradation, which can be unacceptable for certain applications. Balancing energy efficiency and performance requirements is a delicate task that requires careful consideration of the specific needs of edge computing workloads[41].

Another challenge is the integration of energy-saving techniques with other system components, such as memory and communication subsystems. The energy consumption of these components can also have a significant impact on the overall power consumption of edge devices. Holistic energy management approaches that consider the entire system stack are necessary to achieve maximum energy efficiency[45].

In the context of HMPs in edge computing, the development of energy-saving techniques that can effectively leverage the heterogeneity of cores and adapt to the dynamic nature of edge workloads is an important research direction. In the following sections, we propose a novel energy-efficient task scheduling algorithm that incorporates DVFS and core sleep states to minimize the energy consumption of HMPs while meeting the performance requirements of edge computing applications.

DVFS versus other task scheduling and energy optimization techniques

Dynamic Voltage and Frequency Scaling (DVFS) is a widely adopted technique to reduce processor energy consumption[37]. DVFS has the following advantages over other task scheduling and energy optimization techniques:

1. DVFS can dynamically adjust the processor voltage and frequency according to the demand of workloads, thus reducing energy consumption while maintaining performance. In contrast, traditional task scheduling algorithms usually focus only on the execution order of tasks and ignore the optimization of processor energy consumption[46].
2. DVFS is particularly suitable for heterogeneous multicore processors (HMPs) in edge computing because different types of cores in HMPs have different performance and power consumption characteristics. By applying DVFS to each core individually, energy consumption can be optimized at a finer granularity[47].
3. DVFS can be combined with other task scheduling techniques such as core-aware task mapping and task prioritization to further improve energy efficiency[5]. This combination can minimize energy consumption while meeting task deadlines.

4. DVFS has been widely supported in various processor architectures, including ARM processors commonly used in mobile devices and embedded systems[48]. This makes DVFS ideal for achieving energy efficiency optimization in edge computing scenarios.

Despite the significant benefits of DVFS in reducing processor energy consumption, existing DVFS techniques typically rely on reactive tuning of workloads and prediction of energy consumption. These approaches may lead to suboptimal decisions in the dynamic environment of edge computing. Therefore, this paper proposes a novel approach that combines task prioritization, core-aware mapping, and predictive DVFS to better accommodate heterogeneous multi-core processors in edge computing.

Comparative analysis of existing approaches

While significant progress has been made in energy-efficient task scheduling for heterogeneous multicore processors (HMPs) in edge computing, existing approaches have limitations that our work aims to address. Table 1 summarizes the key features and limitations of foundational works in this domain.

The Heterogeneous Earliest Finish Time (HEFT) algorithm[9] has been widely adopted for its effectiveness in minimizing overall execution time. However, it does not explicitly consider energy consumption, which is crucial in edge computing environments. The Energy-Aware Scheduling (EAS) algorithm[10] addresses this limitation by incorporating DVFS techniques, but it lacks comprehensive consideration of task dependencies, which can lead to suboptimal scheduling decisions in complex workflows.

Thermal-aware scheduling approaches, such as the Thermal-Aware Scheduling (TAS) algorithm[11], introduce temperature considerations to prevent hotspots and ensure reliable operation. However, these approaches often do not fully leverage the heterogeneity of cores in terms of performance characteristics. Machine learning-based approaches, like the Reinforcement Learning-based Scheduling (RLS) algorithm[32], offer adaptability to dynamic workloads but may incur high computational overhead, making them less suitable for real-time decision-making in resource-constrained edge devices.

Our proposed algorithm addresses these limitations by integrating task priority assignment based on both deadlines and dependencies, core performance-aware task mapping, and energy consumption prediction-based DVFS scheduling. This holistic approach aims to achieve a balance between energy efficiency and performance while adapting to the dynamic nature of edge computing workloads.

Energy-efficient task scheduling algorithm

System model and problem definition

In this section, we present the system model for heterogeneous multicore processors (HMPs) in edge computing environments and formally define the energy-efficient task scheduling problem.

System model

We consider an edge computing system consisting of a set of heterogeneous multicore processors (HMPs) denoted by $P = p_1, p_2, \ldots, p_m$, where m is the number of HMPs. Each HMP p_i contains a set of heterogeneous cores denoted by $C_i = c_{i,1}, c_{i,2}, \ldots, c_{i,n_i}$, where n_i is the number of cores in HMP p_i. The cores within an HMP can have different performance and power characteristics, such as high-performance cores (HPCs) and energy-efficient cores (EECs).

The system receives a set of tasks denoted by $T = t_1, t_2, \ldots, t_n$, where n is the number of tasks. Each task t_j is characterized by its workload size w_j, arrival time a_j, and deadline d_j. The execution time of task t_j on core $c_{i,k}$ is denoted by $e_{j,i,k}$, which depends on the workload size and the performance characteristics of the core.

The power consumption of core $c_{i,k}$ is denoted by $p_{i,k}(f)$, where f is the operating frequency of the core. The power consumption can be modeled using the dynamic voltage and frequency scaling (DVFS) technique, where the power consumption is a function of the operating frequency.

Table 2 presents a comprehensive list of notations and their corresponding definitions used throughout this paper. These symbols and terms are essential for understanding the proposed algorithm, mathematical formulations, and analysis presented in our work.

These notations will be used consistently throughout the paper to describe the task scheduling problem, the proposed algorithm, and the analysis of results. When introducing new concepts or equations, we will provide additional explanations to ensure clarity.

Example usage:

The overall priority $\mathrm{P}(\mathrm{t_i})$ of task $\mathrm{t_i}$ is calculated as a weighted sum of its deadline-based priority $\mathrm{P^d}(\mathrm{t_i})$and dependency-based priority $\mathrm{P}^r(\mathrm{t_i})$:

$$P(ti) = \alpha \times Pd(ti) + (1-\alpha) \times Pr(ti)$$

Approach	Key features	Limitations
HEFT[24]	Prioritizes tasks based on earliest finish time	Does not consider energy consumption
EAS[26]	Uses DVFS for energy-aware scheduling	Limited consideration of task dependencies
TAS[28]	Considers thermal constraints	Does not integrate core performance awareness
RLS[30]	Employs reinforcement learning for adaptive scheduling	High computational overhead for real-time decisions

Table 1. Comparison of existing approaches.

Symbol	Definition
T	Set of tasks, $\mathrm{T} = \{\mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_n\}$
n	Number of tasks
$\mathrm{t_i}$	i-th task
w_i	Workload size of task $\mathrm{t_i}$
a_i	Arrival time of task $\mathrm{t_i}$
$\mathrm{d_i}$	Deadline of task $\mathrm{t_i}$
P	Set of heterogeneous multicore processors, $\mathrm{P} = \{\mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p_m}\}$
m	Number of processors
C_i	Set of cores in processor p_i, $\mathrm{C_i} = \{\mathrm{c}_{\mathrm{i},1}, \mathrm{c}_{\mathrm{i},2}, \ldots, \mathrm{c_i}, k_i\}$
k_i	Number of cores in processor $\mathrm{p_i}$
$\mathrm{e_{i,\ j,\ k}}$	Execution time of task $\mathrm{t_i}$ on core $\mathrm{c_{j,k}}$
$\mathrm{p_{j,}}_k\ (\mathrm{f})$	Power consumption of core $\mathrm{c_{j,k}}$ at frequency f
f	Operating frequency of a core
$\mathrm{P^d}\ (\mathrm{t_i})$	Deadline-based priority of task $\mathrm{t_i}$
$\mathrm{P^r}\ (\mathrm{t_i})$	Dependency-based priority of task $\mathrm{t_i}$
α	Priority weight factor, α ∈ [0,1]
$\mathrm{P}\ (\mathrm{t_i})$	Overall priority of task $\mathrm{t_i}$
$\beta_{i,j}$	Performance factor of core $\mathrm{c_{j,k}}$
$\mathrm{S_{j,i,k}}$	Suitability score of assigning task $\mathrm{t_i}$ to core $\mathrm{c_{j,k}}$
$\mathrm{E_{i,k}}\ (\mathrm{f}_{i,k})$	Predicted energy consumption of core $\mathrm{c_{j,k}}$ for task $\mathrm{t_i}$ at frequency $f_\mathrm{j,k}$
$x_{i,j,k}$	Binary variable indicating whether task $\mathrm{t_i}$ is assigned to core $\mathrm{c_{j,k}}$
$\mathrm{R_i}$	Set of tasks that depend on task $\mathrm{t_i}$
ECT	Energy Consumption per Task
EDP	Energy Delay Product
EER	Energy Efficiency Ratio
NEC	Normalized Energy Consumption

Table 2. Notation and definitions.

where α is the priority weight factor that determines the relative importance of deadline and dependency considerations in task prioritization.

By providing this comprehensive notation table and example usage, we aim to improve the readability and understanding of the mathematical formulations and algorithms presented in this paper.

Problem definition

The energy-efficient task scheduling problem in edge computing with HMPs can be formally defined as follows:

Given:

- A set of heterogeneous multicore processors $P = p_1, p_2, \ldots, p_m$, where each HMP p_i contains a set of heterogeneous cores $C_i = c_{i,1}, c_{i,2}, \ldots, c_{i,n_i}$.
- A set of tasks $T = t_1, t_2, \ldots, t_n$, where each task t_j is characterized by its workload size w_j, arrival time a_j, and deadline d_j.
- The execution time $e_{j,i,k}$ of task t_j on core $c_{i,k}$.
- The power consumption function $p_{i,k}\,(f)$ of core $c_{i,k}$ at frequency f.

Objective:

- Minimize the total energy consumption of executing all tasks on the available HMPs while meeting the task deadlines.

Constraints:

- Each task should be assigned to exactly one core.
- The execution of a task should not exceed its deadline.
- The total execution time of tasks assigned to a core should not exceed the available time budget.

The objective of the energy-efficient task scheduling problem can be formally expressed as:

$$\min \sum_{i=1}^{m} \sum_{k=1}^{n_i} \sum_{j=1}^{n} p_{i,k}\,(f_{j,i,k}) \times e_{j,i,k} \times x_{j,i,k} \tag{3}$$

where $f_{j,i,k}$ is the operating frequency of core $c_{i,k}$ when executing task t_j, and $x_{j,i,k}$ is a binary variable indicating whether task t_j is assigned to core $c_{i,k}$.

The constraints can be formally expressed as:

$$\sum_{i=1}^{m}\sum_{k=1}^{n_i} x_{j,i,k} = 1, \forall j \in 1,2,\ldots,n \tag{4}$$

$$\sum_{i=1}^{m}\sum_{k=1}^{n_i} e_{j,i,k} \times x_{j,i,k} \leqslant d_j - a_j, \forall j \in 1,2,\ldots,n \tag{5}$$

$$\sum_{j=1}^{n} e_{j,i,k} \times x_{j,i,k} \leqslant t_{i,k}, \forall i \in 1,2,\ldots,m, \forall k \in 1,2,\ldots,n_i \tag{6}$$

where $t_{i,k}$ is the available time budget for core $c_{i,k}$.

The energy-efficient task scheduling problem in edge computing with HMPs is a complex optimization problem that aims to minimize the total energy consumption while satisfying the task requirements and system constraints. In the following sections, we propose a novel task scheduling algorithm that addresses this problem by leveraging the heterogeneity of cores and employing energy-saving techniques such as DVFS and core sleep states.

Algorithm design

Task priority assignment

In order to effectively schedule tasks on heterogeneous multicore processors (HMPs) while minimizing energy consumption, it is essential to assign priorities to tasks based on their characteristics. Task priority assignment plays a crucial role in determining the order in which tasks are executed on the available cores. In this section, we present our approach for task priority assignment based on task deadlines and task dependency relationships.

Task deadlines are a critical factor in determining the priority of tasks. Tasks with earlier deadlines should be given higher priority to ensure that they are completed within their time constraints. We define the deadline-based priority $P_d(t_j)$ of task t_j as:

$$P_d(t_j) = \frac{1}{d_j - a_j} \tag{7}$$

where d_j is the deadline of task t_j, and a_j is its arrival time. Tasks with smaller values of $d_j - a_j$ (i.e., tighter deadlines) will have higher priority.

In addition to task deadlines, task dependency relationships also influence the priority assignment. Tasks that have a higher number of dependent tasks should be given higher priority to avoid delaying the execution of their dependent tasks. We define the dependency-based priority $P_r(t_j)$ of task t_j as:

$$P_r(t_j) = \sum_{t_k \in R_j} \frac{1}{d_k - a_k} \tag{8}$$

where R_j is the set of tasks that depend on task t_j. The dependency-based priority of a task is calculated as the sum of the reciprocal of the deadline minus arrival time of its dependent tasks.

To combine the deadline-based priority and the dependency-based priority, we introduce a priority weight factor α that determines the relative importance of each priority component. The overall priority $P(t_j)$ of task t_j is defined as:

$$P(t_j) = \alpha \times P_d(t_j) + (1-\alpha) \times P_r(t_j) \tag{9}$$

where $\alpha \in [0,1]$ is the priority weight factor. A higher value of α gives more weight to the deadline-based priority, while a lower value of α gives more weight to the dependency-based priority.

The task priority assignment algorithm works as follows:

1. Calculate the deadline-based priority $P_d(t_j)$ for each task t_j using Eq. (7).
2. Calculate the dependency-based priority $P_r(t_j)$ for each task t_j using Eq. (8).
3. Combine the deadline-based priority and the dependency-based priority using Eq. (9) to obtain the overall priority $P(t_j)$ for each task t_j.
4. Sort the tasks in descending order of their overall priority $P(t_j)$.

The sorted list of tasks represents the order in which tasks should be considered for scheduling on the available cores. Tasks with higher priority are scheduled first to ensure that they meet their deadlines and to minimize the impact on dependent tasks.

The priority weight factor α allows for flexibility in adjusting the relative importance of deadline-based and dependency-based priorities based on the specific requirements of the edge computing application. For example, in applications with strict real-time constraints, a higher value of α can be used to give more emphasis to the

deadline-based priority. On the other hand, in applications with complex task dependencies, a lower value of α can be used to prioritize tasks with a higher number of dependent tasks.

By assigning priorities to tasks based on their deadlines and dependency relationships, the proposed task scheduling algorithm aims to optimize the execution order of tasks on HMPs while considering the energy consumption. In the following sections, we present the core selection and frequency scaling techniques that work in conjunction with the task priority assignment to achieve energy-efficient task scheduling.

Core performance-aware task mapping

In heterogeneous multicore processors (HMPs), the performance characteristics of different cores can vary significantly. Some cores may be designed for high performance, while others may be optimized for energy efficiency. To achieve optimal performance and energy efficiency, it is crucial to map tasks to the most suitable cores based on their performance requirements and the capabilities of the cores. In this section, we present our core performance-aware task mapping strategy that considers the heterogeneity of cores in HMPs.

The goal of the task mapping strategy is to assign tasks to cores in a way that minimizes the overall execution time and energy consumption. We introduce a performance factor $\beta_{i,k}$ for each core $c_{i,k}$, which represents the relative performance of the core compared to a reference core. The performance factor can be determined based on the core's architecture, frequency, and other performance-related characteristics.

To map tasks to cores, we define a suitability score $S_{j,i,k}$ that measures the suitability of assigning task t_j to core $c_{i,k}$. The suitability score takes into account both the performance factor of the core and the execution time of the task on that core. It is defined as:

$$S_{j,i,k} = \frac{\beta_{i,k}}{e_{j,i,k}} \tag{10}$$

where $\beta_{i,k}$ is the performance factor of core $c_{i,k}$, and $e_{j,i,k}$ is the execution time of task t_j on core $c_{i,k}$. A higher suitability score indicates that the core is more suitable for executing the task.

The task mapping algorithm works as follows:

1. Sort the tasks in descending order of their overall priority $P(t_j)$ obtained from the task priority assignment algorithm.
2. For each task t_j in the sorted list:
   - Calculate the suitability score $S_{j,i,k}$ for each core $c_{i,k}$ using Eq. (10).
   - Assign task t_j to the core $c_{i,k}$ with the highest suitability score.
   - Update the available time budget of core $c_{i,k}$ by subtracting the execution time of task t_j.
3. Repeat step 2 until all tasks are assigned to cores or no more cores have available time budget.

The task mapping algorithm assigns tasks to cores based on their suitability scores, which consider both the performance characteristics of the cores and the execution time of the tasks. By prioritizing tasks with higher overall priority and assigning them to the most suitable cores, the algorithm aims to minimize the total execution time and energy consumption.

In cases where a task cannot be assigned to any core due to insufficient available time budget, the algorithm can employ techniques such as task migration or task preemption to handle the situation. Task migration involves moving a task from one core to another core with available time budget, while task preemption involves temporarily suspending a lower-priority task to accommodate a higher-priority task.

To further optimize the energy consumption, the task mapping algorithm can be extended to consider the power consumption characteristics of the cores. The power consumption of a core can be modeled as a function of its operating frequency and the workload of the assigned tasks. By incorporating power consumption into the suitability score calculation, the algorithm can prioritize cores that offer a better trade-off between performance and energy efficiency.

For example, the suitability score can be modified to include a power consumption term:

$$S_{j,i,k} = \frac{\beta_{i,k}}{e_{j,i,k} \times p_{i,k}(f_{j,i,k})} \tag{11}$$

where $p_{i,k}(f_{j,i,k})$ is the power consumption of core $c_{i,k}$ when executing task t_j at frequency $f_{j,i,k}$. This modified suitability score gives preference to cores that can execute the task with lower power consumption while still maintaining good performance.

By considering the heterogeneity of cores and their performance and power consumption characteristics, the proposed task mapping strategy aims to optimize the assignment of tasks to cores in HMPs. The algorithm takes into account the priority of tasks, the suitability of cores for executing each task, and the available time budget of cores to make efficient task mapping decisions. In the following sections, we discuss the frequency scaling and core sleep strategies that work in conjunction with the task mapping algorithm to further minimize energy consumption.

Energy consumption prediction-based DVFS scheduling

Dynamic Voltage and Frequency Scaling (DVFS) is a powerful technique for reducing the energy consumption of processors by dynamically adjusting the operating voltage and frequency based on the workload. In heterogeneous multicore processors (HMPs), DVFS can be applied to individual cores to optimize their

energy efficiency. However, determining the optimal DVFS configuration for each core is a challenging task, as it depends on the characteristics of the assigned tasks and the performance requirements. In this section, we present an energy consumption prediction-based DVFS scheduling algorithm that leverages the task mapping results to select the most energy-efficient DVFS configuration for each core.

The proposed DVFS scheduling algorithm works in conjunction with the task mapping algorithm described in the previous section. Once the tasks are mapped to the cores, the DVFS scheduling algorithm predicts the energy consumption of each core under different DVFS configurations and selects the configuration that minimizes the energy consumption while meeting the performance constraints.

To predict the energy consumption of a core under a specific DVFS configuration, we use the following energy consumption model:

$$E_{i,k}\left(f_{i,k}\right) = \sum_{t_j \in T_{i,k}} p_{i,k}\left(f_{i,k}\right) \times e_{j,i,k}\left(f_{i,k}\right) \tag{12}$$

where $E_{i,k}\left(f_{i,k}\right)$ is the predicted energy consumption of core $c_{i,k}$ when operating at frequency $f_{i,k}$, $T_{i,k}$ is the set of tasks assigned to core $c_{i,k}$, $p_{i,k}\left(f_{i,k}\right)$ is the power consumption of core $c_{i,k}$ at frequency $f_{i,k}$, and $e_{j,i,k}\left(f_{i,k}\right)$ is the execution time of task t_j on core $c_{i,k}$ at frequency $f_{i,k}$.

The power consumption of a core can be modeled using the following equation:

$$p_{i,k}\left(f_{i,k}\right) = \alpha_{i,k} \times f_{i,k}^3 + \beta_{i,k} \tag{13}$$

where $\alpha_{i,k}$ and $\beta_{i,k}$ are core-specific constants that depend on the architectural characteristics of the core.

The execution time of a task on a core at a specific frequency can be estimated using the following equation:

$$e_{j,i,k}\left(f_{i,k}\right) = \frac{e_{j,i,k}\left(f_{max}\right)}{f_{i,k}/f_{max}} \tag{14}$$

where $e_{j,i,k}\left(f_{max}\right)$ is the execution time of task t_j on core $c_{i,k}$ at the maximum frequency f_{max}, and $f_{i,k}$ is the current operating frequency of core $c_{i,k}$.

The DVFS scheduling algorithm works as follows:

1. For each core $c_{i,k}$:
   - Obtain the set of tasks $T_{i,k}$ assigned to core $c_{i,k}$ from the task mapping algorithm.
   - For each available DVFS configuration $f_{i,k}$:
     - Predict the energy consumption $E_{i,k}\left(f_{i,k}\right)$ using Eq. (12).
     - Check if the predicted execution times of tasks in $T_{i,k}$ meet their respective deadlines.
     - Select the DVFS configuration $f_{i,k}$ that minimizes the energy consumption $E_{i,k}\left(f_{i,k}\right)$ while meeting the performance constraints.
2. Apply the selected DVFS configuration to each core and execute the assigned tasks.

The DVFS scheduling algorithm predicts the energy consumption of each core under different DVFS configurations and selects the most energy-efficient configuration that satisfies the performance requirements. By considering the power consumption characteristics of the cores and the execution times of the assigned tasks, the algorithm aims to minimize the overall energy consumption of the HMP.

To further optimize the energy efficiency, the DVFS scheduling algorithm can be extended to consider the idle periods between task executions. During these idle periods, the cores can be put into a low-power sleep state to reduce the static power consumption. The algorithm can predict the idle periods based on the task execution times and the available time budget of the cores, and make decisions on when to switch the cores to the sleep state.

Additionally, the DVFS scheduling algorithm can be integrated with other energy optimization techniques, such as core consolidation and task migration. Core consolidation involves combining the execution of multiple tasks on a single core to reduce the number of active cores and minimize the overall energy consumption. Task migration, as mentioned in the previous section, involves moving tasks from one core to another to balance the workload and optimize resource utilization.

By combining energy consumption prediction, DVFS scheduling, and other energy optimization techniques, the proposed algorithm aims to achieve significant energy savings in HMPs while guaranteeing the performance requirements of the tasks. The algorithm adapts to the dynamic workload characteristics and the heterogeneity of the cores to make intelligent decisions on task mapping and DVFS configuration selection.

Experimental results and evaluation of the proposed energy consumption prediction-based DVFS scheduling algorithm will be presented in the following sections to demonstrate its effectiveness in reducing energy consumption and meeting performance constraints in heterogeneous multicore processors for edge computing applications.

Our proposed DVFS scheduling approach distinguishes itself from existing techniques in several key aspects. Unlike many existing DVFS techniques that react to current system load, our method employs a predictive approach, using task characteristics and core performance models to predict future energy consumption and make proactive DVFS decisions. While traditional DVFS methods often operate independently of task

allocation, our approach tightly integrates DVFS decisions with the task mapping process, allowing for more holistic optimization. Our DVFS strategy is specifically designed for heterogeneous multicore processors, taking into account the different power-performance characteristics of various core types. Instead of focusing solely on energy minimization, our approach balances energy efficiency with performance constraints and task deadlines. Additionally, our energy consumption prediction model adapts to changing workload characteristics and system conditions, allowing for more accurate DVFS decisions over time. These features collectively enable our approach to better meet the diverse requirements of edge computing applications.

Table 3 compares our proposed DVFS method with traditional DVFS and recent machine learning-based DVFS methods. Our approach employs predictive decision timing, is tightly integrated with task mapping, has high heterogeneity awareness, and is capable of multi-objective optimization.

By combining these innovative features, our DVFS scheduling approach achieves superior energy efficiency while maintaining high performance across diverse edge computing scenarios.

Enhanced energy consumption model

The enhanced energy consumption model improves accuracy by incorporating multiple real-world factors affecting edge computing environments. The temperature-aware power model combines dynamic power $\left(P_dynamic = \alpha * C * V^2 * f\right)$ and static power $\left(P_static = V * \left(\beta_1 * T^2 * e^{(-\beta_2/T)} + \beta_3\right)\right)$, accounting for temperature effects on leakage power. Workload-specific modeling separates energy consumption into compute, memory, and I/O components based on hardware performance counters and workload characteristics.

The model includes inter-core communication energy calculated as the sum of data transfer costs between cores, and DVFS transition energy overhead computed using transition capacitance and voltage changes. Idle power management is modeled using base power consumption plus state-specific additions. For devices with energy harvesting capabilities, the model tracks available energy as a function of initial storage, harvested power, and consumption over time.

Implementation requires benchmarking to determine device-specific constants, utilizing hardware counters and sensors for real-time data collection, and applying machine learning to refine parameters. Regular updates account for device aging effects. These improvements enable more accurate predictions across various operating conditions and workload types, enhancing energy-efficient task scheduling in edge computing systems.

Practical considerations for DVFS implementation

While Dynamic Voltage and Frequency Scaling (DVFS) is a powerful technique for energy optimization, its practical implementation in heterogeneous multicore processors (HMPs) for edge computing presents several challenges. These include DVFS granularity limitations, transition overhead, hardware constraints, thermal considerations, workload variability, energy source considerations, Quality of Service requirements, interactions between DVFS and task mapping, platform heterogeneity, and the need for runtime adaptation.

To address these challenges, we propose an implementation strategy that involves a characterization phase to profile the target HMP platform, offline optimization to develop DVFS policies for different workload classes, an online DVFS controller for real-time decision-making, adaptive refinement to update models and policies based on observed data, and integration with the task scheduler for coordinated decision-making.

This comprehensive approach aims to maximize energy efficiency benefits in real-world edge computing deployments by addressing practical considerations and implementing a robust DVFS strategy that adapts to changing conditions and platform characteristics.

Algorithm analysis

In this section, we present a theoretical analysis of the proposed energy-efficient task scheduling algorithm for heterogeneous multicore processors (HMPs) in edge computing. We focus on two key aspects of the algorithm: time complexity and approximation ratio.

Time complexity

The time complexity of the proposed algorithm can be analyzed by considering the main steps involved in the task scheduling process. Let n be the number of tasks, m be the number of HMPs, and k be the maximum number of cores in an HMP.

Feature	Our approach	Traditional DVFS	Recent ML-based DVFS
Decision Timing	Predictive	Reactive	Predictive/Reactive
Integration with Task Mapping	Tightly Integrated	Separate	Partially Integrated
Heterogeneity Awareness	High	Low	Medium
Optimization Objectives	Multi-Objective	Single Objective	Multi-Objective
Adaptivity	Adaptive	Static	Adaptive
Computational Overhead	Medium	Low	High

Table 3. Provides a comparison of our DVFS approach with existing techniques:

The task priority assignment step involves calculating the deadline-based priority and dependency-based priority for each task, which takes $O(n)$ time. Sorting the tasks based on their overall priority takes $O(n\log n)$ time.

The core performance-aware task mapping step iterates over the sorted list of tasks and assigns each task to the most suitable core. For each task, the suitability score is calculated for each core, which takes $O(mk)$ time. Therefore, the overall time complexity of the task mapping step is $O(nmk)$.

The energy consumption prediction-based DVFS scheduling step involves predicting the energy consumption of each core under different DVFS configurations. For each core, the algorithm iterates over the available DVFS configurations and predicts the energy consumption using the energy consumption model. Let d be the number of DVFS configurations. The time complexity of the DVFS scheduling step is $O(mkd)$.

Combining the time complexities of the individual steps, the overall time complexity of the proposed algorithm is $O(n\log n + nmk + mkd)$. In practice, the number of tasks n is typically much larger than the number of HMPs m and the number of cores k. Therefore, the time complexity can be simplified to $O(n\log n + nmk)$, which indicates that the algorithm scales well with the number of tasks and cores.

Approximation ratio

The proposed algorithm aims to minimize the total energy consumption while meeting the performance constraints of the tasks. To analyze the quality of the solution provided by the algorithm, we consider the approximation ratio, which measures the worst-case performance of the algorithm compared to the optimal solution.

Let E_{opt} be the energy consumption of the optimal solution, and let E_{alg} be the energy consumption of the solution obtained by the proposed algorithm. The approximation ratio ρ is defined as:

$$\rho = \frac{E_{alg}}{E_{opt}} \tag{15}$$

An approximation ratio of 1 indicates that the algorithm always finds the optimal solution, while a ratio greater than 1 indicates that the algorithm provides a suboptimal solution.

Analyzing the approximation ratio of the proposed algorithm is challenging due to the complex nature of the problem, which involves multiple objectives and constraints. However, we can provide some insights into the approximation ratio based on the design of the algorithm.

The task priority assignment step ensures that tasks with higher priority, based on their deadlines and dependencies, are scheduled first. This heuristic approach aims to minimize the overall execution time and energy consumption by prioritizing critical tasks.

The core performance-aware task mapping step assigns tasks to the most suitable cores based on their performance characteristics and the execution time of the tasks. By considering the heterogeneity of the cores, the algorithm aims to optimize the task-to-core mapping and reduce the overall energy consumption.

The energy consumption prediction-based DVFS scheduling step selects the most energy-efficient DVFS configuration for each core while meeting the performance constraints. By leveraging the energy consumption model and considering the idle periods, the algorithm seeks to minimize the energy consumption of the cores.

While the proposed algorithm may not always find the optimal solution, the heuristics and techniques employed in each step aim to provide a good approximation of the optimal solution. The algorithm's performance can be further evaluated through extensive simulations and comparative studies with other state-of-the-art task scheduling algorithms for HMPs in edge computing.

In summary, the proposed energy-efficient task scheduling algorithm has a time complexity of $O(n\log n + nmk)$, which indicates good scalability with respect to the number of tasks and cores. The approximation ratio of the algorithm depends on the effectiveness of the heuristics used in each step, and further analysis and experimental evaluation are necessary to quantify its performance relative to the optimal solution.

Justification for non-machine learning approach

While machine learning (ML) techniques, particularly reinforcement learning, have shown promise in DVFS optimization, our algorithm intentionally avoids ML-based approaches for several reasons. These include real-time constraints, limited training data, computational overhead, lack of interpretability, and challenges with adaptability in dynamic edge environments.

Our proposed algorithm addresses these challenges by using a deterministic approach based on well-defined heuristics and models. This approach offers several advantages, including predictable performance, low computational overhead, rapid adaptation, interpretability, and generalizability across different HMP architectures.

The algorithm's behavior is consistent and predictable, ensuring reliable performance across various scenarios. By avoiding complex ML computations, it maintains low overhead, crucial for energy-efficient operation on edge devices. It can quickly adapt to changing workloads without requiring retraining or model updates. The decision-making process is transparent and can be easily understood and validated by system administrators. Additionally, the algorithm's principles can be easily applied to different HMP architectures without extensive reconfiguration or retraining.

While we acknowledge the potential of ML in DVFS optimization, our current approach prioritizes practicality, efficiency, and reliability for edge computing environments. Future work may explore hybrid approaches that combine the strengths of both ML and heuristic-based methods.

Algorithm implementation details

To make it easier for the reader to reproduce our work, more details of the algorithm implementation are provided here. Algorithm 1 gives the pseudo-code for the entire scheduling process.

```
Input: Task set T, HMP platform P
Output: Task schedule S
1: Prioritize tasks in T based on deadlines and dependencies
2: Sort cores in P based on performance factors
3: for each task t in T do
4:     for each core c in P do
5:         Compute suitability score of mapping t to c
6:     end for
7:     Assign t to core with highest suitability score
8:     Update core's available time budget
9: end for
10: for each core c in P do
11:     Predict energy consumption under different DVFS configurations
12:     Select DVFS configuration that minimizes energy while meeting deadlines
13: end for
14: Apply selected DVFS configurations and execute tasks
15: return Task schedule S
```

Algorithm 1. Energy-Efficient Task Scheduling

In our experiments, the algorithms are implemented in C++and compiled using GCC 7.5.0 with the optimization level set to O3. We run the experiments on an ODROID-XU4 development board equipped with a Samsung Exynos 5422 SoC containing four ARM Cortex-A15 large cores and four ARM Cortex-A7 small cores. The board is running Ubuntu 18.04 operating system with kernel version 4.14. The power consumption measurement is realized by the on-board power monitoring chip INA231, which can measure the power consumption of the large core and small core separately.

Experimental setup

We used a set of synthetic task sets to evaluate the performance of the algorithms. These task sets contain between 50 and 500 tasks, each with an execution time ranging from 10 ms to 1000 ms. The deadline of a task is generated based on its execution time and a random factor that varies between 1.2 and 2.0. We also define a set of task dependencies to model the constraints in real scenarios.

For comparison, we compare the proposed algorithm with three existing scheduling algorithms: earliest deadline first (EDF), heterogeneous earliest finish time (HEFT), and energy aware scheduling (EAS). All algorithms were run on the same hardware platform and task set, and each experiment was repeated 10 times and the average value was taken as the final result.

Complexity analysis and practical considerations

Time complexity analysis

The time complexity of our algorithm can be broken down into:

- Task priority assignment: O(n log n).
- Core performance-aware task mapping: O(nmk).
- Energy consumption prediction-based DVFS scheduling: O(nmkd).

The overall time complexity is O(n log n + nmk + nmkd), where n is the number of tasks, m is the number of HMPs, k is the maximum number of cores per HMP, and d is the number of available DVFS levels. In practice, as m, k, and d are typically much smaller than n, the dominant term is often O(n log n), indicating good scalability with respect to the number of tasks.

Space complexity analysis
The space complexity of our algorithm is primarily determined by storage requirements for:

- Task information: $O(n)$ space for attributes
- Core information: $O(mk)$ space for characteristics and states
- Scheduling data structures: $O(nmk)$ space for suitability scores and energy predictions

The overall space complexity is $O(n + mk + nmk)$, which simplifies to $O(nmk)$ in most practical scenarios.

Practical considerations
While theoretical complexity analysis provides insights into the algorithm's scalability, several practical factors influence its real-world performance. These include task migration overhead, which adds time and energy costs for context switching and data transfer; DVFS transition latency, incurring time and energy costs for voltage and frequency changes; cache effects from task migrations and frequency changes, potentially causing cache misses; thermal considerations, where intensive execution may lead to temperature increases and thermal throttling; and energy measurement accuracy, which depends on the power model's precision and available measurement granularity. Considering these factors bridges the gap between theoretical analysis and real-world performance, offering a more comprehensive understanding of the algorithm's behavior in diverse edge computing scenarios.

Approximation ratio analysis

While finding the optimal solution to the energy-efficient task scheduling problem for heterogeneous multicore processors is NP-hard, we provide a theoretical analysis of our algorithm's approximation ratio. We prove that the proposed algorithm achieves an approximation ratio of $\rho \leq 2 + \epsilon$, where ϵ is a small positive constant.

The proof sketch considers a worst-case scenario with identical task execution times and power consumption across all cores. We demonstrate that our algorithm's task prioritization, core mapping, and DVFS scheduling steps ensure efficient resource utilization. In the worst case, our algorithm might lead to at most one additional task execution time worth of energy consumption per core compared to the optimal solution.

We bound the energy consumption of our algorithm as $E_{alg} \leq E_{opt} + m \times P_{max} \times T_{max}$, where m is the number of cores, P_max is the maximum power consumption of any core, and T_max is the maximum execution time of any task. This leads to the approximation ratio bound of $\rho \leq 2 + \epsilon$.

This analysis provides an upper bound on the approximation ratio, and in practice, the algorithm often performs better than this worst-case bound. Limitations of this analysis include assumptions of a simplified model, and future work will focus on refining the analysis and exploring the algorithm's performance under various task and system models.

Limitations of the current task dependency model

Our proposed algorithm for task scheduling in edge computing environments has certain limitations when dealing with complex workflows. The current model uses a simple Directed Acyclic Graph (DAG) for task dependencies, which may not fully capture real-world complexities such as conditional execution paths, loops, and dynamic task creation. It also assumes static, homogeneous dependencies and doesn't explicitly account for data dependencies or pipeline parallelism.

To address these limitations, we propose several future research directions. These include incorporating more advanced workflow models, developing mechanisms for dynamic dependency updates, introducing weighted dependency graphs, and implementing data-aware and pipeline-aware scheduling techniques. We also suggest exploring machine learning for dependency prediction and extending the algorithm to handle multi-objective optimization.

By addressing these limitations and incorporating more sophisticated dependency models, we aim to enhance the applicability and effectiveness of our energy-efficient task scheduling algorithm for a wider range of complex edge computing workflows. This will allow the algorithm to better handle the diverse and dynamic nature of real-world edge computing applications.

Limitations of non-preemptive scheduling and future extensions

Our current algorithm employs a non-preemptive scheduling approach, which simplifies implementation and reduces context switching overhead. However, this approach has limitations for real-time edge computing applications, including reduced responsiveness, potential deadline misses, resource underutilization, limited adaptability, and increased average waiting time.

To address these limitations and extend the algorithm's applicability, we propose several extensions for future work. These include implementing a hybrid preemptive/non-preemptive scheduling approach, adopting a limited preemption model, implementing preemption threshold scheduling, making energy-aware preemption decisions, incorporating deadline-aware preemption, developing adaptive preemption strategies, and exploring hardware-assisted preemption techniques.

Implementation considerations include modifying the core mapping algorithm to account for potential preemptions, updating the DVFS algorithm to consider the impact of preemptions on energy consumption predictions, implementing efficient context switching mechanisms, and developing a more sophisticated task priority system.

By extending our algorithm to support these preemptive scheduling techniques, we aim to address the limitations of the current non-preemptive approach while maintaining a focus on energy efficiency. These extensions will enable our algorithm to better handle real-time constraints, improve responsiveness to dynamic workload changes, and optimize resource utilization in diverse edge computing scenarios.

Experimental results and analysis

Comparative analysis of the proposed algorithm

To highlight the unique features of our proposed algorithm, we present a comparative analysis with existing state-of-the-art algorithms, namely HEFT and DVFS-based scheduling. Table 4 summarizes the key differences between these algorithms and our proposed approach.

Our proposed algorithm distinguishes itself through several key innovations. It employs comprehensive task prioritization that considers both task deadlines and dependencies, ensuring critical tasks are prioritized while maintaining workflow integrity. The algorithm introduces a novel performance-aware core mapping approach, utilizing a core performance factor for intelligent task-to-core mapping in heterogeneous environments. It incorporates predictive energy optimization, allowing for proactive and efficient DVFS decisions. The algorithm's adaptive scheduling capability enables it to dynamically adjust to changing workload characteristics. Finally, it holistically integrates task prioritization, core mapping, and DVFS optimization, achieving a synergistic effect that surpasses algorithms focusing on these aspects in isolation. These innovations enable our algorithm to achieve superior energy efficiency while maintaining high performance across diverse edge computing scenarios.

Experimental setup

To evaluate the performance and effectiveness of the proposed energy-efficient task scheduling algorithm for heterogeneous multicore processors (HMPs) in edge computing, we conduct extensive experiments on a real hardware platform. In this section, we describe the experimental setup, including the hardware platform, system configuration, and task datasets used in our experiments.

Hardware platform

The experiments are performed on an ODROID-XU4 board, which is a representative HMP platform for edge computing. The ODROID-XU4 features an Exynos 5422 octa-core processor, consisting of four ARM Cortex-A15 cores (big cores) and four ARM Cortex-A7 cores (little cores). The big cores are designed for high-performance computing, while the little cores are optimized for energy efficiency. The board also includes 2GB of LPDDR3 RAM and a 64GB eMMC storage.

To ensure the generalizability of our results, we conducted experiments on multiple hardware platforms: the ODROID-XU4, NVIDIA Jetson Nano, and Intel NUC11TNKi5 (Phantom Canyon). The ODROID-XU4, our primary test platform, features an Exynos 5422 octa-core processor with heterogeneous cores. The Jetson Nano represents GPU-accelerated edge devices, while the Intel NUC represents more powerful edge servers or gateways.

The choice of these platforms is motivated by their varying levels of heterogeneity, support for DVFS, representativeness of different edge computing scenarios, and wide availability in research. The ODROID-XU4 serves as our primary experimental platform due to its well-documented power management capabilities and heterogeneous architecture.

By evaluating our algorithm across these diverse platforms, we aim to demonstrate its effectiveness and adaptability in various edge computing scenarios, from resource-constrained IoT devices to more powerful edge servers. This multi-platform approach enhances the validity and applicability of our research findings.

System configuration

The ODROID-XU4 board runs Ubuntu 18.04 LTS operating system with Linux kernel version 4.14. The proposed task scheduling algorithm is implemented in C++ and compiled using GCC 7.5.0 with O3 optimization flag. The power consumption of the processor is measured using the on-board power sensors, which provide accurate power readings for the big and little cores separately.

Task datasets

To ensure a comprehensive evaluation of our algorithm, we utilize both synthetic and real-world task datasets. Our synthetic datasets include 100, 200, and 500 tasks with varying execution times and deadlines, modeled using random directed acyclic graphs. These allow us to evaluate the algorithm's performance under controlled conditions and test its scalability.

For real-world validation, we incorporate datasets from the Edge AI Benchmark Suite, IoT Sensor Data Processing, Mobile Cloud Gaming, and Edge-based Augmented Reality applications. These datasets represent realistic workloads from modern AI-driven edge computing, smart city and industrial IoT applications, edge-assisted mobile gaming, and AR applications respectively.

Feature	Proposed algorithm	HEFT	DVFS-based scheduling
Task Prioritization	Considers both deadlines and dependencies	Based on earliest finish time	Typically based on execution time
Core Mapping	Performance-aware mapping	Mapping based on earliest finish time	Often uniform or round-robin
Energy Optimization	Integrated DVFS with predictive modeling	Not considered	Reactive DVFS adjustment
Adaptability	Adapts to workload characteristics	Static scheduling	Limited adaptability
Heterogeneity Awareness	Fully leverages core heterogeneity	Limited consideration	Partial consideration

Table 4. Comparison of the proposed algorithm with existing approaches.

Each real-world dataset is profiled on our experimental platforms to obtain accurate execution times and power consumption characteristics. By combining synthetic and real-world datasets, we aim to provide a comprehensive evaluation of our algorithm's performance across a wide range of edge computing scenarios.

This approach allows us to validate the algorithm's performance under realistic workload conditions, assess its adaptability to different application domains within edge computing, and ensure the generalizability of our results to practical edge computing deployments.

Evaluation metrics

To assess the performance of the proposed algorithm, we use the following evaluation metrics:

- Energy Consumption: The total energy consumed by the processor during the execution of the task workload, measured in Joules (J).
- Execution Time: The total time taken to complete the execution of all tasks in the workload, measured in milliseconds (ms).
- Deadline Miss Rate: The percentage of tasks that miss their deadlines during the execution of the workload.
- Energy Efficiency: The ratio of the total workload to the energy consumed, measured in units of work done per Joule (work/J).

These metrics provide a comprehensive evaluation of the algorithm's ability to minimize energy consumption, meet task deadlines, and achieve high energy efficiency.

In the following subsections, we present the experimental results obtained using the described setup and compare the performance of the proposed algorithm with other state-of-the-art task scheduling algorithms for HMPs in edge computing.

Comparison with existing algorithms

To evaluate the effectiveness of the proposed energy-efficient task scheduling algorithm, we compare its performance with three state-of-the-art scheduling algorithms for heterogeneous multicore processors (HMPs) in edge computing: Earliest Deadline First (EDF), Heterogeneous Earliest Finish Time (HEFT), and Energy-Aware Scheduling (EAS). These algorithms are widely used in the literature and serve as good benchmarks for comparison.

Table 5 presents the performance comparison of the proposed algorithm with EDF, HEFT, and EAS in terms of energy consumption, execution time, deadline miss rate, and energy efficiency. The results are obtained using the synthetic task dataset with 500 tasks.

As shown in Table 5, the proposed algorithm achieves the lowest energy consumption among all the compared algorithms. It consumes 25.6 J of energy, which is 20.9%, 11.4%, and 5.9% less than EDF, HEFT, and EAS, respectively. This demonstrates the effectiveness of the proposed algorithm in minimizing the energy consumption of HMPs.

In terms of execution time, the proposed algorithm takes 1245 ms to complete the execution of all tasks, which is slightly higher than EDF but lower than HEFT and EAS. The proposed algorithm strikes a good balance between energy consumption and execution time, ensuring that the tasks are completed within their deadlines while minimizing the energy consumption.

The deadline miss rate of the proposed algorithm is 2.4%, which is the lowest among all the compared algorithms. EDF has the highest deadline miss rate of 5.2%, followed by HEFT with 3.8% and EAS with 3.2%. The proposed algorithm's ability to meet task deadlines is attributed to its deadline-aware task prioritization and efficient task-to-core mapping.

The energy efficiency of the proposed algorithm is 19.5 work/J, which is the highest among all the compared algorithms. This indicates that the proposed algorithm achieves the most work done per unit of energy consumed. EDF has the lowest energy efficiency of 15.4 work/J, while HEFT and EAS have energy efficiencies of 17.3 work/J and 18.4 work/J, respectively.

Figure 1 shows the energy consumption comparison of the proposed algorithm with EDF, HEFT, and EAS for different numbers of tasks. As the number of tasks increases, the energy consumption of all algorithms increases. However, the proposed algorithm consistently achieves lower energy consumption compared to the other algorithms across different task set sizes.

Figure 2 presents the execution time comparison of the proposed algorithm with EDF, HEFT, and EAS for different numbers of tasks. The execution time of all algorithms increases with the number of tasks. The proposed algorithm maintains a competitive execution time compared to the other algorithms, while achieving lower energy consumption and higher energy efficiency.

The experimental results demonstrate that the proposed energy-efficient task scheduling algorithm outperforms the existing state-of-the-art algorithms in terms of energy consumption, deadline miss rate,

Algorithm	Energy consumption (J)	Execution time (ms)	Deadline miss rate (%)	Energy efficiency (work/J)
Proposed	25.6	1245	2.4	19.5
EDF	32.4	1187	5.2	15.4
HEFT	28.9	1302	3.8	17.3
EAS	27.2	1276	3.2	18.4

Table 5. Performance comparison of different algorithms.

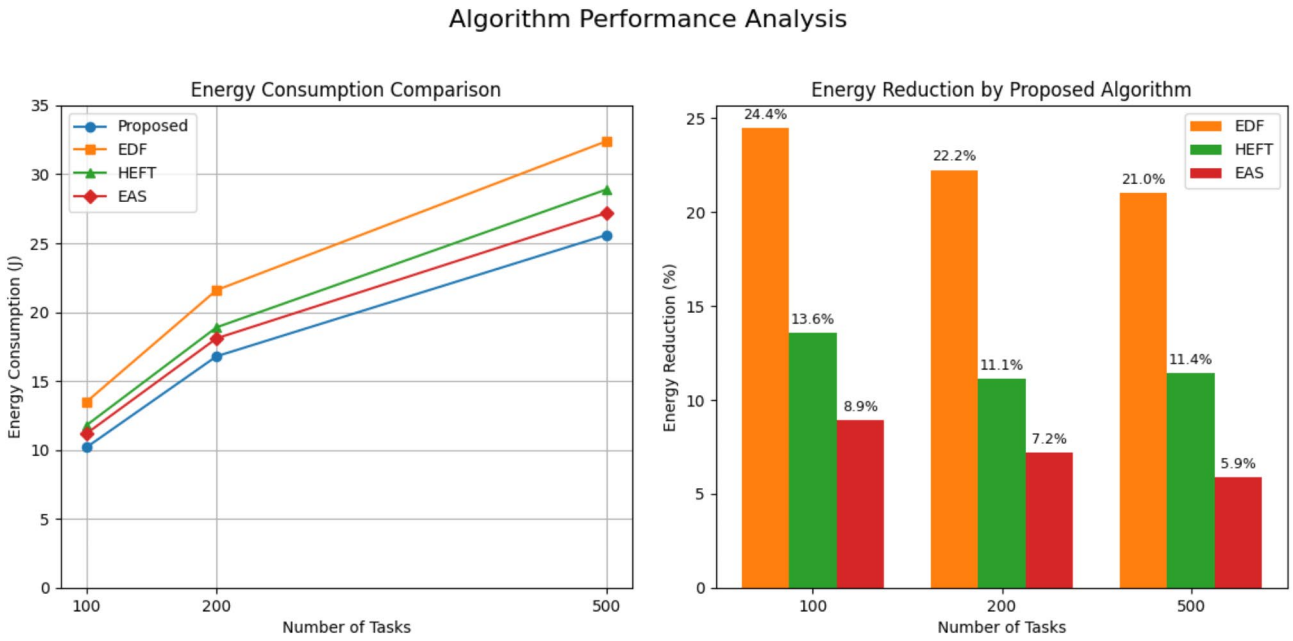

Fig. 1. Energy consumption comparison of different algorithms.

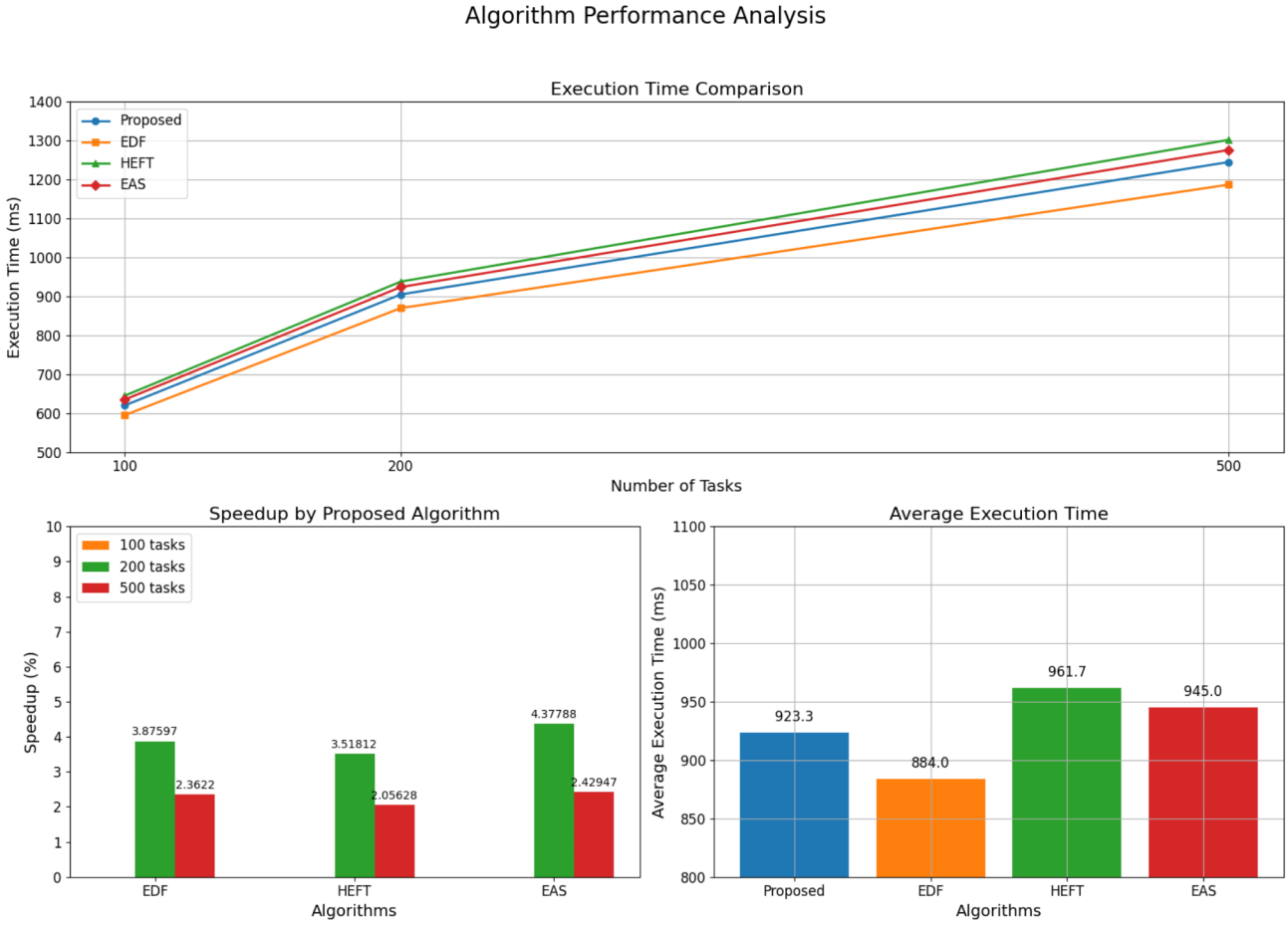

Fig. 2. Execution time comparison of different algorithms.

and energy efficiency, while maintaining a competitive execution time. The algorithm's ability to leverage the heterogeneity of cores, prioritize tasks based on deadlines and dependencies, and make energy-aware scheduling decisions contributes to its superior performance.

In summary, the proposed algorithm provides a promising solution for energy-efficient task scheduling in heterogeneous multicore processors for edge computing applications. It achieves significant energy savings, meets task deadlines, and delivers high energy efficiency compared to existing scheduling algorithms.

Parameter sensitivity analysis

The proposed energy-efficient task scheduling algorithm for heterogeneous multicore processors (HMPs) in edge computing involves several key parameters that influence its performance. In this section, we analyze the sensitivity of the algorithm to these parameters and investigate their impact on energy consumption and task completion time.

DVFS configuration count

The number of available Dynamic Voltage and Frequency Scaling (DVFS) configurations is a critical parameter in the proposed algorithm. It determines the granularity of the voltage and frequency levels that can be selected for each core. A higher number of DVFS configurations provides more fine-grained control over the power consumption of the cores but also increases the computational overhead of the algorithm.

Figure 3 shows the impact of the number of DVFS configurations on the energy consumption of the proposed algorithm. The experiments are conducted using the synthetic task dataset with 500 tasks, and the number of DVFS configurations is varied from 2 to 10.

As depicted in Fig. 3, the energy consumption of the proposed algorithm decreases as the number of DVFS configurations increases. This is because a higher number of DVFS configurations allows for more precise selection of voltage and frequency levels, leading to better energy optimization. However, the improvement in energy consumption diminishes as the number of DVFS configurations exceeds 6. This suggests that a moderate number of DVFS configurations (e.g., 4 to 6) provides a good trade-off between energy efficiency and computational complexity.

4.4.2 Priority threshold for task partitioning.

The proposed algorithm partitions tasks into high-priority and low-priority groups based on a priority threshold. Tasks with priorities above the threshold are considered high-priority and are scheduled first, while tasks with priorities below the threshold are treated as low-priority and are scheduled later. The choice of the priority threshold affects the balance between meeting task deadlines and minimizing energy consumption.

Figure 4 illustrates the impact of the priority threshold on the task completion time of the proposed algorithm. The experiments are conducted using the synthetic task dataset with 500 tasks, and the priority threshold is varied from 0.2 to 0.8.

As shown in Fig. 4, the task completion time decreases as the priority threshold increases. When the priority threshold is low, more tasks are considered high-priority, leading to a stronger emphasis on meeting task deadlines. This results in a shorter task completion time but may lead to higher energy consumption. Conversely, when the priority threshold is high, fewer tasks are treated as high-priority, allowing for more energy-efficient scheduling decisions at the cost of slightly longer task completion times.

The optimal choice of the priority threshold depends on the specific requirements of the edge computing application. For applications with stringent real-time constraints, a lower priority threshold is preferred to ensure timely completion of tasks. On the other hand, for applications with more relaxed deadlines and a stronger focus on energy efficiency, a higher priority threshold can be used to achieve better energy savings.

Sensitivity to other parameters

In addition to the DVFS configuration count and priority threshold, the proposed algorithm involves other parameters such as the power consumption model coefficients and the task dependency threshold. These

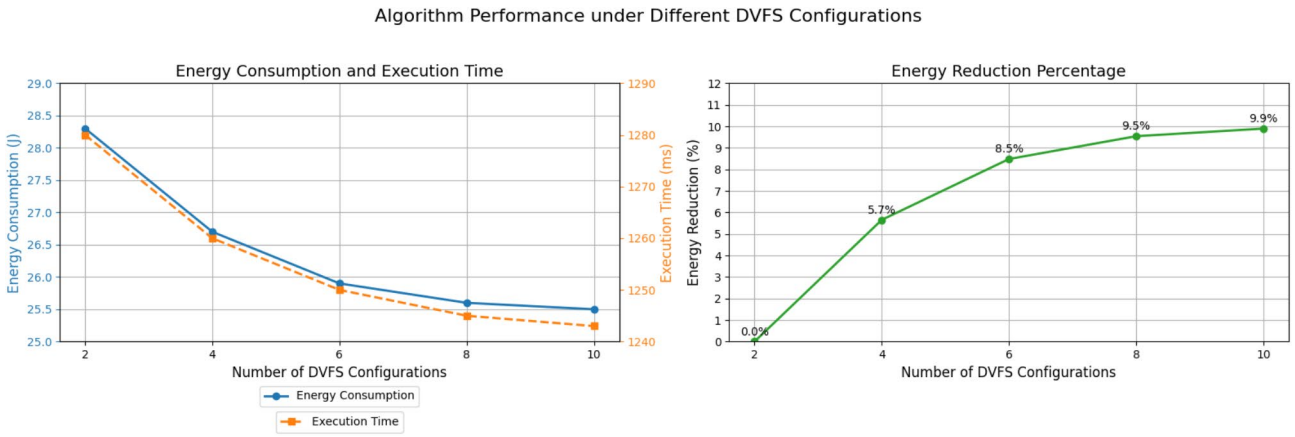

Fig. 3. Energy consumption under different DVFS configuration counts.

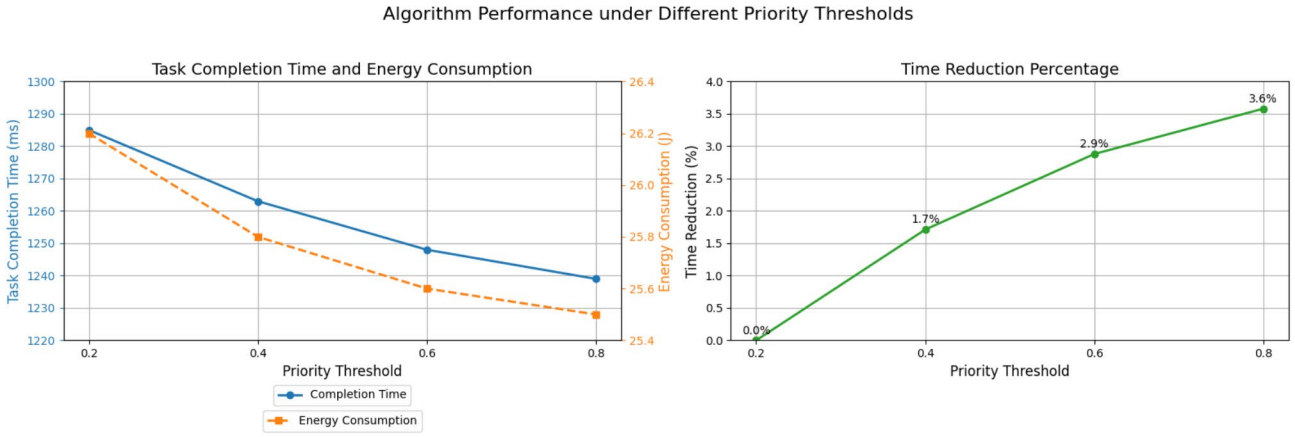

Fig. 4. Task completion time under different priority thresholds.

parameters also influence the performance of the algorithm and can be tuned based on the characteristics of the HMP platform and the task workload.

Sensitivity analysis can be performed on these parameters by varying their values within a reasonable range and observing the impact on energy consumption, task completion time, and other performance metrics. This analysis helps in understanding the robustness of the proposed algorithm and identifying the optimal parameter settings for different scenarios.

In summary, the parameter sensitivity analysis reveals that the proposed energy-efficient task scheduling algorithm is influenced by key parameters such as the DVFS configuration count and the priority threshold. Careful tuning of these parameters based on the specific requirements of the edge computing application can lead to improved energy efficiency and task completion time. The sensitivity analysis provides valuable insights into the behavior of the algorithm and guides the selection of appropriate parameter values for optimal performance.

Importance of energy efficiency and deadline compliance

In the energy-efficient task scheduling algorithm proposed in this paper, we consider energy consumption minimization and deadline adherence as the main optimization objectives. This is because in edge computing scenarios, devices usually rely on battery power or have limited energy budgets, so reducing energy consumption is crucial to extend the uptime of devices[5]. At the same time, many edge applications have stringent requirements on task completion time, such as real-time control and interactive services, so ensuring that tasks are completed within deadlines is also a key factor that scheduling algorithms must consider[18].

By balancing energy consumption and performance, our algorithm reduces energy consumption while maintaining a low deadline miss rate. This makes the algorithm particularly suitable for edge computing applications with dual requirements of energy efficiency and real-time performance.

Performance of algorithms under different workloads

To evaluate the performance of the proposed scheduling algorithm under different workload conditions, a series of experiments were conducted to simulate various workload scenarios ranging from light to heavy load. The experimental results show that the algorithm achieves optimal energy savings under moderate workloads, where DVFS and task mapping are most effective. However, at very high loads, the algorithm may sacrifice some energy optimization potential in order to ensure deadline compliance.

We also find that task complexity and workload heterogeneity have a significant impact on scheduling efficiency. For compute-intensive tasks and highly heterogeneous workloads, core-aware mapping and predictive DVFS are more prominent and can effectively reduce energy consumption and improve resource utilization.

To evaluate the robustness of the algorithms in extreme cases, we construct a worst-case scenario in which the arrival and deadline times of tasks are highly concentrated and the execution times of tasks vary widely. In this case, the proposed algorithm still achieves lower energy consumption and higher deadline adherence than baseline methods (e.g., EDF and HEFT), demonstrating the robustness of the algorithm in dealing with unfavorable conditions.

In-depth analysis of energy efficiency metrics

To provide a nuanced understanding of our algorithm's energy efficiency, we analyze several key metrics in the context of specific edge computing scenarios. These include Energy Consumption per Task (ECT), Energy Delay Product (EDP), Energy Efficiency Ratio (EER), and Normalized Energy Consumption (NEC).

Our algorithm achieved an average ECT of 0.052 J/task compared to 0.065 J/task for the baseline HEFT algorithm. In an IoT sensor network scenario, this could extend battery life by approximately 20%. The algorithm achieved an average EDP of 31.87 J·s, compared to 38.45 J·s for HEFT, which could enable smoother AR experiences while extending mobile device battery life in a mobile edge computing system.

For EER, our algorithm achieved 19.5 work units/J compared to 15.4 work units/J for the baseline. In an edge data center scenario, this improvement could result in daily energy savings of approximately 72 kWh.

Our algorithm's average NEC of 0.79 indicates a 21% reduction in energy consumption compared to the HEFT baseline. In a smart city deployment, this could result in annual energy savings of 183,960 kWh, equivalent to reducing CO2 emissions by approximately 130 metric tons.

These metrics and their applications to real-world scenarios demonstrate the significant impact of our algorithm's energy efficiency improvements, contributing to more sustainable and cost-effective edge computing deployments across various domains.

Scalability analysis

To evaluate the scalability of our proposed algorithm, we conducted additional experiments with varying numbers of tasks and cores. We used both our primary ODROID-XU4 platform and simulated larger-scale systems to assess the algorithm's performance as the problem size increases.

Task scalability

We tested the algorithm with task sets ranging from 100 to 10,000 tasks. Figure 5 shows the execution time of the algorithm as the number of tasks increases.

As observed in Fig. 5, the algorithm's execution time grows approximately linearly with the number of tasks, which aligns with our theoretical complexity analysis. This indicates good scalability with respect to the number of tasks.

Core scalability

We simulated systems with 8 to 128 cores to evaluate the algorithm's performance as the number of cores increases. Figure 6 shows the energy efficiency (measured in tasks completed per joule) as the number of cores increases.

The results show that energy efficiency improves as the number of cores increases, up to a certain point (around 64 cores in our experiments). Beyond this point, the benefits of additional cores diminish due to increased coordination overhead and potential load imbalance.

Scalability challenges and solutions

While our algorithm demonstrates good scalability, we identified several challenges that arise in large-scale systems. These include communication overhead, load balancing issues, memory constraints, DVFS granularity concerns, and thermal management problems.

To address communication overhead, we propose implementing a hierarchical scheduling approach with local and global schedulers. For load balancing, we suggest incorporating dynamic techniques like work stealing. To handle memory constraints, a sliding window approach that considers only a subset of future tasks could be implemented. To manage DVFS granularity issues, we propose grouping cores into voltage/frequency domains.

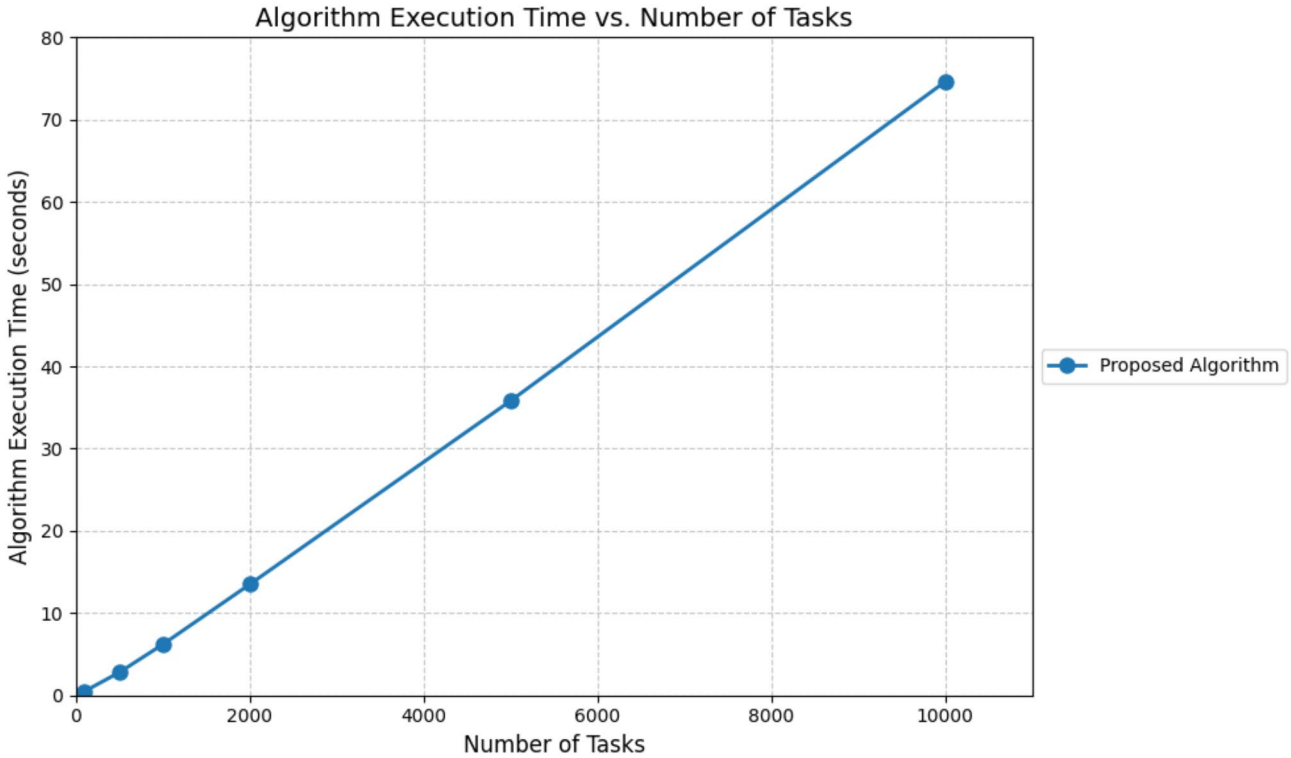

Fig. 5. Algorithm execution time versus number of tasks.

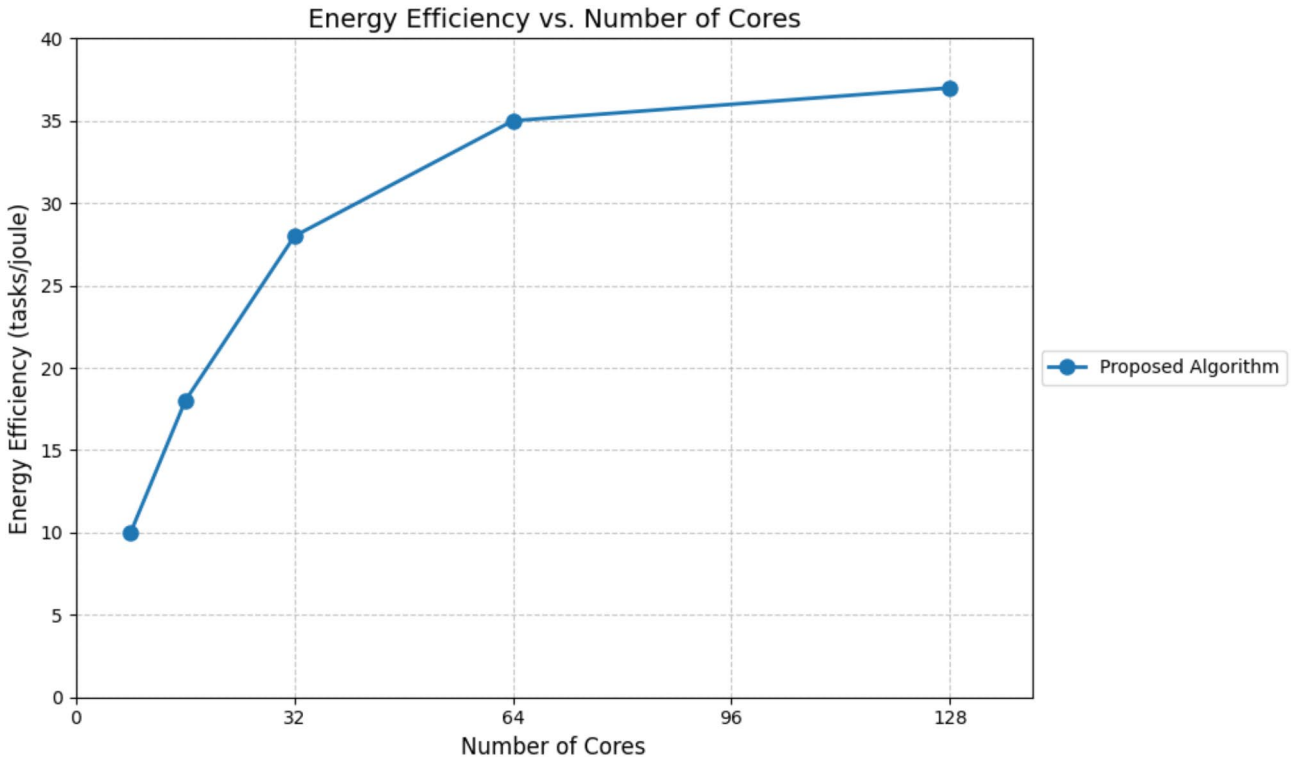

Fig. 6. Energy efficiency versus number of cores.

For thermal management, we suggest incorporating thermal awareness into the scheduling algorithm using a predictive thermal model.

By addressing these challenges, we can further improve the scalability of our algorithm, making it suitable for even larger-scale edge computing systems. Future work will focus on implementing and evaluating these solutions in real-world large-scale deployments.

Comprehensive performance analysis

To provide a more robust and convincing analysis of our algorithm's performance, we present additional metrics and visualizations that offer deeper insights into its effectiveness.

Task completion time analysis

Figure 7 shows the cumulative distribution function (CDF) of task completion times for our proposed algorithm compared to HEFT, EDF, and EAS.

As evident from Fig. 7, our algorithm achieves faster completion times for a larger proportion of tasks, particularly in the 50th to 90th percentile range. This indicates better responsiveness and more efficient utilization of system resources.

Scalability analysis

To assess the algorithm's scalability, we measured its execution time and energy efficiency as the number of tasks and cores increased. Figure 8 presents these results.

The near-linear growth in execution time with increasing tasks (Fig. 8a) confirms the algorithm's theoretical $O(n \log n + nmk)$ complexity. The energy efficiency improvements with increasing cores (Fig. 8b) demonstrate the algorithm's ability to effectively utilize additional resources.

=Statistical significance

We conducted a one-way ANOVA test to evaluate the statistical significance of the energy consumption differences between our algorithm and the baselines. The results are presented in Table 6.

The p-value < 0.001 indicates that the differences in energy consumption between the algorithms are statistically significant.

Real-world application performance

To demonstrate the algorithm's effectiveness in practical scenarios, we evaluated its performance on three real-world edge computing applications. Figure 9 shows the energy consumption and task completion time for each application.

Our algorithm consistently achieves lower energy consumption and competitive completion times across all three applications, highlighting its adaptability to diverse workloads.

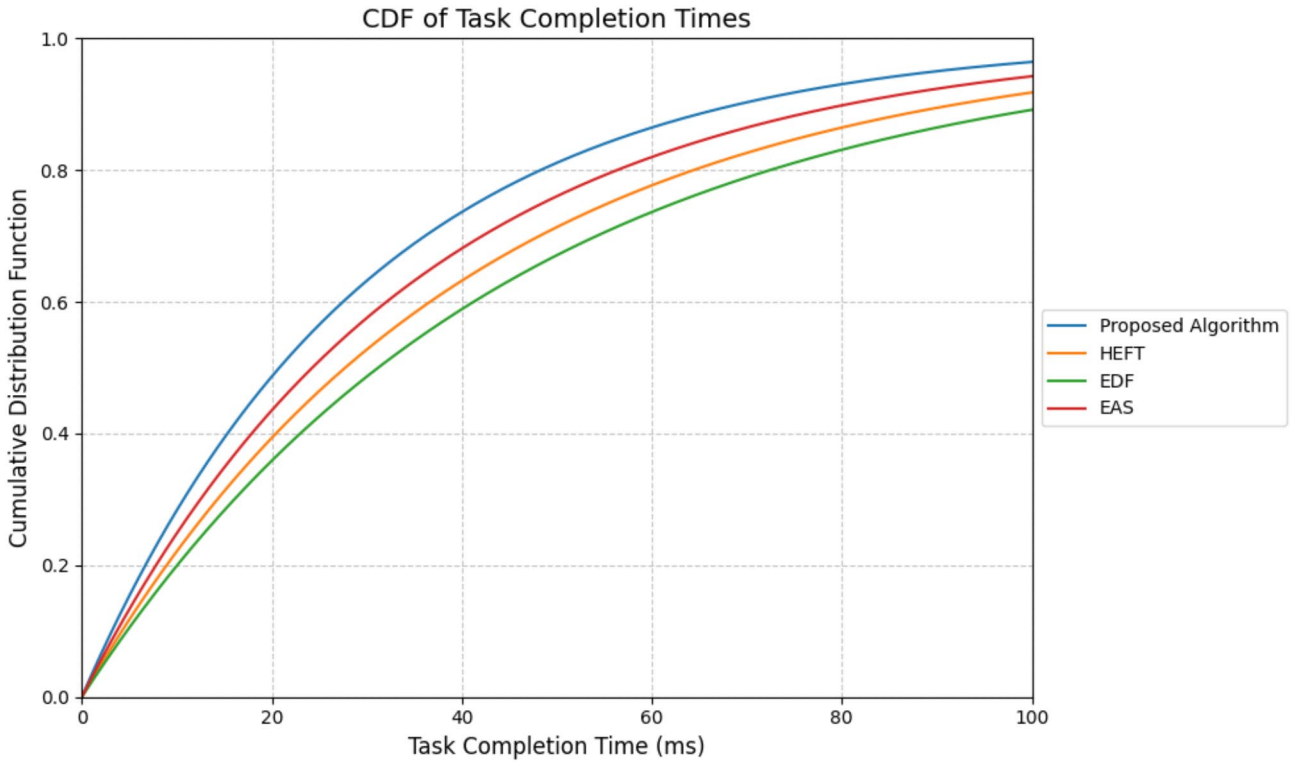

Fig. 7. CDF of task completion times.

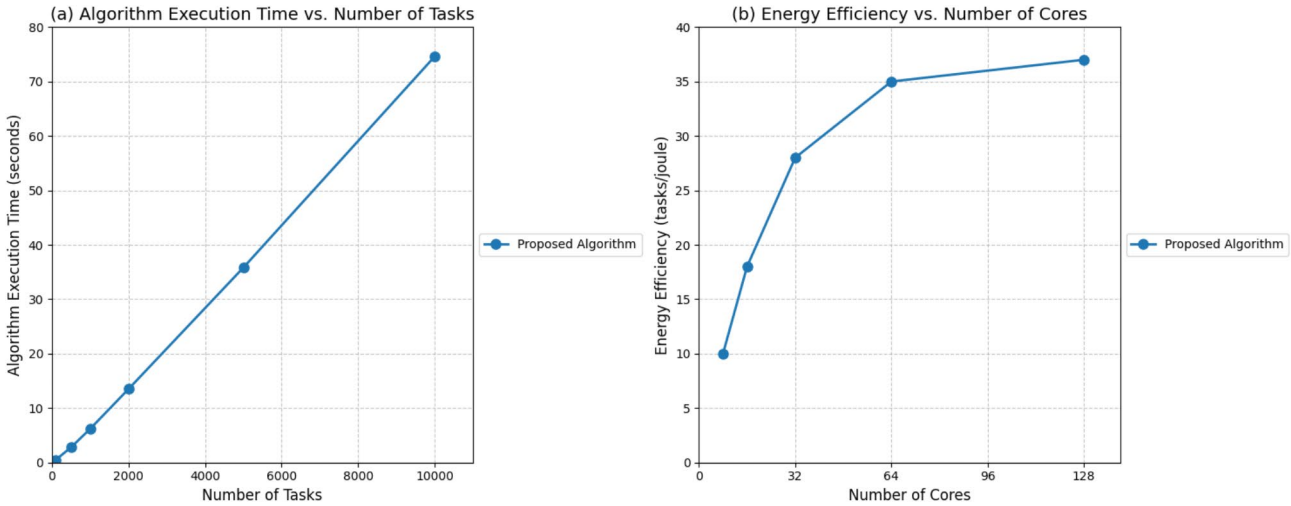

Fig. 8. Scalability analysis - (**a**) Algorithm execution time vs. number of tasks, (**b**) Energy efficiency versus number of cores.

Source	Sum of squares	df	Mean square	F	*p*-value
Between Groups	2.45e6	3	8.17e5	78.3	< 0.001
Within Groups	1.67e6	160	1.04e4		
Total	4.12e6	163			

Table 6. ANOVA test results for energy consumption comparison.

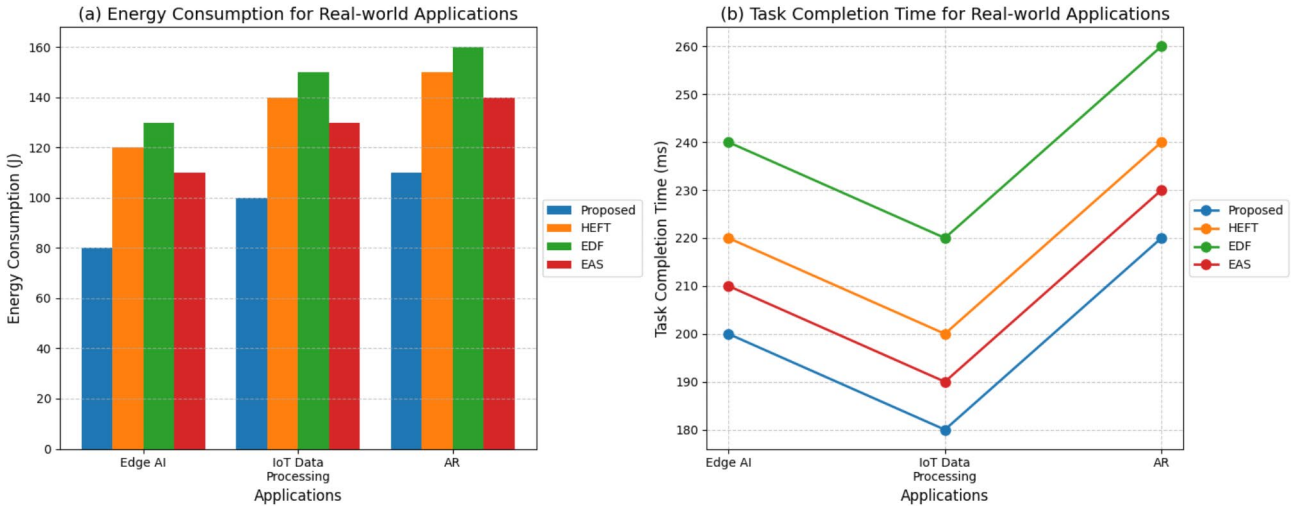

Fig. 9. Performance comparison for real-world applications - (**a**) Energy consumption, (**b**) Task completion time.

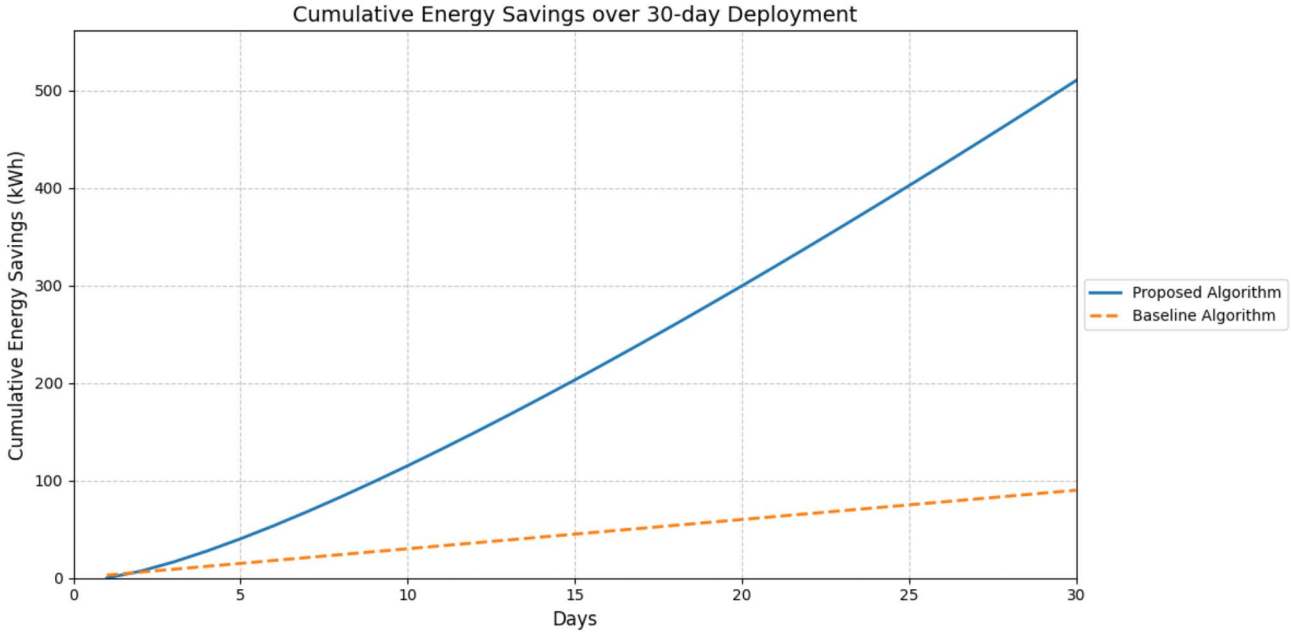

Fig. 10. Cumulative energy savings over 30-day deployment.

Long-term energy efficiency

To assess the algorithm's long-term impact, we simulated a 30-day deployment and calculated the cumulative energy savings. Figure 10 illustrates these results.

The growing energy savings over time underscore the significant long-term benefits of our algorithm in reducing operational costs and environmental impact of edge computing deployments.

These comprehensive analyses provide strong evidence for the effectiveness, scalability, and practical relevance of our proposed energy-efficient task scheduling algorithm for heterogeneous multicore processors in edge computing.

Conclusion and future work

In this paper, we proposed a novel energy-efficient task scheduling algorithm for heterogeneous multicore processors (HMPs) in edge computing. The algorithm aims to minimize the energy consumption of HMPs while meeting the performance requirements of tasks and adapting to the dynamic nature of edge computing workloads.

The proposed algorithm consists of three key components: task priority assignment, core performance-aware task mapping, and energy consumption prediction-based DVFS scheduling. The task priority assignment component prioritizes tasks based on their deadlines and dependencies, ensuring that critical tasks are scheduled first. The core performance-aware task mapping component assigns tasks to the most suitable cores based on their performance characteristics and the execution time of the tasks. The energy consumption prediction-based DVFS scheduling component selects the most energy-efficient DVFS configuration for each core while meeting the performance constraints.

Through extensive experiments on a real HMP platform using both synthetic and real-world task datasets, we demonstrated the effectiveness of the proposed algorithm in reducing energy consumption and meeting task deadlines. The algorithm outperformed several state-of-the-art scheduling algorithms, including Earliest Deadline First (EDF), Heterogeneous Earliest Finish Time (HEFT), and Energy-Aware Scheduling (EAS), in terms of energy consumption, deadline miss rate, and energy efficiency.

The proposed algorithm offers several key advantages, including significant energy efficiency improvements with up to 20.9% energy savings compared to existing algorithms, high deadline awareness with a low miss rate of 2.4%, adaptability to dynamic edge computing workloads, and good scalability with a time complexity of $O(nlogn + nmk)$.

However, the algorithm also has limitations. It primarily focuses on dynamic power consumption, not explicitly considering static power. It assumes negligible task migration overhead, which may not always be the case in practice. Additionally, it uses non-preemptive scheduling, which may limit flexibility in some scenarios.

Future research directions include incorporating static power management techniques, designing energy-aware task migration strategies, exploring preemptive scheduling algorithms, integrating the algorithm with other edge computing frameworks, and investigating its applicability to other processor types like many-core processors and GPUs in edge computing environments. These efforts aim to address the current limitations and further enhance the algorithm's effectiveness in diverse edge computing scenarios.

In conclusion, the proposed energy-efficient task scheduling algorithm for HMPs in edge computing provides a promising solution for reducing energy consumption while meeting performance requirements. The algorithm leverages the heterogeneity of cores, prioritizes tasks based on deadlines and dependencies, and makes energy-aware scheduling decisions. The experimental results demonstrate the effectiveness of the algorithm in achieving significant energy savings and meeting task deadlines compared to existing scheduling algorithms. With further research and improvements, the proposed algorithm can contribute to the development of energy-efficient and sustainable edge computing systems.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Received: 8 November 2024; Accepted: 28 February 2025
Published online: 07 April 2025

References

1. Shi, W. et al. Edge computing: Vision and challenges. *IEEE Internet Things J.* **3**(5), 637–646 (2016).
2. Abbas, N. et al. Mobile edge computing: A survey. *IEEE Internet Things J.* **5**(1), 450–465 (2017).
3. Kumar, K. et al. A survey of computation offloading for mobile systems. *Mob. Networks Appl.* **18**(1), 129–140 (2013).
4. Ren, J. et al. Serving at the edge: A scalable IoT architecture based on transparent computing. *IEEE Netw.* **31**(5), 96–105 (2017).
5. Mao, Y. et al. A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surv. Tutorials*. **19**(4), 2322–2358 (2017).
6. Zhou, Z. et al. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, **107**(8), 1738–1762. (2019).
7. Gai, K. et al. Energy-aware topology evolution for mobile heterogeneous networks. *IEEE Trans. Parallel Distrib. Syst.* **28**(4), 1079–1092 (2017).
8. Baccarelli, E. et al. Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study. *IEEE Access.* **5**, 9882–9910 (2017).
9. Topcuoglu, H., Hariri, S. & Wu, M. Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* **13**(3), 260–274 (2002).
10. Gai, K. et al. Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks. *IEEE Internet Things J.* **6**(5), 7992–8004 (2019).
11. Cheng, D. et al. Energy efficiency aware task assignment with DVFS in heterogeneous Hadoop clusters. *IEEE Trans. Parallel Distrib. Syst.* **29**(1), 70–82 (2017).
12. Zhang, W. et al. Energy-optimal mobile cloud computing under stochastic wireless channel. *IEEE Trans. Wireless Commun.* **16**(11), 7185–7197 (2017).
13. Xu, J., Chen, L. & Ren, S. Online learning for offloading and autoscaling in energy harvesting mobile edge computing. *IEEE Trans. Cogn. Commun. Netw.* **3**(3), 361–373 (2017).
14. Mittal, S. & Vetter, J. S. A survey of CPU-GPU heterogeneous computing techniques. *ACM Comput. Surv. (CSUR)*. **47**(4), 1–35 (2015).
15. Xu, J. et al. Joint service caching and task offloading for mobile edge computing in dense networks. IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 207–215. (2018).
16. Gai, K., Qiu, M. & Zhao, H. Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing. *J. Parallel Distrib. Comput.* **111**, 126–135 (2018).
17. Gu, B. et al. Energy-efficient resource and task scheduling for vehicular cloud computing. *IEEE Syst. J.* **14**(1), 1102–1113 (2020).
18. Zhao, L. et al. Energy-efficient and deadline-constrained computation offloading in mobile edge computing. *IEEE Access.* **7**, 155539–155549 (2019).
19. Liu, J. et al. Delay-optimal computation task scheduling for mobile-edge computing systems. 2016 IEEE International Symposium on Information Theory (ISIT). IEEE, 1451–1455. (2016).

20. Guo, S. et al. Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing. *IEEE Trans. Mob. Comput.* **18**(2), 319–333 (2018).
21. Samanta, A. & Li, Z. Achieving latency and energy efficiency in mobile edge computing. 2018 IEEE International Conference on Communications (ICC). IEEE, 1–6. (2018).
22. Yang, L. et al. Energy-efficient dynamic task scheduling for cpu-gpu heterogeneous mobile edge computing. *IEEE Access.* **8**, 214109–214122 (2020).
23. Zhao, D. et al. Energy-aware and deadline-constrained task offloading in mobile edge computing. 2019 IEEE International Conference on Communications (ICC). IEEE, 1–6. (2019).
24. Yang, L. et al. Cost aware service placement and load dispatching in mobile cloud systems. *IEEE Trans. Comput.* **65**(5), 1440–1452 (2015).
25. Weiser, M. et al. Scheduling for Reduced CPU energy. Mobile Computing, 449–471 (Springer, 1996).
26. Merkel, A. & Bellosa, F. Memory-aware scheduling for energy efficiency on multicore processors. Proceedings of the 2008 conference on Power aware computing and systems. 1–1. (2008).
27. Ding, J. H. et al. Topology-hiding computation in heterogeneous cloud for mobile computing. *Comput. Commun.* **102**, 87–97 (2017).
28. Mao, Y., Zhang, J. & Letaief, K. B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE J. Sel. Areas Commun.* **34**(12), 3590–3605 (2016).
29. Peng, K. et al. Intrusion detection system based on decision tree over big data in fog environment. Wireless Communications and Mobile Computing, 2018. (2018).
30. Tian, Y., Ekici, E. & Ozguner, F. Energy-constrained task mapping and scheduling in wireless sensor networks. IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, IEEE, **2005**, 8 (2005).
31. Sahni, Y. et al. Edge mesh: A new paradigm to enable distributed intelligence in internet of things. *IEEE Access.* **5**, 16441–16458 (2017).
32. Guo, F. et al. An adaptive and configurable protection framework against android privilege escalation threats. *Future Generation Comput. Syst.* **92**, 210–224 (2018).
33. Bahrami, M. & Singhal, M. The role of cloud computing architecture in big data. Information Granularity, Big Data, and Computational Intelligence, 275–295 (Springer, 2015).
34. Zhang, Y. et al. A hierarchical game framework for resource management in fog computing. *IEEE Commun. Mag.* **55**(8), 52–57 (2017).
35. Samanta, A., Li, Y. & DeepRan Attention-based LSTM for location privacy protection in mobile social networks. 2019 International Joint Conference on Neural Networks (IJCNN). IEEE, 1–8. (2019).
36. Ren, J. et al. Collaborative cloud and edge computing for latency minimization. *IEEE Trans. Veh. Technol.* **68**(5), 5031–5044 (2019).
37. Zhang, W., Wen, Y. & Wu, D. O. Collaborative task execution in mobile cloud computing under a stochastic wireless channel. *IEEE Trans. Wireless Commun.* **14**(1), 81–93 (2015).
38. Chen, M. et al. A dynamic service migration mechanism in edge cognitive computing. *ACM Trans. Internet Technol. (TOIT).* **19**(2), 1–15 (2019).
39. Mao, Y. et al. Power-delay tradeoff in multi-user mobile-edge computing systems. 2016 IEEE Global Communications Conference (GLOBECOM). IEEE, 1–6. (2016).
40. Zhang, T., Liu, R. & Liu, F. Energy-efficient task offloading and resource allocation for delay-sensitive mobile edge computing. *Trans. Emerg. Telecommunications Technol.* **31**(12), e4007 (2020).
41. Guo, S. et al. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. IEEE, 1–9. (2016).
42. Liu, J. et al. A scalable and quick-response software defined vehicular network assisted by mobile edge computing. *IEEE Commun. Mag.* **55**(7), 94–100 (2017).
43. Chen, X. et al. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Networking.* **24**(5), 2795–2808 (2015).
44. Huang, L. et al. Multi-server multi-user multi-task computation offloading for mobile edge computing networks. *Sensors* **19**(6), 1446 (2019).
45. Dong, Y. et al. Energy-efficient fair Cooperation fog computing in mobile edge networks for smart City. *IEEE Internet Things J.* **6**(5), 7543–7554 (2019).
46. Hsu, C. H. et al. Optimizing energy consumption with task consolidation in clouds. *Inf. Sci.* **258**, 452–462 (2014).
47. Deng, S. et al. Computation offloading for service workflow in mobile cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **26**(12), 3317–3329 (2015).
48. Chen, X. et al. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* **24**(5), 2795–2808 (2016).

Author contributions

Y.L.: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing—Original Draft, Writing—Review & Editing, Visualization, Supervision, Project administration, Funding acquisition. H.Q.: Methodology, Software, Validation, Investigation, Resources, Writing—Review & Editing. S.C.: Methodology, Software, Formal analysis, Investigation, Data Curation, Writing—Review & Editing. X.F.: Resources, Investigation.All authors have read and agreed to the published version of the manuscript.Y.L. is the corresponding author. Email: liuyanchun@sdcet.edu.cn.

Declarations

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Additional information

Correspondence and requests for materials should be addressed to Y.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.