

A Multi-Objective Optimization Algorithm for Batching Tasks in MEC

Xuemei Yang*, Junjun Zhang

School of Public Safety and Emergency Management, Anhui University of Science and Technology, Hefei, China

Abstract—For multi-source data fusion analysis applications in Multi-access Edge Computing (MEC) systems, an edge node processes the tasks offloaded by multiple devices within the same time window in the form of task batches. Batching tasks amortizes the computational overhead on the edge, but it also prolongs task waiting time and therefore reduces the system throughput within a given time budget. This paper investigates the offloading problem for batching tasks and jointly optimizes transmission power of each device and edge computing resource allocation. The trade-off between system throughput maximization and total energy consumption minimization is modeled as a multi-objective optimization problem. Then, a two-phase algorithm is proposed to obtain a Pareto optimal solution. In the first phase, the transmission power and edge computing resource are set to their upper bounds so as to obtain the maximum system throughput. In the second phase, by setting the task batch number and taking variable substitution, the energy minimization problem is transformed into an easily solved convex problem. Simulation results show that the proposed algorithm attains 79.69%~199.70% higher system throughput than *Batch-MinEnergy*, and 9.10%~41.15% lower energy consumption than *Batch-MaxTps* for batching tasks.

Keywords—MEC; batching tasks; throughput maximization; energy consumption minimization

I. INTRODUCTION

Nowadays, MEC has been widely applied to provide computation and storage services closer to terminal devices, and thus Internet of Things (IoT) devices with limited computing capability can offload their tasks to the edge node for lower latency or energy consumption. Meanwhile, with the development of IoT networks, an increasing number of devices are deployed to collect multi-source and heterogeneous data. In many application cases, such as augmented reality, intelligent surveillance, intelligent vehicles and so on, devices continuously generate and offload tasks to the edge node, and the edge node aggregates tasks from different devices into batches within time windows and processes each task batch for multi-source data fusion analysis. Therefore, it is necessary to investigate the task batch offloading problem in a MEC system.

For task offloading, existing studies have fully investigated the task offloading problem aiming at latency minimization [1] [2] [3], energy consumption minimization [4] [5], or multiple objectives [6] [7] [8], but they rarely considered the scenario of batching tasks on the edge. For processing multiple Deep Neural Networks (DNN) inference tasks in parallel, a number of studies have proposed solutions to reduce the inference

delay and improve the inference accuracy [9] [10]. However, these existing studies are not suitable for the scenario of multi-source data fusion analysis due to the differentiation of task aggregation and batching.

This paper illustrates the process of how to aggregate tasks into a task batch and execute each batch on the edge node, and optimizes the joint strategy of setting transmission power of each device and computing resource of the edge node to maximize the system throughput and minimize the total energy consumption. This faces two challenges. First, batching tasks brings computational efficiency but also introduces a new trade-off. The edge node cannot start batch execution until all tasks in a batch have arrived, so the slowest device in a batch delays the whole batch, which hinders throughput improvement but saves energy. Second, the formulated multi-objective problem **P** is an intractable Mixed Integer Nonlinear Programming (MINLP) problem.

Based on the analysis of problem **P**, a two-phase algorithm is proposed to obtain a joint strategy of setting transmission power of each device and computing resource of the edge node, and this joint strategy is a Pareto optimal solution which can find a point on the Pareto frontier of **P**. Specifically, the proposed algorithm follows a lexicographic strategy and it maximizes the system throughput first and then minimizes the total energy consumption. In the first phase of the proposed algorithm, a system throughput maximization problem without considering energy consumption is formulated, and this simple integer programming problem is easily solved to obtain the maximum throughput. In the second phase of the proposed algorithm, a total energy consumption minimization problem is formulated with a given system throughput, but it is still difficult to solve because both the objective and constraints are highly non-convex. By introducing a substitution of transmission delay variable and execution delay variable for transmission power and edge computing resource allocation, the previous minimization problem is transformed into a convex problem that can be easily solved by many methods such as an interior-point method. At the end, the proposed algorithm obtains a Pareto optimal solution by combining the solutions of two phases.

The main contributions are summarized as follows:

- In the scenario of multi-source data fusion analysis, a multi-objective optimization problem for batching offloaded tasks is formulated, and the transmission power of each device and edge computing resource allocation are jointly optimized.
- A two-phase algorithm is proposed to maximize system throughput and minimize total energy consumption.

*Corresponding author (yangxuemei@aust.edu.cn)

†This work is supported by Scientific Research Foundation for High-level Talents of Anhui University of Science and Technology under Grant 2023YJRC94.

tion based on problem decomposition and convex optimization theory, and it is proved that the proposed algorithm can obtain a Pareto optimal solution.

- Simulation results show that, the proposal obtains the highest system throughput and achieves significant energy consumption reduction for batching tasks. It is also observed that the smaller the batch efficiency coefficient, the greater the benefits of batching.

The rest of this paper is organized as follows: Section II summarizes the related work, Section III introduces the system model and the multi-objective problem, Section IV presents the proposed algorithm in detail, and Section V illustrates the performance evaluation. Finally, the conclusion of this paper is summarized in Section VI.

II. RELATED WORK

Task offloading has been a hot research topic in recent years. Numerous studies have proposed solutions to optimize multiple objectives including task completion time reduction, energy consumption minimization, cost minimization and so on. Considering that edge clients can offload their tasks to other nearby edge clients and edge servers, Bozorgchenani et al. [11] formulated and solved a constrained multi-objective optimization problem to minimize both the energy consumption and task processing delay of the mobile devices. To cope with the challenge of increasing the number of tasks and resources providers in MEC, Li et al. [12] proposed a new multi-objective strategy based on the biogeography-based optimization algorithm to satisfy users' multiple requirements including the execution time, energy consumption and cost. In a multi-user multi-server, cloud-edge-end collaborative offloading scenario, Gong et al. [13] proposed an efficient multi-objective evolutionary optimization algorithm based on MOEA/D to maximize rewards and minimize latency. In the scenario of multiple edge base stations collaboration, Li et al. [7] proposed a two-stage hybrid multi-objective optimization evolutionary algorithm to minimize the delay and energy consumption. In an unmanned aerial vehicles (UAVs) aided MEC system, Sun et al. [14] formulated a multi-objective optimization problem aiming at minimizing the total task completion delay, reducing the total UAV energy consumption, and maximizing the total number of offloaded tasks, and solved it based on convex optimization theory. Considering the new requirements for mobile edge networks to handle multimodal data, Shi et al. [15] formulated a sensor activation time, computation offloading, and resource allocation joint optimization problem to improve throughput and reduce mobile terminal energy consumption. Considering that context-aware service recommendations were most significant for maximizing system efficiency and resource utilization in fog-enabled IoT networks, Mengya et al. [16] proposed an enhanced grey wolf optimization algorithm to optimize multiple conflicting objectives, including latency, energy consumption, trust, and service utility. For scalable IoT edge-cloud systems, Ayman [17] designed an explainable online score model and proposed a lightweight load-deadline-aware task offloading method to achieve LEAOT (Load-, Deadline-, Latency-, and Energy-Aware Task Offloading).

Additionally, there have been a lot of studies on task throughput improvement in MEC. Considering that the system-wide throughput under latency bounds was important in a

dynamic MEC system, Deng et al. [18] proposed a joint user association and resource allocation algorithm to maximize the long-term throughput. In an MEC system with multiple energy harvesting devices, Mei et al. [19] considered the challenges of dynamic task queues and battery levels, and employed the Lyapunov optimization theory to solve the formulated system throughput maximization problem. To improve the long-term throughput in a dynamic MEC system, Dai et al. [20] combined Lyapunov theory and federated deep reinforcement learning, and proposed a LyFOTO for adaptively boosting system throughput. To obtain high throughput and low average latency, Ning et al. [21] proposed MARHO, a Multi-Agent Reinforcement learning-based Hybrid task Offloading framework designed for a maritime MEC environment, with the consideration of capturing queueing dynamics, wireless transmission latency, computation delay, and battery constraints. Considering the unstable computing performance and transmission bandwidth, Fan et al. [22] developed a novel cloud-fog-edge multi-layer collaboration resource scheduling framework to reduce delay and energy consumption and improve throughput.

With the development of edge intelligence, batching multiple DNN inference tasks into a single batch for parallel processing has become popular in MEC systems. To reduce time-consuming memory access and boost the throughput, Liu et al. [23] studied a communication-and-computation resource allocation problem with batching and early exiting. Considering that each inference request may have different service level objectives, Zhang et al. [24] proposed a novel learning-based scheduling framework that incorporates adaptive batching and concurrent execution of DNN inference services on edge devices to optimize system utilization. Yang et al. [9] proposed an adaptive and cooperative batching framework for edge inference, wherein temporal adaptive batching and spatial task steering were jointly devised to balance the trade-off between batch latency and computational efficiency. To reduce the inference delay and improve the inference accuracy, Zou et al. [10] proposed an adaptive image batching and slicing framework for delay-sensitive small object detection in distributed edge environments. To accelerate computation-intensive edge cloud AI operations, Liu et al. [25] proposed an online optimization framework for joint resource provisioning and batch scheduling for more cost-efficient AIoT inference serving in an XPU edge cloud.

However, these task offloading studies for multi-objective optimization and throughput improvement mainly focused on processing each single task, and studies for batching tasks mainly focused on optimizing DNN inference strategy. This paper constructs a model of batching tasks in the scenario of multi-source data fusion and proposes a two-phase algorithm to maximize the system throughput and minimize the total energy consumption.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. System Model

In the scenario of multi-source data fusion, a MEC system consists of one edge node and N devices. The device set is denoted by $\mathcal{N} = \{1, 2, \dots, N\}$. For any device $n \in \mathcal{N}$, α^n denotes the task data size, and β^n denotes the required computing resources for executing the whole task. Each device

continuously generates tasks, and the edge node aggregates the tasks reported by each device within the same time window into a task batch. In practice, batching tasks usually achieves lower execution delay compared with individual task execution [26] [27]. Considering the resource efficiency gain brought by batching tasks, the computing resource required for a task batch is modeled as

$$\Theta = \xi \sum_{n \in \mathcal{N}} \beta^n, \quad (1)$$

where, $\xi \in (0, 1]$ is the batch efficiency coefficient. $\xi = 1$ means that batching does not bring any computational savings. The smaller ξ is, the greater the computational savings.

The communication between devices and edge node is implemented via wireless communication technology. The edge node starts executing a task batch only if all devices have finished the transmission of all tasks in this batch. The transmission power decision of device n is denoted by $p_n \in [P_n^{min}, P_n^{max}]$, where, P_n^{min} and P_n^{max} are the minimum and maximum transmission power, respectively. The computing capability allocation decision of the edge node is denoted by $f^e \in [F^{min}, F^{max}]$, where, F^{min} and F^{max} are the minimum and maximum computing capability, respectively. Each device employs a First-Come First-Served (FCFS) mechanism to sequentially transmit its tasks, and the edge node also adopts a FCFS mechanism to sequentially execute the successive task batches. Let non-negative integer variable s denote the number of task batches executed during 1 second, and then the system throughput, i.e., the total tasks completed during 1 second, is Ns . The process of transmitting each task and executing task batches is shown in Fig. 1.

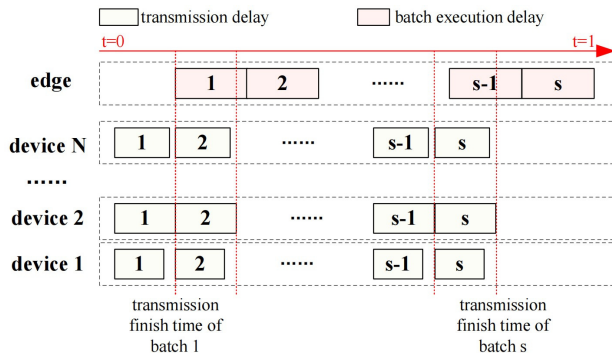


Fig. 1. The process of batching tasks.

The uplink transmission rate r_n of device n is given by Shannon's theorem as follows:

$$r_n = B \log_2(1 + \frac{p_n g^n}{BN_0}) \quad (2)$$

where, B denotes the bandwidth assigned to each device n , g^n denotes the channel gain, N_0 denotes the noise power spectral density. Correspondingly, given a transmission rate r_n , the required transmission power of device n is computed by

$$p_n = (2^{\frac{r_n}{B}} - 1) \frac{BN_0}{g^n} \quad (3)$$

The task transmission delay of device n is

$$d_n^{tr} = \frac{\alpha^n}{r_n} \quad (4)$$

For a task batch, the batch transmission delay is determined by the slowest device and is calculated by

$$d^{tr} = \max\{d_n^{tr} \mid n \in \mathcal{N}\} \quad (5)$$

The batch execution delay is

$$d^{ed} = \frac{\Theta}{f^e} \quad (6)$$

In the scenario of batching tasks, each device generates tasks periodically according to the transmission delay, and the edge node starts executing a task batch only after all tasks in this batch have arrived at the edge and previous batches have been completed. In other words, the completion time of the s th task batch $T_{(s)}$ depends on the transmission finish time of the s th task batch and the execution finish time of the $(s-1)$ th task batch. Through inductive reasoning, $T_{(s)}$ is calculated by

$$T_{(s)} = d^{tr} + (s-1) \max\{d^{tr}, d^{ed}\} + d^{ed} \quad (7)$$

The energy consumption consists of transmission energy consumption on the device side and execution energy consumption on the edge side. The transmission energy consumption for transmitting a task batch is

$$e^{tr} = \sum_{n \in \mathcal{N}} p_n d_n^{tr} = \sum_{n \in \mathcal{N}} p_n \frac{\alpha^n}{r_n} \quad (8)$$

e^{tr} increases with p_n because the first order derivative regarding p_n is always positive when $p_n > 0$.

For the task batch execution, given the computing capability f^e , the power consumption of the edge node is modeled as $\kappa(f^e)^3$ (joule per second), where, κ represents the computation energy efficiency coefficient related to the processor's chip of the edge node [27] [28]. The execution energy consumption of a task batch is

$$e^{ed} = \kappa(f^e)^3 d^{ed} = \kappa(f^e)^2 \Theta \quad (9)$$

e^{ed} increases with f^e because the first order derivative regarding f^e is always positive when $f^e > 0$.

Then, the total energy consumption of s task batches is calculated by

$$E_{(s)} = s(e^{tr} + e^{ed}) \quad (10)$$

B. Problem Formulation

Considering the demand for batching tasks, this paper investigates the task batch offloading problem, and formulates a multi-objective optimization problem to maximize the system throughput $S = Ns$ and minimize the total energy consumption $E_{(s)}$ as follows.

$$\begin{aligned} \mathbf{P}: & \max_{\mathbf{p}, f^e} Ns \\ & \min_{\mathbf{p}, f^e} E_{(s)} \end{aligned} \quad (11)$$

s.t.

$$T_{(s)} \leq 1 \quad (11a)$$

$$F^{min} \leq f^e \leq F^{max} \quad (11b)$$

$$P_n^{min} \leq p_n \leq P_n^{max}, \forall n \in \mathcal{N} \quad (11c)$$

where, $\mathbf{p} = (p_1, p_2, \dots, p_N)$, and constraint (11a) means that s task batches must be completed during 1 second. Because the integrality of s makes $\mathbf{P}$ a mixed-integer program and the max term couples all devices, $\mathbf{P}$ is a non-convex MINLP and it is hard to solve. Here, a group of $(\mathbf{p}, f^e)$ is called a solution or joint strategy for $\mathbf{P}$.

C. Problem Analysis

Usually, the goal of solving a multi-objective optimization problem is to obtain a Pareto optimal solution, i.e., a solution for which no objective can be improved without degrading another objective. For the objective Ns , s increases with the decrease of d_n^{tr} and d^{ed} , d^{tr} decreases with the increase of each p_n , and d^{ed} decreases with the increase of f^e , so Ns increases with the increase of each p_n and f^e . As for the objective $E_{(s)}$, for a given s , $E_{(s)}$ decreases with the decrease of each p_n and f^e . Therefore, there is no feasible solution to simultaneously increase Ns and decrease $E_{(s)}$. But, for a fixed batch number s , as long as $T_{(s)} \leq 1$, each p_n and f^e can be reduced to decrease $E_{(s)}$ without any loss of the system throughput Ns .

Therefore, this paper adopts a lexicographic strategy and considers obtaining a Pareto optimal solution by first optimizing the system throughput and then optimizing the total energy consumption. Specifically, the following two-phase strategy is considered. In the first phase *Phase 1*, by setting $p_n = P_n^{max}$ and $f^e = F^{max}$, problem $\mathbf{P}$ degrades into a batch number s maximization problem, and then the maximum batch number s^* can be easily obtained. In the second phase *Phase 2*, by fixing $s = s^*$, problem $\mathbf{P}$ degrades into a total energy consumption $E_{(s^*)}$ minimization problem, and then the minimum $E_{(s^*)}$ can be obtained by solving this minimization problem.

Proposition III.1. *If s^* is the maximum feasible value of s under (11a) and $E_{(s^*)}^*$ is the minimum value of $E_{(s^*)}$ with $s = s^*$, $(Ns^*, E_{(s^*)}^*)$ is a point on the Pareto frontier of problem (11) and the solution that yields $(Ns^*, E_{(s^*)}^*)$ is a Pareto optimal solution.*

Proof: Pareto optimality requires that there exists no feasible solution that is no worse in both objectives and strictly better in at least one. Because s^* is the maximum feasible value, every feasible solution satisfies $s \leq s^*$, i.e., the system throughput is no larger than Ns^* ; hence there is no feasible solution can yield a larger system throughput. For any feasible solution with the same throughput Ns^* , its batch number must be s^* , and since $E_{(s^*)}^*$ is the minimum energy under s^* , there is no feasible solution with a lower energy under the same throughput. ■

IV. THE TWO-PHASE ALGORITHM

As stated before, the proposed two-phase algorithm consists of two phases, *Phase 1* and *Phase 2*. In *Phase 1*, without considering the energy consumption, the batch number s maximization problem is formulated and solved to obtain the maximum system throughput. In *Phase 2*, with a given batch number s^* , the total energy consumption $E_{(s^*)}$ minimization problem is formulated and solved to obtain a Pareto optimal solution of problem $\mathbf{P}$.

A. Phase 1: Maximizing Batch Number

By setting each $p_n = P_n^{max}$, the lower bound of batch transmission delay is $d_n^{tr,L} = \max\{d_n^{tr,L} | n \in \mathcal{N}\}$, and $d_n^{tr,L} = \frac{\alpha_n^n}{r_n(P_n^{max})}$. The function $r_n(P_n^{max})$ calculates the transmission rate with power P_n^{max} . By setting $f^e = F^{max}$, the lower bound of batch execution delay $d^{ed,L} = \frac{\Theta}{F^{max}}$. Then, without considering the energy consumption, the batch number s maximization problem is formulated as follows

$$\mathbf{P1:} \max s \quad (12)$$

s.t.

$$d^{tr,L} + d^{ed,L} + (s-1) \max\{d^{tr,L}, d^{ed,L}\} \leq 1 \quad (12a)$$

Obviously, the maximum s^* is obtained by

$$s^* = \left\lceil 1 + \frac{1 - d^{tr,L} - d^{ed,L}}{\max\{d^{tr,L}, d^{ed,L}\}} \right\rceil \quad (13)$$

It is assumed that there is always $s^* \geq 1$ in the considered system.

B. Phase 2: Minimizing Total Energy Consumption

In *Phase 2*, although the batch number has been fixed as $s = s^*$, the total energy consumption minimization problem is still intractable due to the nonlinear terms in $E_{(s^*)}$ and the max operation in $T_{(s^*)}$. Because of the one-to-one mapping between d_n^{tr} and p_n of each device n and one-to-one mapping between d^{ed} and f^e , the decision variables p_n and f^e are substituted with d_n^{tr} and d^{ed} , respectively, and further the energy minimization problem is transformed into a convex problem via equivalent reformulations.

By taking $\mathbf{D}^{tr} = (d_1^{tr}, d_2^{tr}, \dots, d_N^{tr})$ and d^{ed} as decision variables, there are

$$e^{tr} = \sum_{n \in \mathcal{N}} H^n d_n^{tr} (2^{I^n/d_n^{tr}} - 1) \quad (14)$$

where, $H^n = \frac{BN_0}{g^n}$ and $I^n = \frac{\alpha_n^n}{B}$, and

$$e^{ed} = \frac{\kappa \Theta^3}{(d^{ed})^2} \quad (15)$$

For constraint (11a), there is the following proposition.

Proposition IV.1. *For $s \geq 1$, the constraint $T_{(s)} \leq 1$ is equivalent to the two linear inequalities*

$$sd^{tr} + d^{ed} \leq 1 \quad \text{and} \quad d^{tr} + sd^{ed} \leq 1. \quad (16)$$

Proof: $T_{(s)} = s \max\{d^{tr}, d^{ed}\} + \min\{d^{tr}, d^{ed}\} = \max\{sd^{tr} + d^{ed}, d^{tr} + sd^{ed}\}$. If $d^{tr} \geq d^{ed}$, then $T_{(s)} = sd^{tr} + d^{ed} = \max\{sd^{tr} + d^{ed}, d^{tr} + sd^{ed}\}$ since $sd^{tr} + d^{ed} - (d^{tr} + sd^{ed}) = (s-1)(d^{tr} - d^{ed}) \geq 0$; the case $d^{ed} \geq d^{tr}$ is symmetric. Therefore $T_{(s)} \leq 1$ holds if and only if both $sd^{tr} + d^{ed} \leq 1$ and $d^{tr} + sd^{ed} \leq 1$ hold. ■

By *Proposition IV.1*, constraint (11a) is exactly replaced by two linear inequalities without introducing any relaxation error.

Algorithm 1 The two-phase algorithm

- 1: **Phase 1:**
 - 2: compute $d_n^{tr,L}, d_n^{tr,U}, d_n^{ed,L}$ by setting $p_n = P_n^{max}, f^e = F^{max}$;
 - 3: obtain s^* with equation (13);
 - 4: **Phase 2:**
 - 5: compute $H^n, I^n, d_n^{tr,U}, d_n^{ed,U}$;
 - 6: solve problem **P2** and obtain the optimal d_n^{tr*}, d_n^{ed*} ;
 - 7: compute the optimal p_n^*, f^{e*} with d_n^{tr*}, d_n^{ed*} ;
 - 8: **return** s^*, p_n^*, f^{e*} .
-

Consequently, the $E_{(s^*)}$ minimization problem is formulated as follows

$$\mathbf{P2}: \min_{\mathbf{D}^{tr}, \mathbf{d}^{ed}} s^* \left(\sum_{n \in \mathcal{N}} H^n d_n^{tr} (2^{I^n/d_n^{tr}} - 1) + \frac{\kappa \Theta^3}{(d_n^{ed})^2} \right) \quad (17)$$

s.t.

$$s^* d_n^{tr} + d_n^{ed} \leq 1, \quad \forall n \in \mathcal{N} \quad (17a)$$

$$d_n^{tr} + s^* d_n^{ed} \leq 1, \quad \forall n \in \mathcal{N} \quad (17b)$$

$$d_n^{tr,L} \leq d_n^{tr} \leq d_n^{tr,U}, \quad \forall n \in \mathcal{N} \quad (17c)$$

$$d_n^{ed,L} \leq d_n^{ed} \leq d_n^{ed,U} \quad (17d)$$

where, $d_n^{tr,L} = \frac{\alpha^n}{r_n(P_n^{max})}$, $d_n^{tr,U} = \frac{\alpha^n}{r_n(P_n^{min})}$, $d_n^{ed,L} = \frac{\Theta}{F_n^{max}}$, and $d_n^{ed,U} = \frac{\Theta}{F_n^{min}}$. Note that function $e_n^{tr}(d) = H^n d(2^{I^n/d} - 1)$ is the perspective function $x\phi(a/x)$ with $\phi(u) = 2^u - 1$ convex and $\phi(0) = 0$, and by the perspective-function rule it is convex on $d > 0$. Then, because $H^n d_n^{tr}(2^{I^n/d_n^{tr}} - 1)$ is convex on $d_n^{tr} > 0$, $\frac{\kappa \Theta^3}{(d_n^{ed})^2}$ is convex on $d_n^{ed} > 0$, and constraints (17a)~(17d) are all affine functions, **P2** is a convex problem and it can be easily solved by a number of methods, such as interior-point method.

C. Pseudocode and Complexity

The pseudocode of the proposed algorithm is shown in **Algorithm 1**. Furthermore, according to *Proposition III.1*, the solution p_n^*, f^{e*} obtained by **Algorithm 1** is a Pareto optimal solution of problem **P**.

In *Phase 1*, the complexity of calculating s^* is $O(N)$. In *Phase 2*, the complexity of solving problem **P2** via interior-point method is about $O(N^{3.5} \ln(1/\epsilon))$ where, ϵ is the optimality tolerance [29]. Overall, the complexity of **Algorithm 1** mainly depends on *Phase 2*.

V. PERFORMANCE EVALUATION

A. Simulation Settings

The following simulation settings are used. For the edge node, the computing energy coefficient is $\kappa = 10^{-26}$ [28], and the minimum and maximum computing resources are $F^{min} = 5$ GHz and $F^{max} = 10$ GHz, respectively. The channel bandwidth is $B = 1$ MHz, the noise power spectral density is $N_0 = -174$ dBm/Hz. For each device, the transmission power bounds are $P_n^{min} = 0.1$ W and $P_n^{max} = 1.0$ W, and the channel gain is generated according to the path-loss model $g^n = A \left(\frac{c/f}{4\pi d^n} \right)^l$ with the antenna gain $A = 4.11$, the velocity of light $c = 3 \times 10^8$ m/s, the carrier frequency

$f = 915$ MHz, $\pi = 3.14$, the distance between a device and edge node $d^n \in [10, 100]$ m, and the path-loss exponent $l = 2.8$ [30]. The data size of each task is $\alpha^n \in [0.5, 1]$ Mbit, and the computing resource requirement β^n is uniformly generated in $[0.05, 0.2]$ Gcycles. The default batch efficiency coefficient is $\xi = 0.8$. The number of devices is $N \in \{5, 10, 20, 30, 40\}$. To implement the proposed algorithm, CVXPY, a python-embedded modeling language for convex optimization problems, is employed. Moreover, considering the randomness of previous simulation settings, the average result of 5000 simulations is taken as the final result.

B. Methods for Comparison

The proposed method, *Ours* for short, is compared with the following methods.

- *Batch-MaxTps*: A system throughput maximization problem that is equivalent to **P1** is formulated and solved without considering energy consumption.
- *Batch-MinEnergy*: The goal is to minimize the total energy consumption by setting $p_n = P_n^{min}$ and $f^e = F^{min}$ for batching tasks.
- *Batch-Median*: The goal is to minimize the total energy consumption with a given system throughput median. It serves as a simplified version of the weighted-sum optimization method.
- *FCFS-MaxTps*: Unlike the scenario of batching tasks, a FCFS scenario is that the edge node sequentially executes each arrived task based on a FCFS mechanism. The goal is to maximize the system throughput by setting $p_n = P_n^{max}$ and $f^e = F^{max}$ in the FCFS scenario.
- *FCFS-MinEnergy*: The goal is to minimize the total energy consumption by setting $p_n = P_n^{min}$ and $f^e = F^{min}$ in the FCFS scenario.

Although *FCFS-MaxTps* and *FCFS-MinEnergy* are not feasible for batching tasks, they are typical methods for task offloading, and thus they are still taken as two baselines here. In the scenario of batching tasks, according to the core idea of each method, the expected behavior of each method is as follows. *Batch-MaxTps* would be the best in terms of system throughput, *Batch-MinEnergy* would be the best in terms of total energy consumption, the performance of *Batch-Median* would be between that of *Batch-MaxTps* and *Batch-MinEnergy*, and *Ours* would achieve the same system throughput as *Batch-MaxTps* but consume less energy than *Batch-MaxTps*.

C. Results and Analysis

Fig. 2 shows the system throughput and total energy consumption of all methods, and Fig. 3 shows the number of batches and energy consumption per batch of all methods. The error bars in Fig. 2 and Fig. 3 represent the standard deviation of results from 5000 simulations. The batching-based methods exhibit the expected performance trends.

In Fig. 2(a), the system throughput of *Ours* and *Batch-MaxTps* are 79.69%~199.70% higher than that of *Batch-MinEnergy*. In Fig. 2(b), the total energy consumption of *Ours*

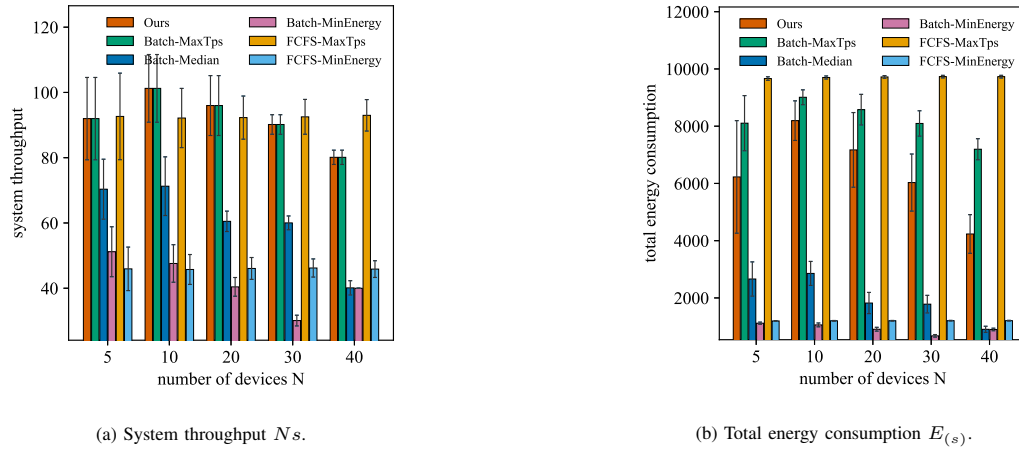


Fig. 2. The system throughput and total energy consumption of all methods.

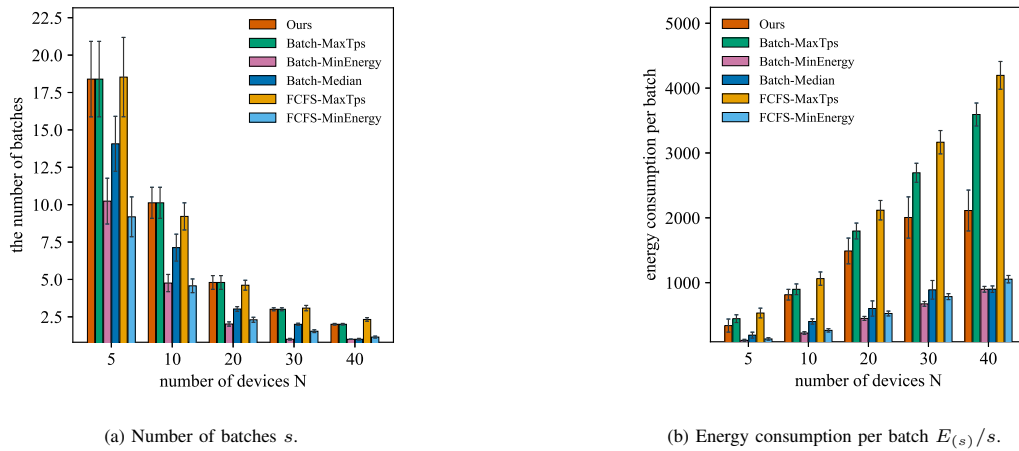


Fig. 3. The number of batches and energy consumption per batch of all methods.

is 9.10%~41.15% lower than *Batch-MaxTps* because of the multi-objective optimization target, and 3.7~7.9 times higher than *Batch-MinEnergy* because of the much higher throughput. The system throughput of *Batch-Median* is lower than that of *Batch-MaxTps* and higher than that of *Batch-MinEnergy*. The total energy consumption of *Batch-Median* is less than that of *Batch-MaxTps* and more than that of *Batch-MinEnergy*. For each batch, because the number of tasks in a batch increases with N , the number of batches s of each method always decreases with the increase of N in Fig. 3(a), and the energy consumption per batch of each method always increases with the increase of N in Fig. 3(b).

Moreover, the system throughput of *Ours* first increases when N increases from 5 to 10, and then decreases when N continues to increase as shown in Fig. 2(a). The reason is that d^{ed} increases with N , which determines the size of a batch, whereas d^{tr} changes relatively little with N , and a much higher d^{ed} can dramatically decrease s and Ns . Because the total energy consumption is directly related to N and s , changing trends of *Ours* in Fig. 2(a) and Fig. 2(b) are basically the same.

As for the two methods for FCFS scenario, in Fig. 2(a), *FCFS-MaxTps* achieves higher throughput than *Ours* and *Batch-MaxTps* when $N \in \{5, 30, 40\}$, and *FCFS-MinEnergy* achieves higher throughput than *Batch-MinEnergy* when $N \in \{20, 30, 40\}$, which is because the edge node does not need to wait for the slowest task in a batch. For the same reason, there is no significant fluctuation in the performance of *FCFS-MaxTps* and *FCFS-MinEnergy* in Fig. 2(a). In Fig. 2(a) and Fig. 2(b), changing trends of *FCFS-MaxTps* and *FCFS-MinEnergy* are also basically the same.

To evaluate the effect of batch efficiency coefficient ξ , simulations are conducted under the settings $N = 20$ and $\xi \in \{0.4, 0.6, 0.8, 1.0\}$. Fig. 4 shows the system throughput and total energy consumption under different ξ , and Fig. 5 shows the number of batches and energy consumption per batch under different ξ . With the increase of ξ , the computing resource required by a task batch Θ grows linearly, and the computational savings offered by batching gradually disappear. Thus, the system throughput in Fig. 4(a) and number of batches in Fig. 5(a) decrease with the increase of ξ . In Fig. 4(b), because the system throughput decreases with ξ , the total

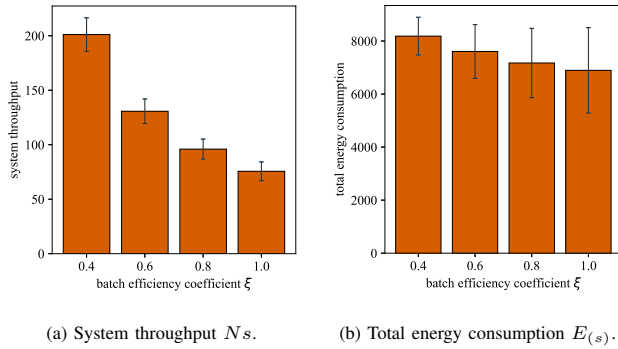


Fig. 4. The system throughput and total energy consumption under different ξ .

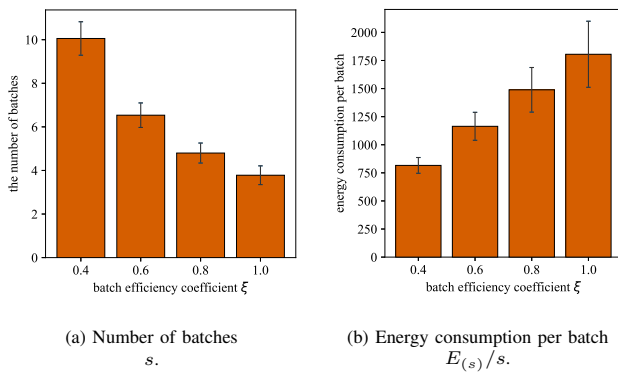


Fig. 5. The number of batches and energy consumption per batch under different ξ .

energy consumption also decreases with ξ . As for the energy consumption per batch shown in Fig. 5(b), because larger ξ brings higher batch execution delay, the energy consumption per batch increases.

VI. CONCLUSION

This paper studies an offloading problem for batching tasks in the scenario of multi-source data fusion analysis. To maximize the system throughput and minimize the total energy consumption, this paper formulates a multi-objective optimization problem and jointly optimizes the transmission power of each device and edge computing resource allocation. Through problem analysis and decomposition, a two-phase algorithm is proposed to obtain a Pareto optimal solution. In *Phase 1*, the maximum system throughput is obtained by taking the upper bounds of the transmission power and the edge computing resource. In *Phase 2*, by setting a task batch number and taking variable substitution, the energy consumption minimization problem is formulated and transformed into a convex problem that can be easily solved. Simulation results show that, for batching tasks, the proposed algorithm attains 79.69%~199.70% higher system throughput than *Batch-MinEnergy* and 9.10%~41.15% lower energy consumption than *Batch-MaxTps*, and the smaller the batch-efficiency coefficient, the greater the throughput and energy benefits of batching.

REFERENCES

- [1] W. Feng, N. Zhang, S. Li, S. Lin, R. Ning, S. Yang, and Y. Gao, "Latency minimization of reverse offloading in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5343–5357, 2022.
- [2] B. Zhang, Z. Du, and L. Yu, "Multi-service dependent task offloading based on multi-agent drl for vehicular edge computing," *IEICE Transactions on Communications*, vol. E109-B, no. 2, pp. 305–316, 2026.
- [3] N. H. Nguyen, N. V. Thieu, Q.-T. Luu, A. T. Nguyen, S. H. Wanasekara, N. C. Luong, F. Kavehmadavani, and V.-D. Nguyen, "Oranits: Mission assignment and task offloading in open ran-based its using metaheuristic and deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 27, no. 10, pp. 12 515–12 530, 2026.
- [4] B. Hu, Y. Shi, and Z. Cao, "Adaptive energy-minimized scheduling of real-time applications in vehicular edge computing," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 5, pp. 6895–6906, 2023.
- [5] D. Yang, B. Li, and D. Niyato, "Energy-aware task offloading for rotatable star-ris-enhanced mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 12, no. 12, pp. 20 239–20 250, 2025.
- [6] M. Kim, J. Jang, Y. Choi, and H. J. Yang, "Distributed task offloading and resource allocation for latency minimization in mobile edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 15 149–15 166, 2024.
- [7] L. Li, Q. Qiu, Z. Xiao, Q. Lin, J. Gu, and Z. Ming, "A two-stage hybrid multi-objective optimization evolutionary algorithm for computing offloading in sustainable edge computing," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 735–746, 2024.
- [8] I. Bisio, C. Garibotto, F. Lavagetto, A. Sciarone, and M. Zerbino, "Distributed multiobjective optimization for edge computing in resource-constrained social iot networks," *IEEE Internet of Things Journal*, vol. 13, no. 8, pp. 15 549–15 558, 2026.
- [9] Z. Yang, Z. Zheng, L. Deng, S. Zhang, Z. Wang, and H. Luo, "Acbatch: Adaptive and cooperative batching for edge inference," in *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications*, 2025, pp. 1–10.
- [10] H. Zou, J. Guo, S. Sun, Y. Liang, C. Xu, Y. Zhao, S. Zhao, and T. Wang, "Adaptive image batching and slicing in edge networks for delay-critical small object detection," *IEEE Transactions on Parallel and Distributed Systems*, vol. 37, no. 7, pp. 1657–1670, 2026.
- [11] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. Salinas Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Transactions on Mobile Computing*, vol. 20, no. 10, pp. 2992–3005, 2021.
- [12] H. Li, P. Zheng, T. Wang, J. Wang, and T. Liu, "A multi-objective task offloading based on bbo algorithm under deadline constrain in mobile edge computing," *Cluster Computing*, vol. 26, no. 6, pp. 4051–4067, 2023.
- [13] Y. Gong, K. Bian, F. Hao, Y. Sun, and Y. Wu, "Dependent tasks offloading in mobile edge computing: A multi-objective evolutionary optimization strategy," *Future generation computer systems*, vol. 148, pp. 314–325, 2023.
- [14] G. Sun, Y. Wang, Z. Sun, Q. Wu, J. Kang, D. Niyato, and V. C. M. Leung, "Multi-objective optimization for multi-uav-assisted mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 14 803–14 820, 2024.
- [15] T. Shi, T. Zhang, X. Xu, Y. Liu, and R. Huang, "Multimodal mobile edge computing: Multi-objective optimization with synchronization constraint," *IEEE Transactions on Mobile Computing*, vol. 25, no. 6, pp. 8034–8049, 2026.
- [16] M. NIE and F. JIAO, "A novel multi-objective service scheduling framework for fog-driven agricultural internet of things," *International Journal of Advanced Computer Science and Applications*, vol. 17, no. 8, 2026. [Online]. Available: <https://doi.org/10.14569/IJACSA.2026.0170863>
- [17] A. Noor, "Leaot: Load-, deadline-, latency-, and energy-aware task offloading for scalable iot edge-cloud systems," *International Journal of Advanced Computer Science and Applications*, vol. 17, no. 8, 2026. [Online]. Available: <https://doi.org/10.14569/IJACSA.2026.0170877>

- [18] Y. Deng, Z. Chen, X. Chen, and Y. Fang, "Throughput maximization for multi-edge multiuser edge computing systems," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 68–79, 2022.
- [19] J. Mei, L. Dai, Z. Tong, X. Deng, and K. Li, "Throughput-aware dynamic task offloading under resource constant for mec with energy harvesting devices," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3460–3473, 2023.
- [20] L. Dai, F. Zeng, H. Kong, J. Cai, H. Jiang, and K. Li, "Throughput-aware cooperative task offloading in dynamic mobile edge computing systems," *IEEE Transactions on Mobile Computing*, vol. 24, no. 12, pp. 13 276–13 292, 2025.
- [21] J. Ning, A. Li, G. C. F. Lee, S. Sun, and T. Yang, "Marho: Hybrid task offloading in maritime mec via multi-agent reinforcement learning," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 10 322–10 337, 2025.
- [22] W. Fan, F. Xiao, Y. Pan, X. Chen, L. Han, and S. Yu, "Latency-aware joint task offloading and energy control for cooperative mobile edge computing," *IEEE Transactions on Services Computing*, vol. 18, no. 3, pp. 1515–1528, 2025.
- [23] Z. Liu, Q. Lan, and K. Huang, "Resource allocation for multiuser edge inference with batching and early exiting," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 1186–1200, 2023.
- [24] Z. Zhang, Y. Zhao, H. Li, and J. Liu, "Bcedge: Slo-aware dnn inference services with adaptive batch-concurrent scheduling on edge devices," *IEEE Transactions on Network and Service Management*, vol. 21, no. 4, pp. 4131–4145, 2024.
- [25] R. Liu, Y. Wu, K. Zhao, Z. Zhou, X. Gao, X. Lin, X. Zhang, X. Chen, and G. Lu, "Online resource provisioning and batch scheduling for ai-ot inference serving in an xpu edge cloud," *IEEE Transactions on Emerging Topics in Computing*, vol. 13, no. 1, pp. 234–249, 2025.
- [26] V. Pentsos, S. Tragoudas, K. N. Gowda, and M. Schmit, "Energy-aware dnn task scheduling with dynamic batching and frequency adjustment," in *2025 26th International Symposium on Quality Electronic Design (ISQED)*, 2025, pp. 1–8.
- [27] Y. Xu, S. Zhou, and Z. Niu, "Joint optimization of offloading, batching and dvfs for multiuser co-inference," in *ICC 2025 - IEEE International Conference on Communications*, 2025, pp. 1518–1523.
- [28] M. Zhao, J.-J. Yu, W.-T. Li, D. Liu, S. Yao, W. Feng, C. She, and T. Q. S. Quek, "Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10 925–10 940, 2021.
- [29] J. Gondzio, "Interior point methods 25 years later," *European Journal of Operational Research*, vol. 218, no. 3, pp. 587–601, 2012. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221711008204>
- [30] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.