Online Scheduling of Battery-Aware Speculative Decoding for Energy-Efficient Cloud-Edge Collaborative LLM Inference

Hengdi Wang
School of Artificial Intelligence
Beijing University of Posts and Telecommunications
Beijing, China
wanghengdi@bupt.edu.cn

Lei Jiao*
Center for Cyber Security and Privacy
University of Oregon
Eugene, OR, USA
ljiao2@uoregon.edu

Konglin Zhu*
School of Artificial Intelligence
Beijing University of Posts and Telecommunications
Beijing, China
klzhu@bupt.edu.cn

Lin Zhang
School of Artificial Intelligence
Beijing University of Posts and Telecommunications
Beijing, China
zhanglin@bupt.edu.cn

Abstract

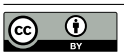
While distributed speculative decoding can offer efficient acceleration for Large Language Model (LLM) inference in cloud-edge environments, unleashing its full potential confronts significant challenges, including complex token draft-length management, uncertain prompt arrivals and system conditions, and joint edge battery orchestration under fluctuating grid-energy prices. To systematically address these challenges, we conduct an intensive mathematical and algorithmic study. We first formulate a long-term cost minimization problem that captures all the challenges. Then, we decompose this problem into two subproblems and solve them alternately as time goes in an online manner. Particularly, we design a learning-centric algorithmic framework which employs bandit learning to address the intractability and manage draft length selection in the first subproblem and applies online learning to address the online uncertainty and manage battery states and grid-energy use in the second subproblem, both without relying on future inputs. We further conduct rigorous theoretical analysis, proving sub-linear regrets against offline optimal objectives and asymptotically-diminishing violation of long-term constraints. Extensive evaluations with real-world LLM inference demonstrate that our approach substantially outperforms several state-of-the-art methods, with many additional advantages.

Keywords

Cloud-Edge Computing, Large Language Model, Speculative Decoding, Energy Efficiency, Bandit Learning, Online Learning.

ACM Reference Format:
Hengdi Wang, Lei Jiao, Konglin Zhu, and Lin Zhang. 2026. Online Scheduling of Battery-Aware Speculative Decoding for Energy-Efficient Cloud-Edge Collaborative LLM Inference. In *Proceedings of the 55th International Conference on Parallel Processing (ICPP '26), September 28-October 01, 2026, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3832810.3832825

*Corresponding authors.



ICPP '26, Singapore

ACM ISBN 979-8-4007-2657-6/2026/09
https://doi.org/10.1145/3832810.3832825

1 Introduction

To provision high-quality low-latency Large Language Model (LLM) inference services, the service provider leveraging cloud and edge infrastructures faces a critical trade-off: edge-only deployment is constrained by limited computational resources, while cloud-only processing introduces substantial wide-area network delay [1, 2]. Cloud-edge *speculative decoding* [3] resolves this by strategically distributing the inference pipeline, as visualized in Figure 1. That is, a lightweight *draft LLM* at the network edge auto-regressively generates an initial sequence of tokens locally, delivering rapid output to the end user; this draft is then sent to the remote cloud, where a powerful *target LLM* verifies all tokens in a single parallel forward pass, accepting correct ones and correcting the first mismatch. This architecture preserves the quality of cloud-scale models while significantly reducing the time-to-first-token and amortizing the latency over multiple tokens per communication round [4].

However, efficiently managing this distributed pipeline across the cloud-edge continuum requires the joint optimization of communication, computation, and energy usage, which is a non-trivial task due to the unique and fundamental challenges as follows.

First, the LLM inference service must dynamically manage the draft length, which directly trades computation against communication in the speculative-decoding process. The draft length impacts the number of speculative-decoding iterations between each edge and the cloud, while in every single iteration, sending too many draft tokens strains the network link and wastes computation at the edge but sending too few decreases parallelism utilization in the cloud [5, 6]. Further, the draft length at the edge needs to adapt to the target LLM's token-acceptance rate in the cloud, requiring dynamically balancing the exploration of new different length options with the exploitation of previously-chosen lengths [7, 8]. These are all exacerbated by unpredictable inference-request arrivals, fluctuating network conditions, and pre-determined communication traffic budget, demanding smart learning-based coordination that can operate effectively without any prior knowledge.

Second, the LLM inference service must also dynamically manage the energy of the joint cloud-edge system, which constitutes a major

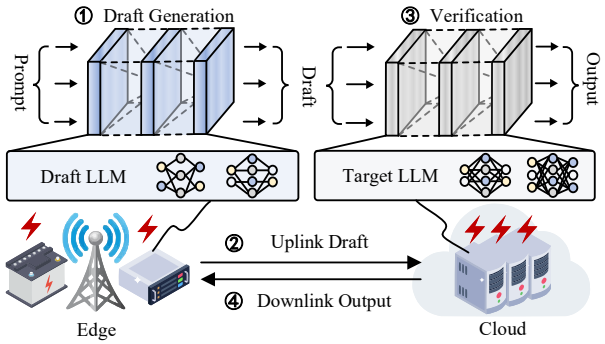


Figure 1: System scenario

source of the operational expenditure nowadays. In particular, this is not just only about how the complexity of speculative decoding influences the grid-energy consumption for the cloud, but also about how it could entail the charging and discharging operations for the batteries widely equipped at today's edges (e.g., 5G edge servers) [9, 10]. Edge battery-energy storage has the advantage of shifting energy footprint to low-price periods, but has limited capacity and degradation with use [11]. The core dilemma is whether to conserve energy from the grid now even at a high price, or to use the stored energy now which risks battery depletion and even higher grid-energy cost in the future, while jointly managing the speculative-decoding-based LLM inference. Decisions must be made online without future energy price and request loads.

Unfortunately, existing research has inadequately addressed the aforementioned challenges. Studies [1, 4, 12–14] that provide LLM inference services through cloud-edge collaboration often overlook the energy consumption of LLMs and the potential of speculative decoding to accelerate inference. Research [5, 6, 15–17] on LLM speculative decoding in resource-constrained networks either focuses solely on communication without considering the impact of energy consumption, or never accounts for limited communication resources. Those [9, 10, 18–20] investigating edge battery management also neglect the cloud-edge LLM inference scenario. Detailed discussions of related work are in Section 6.

This paper, for the first time, presents a rigorous modeling and mathematical study on the joint online scheduling of speculative decoding and battery energy for the cloud-edge collaborative LLM inference systems. We make four-fold contributions:

First, we model and formulate a long-term optimization problem that minimizes the total cost of cloud-edge LLM inference. Our models capture the end-to-end distributed speculative decoding process, including the computation delay of edge generation and cloud verification, the communication delay of token transference between edge and cloud, and the cloud-edge energy expense with edge battery degradation. Our problem turns out to be a nonlinear mixed-integer program, pursuing optimization over diverse inference requests while respecting the communication budget with sufficient energy provisioning. Our formulation is general and readily accommodates the dynamism and heterogeneity of inputs such as inference requests, energy prices, and battery degradation costs.

Second, we devise a novel learning-centric decomposition-based algorithmic framework that operates in polynomial time to solve this problem online. Our approach decomposes the original problem into two subproblems, containing draft length control and energy control, respectively. For the former subproblem, we design a *bandit learning* [21, 22] approach that, for each edge, constructs a feasible set of draft length options under the communication budget and updates the weights of the selected draft lengths through exponential weighting as time goes, using such weights to dynamically manage the trade-off between exploring new draft lengths and exploiting existing draft lengths. For the latter subproblem, with the given draft length, we design an *online learning* [23, 24] approach to manage battery operations and grid-energy use, which utilizes the convex-concave reformulation and the alternate primal-dual updates facilitated by a virtual queue, without relying on future inputs. Overall, our key algorithmic strategy is to group and isolate the different challenges, namely *integrality*, *posterior*, *state transitions*, and *online uncertainty*, in separate subproblems, and identify and leverage different algorithmic techniques to address them.

Third, we derive rigorous theoretical analysis for our algorithms. For the first subproblem, we prove that the *regret*, i.e., the gap between the expected inference delay incurred by our online draft length selections and that incurred by the single best draft length grows slower than the natural progression of time. For the second subproblem, we also prove that, for the joint energy expense and battery degradation, the *regret*, i.e., the difference between the objective value from our online energy decisions and that from the series of the per-slot optimal energy decisions, possesses sub-linear growth with regard to the length of the time horizon, and that the *violation* of the long-term constraint due to battery state transitions is sub-linear, i.e., diminishing asymptotically. Finally, combining these results through introducing intermediate auxiliary terms, we prove that our original problem can adopt a sub-linear upper bound upon its *regret* with regard to overall static offline optimal decisions.

Fourth, we conduct extensive experiments using real LLM inference datasets [25–29], draft-target LLM pairs [7, 8], workload traces [30], grid-energy prices [31], and edge locations [32], as well as cloud-edge communication conditions [1, 33], capacity [11], efficiency [9], and battery degradation parameters [34]. We compare our proposed approach against multiple state-of-the-art alternatives under various settings, and obtain the following results: (i) our approach reduces the total cost by about 20%∼50% compared to others; (ii) the regrets and the violation grow sub-linearly in practice, aligned with our theoretical analysis; (iii) our approach scales with robustness as workload and communication budget vary; (iv) by adjusting weights and other parameters, our approach can successfully navigate optimization trade-offs between inference latency and energy consumption; (v) our algorithms finish execution within tens of milliseconds in each one-minute time slot, fully acceptable in reality; (vi) as grid-energy price fluctuates, the battery energy is well utilized by our algorithms to jointly supply adequate energy to LLM inference at minimum total expense.

2 Modeling and Formulation

2.1 System Settings and Models

We summarize our main notations and descriptions in Table 1.

Cloud-Edge LLM Infrastructure: We target an LLM service provider that owns a cloud-edge infrastructure and uses this infrastructure to offer the LLM inference service to the end users. This

Table 1: Notations and Descriptions

Input	Description
$\mathcal{I}$	Set of inference requests
$\mathcal{N}$	Set of edges
$\mathcal{T}$	Set of time slots
$\mathbf{h}_{i,n}$	Context for request i on edge n
$\mathbf{u}^{\text{draft}}$	Sequence of draft tokens on edge
$\mathbf{q}^{\text{draft}}$	Sequence of probabilities for draft tokens
$\mathbf{u}^{\text{target}}$	Output sequence in cloud
u_k	k-th draft token in draft sequence $\mathbf{u}^{\text{draft}}$
κ	First position where rejection occurs in acceptance test
$\tilde{u}_\kappa$	Corrected or newly-generated token at position κ
v_k	Random value sampled for acceptance at position k
p_k^{draft}	Probability of k-th draft token by draft LLM (abbreviated as q_k)
p_k^{target}	Probability of k-th draft token by target LLM (abbreviated as p_k)
f_n^{draft}	Conditional probability distribution function of draft LLM on edge n
f^{target}	Conditional probability distribution function of target LLM in cloud
α	Probability of accepting a draft token by target LLM
U_{nt}	Number of accepted draft tokens per iteration on edge n at t
G_{nt}	Number of output tokens per iteration on edge n at t
M_{int}	Total length of output sequence for request i on edge n at t
T_n'	Time consumed for auto-regressively generating a token on edge n
D_n	Communication set-up time between edge n and cloud
β_n	Time consumed for edge n to transfer a token with probability to cloud
β_n'	Time per token for cloud to transfer output sequence to edge n
β_n''	Network traffic of sending a draft token and probability from edge n
T''	Time consumed by a parallel verification for draft tokens in cloud
φ_{int}	Total delay for resolving request i on edge n at t
T_{nt}	Total delay of all steps per iteration on edge n at t
ρ_{int}	Network traffic incurred between edge n and cloud for request i at t
A_n	Budget on total cumulative traffic of edge n
ξ_{int}'	Energy of edge n for generating draft tokens for request i at t
ξ_{int}	Energy of edge n for transmitting draft tokens for request i at t
ψ_{int}	Energy of cloud for handling draft tokens of request i from edge n at t
P_n, Q_n	Energy efficiency constants depending on draft LLM on edge n
P, Q	Energy efficiency constants depending on target LLM in cloud
r_n	Energy of edge n per single-token transmission
s_t	Energy price of power grid at t
a_n	Minimum battery energy limit of edge n
b_{nt}	Available battery energy of edge n at t
c_n	Maximum battery energy limit of edge n
m_n	Acquisition price of battery for edge n
o_n	Maximum number of cycles that battery of edge n can operate
d_n	Degradation cost per unit energy charged or discharged for edge n
η_c	Battery charging efficiency
η_d	Battery discharging efficiency
Decision	**Description**
γ_{nt}	Number of draft tokens to generate per iteration on edge n at t
x_{nt}	Amount of grid energy to charge battery of edge n at t
y_{nt}	Amount of energy to discharge from battery of edge n at t
z_{nt}	Amount of grid energy for supporting LLM inference on edge n at t
e_t	Amount of grid energy for supporting LLM inference in cloud at t

infrastructure consists of a cloud data center and a group of distributed and potentially heterogeneous "edges", where each edge is a micro data center or server cluster co-located with a cellular base station or WiFi access point in close proximity to the end users. The cloud and the edges are interconnected via wired networks, and the end users connect to the edges via wireless or wired networks. We denote the set of edges as $\mathcal{N} = \{1, 2, \ldots, N\}$. We consider the system operating over a series of time slots $\mathcal{T} = \{1, 2, \ldots, T\}$.

LLM Inference via Speculative Decoding: Without loss of generality, the distributed speculative-decoding-based LLM inference service has a full-fledged powerful *target LLM* run in the cloud and one lightweight and often smaller *draft LLM* run on each edge.

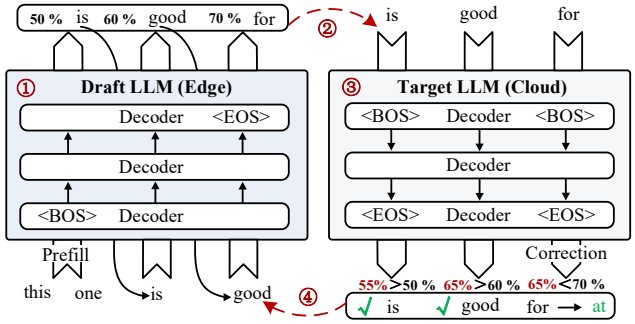


Figure 2: Indicative example of speculative decoding: The draft LLM autoregressively generates draft tokens "is good for" (Step 1), and sends them to the target LLM (Step 2); The target LLM verifies and corrects them (Step 3), and returns output tokens "is good at" to the draft LLM (Step 4).

At each time slot t, for each inference request $i \in \mathcal{I} = \{1, 2, \ldots, I\}$ (i.e., a sequence of input tokens) that arrives at the edge n from the end user, the edge n sends the context $\mathbf{h}_{i,n}$ (e.g., input tokens from the user, plus other system prompts if needed) to the cloud and afterwards the LLM inference (i.e., generating the sequence of output tokens) using *speculative decoding* is an iterative process between the edge n and the cloud, where each iteration is composed of multiple steps described as follows (also see Figure 2):

- *Step 1*: The draft LLM on the edge n auto-regressively generates a speculative draft token sequence of the length $\gamma_{nt} \in \{1, ..., \gamma_{\max}\}$ as pre-specified, denoted as $\mathbf{u}^{\text{draft}} \triangleq \{u_1, ..., u_{\gamma_{nt}}\}$, where $u_k, \forall k \in \{1, ..., \gamma_{nt}\}$ refers to the k-th draft token. Each $u_k, \forall k$ is sampled from the conditional probability distribution $p_k^{\text{draft}}(\cdot) \triangleq f_n^{\text{draft}}(\cdot|\mathbf{h}_{i,n}, \mathbf{u}_{1:k-1}^{\text{draft}})$, where f_n^{draft} depends on the draft LLM; $\mathbf{h}_{i,n}$ is the context so far; and $\mathbf{u}_{1:k-1}^{\text{draft}}$ is the sequence of draft tokens before the k-th draft token. The edge n also extracts the actual probability of the draft token $u_k, \forall k$ as $q_k \triangleq p_k^{\text{draft}}(u_k)$, thereby producing $\mathbf{q}^{\text{draft}} \triangleq \{q_1, ..., q_{\gamma_{nt}}\}$.
- *Step 2*: The edge n sends $\mathbf{u}^{\text{draft}}$ and $\mathbf{q}^{\text{draft}}$ to the cloud.
- *Step 3*: The target LLM in the cloud processes the entire sequence $\mathbf{u}^{\text{draft}}$ in one parallel forward pass to compute the conditional probability distribution for each position k as $p_k^{\text{target}}(\cdot) \triangleq f^{\text{target}}(\cdot|\mathbf{h}_{i,n}, \mathbf{u}_{1:k-1}^{\text{draft}})$, where f^{target} depends on the target LLM. The cloud then performs the acceptance test sequentially. That is, for each position k, it calculates the probability $p_k \triangleq p_k^{\text{target}}(u_k)$, samples a random value $v_k \sim \text{Uniform}(0, 1)$, and accepts the draft token u_k if $v_k \leq \min\{1, \frac{p_k}{q_k}\}$. Let κ be the first position where rejection occurs, i.e., $\kappa \leq \gamma_{nt}$ or $\kappa = \gamma_{nt} + 1$. The output sequence is $\mathbf{u}^{\text{target}} \triangleq \{u_1, ..., u_{\kappa-1}, \tilde{u}_\kappa\}$, where $\tilde{u}_\kappa \sim p_\kappa^{\text{target}}(\cdot)$ is the corrected or newly-generated token. The cloud updates the context as $\mathbf{h}_{i,n} \leftarrow \mathbf{h}_{i,n} \oplus \mathbf{u}^{\text{target}}$.
- *Step 4*: The cloud sends the output sequence $\mathbf{u}^{\text{target}}$ to the edge n, and the edge n updates the context as $\mathbf{h}_{i,n} \leftarrow \mathbf{h}_{i,n} \oplus \mathbf{u}^{\text{target}}$, preparing for the next iteration.

Without loss of generality and following existing research [3], we consider the expected number of tokens in the output sequence produced per iteration in speculative decoding for each inference

request from the edge n at the time slot t. Denote α as the probability for the cloud to accept a single draft token. Let U_{nt} be the number of accepted draft tokens, and G_{nt} be the number of tokens in the output sequence. Then, we have[1] $\mathbb{E}[G_{nt}] = 1 + \mathbb{E}[U_{nt}] = 1 + \sum_{k=1}^{\gamma_{nt}} \mathbb{P}[U_{nt} \geq k] = 1 + \sum_{k=1}^{\gamma_{nt}} \alpha^k = 1 + \alpha \cdot \frac{1-\alpha^{\gamma_{nt}}}{1-\alpha} = \frac{1-\alpha^{\gamma_{nt}+1}}{1-\alpha}$.

Delay of LLM Inference: We account for the total computation and communication delay for resolving the inference request i via distributed speculative decoding between the edge n and the cloud at the time slot t. Note that the delay and the traffic incurred by transferring the *context* of each inference request to the cloud naturally relies on system inputs and not on the control decisions considered in this work, also explained next.

First, given M_{int} as the total length of the output sequence, i.e., the total number of output tokens for responding to this inference request, the number of cloud-edge speculative decoding iterations for this inference request is $M_{int}/\mathbb{E}[G_{nt}]$.

Second, each iteration consists of *Steps 1~4* as stated, and for each of such iterations, we have the following: (i) Computation delay of *Step 1* is $\gamma_{nt} T_n^{'}$, where $T_n^{'}$ represents the time consumed for all operations related to auto-regressively generating a single token on the edge n; (ii) Communication delay of *Step 2* is $D_n + \beta_n \gamma_{nt}$, where D_n is the communication set-up time between the edge n and the cloud, and β_n is the time consumed for the edge n to transfer a single token with its probability to the cloud; (iii) Computation delay of *Step 3* is $T^{''}$, which is the time consumed in the cloud by a single parallel forward pass for draft token verifications (note that the sequential acceptance test is often negligible in time compared to the forward pass); (iv) Communication delay of *Step 4* is $D_n + \beta_n^{'} \mathbb{E}[G_{nt}]$, where $\beta_n^{'}$ is the time per token for the cloud to transfer the output sequence to the edge n.

Therefore, the total delay, written as a function of γ_{nt}, is

$$\varphi_{int}(\gamma_{nt}) \triangleq M_{int}/\mathbb{E}[G_{nt}] \cdot T_{nt}, \tag{1}$$

where $T_{nt} = \gamma_{nt} T_n^{'} + 2D_n + \beta_n \gamma_{nt} + T^{''} + \beta_n^{'} \mathbb{E}[G_{nt}]$ is the total delay of all steps per iteration.

Traffic of LLM Inference: For the speculative-decoding-based LLM inference for each inference request i from the edge n at the time slot t, the network traffic incurred between the edge n and the cloud, written as a function of γ_{nt}, is

$$\rho_{int}(\gamma_{nt}) \triangleq M_{int}/\mathbb{E}[G_{nt}] \cdot (\beta_n^{''} \gamma_{nt} + \mathbb{E}[G_{nt}]), \tag{2}$$

where $\beta_n^{''}$ refers to the amount of network traffic incurred by sending a single draft token and its probability. To prevent the consumption of excessive communication resource, we impose a budget A_n on the total cumulative traffic of edge n at every t.

Energy of LLM Inference: First, the energy consumption of the edge includes the energy of computation and the energy of transmission. For the inference request i that arrives at the edge n at the time slot t, the draft LLM performs "prefill" and "decode" operations to auto-regressively generate the draft tokens, which consumes the energy of $\xi_{int}^{'}(\gamma_{nt}) \triangleq P_n \cdot |\mathbf{h}_{i,n}|^2 + Q_n \cdot |\mathbf{h}_{i,n}| \cdot M_{int}/\mathbb{E}[G_{nt}] \cdot \gamma_{nt}$, where P_n and Q_n are constants depending on the draft LLM, the server energy efficiency, etc.; the energy for transmitting all draft tokens to the cloud is $\xi_{int}^{''}(\gamma_{nt}) \triangleq r_n \cdot M_{int}/\mathbb{E}[G_{nt}] \cdot \gamma_{nt}$, where r_n is the energy used by the edge n per single-token transmission.

[1]The notation $\mathbb{E}[G_{nt}]$ could be $\mathbb{E}[G_{nt}|\alpha]$ to be more precise, as it is a conditional expectation for a given α.

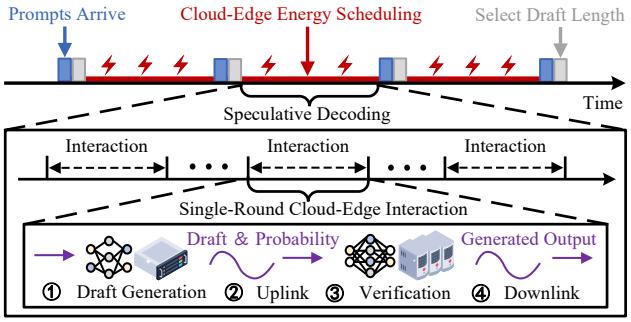


Figure 3: Time flow of cloud-edge system

Second, the energy consumption of the cloud also includes the energy of computation and the energy of transmission. We only consider the former because the energy for transmitting the entire output sequence to the edge n depends on system inputs and is thus not the target of our optimization. For the inference request i from the edge n at the time slot t, due to the essential similarity between the forward pass and acceptance test operations and the prefill and decode operations, the computation energy can be expressed as $\psi_{int}(\gamma_{nt}) \triangleq M_{int}/\mathbb{E}[G_{nt}] \cdot (P(|\mathbf{h}_{i,n}| + \gamma_{nt})^2 + Q(|\mathbf{h}_{i,n}| + \gamma_{nt}))$, where P and Q are constants depending on the target LLM, the cloud energy efficiency, etc.

Energy Expense and Battery Management: The cloud-edge LLM inference infrastructure connects to a power grid; each edge also has its local battery or battery groups equipped, where the batteries can be charged or discharged dynamically. To support the LLM inference operations, an edge can draw energy from both the grid and its battery, and the cloud can draw energy form the grid. We denote the grid energy price at the time slot t as s_t, i.e., drawing grid energy to either charge the batteries or directly support LLM inference incurs energy expense. For the battery of the edge n at the time slot t, we denote its available battery energy as b_{nt}, enforcing $a_n \leq b_{nt} \leq c_n$, where a_n and c_n are the desired minimum and maximum battery limits, respectively, and can be specified by the system or the battery manufacturer. As every charge or discharge operation harms the battery's lifespan, we consider the degradation [1] per unit energy charged or discharged as $d_n \triangleq m_n/\{2o_n(c_n - a_n)\}$, where m_n is the acquisition price of the battery and o_n is the maximum number of cycles that the battery can be operated for. We also use η_c and η_d to represent the battery charging and discharging efficiency. That is, if the battery of the edge n charges x_n^t and discharges y_n^t amount of energy, then the net charged and discharged energy is $x_n^t \eta_c$ and y_n^t/η_d. See Figure 3 for the time flow visualization.

Control Decisions: At each time slot t, the cloud-edge speculative-decoding-based LLM inference system makes the following control decisions: (i) $\gamma_{nt} \in \{1, ..., \gamma_{\max}\}$, denoting the number of draft tokens to generate per iteration at the edge n with regard to all inference requests that arrive at the edge n; (ii) $x_{nt} \geq 0$, denoting the amount of grid energy used to charge the battery of the edge n; (iii) $y_{nt} \geq 0$, denoting the amount of energy discharged from the battery of the edge n to support the LLM inference on the edge n; (iv) $z_{nt} \geq 0$,

denoting the amount of grid energy used to support the LLM inference on the edge n; (v) $e_t \geq 0$, denoting the amount of grid energy used to support the LLM inference in the cloud.

Total Cost of Cloud-Edge LLM Inference: The total cost of LLM inference contains multiple components as follows: (i) the total latency, i.e., $\sum_t \sum_i \sum_n \varphi_{int}(\gamma_{nt})$; (ii) the total energy expense of all edges, i.e., $\sum_t \sum_n s_t(x_{nt} + z_{nt})$; (iii) the total energy expense of the cloud, i.e., $\sum_t s_t e_t$; (iv) the degradation cost incurred by all edge batteries, i.e., $\sum_t \sum_n (d_n(x_{nt}\eta_c + y_{nt}/\eta_d))$.

2.2 Problem Formulation and Challenges

Problem Formulation: Using the models shown above, we formulate the optimization problem that controls the speculative decoding process to minimize the total cost:

$$\begin{aligned}
\min \quad & \mathcal{P} \triangleq \textstyle\sum_t \sum_i \sum_n \varphi_{int}(\gamma_{nt}) + \sum_t (s_t(\sum_n (x_{nt} + z_{nt}) + e_t)) \\
& + \textstyle\sum_t \sum_n (d_n(x_{nt}\eta_c + y_{nt}/\eta_d)) && (3)\\
\text{s.t.} \quad & \textstyle\sum_i \rho_{int}(\gamma_{nt}) \leq A_n, \forall t, \forall n, && (3a)\\
& b_{nt} = b_{n,t-1} + x_{nt}\eta_c - y_{nt}/\eta_d, \forall t, \forall n, && (3b)\\
& a_n \leq b_{nt} \leq c_n, \forall t, \forall n, && (3c)\\
& \textstyle\sum_i (\xi'_{int}(\gamma_{nt}) + \xi''_{int}(\gamma_{nt})) \leq y_{nt} + z_{nt}, \forall t, \forall n, && (3d)\\
& \textstyle\sum_i \sum_n \psi_{int}(\gamma_{nt}) \leq e_t, \forall t, && (3e)\\
\text{var.} \quad & \gamma_{nt} \in \{1, ..., \gamma_{\max}\}, x_{nt} \geq 0, y_{nt} \geq 0, z_{nt} \geq 0, e_t \geq 0, \forall t, \forall n. && (3f)
\end{aligned}$$

The objective (3) minimizes the total cost of LLM inference. Constraint (3a) ensures the network traffic remains within the budget. Constraint (3b) reflects the state transition of battery energy. Constraint (3c) applies the battery limits. Constraint (3d) ensures the power grid and the edge battery jointly offer sufficient energy to each edge to support LLM inference. Constraint (3e) ensures the power grid offers sufficient energy to the cloud to support LLM inference. Constraint (3f) specifies the domains of decision variables.

Problem Challenges: It is not trivial to solve the problem $\mathcal{P}$ online.[2] First, $\mathcal{P}$ is online optimization with time-coupled state transitions. Constraints (3b) and (3c) impose a decisive relation upon the control decisions x_{nt} and y_{nt} across time slots. Every decision made currently based on current inputs will inevitably impact what decisions can be made in the future; however, future inputs remain uncertain currently, making anticipatory decision-making hard. Second, many inputs, including $\varphi_{int}(\cdot)$, to $\mathcal{P}$ are posterior, i.e., only revealed at t after a concrete control action is taken at t, which escalates the difficulty of online decision-making. Third, the mixed-integer nonlinearity makes $\mathcal{P}$ intractable even in an offline setting, not to mention we desire to solve it in an online manner. Overall, this problem is difficult to solve because of its discrete decisions, online uncertainty, and time-coupled constraints.

3 Online Algorithms Design

3.1 Decomposed Scheduling: Algorithm 1

Algorithmic Rationale: Our core idea is decomposing our original problem $\mathcal{P}$ into two subproblems $\mathcal{P}_1$ and $\mathcal{P}_2$, and then addressing the *integrality* and *posterior*, contained in $\mathcal{P}_1$, and the *state transitions* and *online uncertainty*, contained in $\mathcal{P}_2$, separately. Based on this idea, we design Algorithm 1, which we call the Energy-Aware Online Scheduling framework (EAOS), as our overall control algorithm. In each time slot, Algorithm 1 invokes Algorithms 2 and 3 to address $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, which will be elaborated in the next two subsections, and performs the speculative-decoding-based LLM inference following the control decisions made.

Algorithm 1 Energy-Aware Online Scheduling Framework

1: **Initialize:** Set $x_{n0} = y_{n0} = z_{n0} = e_0 = 0$;
2: **for** $t = 1, 2 \cdots T$ **do**
3: Invoke **Algorithm 2** to get draft length γ_{nt};
4: Given draft length $\bar{\gamma}_{nt}$, invoke **Algorithm 3** to get e_t for cloud energy and $\{x_{nt}, y_{nt}, z_{nt}\}$ for edge energy;
5: Execute *speculative decoding* on the cloud and edges;
6: **end for**

Problem Decomposition: We derive the two subproblems.

The first subproblem $\mathcal{P}_1$ is as follows:

$$\begin{aligned}
\min \quad & \mathcal{P}_1 \triangleq \textstyle\sum_t \sum_i \sum_n \varphi_{int}(\gamma_{nt}) && (4)\\
\text{s.t.} \quad & \textstyle\sum_i \rho_{int}(\gamma_{nt}) \leq A_n, \forall t, \forall n, && (4a)\\
\text{var.} \quad & \gamma_{nt} \in \{1, ..., \gamma_{\max}\}, \forall t, \forall n. && (4b)
\end{aligned}$$

The objective (4) is the total latency. Constraint (4a) originates from (3a). Constraint (4b) specifies the domain for the draft length.

The second subproblem $\mathcal{P}_2$ is as follows:

$$\begin{aligned}
\min \quad & \mathcal{P}_2 \triangleq \textstyle\sum_t (s_t(\sum_n (x_{nt} + z_{nt}) + e_t)) \\
& + \textstyle\sum_t \sum_n (d_n(x_{nt}\eta_c + y_{nt}/\eta_d)) && (5)\\
\text{s.t.} \quad & \textstyle\sum_t g_t \triangleq \sum_t (T-t)(x_{nt}\eta_c - y_{nt}/\eta_d) \\
& - T(c_n - a_n) \leq 0, \forall n, && (5a)\\
& h_{1t} \triangleq \textstyle\sum_i (\xi'_{int}(\bar{\gamma}_{nt}) + \xi''_{int}(\bar{\gamma}_{nt})) \\
& - y_{nt} - z_{nt} \leq 0, \forall t, \forall n, && (5b)\\
& h_{2t} \triangleq \textstyle\sum_i \sum_n \psi_{int}(\bar{\gamma}_{nt}) - e_t \leq 0, \forall t, && (5c)\\
\text{var.} \quad & e_t \geq 0, x_{nt} \geq 0, y_{nt} \geq 0, z_{nt} \geq 0, \forall t, \forall n. && (5d)
\end{aligned}$$

The objective (5) captures the energy expense and the degradation cost of edge batteries. Constraint (5a) is derived from the combination of (3b) and (3c), that is, $b_{n,T+1} = b_{nT} + x_{nT}\eta_c - y_{nT}/\eta_d = b_{n,T-1} + \sum_{t=T-1}^{T}(x_{nt}\eta_c - y_{nt}/\eta_d) = ... = b_{n0} + \sum_{t=1}^{T}(x_{nt}\eta_c - y_{nt}/\eta_d)$, $\forall n$, which leads to $\sum_{t=1}^{t}(x_{nt}\eta_c - y_{nt}/\eta_d) = b_{n,t+1} - b_{n0} \leq c_n - a_n, \forall n$. Summing them from $t = 1$ to $t = T-1$, we obtain Constraint (5a). Constraints (5b) and (5c) are respectively transformed from (3d) and (3e), where $\bar{\gamma}_{nt}$, as solved in the problem $\mathcal{P}_1$,[3] is applied as input. The decision variables of energy scheduling are listed in (5d).

Remarks and Insights: (i) In the objective of $\mathcal{P}$, the draft length γ_{nt} affects latency through the expected output token count $\mathbb{E}[G_{nt}] = \frac{1-\alpha^{\gamma_{nt}+1}}{1-\alpha}$; in the constraints of $\mathcal{P}$, the traffic and energy also depend on γ_{nt} through $\mathbb{E}[G_{nt}]$. This motivates us to leverage it as a bridge between the two subproblems in our algorithm design. (ii) Given the aforementioned challenges for $\mathcal{P}$, we use decomposition to prioritize online tractability and operational feasibility over absolute optimality, while rigorously bounding the resulting sub-optimality. Section 4 offers the performance analysis, bounding

[2] We use the notation like $\mathcal{P}$ to refer to either an optimization problem or just its objective function, which is clear from the context.

[3] For decisions, we use the notation like γ_{nt} to refer to the decision *variable*, and the notation like $\bar{\gamma}_{nt}$, i.e., decorated symbol, to refer to a specific *value* that the corresponding decision variable takes.

Algorithm 2 Draft Length Control Algorithm

Input: Inference requests $i \in \mathcal{I}_t = \{1, \cdots, I_t\}$, $I = \max I_t$.
Output: Selected draft length $\bar{\gamma}_{nt}$.
1: **Initialize:** weight $v^j_{n1} = 1$ for $\forall j \in \{1, ..., \gamma_{\max}\}$, $\forall n$;
2: **for** $\forall t$, $\forall n$ **do**
3: Construct feasible set $\mathcal{F}_{nt} = \{j : \sum_i \rho_{int}(\gamma^j_{nt}) \le A_n\}$;
4: Compute probability $p^j_{nt} = (1-\varsigma)\frac{v^j_{nt}}{\sum_{j\in\mathcal{F}_{nt}} v^j_{nt}} + \frac{\varsigma}{|\mathcal{F}_{nt}|}$;
5: Sample arm $\bar{j}$ from p^j_{nt}, $\forall j \in \mathcal{F}_{nt}$, then $\bar{\gamma}_{nt} \leftarrow \bar{j}$;
6: Observe latency $\varphi_{1nt}(\gamma^{\bar{j}}_{nt}), \cdots, \varphi_{I_t nt}(\gamma^{\bar{j}}_{nt})$
7: Update estimator $\ell^{\bar{j}}_{nt} = \frac{\sum_{i\in\mathcal{I}_t}\varphi_{int}(\gamma^{\bar{j}}_{nt})}{p^{\bar{j}}_{nt}}$, $\ell^j_{nt} = 0$ for $j \ne \bar{j}$;
8: Update weight $v^j_{n,t+1} = v^j_{nt}\exp(-\frac{\varsigma}{I\gamma_{\max}}\ell^j_{nt})$.
9: **end for**

such sub-optimality, and serves as the theoretical evidence that the decomposition-based approach works.

3.2 Control of Draft Length: Algorithm 2

For the problem $\mathcal{P}_1$, as the number of users' prompts varies over time and Constraint (4a) limits the feasible draft length options, we propose to design *adversarial bandit learning* in Algorithm 2. Unlike standard adversarial bandit learning where only one *reward* is disclosed after one *arm* is selected, in our case here, we have a varying number of multiple rewards (due to processing multiple inference requests) after one draft length option is chosen. We will need to handle this non-trivial difference in our algorithm.

Algorithm 2 addresses the bandit problem for each edge n separately, and treats different draft lengths as different arms while maintaining and updating a set of weights v^j_{nt} for each arm j as time goes. In Line 1, all arm weights are initialized to 1 to reflect no prior preference among arms. For each edge n, at each time slot t, in Line 3, we first construct a feasible set $\mathcal{F}_{nt}$ that includes only those arms whose cumulative traffic footprint across all prompts does not exceed the edge's budget. In Line 4, based on the weights, we compute a probability distribution p^j_{nt} over $\mathcal{F}_{nt}$, which is a mixture of an exploitation term proportional to the weights and a uniform exploration term scaled by a parameter ς. This exploration-exploitation trade-off is essential in adversarial environments where the inference processing latency patterns may change unpredictably. Afterwards, we sample an arm $\bar{j}$ according to p^j_{nt} in Line 5, observe the resulting latencies $\varphi_{int}(\gamma^{\bar{j}}_{nt})$ for all prompts in Line 6, and construct an unbiased estimator ℓ^j_{nt} by dividing the total observed latency by the selection probability in Line 7. This importance-weighted estimator is crucial for handling the partial feedback nature of the bandit problem, where only the latency of the selected arm is observed, yet we need to estimate the performance of all arms. Finally, in Line 8, the weights of all the arms are updated using exponential weighting, where the magnitude of updating is scaled by $\varsigma/(I\gamma_{\max})$ to ensure stability. Arms that yield higher latency receive larger estimated losses, causing their weights to decrease exponentially, making them less likely to be selected in future time slots. The time complexity of Algorithm 2 is $O(IN\gamma_{max})$.

Remarks and Insights: (i) Our method enforces Constraint (4a) by restricting the action space to $\mathcal{F}_{nt}$, which eliminates constraint violations without parameter tuning and makes the feasibility check computationally light. (ii) We have successfully handled the setting where each arm selection yields multiple rewards by aggregating such rewards through inverse probability weighting to obtain a single unbiased estimator for the chosen arm.

Algorithm 3 Energy Control Algorithm

Input: $f_t(\cdot), g_t(\cdot), h_{1t}(\cdot), h_{2t}(\cdot), \omega_t = \{x_{nt}, y_{nt}, z_{nt}, e_t\}$.
Output: $\omega_{t+1} = \{x_{n,t+1}, y_{n,t+1}, z_{n,t+1}, e_{t+1}\}$.
1: **Initialize:** $\omega_0 = 0$, $\lambda_0 = 0$, $g_0(\cdot) = 0$;
2: Update decision ω_{t+1} according to (6);
3: Update virtual queues $\lambda_{n,t+1}$ according to (7).

3.3 Control of Energy: Algorithm 3

For the problem $\mathcal{P}_2$, to facilitate our algorithm design, we donate $\omega_t \triangleq \{x_{nt}, y_{nt}, z_{nt}, e_t\}$ as the control variables. We decompose $\mathcal{P}_2$ into one-shot problems $\mathcal{P}_{2,t}$, $\forall t$ and solve it with a primal-dual-based method that progresses online without future inputs.

At any time slot t, we first solve the following one-shot problem and assign its solution to ω^{t+1}:

$$\min \mathcal{P}_{2,t} \triangleq f_t(\omega) + (\lambda_t + g_{t-1}(\omega_t))^\top g_t(\omega) + \mu\|\omega - \omega_t\|_2^2 \quad (6)$$
$$\text{s.t.} \quad h_{1t}(\omega_t) \le 0, h_{2t}(\omega_t) \le 0, \forall t, \quad (6a)$$
$$\text{var.} \quad \omega_t \in \mathbb{W}, \forall t, \quad (6b)$$

where ω_t is the primal decision variable and λ_t is the dual decision variable. The parameter μ is a positive step size, and the term $\mu\|\omega - \omega_t\|_2^2$ is a regularization and proximal term. Next, we construct a "virtual queue" that is updated as

$$\lambda_{n,t+1} = \max\{-g_{nt}(\omega_{t+1}),\ \lambda_{nt} + g_{nt}(\omega_{t+1})\}, \forall t, \forall n. \quad (7)$$

Here, $\lambda_t = [\lambda_{1t}, \lambda_{2t}, \cdots, \lambda_{Nt}]^\top$ and $g_t = [g_{1t}, g_{2t}, \cdots, g_{Nt}]^\top$. If we substitute the initial term inside the maximization with zero, the update for $\lambda_{n,t+1}$ becomes equivalent to conventional queue updates using the increments $g_{nt}(\omega_{t+1})$. Consequently, we can find that $\lambda_{n,t+1}, \forall t, \forall n$ serve to quantify the accumulated constraint violations and that the bounds for the queue backlogs can be directly converted into corresponding bounds for the constraint violations.

Algorithm 3 adopts the above idea. In such alternate steps, i.e., updating ω_{t+1} in (6) and λ_{t+1} in (7), we only need the inputs before (excluding) $t+1$ rather than any future information. $\mathcal{P}_{2,t}$ can be solved through standard convex optimization solvers, such as the interior-point method [35]. The time complexity of Algorithm 3 is thus $O(N^2\log(1/\sigma))$ for calculating a σ-accurate solution.

Remarks and Insights: (i) Our step size μ is designed independently of the time horizon length T, making it suitable for scenarios where the total length of the entire time horizon is unknown in advance and the online optimization may terminate at any arbitrary time slot. (ii) When computing ω_t for t, no information at or after t is needed, making the computation process entirely on the fly.

4 Theoretical Performance Analysis

We first study the theoretical guarantees that come with our Algorithms 2 and 3 that solve $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. Then, we combine them to obtain the performance of our entire approach, i.e., Algorithm 1, which solves the original problem $\mathcal{P}$. Note that

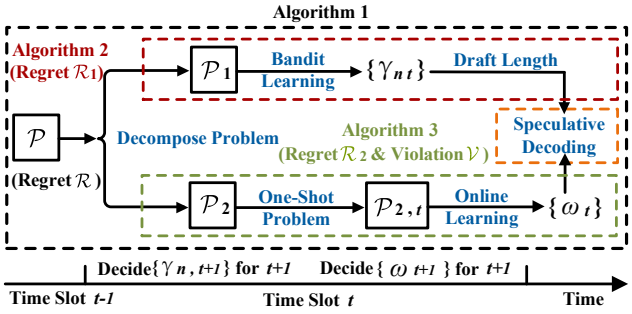


Figure 4: Roadmap for theoretical analysis

such combination analysis is non-trivial. The roadmap for such theoretical analysis is shown in Figure 4.

For the problem $\mathcal{P}_1$, the *regret* captures the gap between the expected objective value evaluated with the online solutions from our bandit-learning-based approach and the objective value evaluated with the optimum that selects the best arm for the entire time horizon in hindsight. We prove that such regret accumulated over time is sub-linear with regard to the length of the time horizon T, i.e., it grows slower than the natural passage of time.

Theorem 4.1. *The regret of $\mathcal{P}_1$ is upper-bounded as*

$$\mathcal{R}_1 \triangleq \mathbb{E}[\mathcal{P}_1(\bar{\gamma}_{nt})] - \mathcal{P}_1(\gamma_n^*) \leq 2IN\sqrt{2T\gamma_{max}\ln\gamma_{max}},$$

where $\bar{\gamma}_{nt}$ is the online solution obtained from Algorithm 2 and γ_n^* is the static time-invariant offline optimal decision for $\mathcal{P}_1$ over the whole time horizon.

Proof. See Appendix A. □

For the problem $\mathcal{P}_2$, the *regret* quantifies the gap between the objective value gained by our primal-dual-based approach and the sum of the optimal objective values obtained from solving each corresponding one-shot problem independently. We prove the sub-linear growth of the regret. The *constraint violation* captures the cumulative violation of Constraint (5a) over time. That is, this long-term constraint defined for the entire time horizon (in contrast to any instantaneous constraint defined for each single time slot) is generally hard to satisfy; therefore, we allow its violation but prove such violation is also sub-linear, i.e., our approach is asymptotically feasible and the cumulative violation is decaying effectively.

Theorem 4.2. *The regret of $\mathcal{P}_2$ is upper-bounded as*

$$\mathcal{R}_2 \triangleq \mathcal{P}_2(\bar{\omega}_t) - \mathcal{P}_2(\omega_t^*) \leq o(T),$$

where $\bar{\omega}_t$ is the online solution obtained from Algorithm 3, and ω_t^* is the per-slot offline optimal decision for $\mathcal{P}_2$ given $\bar{\gamma}_{nt}$ at time slot t.

Proof. See Appendix B. □

Theorem 4.3. *The constraint violation of $\mathcal{P}_2$ is upper-bounded as*

$$\mathcal{V} \triangleq \textstyle\sum_t g_t(\bar{\omega}_t) \leq o(T),$$

where $\bar{\omega}_t$ is the online solution obtained from Algorithm 3.

Proof. See Appendix C. □

For the problem $\mathcal{P}$, regarding its *regret*, we choose to compare the expected objective value incurred by our entire online approach, i.e., Algorithm 1, against that incurred by the static offline optimum, i.e., the single best draft length and energy policy chosen with full hindsight for the entire time horizon. This is a standard practice in existing research. Yet, note that simply summing the outcomes of Theorems 4.1 and 4.2 is not valid, since their respective optima are defined in distinct senses. Instead, establishing Theorem 4.4 necessitates breaking down the overall regret into several parts, with auxiliary terms to facilitate and complete the reasoning.

Theorem 4.4. *The static regret of $\mathcal{P}$ is upper-bounded as*

$$\mathcal{R} \triangleq \mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \bar{\omega}_t)] - \mathcal{P}(\hat{\gamma}_n, \hat{\omega}) \leq o(T),$$

where $\bar{\gamma}_{nt}, \bar{\omega}_t$ are the online decisions obtained from our algorithms, and $\hat{\gamma}_n, \hat{\omega}$ are the joint static time-invariant offline optimal decisions for $\mathcal{P}$ over the whole time horizon.

Proof. See Appendix D. □

Remarks and Insights: (i) *Comparators:* For the subproblem $\mathcal{P}_1$, the comparator is the static offline optimal solution γ_n^*. For the subproblem $\mathcal{P}_2$, the comparator is the series of the per-slot optimal solution ω_t^* given the online solution $\bar{\gamma}_{nt}$ at each t from Algorithm 2. For the original problem $\mathcal{P}$, the comparator is the static offline optimal solution $\hat{\gamma}_n, \hat{\omega}$. (ii) *Coupling:* In Appendix D, to upper-bound the full regret $\mathcal{R}$, we decompose it into three gaps: the first two are bounded by exploiting the sub-linear regrets $\mathcal{R}_1$ and $\mathcal{R}_2$; the third is automatically non-positive. Consequently, $\mathcal{R}$ is also sub-linear. (iii) *Alternative:* Note that for $\mathcal{P}$, an alternative comparator can be the dynamic offline optimum, i.e., the solution to the full time-coupled problem $\mathcal{P}$ with perfect future knowledge and time-varying decisions. The gap between the aforementioned static comparator and such a dynamic comparator could be generally linear in T without additional restrictive assumptions, which we intend to investigate further in the future.

5 Experimental Evaluations

5.1 Experimental Settings

Datasets and LLMs: We adopt well-known datasets spanning diverse task categories, including the translation task (IWLST17-de-en [25], WMT14-de-en [26]), the summarization task (CNN/Daily [27]), and the question-answering task (SQuAD v2 [28], TruthfulQA [29]). Specifically, the model pairs are (LLaMA-68M, LLaMA-7B), (LLaMA-1.1B, LLaMA-7B), and (LLaMA-1.1B, LLaMA-13B), which are widely used for evaluating speculative decoding [7, 8]. The maximum draft length γ_{max} is set as 20 tokens commonly [7].

Cloud-Edge System: We use the Australian EUA dataset [32], containing 125 edge nodes and 816 mobile users in the Melbourne central business district (CBD) area. We select the location with the highest density of mobile users as the site for the cloud server. The cloud is equipped with NVIDIA A800 PCIe 80GB, and each edge is equipped with NVIDIA GeForce RTX 4090 24GB. We consider the time horizon $T = 300$ and each user sends 0~5 inference requests at each time slot, where a single time slot is set to 1 minute [36]. The number of inference requests that each user submits to her edge is set in proportion to the dynamic job arrival trace of Google clusters [30]. Based on user density, the traffic parameter β_n'' is set as 2~4 B,

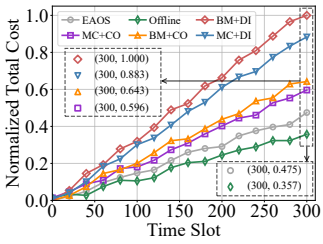


Figure 5: Total cost

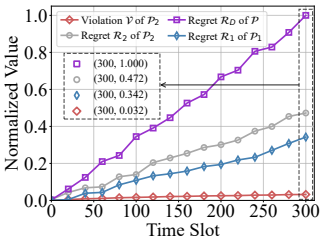


Figure 6: Regret and violation

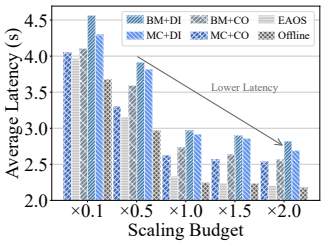


Figure 7: Impact of A_n on $\mathcal{P}_1$

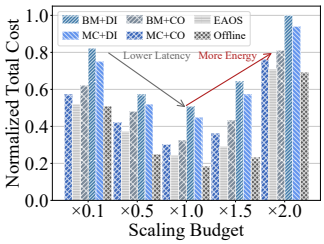


Figure 8: Impact of A_n on $\mathcal{P}$

and the budget A_n is set as 55∼65 MB [33]. The time parameter β_n is set as 0.19∼0.35 ms, $\beta_n^{'}$ set as 0.12∼0.24 ms [1], $T_n^{'}$ set as 1.7∼2.4 ms, and $T^{''}$ set as 2.2∼3.8 ms [15]. The energy parameter r_n is set as 4.2∼4.8×10^{-10} J, P_n set as 0.08∼0.15 J, Q_n set as 0.11∼0.30 J, P set as 0.27∼0.83 J, and Q set as 0.55∼1.32 J based on server efficiency and LLMs, etc [37–39]. We consider lithium-ion batteries as edge batteries, where the charging efficiency η_c is set as 99.9% and the discharging efficiency η_d is set as 85% [9]. The battery capacity at each edge is set to support the peak power demand for 12 hours [11]. The battery degradation parameter d_n is set as 8.3×10^{-5} \$/Wh [34]. We set the energy price s_t using the real-world trace of South Australia during October 26-27, 2021 [31].

Baselines: For draft length selection, we consider `BMOP` [40]: A state-of-the-art approach utilizing bandit learning with a probing budget; `MCEW` [41]: A state-of-the-art approach using bandit play with a operating budget. For cloud-edge energy scheduling, we consider `COMU` [42]: A state-of-the-art approach applying lower-and-upper-bounded virtual queues for online learning; `DINU` [43]: A state-of-the-art approach adopting gradient sampling and online mirror ascent for optimization. We combine these methods and denote all the combined approaches as (i) `BM+CO`; (ii) `BM+DI`; (iii) `MC+CO`; (iv) `MC+DI`. And we have the following offline solution: (v) `Offline`: The dynamic offline optimum (i.e., optimal solution varying over time) that solves the problem $\mathcal{P}$ as a whole via the Gurobi [44] optimization solver with all inputs provided in advance. We then compare our proposed approach `EAOS` with these baselines.

Here, we highlight that we intentionally omit the static offline optimum from our experiments because it is a strictly weaker benchmark than the dynamic offline optimum. The dynamic optimum is the absolute lower bound and a strictly stronger comparator. Any algorithm that performs well against this benchmark will necessarily outperform the static optimum. Including the static line would add visual clutter without providing additional insight into our algorithm's adaptability. Our theory already proves sub-linear static regret as a worst-case guarantee; the experiments are designed to test our algorithms' practical performance against the hardest possible comparator. This dual approach ensures both worst-case robustness (theory) and practical excellence (experiments).

5.2 Evaluation Results

All plots in Figures 5∼16 are measured from our experiments.

Total Cost: Figure 5 shows the total cost of the cloud-edge LLM inference system. Our proposed approach `EAOS` consistently outperforms other alternative approaches and is the closest to the offline optimum. The total cost of our method is ultimately decreased by 26.1% compared to `BM+CO`, by 52.5% compared to `BM+DI`, by 20.3% compared to `MC+CO`, and by 46.2% compared to `MC+DI`.

Regret and Violation: Figure 6 presents the regret $\mathcal{R}_1$ of the problem $\mathcal{P}_1$, the regret $\mathcal{R}_2$ and the constraint violation $\mathcal{V}$ of the problem $\mathcal{P}_2$, and the gap $\mathcal{R}_D$ between the objective value incurred by our online solutions and that incurred by the dynamic offline optimal solutions for the problem $\mathcal{P}$. We observe that $\mathcal{R}_1$ and $\mathcal{R}_2$ are smaller than $\mathcal{R}_D$. We also observe that the constraint violation is one order of magnitude less than the regrets.

Impact of Budget: Figure 7 and Figure 8 exhibit the impact of the communication budget A_n in the problem $\mathcal{P}_1$ and the problem $\mathcal{P}$. As A_n increases, longer draft lengths become feasible, allowing the draft model to be configured with more appropriate draft lengths to reduce the total latency. Yet, as A_n continues to increase, the cloud-edge system consumes more energy to obtain lower latency, leading to an increase in the total cost. We find that our method dynamically adjusts decisions and consistently outperforms other approaches under various restrictions on A_n.

Impact of Workload: Figure 9 visualizes the impact of the edge workload on the total cost. As the number of the LLM inference requests submitted by each user to the edge increases, each edge needs to handle more speculative decoding interactions with the cloud. The total delay and total energy consumption will both increase as a result. Our proposed method demonstrates greater stability at large scales compared to other alternative approaches.

Latency Weight: Figure 10 demonstrates the impact of the total latency weight on the total cost. The total cost includes both latency (measured in seconds) and energy overhead (measured in dollars), and the cloud-edge system can adjust the weights between these two factors to achieve different optimization priorities. We observe that our proposed approach continues to show superior performance across varying total latency weight settings.

Execution Time: Figure 11 reflects the execution time of Algorithm 2 and Algorithm 3 (the latency of speculative decoding is shown in Figures 14 and 15 below). It can be observed that both of them are able to complete within tens of milliseconds, with a gradual increase as the workload grows. This is fully acceptable compared to the response time of typical LLM services in reality.

Impact of Parameter μ: Figure 12 depicts the impact of the parameter μ which makes Algorithm 3 more accurate but adds computation in the process of operation. As shown, as μ increases, the total cost declines sharply and the runtime rises slowly. However, once μ increases beyond a certain value, further increases will cause the total cost to rise, whereas the runtime grows substantially.

Bandit Learning: Figure 13 details the bandit learning. Without violating the communication traffic constraints, Algorithm 2

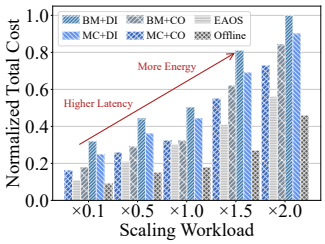


Figure 9: Impact of workload

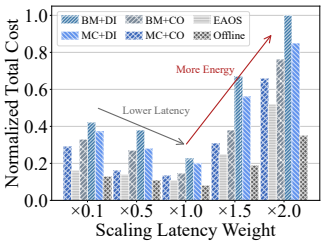


Figure 10: Latency weight

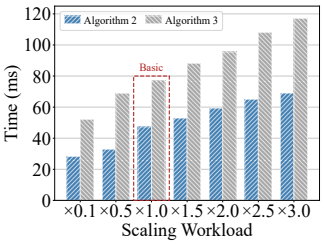


Figure 11: Execution time

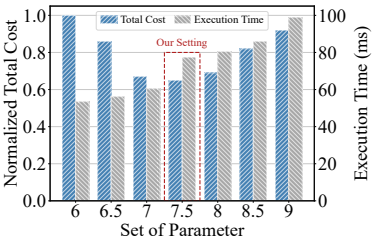


Figure 12: Impact of parameter μ

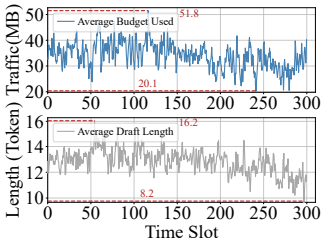


Figure 13: Bandit learning

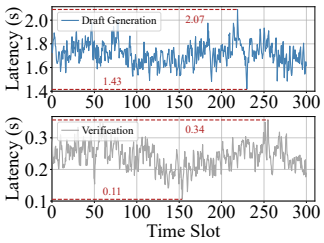


Figure 14: Comp. latency

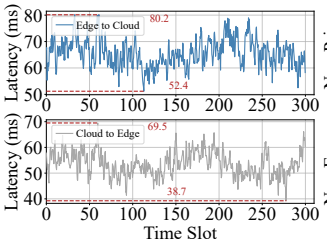


Figure 15: Comm. latency

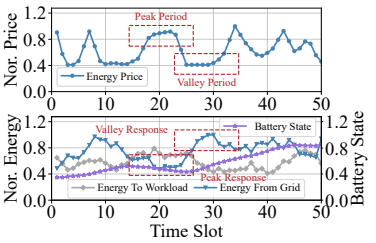


Figure 16: Battery management

strategically selects the draft length to rationally allocate communication resources across each time slot, ensuring that each inference request can be processed via speculative decoding effectively. And we have marked the maximum and minimum values in the figure.

Details of Latency: Figure 14 reflects the average computational latency of speculative decoding. As mentioned earlier, the computational latency includes the processing time at both the edge and the cloud. Figure 15 reveals the average communication latency of speculative decoding. Also, as previously stated, the communication latency includes the time for bidirectional communication between the edge and the cloud. And we have marked the maximum and minimum values of the latency in the Figures 14 and 15.

Battery Management: Figure 16 illustrates the scheduling of edge battery operations. We find that when the energy price from the power grid is at its peak or valley period, the energy from the edge batteries to the workload, and the energy from the grid to the edge batteries, will respond in a regular pattern timely. Correspondingly, the total battery state will also change regularly according to the energy price. This response mechanism can effectively utilize battery buffering to provide continuous energy to the LLM services.

6 Related Work

We review existing research in different groups, and for each group, highlight the insufficiency compared to our work, respectively.

LLM Inference Services with Cloud-Edge Collaboration: He *et al.* [4] study LLM task offloading and resource allocation in cloud-edge settings via active inference. Ma *et al.* [12] address LLM scheduling and data flow in cloud-edge environments. Yao *et al.* [1] improve LLM service quality and response time using multi-agent reinforcement learning within a vector database assisted cloud-edge framework. Ren *et al.* [13] investigate LLM inference offloading and computational load reduction by combining reinforcement learning with decentralized identity management. Huang *et al.* [14] optimize LLM prompt security, latency, and resource use in cloud-edge systems using a multi-stage dynamic Bayesian game model.

This group of works exploit cloud-edge collaboration to enable and provision LLM inference services, but fail to investigate speculative decoding, which is a concrete technique to significantly accelerate LLM inference. They also typically lack considerations of edge battery operations for sustainable LLM services.

Speculative Decoding in Resource-Limited Environments: Xu *et al.* [15] accelerate constrained token generation with adaptive draft branch navigation and pipelined verification. Ye *et al.* [16] optimize constrained prefill and autoregressive stages via a carefully-designed speculative decoding method. Svirschevski *et al.* [17] enable high-throughput decoding on memory-limited hardware via a draft token tree. Ning *et al.* [5] cut communication overhead in device-edge collaboration by split verification. Bhattacharjee *et al.* [6] transmit draft token distributions across constrained computing nodes via sparsification and lattice quantization.

These studies focus on speculative decoding, yet often ignore the aspect of dynamic energy consumption and management in the speculative decoding process. Moreover, they often do not take a rigorous formal approach like ours to model, analyze, and optimize the holistic system performance in an online fashion.

Battery-Aware Energy Management in Edge Computing: Zheng *et al.* [9] propose a deep reinforcement learning based approach in coping with the dynamic power demand for edge-cloud collaboration. Deng *et al.* [18] study stochastic task and energy model in wireless-powered mobile edge computing systems. Meng *et al.* [19] design a learning-driven method for decentralized model training in wireless edge computing systems to overcome limited resources and battery life. Li *et al.* [20] focus on pattern-aware online energy optimization to extend the lifespan of edge batteries. Wang *et al.* [10] transform edge AI infrastructures into virtual power plants via smart scheduling of edge battery energy.

Although these investigations cover edge batteries, they often overlook LLM inference, let alone the energy complexity caused by interactions between the cloud and the edges due to speculative decoding. Our work, to the best of our knowledge, is the first to incorporate battery-energy scheduling in LLM speculative decoding.

7 Conclusion and Future Work

Distributed speculative decoding can greatly speed up LLM inference in cloud-edge environments, but a comprehensive treatment of its delay, traffic, and energy management, especially in battery-powered mobile edge computing settings, has been largely missing. This paper fills this gap through novel analytical modeling, algorithm design, theoretical analysis, and practical experiments, focusing on inventively integrating bandit learning and online learning for system optimization in an online manner in the long run.

Future work could include multiple research directions. We intend to design incentive mechanisms and security protocols for speculative-decoding-based cloud-edge collaborative LLM inference, and also study the optimization of other types of LLM inference acceleration techniques across cloud and edge boundaries.

Acknowledgments

This work was supported in part by the Science and Technology Projects of Xizang Autonomous Region (XZ202401ZY0027), the Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, the Beijing Natural Science Foundation (2026FTZD001 and 4222033), the Beijing Key Laboratory of Multimodal Data Intelligent Perception and Governance, and the U.S. National Science Foundation (CNS-2047719 and CNS-2225949).

References

[1] Zhi Yao, Zhiqing Tang, Wenmian Yang, and Weijia Jia. 2025. Enhancing LLM QoS Through Cloud-Edge Collaboration: A Diffusion-Based Multi-Agent Reinforcement Learning Approach. *IEEE Transactions on Services Computing* 18, 3 (2025), 1412–1427.

[2] Yandi Li, Jianxiong Guo, Zhiqing Tang, Xingjian Ding, Juncheng Wang, Tian Wang, and Weijia Jia. 2025. Cloud-Edge System for Scheduling Unpredictable LLM Requests With Combinatorial Bandit. *IEEE Transactions on Services Computing* 18, 6 (2025), 3567–3580.

[3] Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast Inference from Transformers via Speculative Decoding. In *ICML*.

[4] Ying He, Jingcheng Fang, F. Richard Yu, and Victor C. Leung. 2024. Large Language Models (LLMs) Inference Offloading and Resource Allocation in Cloud-Edge Computing: An Active Inference Approach. *IEEE Transactions on Mobile Computing* 23, 12 (2024), 11253–11264.

[5] JIAHONG NING, Ce ZHENG, and Tingting Yang. 2025. DSSD: Efficient Edge-Device Deployment and Collaborative Inference via Distributed Split Speculative Decoding. In *ICML ML4Wireless Workshop*.

[6] Payel Bhattacharjee, Fengwei Tian, Meiyu Zhong, Guangyi Zhang, Osvaldo Simeone, and Ravi Tandon. 2025. Conformal Sparsification for Bandwidth-Efficient Edge–Cloud Speculative Decoding. In *NeurIPS AI4NextG Workshop*.

[7] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. SpecInfer: Accelerating Large Language Model Serving with Tree-based Speculative Inference and Verification. In *ACM ASPLOS*.

[8] Fahao Chen, Peng Li, Tom H. Luan, Zhou Su, and Jing Deng. 2025. SPIN: Accelerating Large Language Model Inference with Heterogeneous Speculative Models. In *IEEE INFOCOM*.

[9] Dongyu Zheng, Lei Liu, Guoming Tang, Yi Wang, and Weichao Li. 2024. Power Demand Reshaping Using Energy Storage for Distributed Edge Clouds. *IEEE Transactions on Parallel and Distributed Systems* 35, 2 (2024), 362–376.

[10] Hengdi Wang, Lei Jiao, Konglin Zhu, Lin Zhang, and Jin Dong. 2026. Multi-Tenant Edge AI as a Virtual Power Plant via Online Auction-Driven Energy Scheduling. *IEEE Transactions on Mobile Computing* (2026).

[11] Guoming Tang, Hao Yuan, Deke Guo, Kui Wu, and Yi Wang. 2021. Reusing Backup Batteries as BESS for Power Demand Reshaping in 5G and Beyond. In *IEEE INFOCOM*.

[12] Mulei Ma, Chenyu Gong, Liekang Zeng, and Yang Yang. 2025. Multi-Tier Multi-Node Scheduling of LLM for Collaborative AI Computing. In *IEEE INFOCOM*.

[13] Yuzheng Ren, Haijun Zhang, Fei Richard Yu, Wei Li, Pincan Zhao, and Ying He. 2025. Industrial Internet of Things With Large Language Models (LLMs): An Intelligence-Based Reinforcement Learning Approach. *IEEE Transactions on Mobile Computing* 24, 5 (2025), 4136–4152.

[14] Haiyang Huang, Tianhui Meng, and Weijia Jia. 2025. Joint Optimization of Prompt Security and System Performance in Edge-Cloud LLM Systems. In *IEEE INFOCOM*.

[15] Daliang Xu, Wangsong Yin, Hao Zhang, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2025. EdgeLLM: Fast On-Device LLM Inference With Speculative Decoding. *IEEE Transactions on Mobile Computing* 24, 4 (2025), 3256–3273.

[16] Shengyuan Ye, Bei Ouyang, Liekang Zeng, Tianyi Qian, Xiaowen Chu, Jian Tang, and Xu Chen. 2025. Jupiter: Fast and Resource-Efficient Collaborative Inference of Generative LLMs on Edge Devices. In *IEEE INFOCOM*.

[17] Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. 2024. SpecExec: massively parallel speculative decoding for interactive LLM inference on consumer devices. In *NeurIPS*.

[18] Xiumei Deng, Jun Li, Long Shi, Zhiqiang Wei, Xiaobo Zhou, and Jinhong Yuan. 2022. Wireless Powered Mobile Edge Computing: Dynamic Resource Allocation and Throughput Maximization. *IEEE Transactions on Mobile Computing* 21, 6 (2022), 2271–2288.

[19] Zeyu Meng, Hongli Xu, Min Chen, Yang Xu, Yangming Zhao, and Chunming Qiao. 2021. Learning-Driven Decentralized Machine Learning in Resource-Constrained Wireless Edge Computing. In *IEEE INFOCOM*.

[20] Qing Li, Shangguang Wang, Xiao Ma, Ao Zhou, Yue Wang, Gang Huang, and Xuanzhe Liu. 2024. Battery-Aware Energy Optimization for Satellite Edge Computing. *IEEE Transactions on Services Computing* 17, 2 (2024), 437–451.

[21] Ilai Bistritz, Zhengyuan Zhou, Xi Chen, Nicholas Bambos, and Jose Blanchet. 2019. Online exp3 learning in adversarial bandits with delayed feedback. In *NeurIPS*.

[22] Taishi Uchiya, Atsuyoshi Nakamura, and Mineichi Kudo. 2010. Algorithms for adversarial bandit problems with multiple plays. In *ALT*.

[23] Eric C Hall and Rebecca M Willett. 2015. Online convex optimization in dynamic environments. *IEEE Journal of Selected Topics in Signal Processing* 9, 4 (2015), 647–662.

[24] Tianyi Chen, Qing Ling, and Georgios B Giannakis. 2017. An online convex optimization approach to proactive network resource allocation. *IEEE Transactions on Signal Processing* 65, 24 (2017), 6350–6364.

[25] Mauro Cettolo et al. 2017. Overview of the IWSLT 2017 Evaluation Campaign. In *IWSLT*.

[26] Ondřej Bojar et al. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. In *ACL SMT Workshop*.

[27] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

[28] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv preprint* arXiv:1606.05250 (2016).

[29] Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. TruthfulQA: Measuring How Models Mimic Human Falsehoods. *arXiv preprint* arXiv:2109.07958 (2021).

[30] [n. d.]. Google cluster dataset. https://github.com/google/cluster-data

[31] [n. d.]. Enengy dataset. https://aemo.com.au/initiatives/major-programs/nem-distributed-energy-resources-der-program/

[32] [n. d.]. Edge infrastructure dataset. https://aemo.com.au/energy-systems/energy/national-energy-market-nem/data-nem/data-dashboard-nem.

[33] [n. d.]. Data traffic of 5G communication. https://standards.globalspec.com/standards/detail?docId=14565764

[34] Wen Chen, Jing Qiu, Junhua Zhao, Qingmian Chai, and Zhao Yang Dong. 2021. Bargaining Game-Based Profit Allocation of Virtual Power Plant in Frequency Regulation Market Considering Battery Cycle Life. *IEEE Transactions on Smart Grid* 12, 4 (2021), 2913–2928.

[35] Shinji Mizuno and Florian Jarre. 1999. Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation. *Mathematical Programming* 84, 1 (1999), 105–122.

[36] [n. d.]. Observability for NVIDIA NIM for LLMs. https://docs.nvidia.com/nim/large-language-models/1.12.0/observability.html

[37] [n. d.]. LLM Energy. [Online]. Available: https://www.globenewswire.com/news-release/2026/01/20/3221929/0/en/ESG-AI-Launches-Public-Calculator-Measuring-the-Environmental-Impact-of-Major-AI-Models.html.

[38] [n. d.]. Configuration of cloud server. https://technical.city/zh/video/A800-PCIe-80-GB

[39] [n. d.]. Configuration of edge server. https://www.hardware-corner.net/rtx-4090-llm-benchmarks/

[40] Zhizhen Li, Xuanhao Luo, Mingzhe Chen, Chenhan Xu, Shiwen Mao, and Yuchen Liu. 2025. Contextual Combinatorial Beam Management via Online Probing for Multiple Access mmWave Wireless Networks. *IEEE Journal on Selected Areas in Communications* 43, 3 (2025), 959–972.

[41] Jia Liu, Jianbo Shao, Min Sheng, Yang Xu, Tarik Taleb, and Norio Shiratori. 2024. Mobile Crowdsensing Ecosystem With Combinatorial Multi-Armed Bandit-Based Dynamic Truth Discovery. *IEEE Transactions on Mobile Computing* 23, 12 (2024), 13095–13113.

[42] Juncheng Wang, Yituo Liu, Ben Liang, and Min Dong. 2025. Constrained Over-the-Air Model Updating for Wireless Online Federated Learning with Delayed

Information. In *IEEE INFOCOM*.
[43] Rui Li, Tao Ouyang, Liekang Zeng, Guocheng Liao, Zhi Zhou, and Xu Chen. 2024. Online Optimization of DNN Inference Network Utility in Collaborative Edge Computing. *IEEE/ACM Transactions on Networking* 32, 5 (2024), 4414–4426.
[44] [n. d.]. Gurobi. http://www.gurobi.com
[45] Xuanyu Cao, Junshan Zhang, and H. Vincent Poor. 2018. A Virtual-Queue-Based Algorithm for Constrained Online Convex Optimization With Applications to Data Center Resource Allocation. *IEEE Journal of Selected Topics in Signal Processing* 12, 4 (2018), 703–716.

Appendix

A. Proof of Theorem 4.1

Let $\varpi_{nt} = \sum_{j\in\mathcal{F}_{nt}} v_{nt}^j$ and $q_{nt}^j = v_{nt}^j/\varpi_{nt}$, then we have

$$\frac{\varpi_{n,t+1}}{\varpi_{nt}} \le 1 - \frac{\varsigma}{\gamma_{max} I}\sum_{j\in\mathcal{F}_{nt}} q_{nt}^j \ell_{nt}^j + \frac{\gamma^2}{2\gamma_{max}^2 I^2}\sum_{j\in\mathcal{F}_{nt}} q_{nt}^j {\ell_{nt}^j}^2,$$

summing over T and taking ln, then it leads to

$$\ln \varpi_{n,T+1} \le \ln \gamma_{max} + \sum_{t=1}^{T}\sum_{j\in\mathcal{F}_{nt}}\left(\frac{\varsigma^2}{2\gamma_{max}^2 I^2} q_{nt}^j {\ell_{nt}^j}^2 - \frac{\varsigma}{\gamma_{max} I} q_{nt}^j \ell_{nt}^j\right).$$

As $v_{j^*,T+1} = \exp\left(-\frac{\varsigma}{\gamma_{max} I}\sum_{t=1}^{T}\ell_{nt}^{j*}\right) \le \varpi_{n,T+1}$, where j^* is the optimum, we can get $-\frac{\varsigma}{\gamma_{max} I}\sum_{t=1}^{T}\ell_{nt}^{j*} \le \ln \varpi_{n,T+1}$. Then we can obtain

$$\sum_{t=1}^{T}\sum_{j\in\mathcal{F}_{nt}} q_{nt}^j \ell_{nt}^j - \sum_{t=1}^{T}\ell_{nt}^{j^*} \le \frac{\gamma_{max} I \ln \gamma_{max}}{\varsigma} + \frac{\varsigma}{2\gamma_{max} I}\sum_{t=1}^{T}\sum_{j\in\mathcal{F}_{nt}} q_{nt}^j {\ell_{nt}^j}^2.$$

Using $\mathbb{E}[{\ell_{nt}^j}^2] \le I^2/p_{nt}^j$ and $p_{nt}^j \ge (1-\varsigma)q_{nt}^j$, we acquire

$$\mathbb{E}[\sum_{t=1}^{T}\sum_{j\in\mathcal{F}_{nt}} q_{nt}^j {\ell_{nt}^j}^2] - \sum_{t=1}^{T}\ell_{nt}^{j^*} \le \frac{\gamma_{max} I \ln \gamma_{max}}{\varsigma} + \frac{\varsigma T I}{2(1-\varsigma)}.$$

Since $p_{nt}^j = (1-\varsigma)q_{nt}^j + \varsigma/\gamma_{max}$, we can deduce that $\sum_{j\in\mathcal{F}_{nt}} q_{nt}^j \ell_{nt}^j = \frac{\mathbb{E}[\ell_{nt}^{\bar{j}}] - \frac{\varsigma}{\gamma_{max}}\sum_{j\in\mathcal{F}_{nt}}\ell_{nt}^j}{1-\varsigma}$. Due to $\bar{\gamma}_{nt} \leftarrow \bar{j}$, $\gamma_n^* \leftarrow j^*$, and using the inequality $\mathbb{E}[\sum_{t=1}^{T}\frac{1}{\gamma_{max}}\sum_{j\in\mathcal{F}_{nt}}\ell_{nt}^j] \le TI$, we get the following:

$$\mathbb{E}[\mathcal{P}_1(\bar{\gamma}_{nt})] - \mathcal{P}_1(\gamma_n^*)) \le \gamma_{max} I \ln \gamma_{max}/\varsigma + 3\varsigma T I/2,$$

considering $\varsigma = \sqrt{2\gamma_{max}\ln\gamma_{\max}/3T}$ and $n \in \mathcal{N}$, we get $\mathcal{R}_1$.

B. Proof of Theorem 4.2

To facilitate our proof, we first make several technical *assumptions* which are generally easy to be satisfied for all $t \le T$: (i) There exists some $\phi > 0$ such that each g_t is Lipschitz continuous with modulus ϕ, i.e., $\|g_t(\omega_1) - g_t(\omega_2)\|_2 \le \phi\|\omega_1 - \omega_2\|_2$, for any $\omega_1, \omega_2 \in \mathbb{W}$. (ii) There exists constant $\delta_1 > 0$ such that $\|\omega\|_2 \le \delta_1$ for any $\omega \in \mathbb{W}$. (iii) There exists constant $\delta_2 > 0$ such that $|f_t(\omega)| \le \delta_2$ for any $\omega \in \mathbb{W}$. (iv) There exists constant $\delta_3 > 0$ such that $\|g_t(\omega)\|_\infty \le \delta_3$ for each $\omega \in \mathbb{W}$. (v) The algorithmic parameter μ and Lipschitz modulus ϕ satisfy $\mu \ge \phi^2$. Then we define some *notations* to quantify the violation from different perspectives for $\forall t$: (i) $v_\omega(t) \triangleq \sum_{\tau=0}^{t-1}\|\omega_{\tau+1}^* - \omega_\tau^*\|_2$. (ii) $v_f(t) \triangleq \sum_{\tau=0}^{t-1}\|f_\tau - f_{\tau+1}\|_\infty$. (iii) $v_g(t) \triangleq \sum_{\tau=1}^{t-1}\|g_{\tau-1} - g_\tau\|_\infty^2$. (iv) $\tilde{v}_g(t) \triangleq \sum_{\tau=1}^{t-1}\|g_{\tau-1} - g_\tau\|_\infty$. (v) $v_\lambda(t) \triangleq \sum_{\tau=0}^{t-2}\|\lambda_\tau^* - \lambda_{\tau+1}^*\|_2$. Here, the optimal primal and dual variable are denoted as ω_t^* and λ_t^*, respectively.

According to equation (7), we have $\lambda_{n,t+1} \ge 0$. Then, for $\forall t \le T$, $\lambda_{n,t+1} + g_{nt}(\omega_{t+1}) \ge 0$. If $g_{nt}(\omega_{t+1}) \ge 0$, we will gain

$$\lambda_{n,t+1} \ge \lambda_{nt} + g_{nt}(\omega_{t+1}) \ge g_{nt}(\omega_{t+1}) = |g_{nt}(\omega_{t+1})|,$$

If $g_{nt}(\omega_{t+1}) < 0$, then we can acquire

$$\lambda_{n,t+1} \ge -g_{nt}(\omega_{t+1}) = |g_{nt}(\omega_{t+1})|,$$

which leads to $\lambda_{n,t+1}^2 \ge g_{nt}^2(\omega_{t+1})$. Summing this inequation over all n yields $\|\lambda_{t+1}\|_2 \ge \|g_t(\omega_{t+1})\|_2$. Then, in view of the work [45], for $\forall t$, we are able to get the following inequality:

$$\sum_{\tau=0}^{t-1} f_\tau(\omega_\tau) \le \sum_{\tau=0}^{t-1} f_\tau(\omega_\tau^*) + \frac{1}{2}\|g_{t-1}(\omega_t)\|_2^2 - \frac{1}{2}\|\lambda_t\|_2^2 + v_f(t)) + v_g(t + 4\mu\delta_1 v_\omega(t) + \|g_0(\omega_0)\|_2^2 + \delta_2 + f_0(\omega_0) + \mu\|\omega_0 - \omega_0^*\|_2^2.$$

Substituting $\|\lambda_t\|_2 \ge \|g_{t-1}(\omega_t)\|_2$ into this inequality gives

$$\mathcal{R}_2 \le v_f(T) + v_g(T) + 4\mu\delta_1 v_\omega(T) + \delta_2 + \|g_0(\omega_0)\|_2^2 + f_0(\omega_0) + \mu\|\omega_0 - \omega_0^*\|_2^2.$$

Due to v_f, v_g, v_ω are sub-linear in $t \le T$, then $\mathcal{R}_2 \le o(T)$.

C. Proof of Theorem 4.3

Based on equation (7), we have $\lambda_{nt} \ge \sum_{\tau=0}^{t-1} g_{n\tau}(\omega_{\tau+1})$, $\forall t \le T$. Therefore, we can obtain the following inequation, $\forall t \le T$:

$$\sum_{\tau=0}^{t-1} g_{n\tau}(\omega_\tau) \le \sum_{\tau=0}^{t-2}|g_{\tau+1,k}(\omega_{\tau+1}) - g_{n\tau}(\omega_{\tau+1})| + g_{n0}(\omega_0) + \sum_{\tau=0}^{t-2} g_{n\tau}(\omega_{\tau+1}) \le g_{n0}(\omega_0) + \lambda_{n,t-1} + \tilde{v}_g(t)$$

As $\lambda_{n,t-1} \le \|\lambda_{t-1}\|_2$, then, for $\forall t \le T$, we can gain

$$\sum_{\tau=0}^{t-1} g_{n\tau}(\omega_\tau) \le g_{n0}(\omega_0) + \tilde{v}_g(t) + \|\lambda_{t-1}\|_2.$$

Based on the previous work [45], for $\forall t \le T$, we will have

$$\|\lambda_t\|_2 \le 4v_\lambda(t) + 2\sqrt{v_f(t)} + \sqrt{2v_g(t)} + \sqrt{8\mu\delta_1 v_x(t)} + \sqrt{{\delta_3}^2 + 4\delta_2 + 2\mu\|\omega_0 - \omega_0^*\|_2^2 + 2\|g_0(\omega_0)\|_2^2} + 2\|\lambda_0^*\|_2.$$

Making use of this inequation and noting that the R.H.S. for it is monotonically increasing with $t \le T$, we get the bound:

$$\mathcal{V} \le \tilde{v}_g(T) + 4v_\lambda(T) + 2\sqrt{v_f(T)} + \sqrt{2v_g(T)} + \sqrt{{\delta_3}^2 + 4\delta_2 + 2\mu\|\omega_0 - \omega_0^*\|_2^2 + 2\|g_0(\omega_0)\|_2^2} + 2\|\lambda_0^*\|_2 + g_{n0}(\omega_0) + \sqrt{8\mu\delta_1 v_\omega(T)}$$

Considering that $\tilde{v}_g$ and v_λ are sub-linear in $t \le T$ while v_f, v_g and v_ω are subquadratic in $t \le T$, we have $\mathcal{V} \le o(T)$.

D. Proof of Theorem 4.4

We use $\bar{\gamma}_{nt}, \bar{\omega}_t$ to denote the online decisions obtained from our algorithms, and $\hat{\gamma}_n, \hat{\omega}$ to denote the static offline optimal decisions for the original problem $\mathcal{P}$ over the whole time horizon. We have

$$\mathcal{R} = \mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \bar{\omega}_t)] - \mathcal{P}(\hat{\gamma}_n, \hat{\omega}) = \mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \bar{\omega}_t)] - \mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \hat{\omega})] + \mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \hat{\omega})] - \mathcal{P}(\gamma_n^*, \hat{\omega}) + \mathcal{P}(\gamma_n^*, \hat{\omega}) - \mathcal{P}(\hat{\gamma}_n, \hat{\omega}).$$

As the expectation is only for $\bar{\gamma}_{nt}$, the first two terms are bounded by Theorem 4.2, where ω_t^* is the optimum at t given $\bar{\gamma}_{nt}$:

$$\mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \bar{\omega}_t)] - \mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \hat{\omega})] \le \mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \bar{\omega}_t)] - \mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \omega_t^*)] \le o(T).$$

The next two terms are bounded by Theorem 4.1:

$$\mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \hat{\omega})] - \mathcal{P}(\gamma_n^*, \hat{\omega}) \le o(T).$$

The last two terms can also be bounded:

$$\mathcal{P}(\gamma_n^*, \hat{\omega}) - \mathcal{P}(\hat{\gamma}_n, \hat{\omega}) = \mathcal{P}_1(\gamma_n^*) - \mathcal{P}_1(\hat{\gamma}_n) \le 0,$$

because γ_n^* statically minimizes $\mathcal{P}_1$. That is, overall, we obtain

$$\mathcal{R} = \mathbb{E}[\mathcal{P}(\bar{\gamma}_{nt}, \bar{\omega}_t)] - \mathcal{P}(\hat{\gamma}_n, \hat{\omega}) \le o(T) + o(T) + 0 = o(T).$$