LLM Cache Bandits with Query Heterogeneity for Cost-Efficient Inference

Anonymous ACL submission

Abstract

This paper revisits the LLM cache bandit problem, with a special focus on addressing the query heterogeneity for cost-effective LLM inference. Previous works often assume uniform query sizes. Heterogeneous query sizes introduce a combinatorial structure for cache selection, making the cache replacement process more computationally and statistically challenging. We treat optimal cache selection as a knapsack problem and employ an accumulation-based strategy to effectively balance computational overhead and cache updates. In theoretical analysis, we prove that the regret of our algorithm achieves an $O(\sqrt{MNT})$ bound, improving the coefficient of $\sqrt{MN}$ compared to the $O(MN\sqrt{T})$ result in Zhu et al. (2024), where N is the total number of queries and M is the cache size. Additionally, we also provide a problem-dependent bound, which was absent in previous works. The experiment rely on real-world data show that our algorithm reduces the total cost by approximately 12%.

1 Introduction

Prominent Large Language Models (LLMs), such as GPT (Radford and Narasimhan, 2018), Gemini (Team et al., 2024), and DeepSeek (DeepSeek-AI et al., 2025), have demonstrated exceptional capabilities, driving innovations in AI-powered systems. However, as LLMs significantly enhance the capabilities of AI systems, the increasing volume of query processing requests presents challenges for cost-effective inference. In real-world LLM marketplaces, repeated queries are inevitable, leading to higher inference costs and increased latency (Chen et al., 2023; Li et al., 2024). As model scales continue to grow, the impact of such repeated computation becomes more pronounced, resulting in substantial resource waste (Bang, 2023). The study of Vijayakumar et al. (2018) point out that repetition implies computation is redundantly applied to similar outputs. This lack of diversity results in repeated execution of the same computations, leading to significant waste of computational resources. Such resource waste has already hindered the practical deployment of LLMs. Laskar et al. (2024) reports that in meeting summarization tasks, the same transcripts are often repeatedly submitted to LLMs, with each query triggering a new API call—even when the input remains unchanged. These redundant calls significantly increase inference costs and hinder the practical deployment of LLMs. The work of Stogiannidis et al. (2023) also states that user inputs are often highly similar over time, resulting in LLMs frequently processing instances that are very similar to those previously handled (and already paid for). This increases the operational cost for enterprises when relying on external LLM calls. Developing methods to reduce these redundant costs and resource waste would be highly beneficial for the efficient deployment of large models.

Cache techniques are an effective way to reduce repeated queries in LLMs, minimizing the impact of redundant inquiries. Recently, caching strategies tailored for LLMs have emerged. H2o (Zhang et al., 2023) optimize internal data flow in LLMs to reduce delays and resource consumption during inference. GPTcache (Bang, 2023) is known for its simplicity and flexibility, SCALM (Li et al., 2024) improves cache efficiency by leveraging the semantic understanding of queries. Zhu et al. (2024) considered a bandit setting for online cache updates based on estimated values. These works have successfully integrated caching strategies into the deployment of large models, but most of them do not consider the settings with variable size query.

Incorporating cache into LLMs poses challenges due to the variability and unpredictability of query costs. To address this issue, Zhu et al. (2024) suggests using both expected cost and popularity as

Table 1: Summary of the main results

Previously work	Heterogeneous query size	Problem-dependent regret	Problem-independent regret
(Zhu et al., 2024)	×	×	$\widetilde{O}\left(MN\sqrt{T}\right)$
Our Work	**Heterogeneous query size**	**Problem-dependent regret**	**Problem-independent regret**
VSOCB (Algorithm 1)	√	$\widetilde{O}\left(\sum_{q\in\mathcal{M}^*}\frac{P(q)MN}{\Delta_q}\right)$ (Theorem 4.1)	$\widetilde{O}\left(\sqrt{MNT}\right)$ (Theorem 4.1)
VSOCB-APX(Appendix A)	√	$\widetilde{O}\left(\sum_{q\in\mathcal{Q}}\frac{P(q)N(N-l_{min})}{\Delta_q^c}\right)$	$\widetilde{O}\left(\sqrt{N(N-l_{min})T}\right)$

$\mathcal{Q}$ is the set of all queries, $\mathcal{M}^*$ is the optimal cache, N is the number of queries, M is the size of cache, $l_{min}$is the minimum number of queries required to fill the cache as much as possible, $P(q)$ is the sampling probability of query q, T is the time horizon, Δ_q, Δ_q^c represent the complementary gap and the approximate gap between the cache containing q and the optimal cache respectively. From the table, (Zhu et al., 2024) lacks an analysis of the problem-dependent form of regret, and when `Oracle` provides an exact solution, the regret bound of our algorithm outperforms that of (Zhu et al., 2024) the coefficient of $\sqrt{MN}$.

criteria for cache selection in their online algorithm. While this approach works well when query sizes are fixed, it lacks solutions for cases where query sizes vary. In such scenarios, the length of the query must also be considered, making the number of queries that can be stored in the cache no longer constant, a point that Zhu et al. (2024) briefly discusses but leaves open for further improvement.

To tackle this open problem, we revisit the framework of Zhu et al. (2024) and adapt it to accommodate heterogeneous query sizes. Several additional challenges must be addressed: (1) Heterogeneous query sizes transform the cache replacement process into a combinatorial optimization problem. To replace a new query, the cache may need to evict multiple existing queries. This multi-query replacement strategy requires careful selection of queries in order to maximize the cache hit rate while adhering to size constraints. (2) Unlike the CMAB model (Chen et al., 2016), where action selection can occur globally across all queries, online cache updates are restricted to queries currently in the cache or the incoming query.

In this paper, we propose an online algorithm **Variable Size Online Cache Bandits** (VSOCB), an extension variant of Zhu et al. (2024). Our main contributions are:

Algorithmic Framework. Our algorithm addresses the combinatorial optimization challenges and is well-suited for scenarios where cache selection is constrained in an online setting. Furthermore, our algorithm effectively manages the trade-off between additional computational overhead and cache updates. The VSOCB algorithm is composed of the following key components:

1) *Cache recommendation via knapsack problem*: We design an `Oracle` that utilizes a knapsack problem solver to guide the cache replacement. This approach extends beyond single-query replacement to support more sophisticated strategies. However, solving the knapsack problem incurs computational overhead, creating a trade-off between computational cost and cache update efficiency. Calling `Oracle` too frequently leads to excessive overhead, while infrequent updates hinder effective caching. Balancing these trade-offs is essentially the key consideration for our algorithm.

2) *Accumulation-based* `Oracle` *invoking policy*: To address this `Oracle` trade-off, we employ an accumulation-based strategy for invoking `Oracle`. Instead of calling `Oracle` for every update, we delay its invocation until the query has accumulated sufficient information about its cost or sampling probability. This controlled invocation reduces computational overhead while maintaining efficiency. We theoretically demonstrate that the number of `Oracle` calls under this policy scales logarithmically with the number of rounds, ensuring computational efficiency.

3) *Recommend and wait strategy*: To adapt to online cache selection, we propose a recommend-and-wait strategy. This approach introduces a recommended cache that temporarily stores the optimal cache configuration identified by the knapsack problem. Instead of immediately applying the recommendation, the strategy strategically "waits" for the relevant query to arrive. Upon `Oracle` invocation, the recommended cache is provided as a guideline for the policy. In the subsequent rounds, the algorithm incrementally updates the current cache based on the recommended cache

Theoretical Analysis. Heterogeneous query sizes and online updates make cache selection a constrained combinatorial optimization problem, which introduces additional challenges in analysis. Our main theoretical contributions are as follows:

1) *Regret bound and* `Oracle` *calls:* We prove that our algorithm achieves $\widetilde{O}(\sqrt{MNT})$ regret bound with $O(N\log T)$ Oracle calls.

2) *Generalization to heterogeneous query sizes:*

The case of equal-length queries, as studied in previous works (Zhu et al., 2024), is shown to be a special case of our approach. In particular, Zhu et al. (2024) provided a regret bound of $\widetilde{O}(MN\sqrt{T})$ and establish a lower bound of $\Omega(\sqrt{T})$. Our algorithm extends this framework to handle heterogeneous query sizes while improving the analysis by reducing the coefficient of $\sqrt{MN}$, and our regret bound matches the lower bound in terms of T.

3) *Problem-dependent regret bound:* We provide a problem-dependent regret bound of $\widetilde{O}\left(\sum_{q\in\mathcal{M}^*}\frac{P(q)MN}{\Delta_q}\right)$, which explicitly captures the dependence of regret on query-specific parameters. To the best of our knowledge, such a result is absent in previous work.

Table 1 summarizes our key findings and compares our method with prior work, highlighting improvements in both generalization and computational efficiency.

2 Related work

Cache for LLMs: In the field of LLM, cache technology can address the issue of repeated inquiries, thereby reducing the cost of the LLM marketplace (Chen et al., 2023). Xu et al. (2023); Zhang et al. (2023) focus on content reuse during the inference phase, aimed to optimize the internal data flow within LLMs. Bang (2023) is recognized in the industry for its simplicity and flexibility, providing various options for integration with different LLMChat services. (Li et al., 2024) leverage semantic understanding of queries to improve cache hit rates through a semantic-oriented caching approach. Beyond these works, Zhu et al. (2024) propose an online learning-based caching algorithm to minimize resource waste. While they briefly mention extending their method to variable-sized queries, they lack rigorous theoretical analysis or concrete solutions. We build upon their approach by generalizing caching to variable-size constraints, filling a critical gap in dynamic query management.

Knapsack problem: The Knapsack Problem (Kellerer et al., 2004) is a fundamental problem in optimization and computer science, frequently used in resource allocation and selection tasks. In our work, the knapsack problem serves as an offline solver for cache selection, guiding efficient decisions on which queries to store or evict. Specifically, we model our problem as a 0-1 Knapsack Problem and employ dynamic programming (Bertsekas, 2005) to update our cache replacement policy. Additionally, the Online Knapsack Problem (Zhou et al., 2008) is a variant where decisions must be made without knowledge of future inputs, making it particularly relevant to online caching scenarios. Another relevant extension is the Minimum Knapsack Problem (MKP) (Csirik et al., 1991; Santini and Malaguti, 2024), which differs from the traditional 0/1 Knapsack by seeking to minimize an objective while satisfying specific constraints.

Multi-armed bandit: The multi-armed bandit problem is a foundational online decision-making framework with applications in recommendation systems (Zhou et al., 2017). In the classic setting, an agent selects among multiple arms, each associated with an unknown reward distribution, with the goal of maximizing cumulative rewards through exploration and exploitation. We model each query as an arm, with its cost following an unknown distribution. An extension of this framework is the Combinatorial Multi-Armed Bandit (CMAB) (Chen et al., 2016; Wang and Chen, 2017), where a super arm consists of multiple base arms played simultaneously. However, the CMAB model does not directly apply to our setting for two reasons. First, feedback is only available for unselected queries. Second, unlike CMAB, where super arms can be freely composed of all base arms, cache selection is constrained by memory and structural limitations.

3 Problem Setting

3.1 Caching with Variable Size Query

We consider a finite set of queries $\mathcal{Q} = \{q_1, ..., q_N\}$ containing N distinct queries, each query $q \in \mathcal{Q}$ is associated with a size $L(q) \in \mathbb{N}_+$. When query q is input to the LLM system, the LLM processes it and returns a corresponding result with size $A(q) \in \mathbb{N}_+$. We use $S(q) = L(q) + A(q)$ to denote the *total query size* of q. Each distinct query has a sampling probability, which represents its frequency of occurrence. In each round, a query is received, which is sampled from $\mathcal{Q}$ according to the fixed sample distribution. We use $P(q) \in (0, 1]$ to denote the sampling probability that query q be selected such that $\sum_{q\in\mathcal{Q}} P(q) = 1$. Each time the LLM processes q, it will incur a random cost $C(q)$, which is supported on $[c_1, c_2]$ with $c_2 > c_1 > 0$. The cost can be expressed as:

$$C(q) = C^*(q) + \epsilon_q.$$

where ϵ_q is a sub-Gaussian noise that captures the uncertainties in the cost, with $\mathbb{E}[\epsilon_q] = 0$. The

expected true cost $C^*(q) = \mathbb{E}[C(q)]$ is the mean cost of processing query q.

Remark 3.1. *Since the cache needs to store both the query for identification and the answer for direct response, we refer to the total query size as query size for simplicity.*

The cache stores both the query and its result content, and outputs the result of a matching query directly with zero cost. We use $r(q)$ to denote the result content return by LLM correspond to query q. We maintain a cache $\mathcal{M}$ for the LLM system, storing a small subset of queries with their corresponding results, i.e. we store a question-answer pair $(q, r(q))$ in cache, and the size of $(q, r(q))$ is $S(q)$. We use $q \in \mathcal{M}$ to denote $(q, r(q))$ has been stored in cache. The size of cache has a maximum size M, the total size of the stored queries must satisfy $\sum_{q\in\mathcal{M}} S(q) \leq M$. Let $\mathfrak{J}$ be the set of all possible sets that satisfy the size constraint

$$\mathfrak{J} = \left\{\mathcal{M} | \sum_{q\in\mathcal{M}} S(q) \leq M\right\}.$$

To reduce overhead, we should place queries with high expected costs and high sampling frequencies into the cache. Following Zhu et al. (2024), we can define the expected cost of a given cache $\mathcal{M}$ is

$$cost(\mathcal{M}) = \sum_{q\in\mathcal{Q}} C^*(q)P(q)\mathbb{I}\{q \notin \mathcal{M}\},$$

where $\mathbb{I}$ is the indicator function. From another perspective, the cost we saved can be regarded as additional reward we have obtained. Then optimal cache can be defined as

$$\mathcal{M}^* = argmax_{\mathcal{M}\in\mathfrak{J}} \sum_{q\in\mathcal{M}} C^*(q)P(q). \quad (1)$$

Previous works such as (Zhu et al., 2024) assume homogeneous query sizes, where $S(q) = 1$ for all queries, reducing the problem to a top-K selection problem. In contrast, we consider a heterogeneous setting, where queries have variable sizes $S(q)$, extending previous work.

`Oracle`. From the formula of optimal cache, we can model the optimal problem as a 0-1 knapsack problem where $C^*(q)P(q)$ represents the value of q and $S(q)$ is its volume. The cache is treated as a knapsack with capacity of M, and the optimal cache is the optimal solution of a knapsack problem. Inspired by the CMAB model, we assume that the agent has access to an offline `Oracle` to guide cache selection decisions. `Oracle` have an offline solver that can solve the knapsack problem.

In some cases, the solution provided by `Oracle` is approximate. In such scenarios, we reformulate the problem as a minimum knapsack problem (MKP). Further details are provided in Appendix.

3.2 Online Learning

Online cache bandit. The agent's objective is to choose the optimal cache without knowing the expected true cost and sample distribution. We consider a total number of $T \in \mathbb{N}_+$ learning rounds. In round t, agent receives a query q_t which is sampled from $\mathcal{Q}$ with probability $P(q)$. The agent has a *current cache* $\mathcal{M}_t \in \mathfrak{J}$ in round t. $\mathcal{M}_t$ is selected based on the history feedback information from previous rounds. After q_t arrive, the agent will check the current cache $\mathcal{M}_t$. If q_t is found in the cache, i.e. $q_t \in \mathcal{M}_t$, we say that query *hits* the cache. In this case, the result of q_t is directly returned without processing by the LLM. The cost of this query is 0 and will save a potential cost $C(q_t)$, which is unobserved to the agent. If query q_t *miss* the cache, i.e. $q_t \notin \mathcal{M}_t$, the system processes the query, incurring a cost $C(q_t)$, returns the result $r(q_t)$, and delivers the feedback to the agent, which also get the query's size $S(q_t)$ at the same time.

The agent can update the cache based on history information. At the same time, the algorithm will decide whether to use `Oracle`. If `Oracle` is called, the cache will be updated according to the output provided by `Oracle`.

Remark 3.2. *As noted in Zhu et al. (2024), our setting differs from traditional bandit problems because feedback may not be observed in every round.*

Remark 3.3. *In the online mode, the agent's updating of the cache in each round are constrained. At each time step, the cache can only be updated using the currently arriving query and the queries already stored in the cache, i.e. $\mathcal{M}_{t+1} \subseteq \mathcal{M}_t \cup \{q_t\}$. This is because that we discard the results $r(q)$ of queries that are removed from the cache during updates. If a previously dropped query reappears, the $r(q)$ need to be recomputed by LLM.*

Learning objective. Our goal is to minimize the total cost with time horizon T. By applying the indicator function to the regret in Zhu et al. (2024) for an equivalent transformation, we get:

$$Reg(T) = \mathbb{E}\left[\sum_{t=1}^{T} C(q_t)\left(\mathbb{I}\{q_t \in \mathcal{M}^*\} - \mathbb{I}\{q_t \in \mathcal{M}_t\}\right)\right],$$

where $\mathcal{M}^* = argmax_{\mathcal{M}\in\mathfrak{J}} \sum_{q\in\mathcal{M}} C^*(q)P(q)$.

4 Variable Size Online Cache Bandit

4.1 Algorithm with Exact `Oracle`

Our algorithm is presented in Alg.1. When we have an exact solver, $Oracle$ refers to Alg.2.

Query record. In the online setting, the agent does not know which queries exist at the beginning. Therefore, the agent needs to maintain a set to record the queries it has encountered. Only after the agent has seen a query will it have information about that query and estimate its cost and sampling probability. Subsequently, when invoking `Oracle` or updating the cache, the agent will take this query into account. We use a set $\mathcal{Q}_t$ (line 2) in our algorithm to represent this record set, a query $q \in \mathcal{Q}_t$ means that the estimation and information of this query (excluding $r(q)$) have been recorded by algorithm. Clearly, when all the queries in $\mathcal{Q}$ have arrived at least once, we have $\mathcal{Q}_t = \mathcal{Q}$.

`Oracle` guide: For heterogeneous query sizes, the previously straightforward single-query replacement strategy is no longer applicable. This is because, with variable-sized queries, cache replacement must also account for whether the incoming query can physically fit into the available space. Replacement may fail if the query's size exceeds the cache's remaining capacity. Inspired by the `Oracle` in the CMAB model (Chen et al., 2016), we assume that the algorithm has access to an `Oracle` that returns an optimal cache selection subject to size constraints. Specifically, `Oracle` is denoted by $Oracle(\mathcal{Q}', C^*, P, S, M)$, where $C^* = \{C^*(q)\}_{q\in\mathcal{Q}'}$, $P = \{P(q)\}_{q\in\mathcal{Q}'}$ and $S = \{S(q)\}_{q\in\mathcal{Q}'}$. It takes the cost, sample probability and query size of all queries in a given set $\mathcal{Q}'$ as input and output a set $\mathcal{M}$ that yields the solution of knapsack problem with volume constraint of cache size M. We assume that `Oracle` has a knapsack problem solver (e.g. dynamic programming) which can output the exact optimal solution. We denote this solver as $Knap(\boldsymbol{w}, \boldsymbol{v}, m)$. By inputting the item values $\boldsymbol{w}$, volumes $\boldsymbol{v}$, and the knapsack capacity m into $Knap$, it will return the optimal knapsack solution. We encapsulate this solver as our `Oracle` in Alg.2. `Oracle` accepts the estimated values of query costs and sampling probabilities, multiplies them together to compute the value, and then inputs this value into $Knap$. `Oracle` can see global information, allowing it to provide global guidance on cache replacement.

Lower confidence bound: Similar to Zhu et al. (2024), we use a lower-confidence-bound (LCB) for estimating the cost of each query. As stated in Remark 3.2, our queries generate feedback only when they miss the cache. This leads to more precise estimates for queries that are not in the cache compared to those that are. LCB introduce a penalty term for queries placed in the cache, encouraging more exploration and helping to mitigate this imbalance. This LCB approach is akin to adding a penalty to the estimation of queries that remain in the cache for a long time, making them more likely to be replaced, similar to an enforcement of exploration. We use $C_t(q)$ to denote the cost of query q that the agent receives in round t. In algorithm, we use $T_t^{(c)}(q) = \sum_{s=1}^{t} \mathbb{I}\{q_s \notin \mathcal{M}_s, q_s = q\}$ to represent the number of cache misses for query q up to the t-th round, where $\mathbb{I}\{q_s \notin \mathcal{M}_s, q_s = q\}$ denote query q arrive in round s and misses cache. Let $\widetilde{C}_t(q) = \sum_{s=1}^{t} \mathbb{I}\{q_s \notin \mathcal{M}_s, q_s = q\} C_t(q)$ to represent the accumulated cost information of query q up to the t-th round. The lower-confidence-bound based estimator for cost is:

$$\widehat{C}_t(q) = \max\left\{0, \frac{\widetilde{C}_t(q)}{T_t^{(c)}(q)} - (c_2 - c_1)\sqrt{\frac{\ln(8TN/\delta)}{2T_t^{(c)}(q)}}\right\}. \tag{2}$$

Variance confidence bound: Different from Zhu et al. (2024), for the estimation of the sampling probability $P(q)$, we also used an LCB-type estimate rather than a empirical estimate. Additionally, the confidence bound for $\widehat{P}_t(q)$ is derived from the variance bound in Liu et al. (2024). According to the work in Liu et al. (2024), the variance estimate $V_t(q)$ of the sampling probability of a query can be obtained in the following form:

$$V_t(q) = \frac{1}{t}\sum_{s=1}^{t}\left(\mathbb{I}\{q_s = q\} - \frac{\sum_{i=1}^{t}\mathbb{I}\{q_i = q\}}{t}\right)^2.$$

Let $T_t^{(q)}(q) = \sum_{s=1}^{t} \mathbb{I}\{q_s = q\}$, the estimator for sample probability in round t is:

$$\widehat{P}_t(q) = \max\left\{0, \frac{T_t^{(q)}(q)}{t} - LVCB_t(q, \delta)\right\}, \tag{3}$$

$$LVCB_t(q, \delta) = \sqrt{\frac{3V_t(q)R(\delta)}{t}} + \frac{5R(\delta)}{t}, \tag{4}$$

where $R(\delta) = \ln(16TN/\delta)$.

Recommend and wait strategy: In online model, VSOCB does not store the result content corresponding to all queries. When the cache is updated, the results content $r(q)$ of the queries that

are removed from the cache will be cleared. Because cache need to store $(q, r(q))$, cache selection can only access the queries currently in the cache and any newly arriving queries. As mentioned in Remark3.3, the queries chosen for cache updates are not arbitrary. Although `Oracle` can output an optimal solution for the global set of queries, unlike the setting in CMAB (Chen et al., 2016), our cache cannot be directly updated to the selection recommended by `Oracle` due to the online constraint. For example, a query in `Oracle`'s output indicates that the agent should add this query to the cache, but if this query was replaced from the cache during a previous update, the agent will not have $r(q)$ for this query in the current round. In this case, the cache must wait until the query arrives again and the $r(q)$ is re-obtained before it can be added to the cache. Under this setting, we involve a *Recommended cache* denoted by $\widehat{\mathcal{M}}_t$ to temporarily store the results output by our `Oracle` at the current time. When `Oracle` is invoked, the policy does not update the cache immediately. Instead, it first receives a recommended cache $\widehat{\mathcal{M}}_t$, and the cache will retain only the portion that matches the recommended cache, and the differing parts will be removed to free up space for subsequent updates (line 23). In subsequent rounds, the algorithm updates the current cache based on recommended cache. In the following rounds, if a query that belongs to the recommended cache but is not currently in the cache arrives, it will miss the current cache and be processed by the LLM to obtain its response. The query will then be added to the current cache. Otherwise, the cache will not be updated unless there is still available space in the recommended cache. This process continues until the next recommended cache update occurs (line 16-20).

Full cache: The recommended cache may remain partially empty until all queries have been observed at least once (i.e., $\mathcal{Q}_t = \mathcal{Q}$). This implies that some cache space might go unused. To prevent such waste, we check whether there is remaining capacity in the recommended cache. If space is available, we place the additional queries into the cache (as indicated by the 'or' condition in line 16).

Accumulation-based `Oracle` invoking policy: The calling of `Oracle` can gradually update the current cache to the optimal solution, but it also introduces additional computational overhead for the entire system. Especially when solving such knapsack problems, the cost of calling `Oracle` is substantial. Therefore, we need to minimize the number of `Oracle` calls as much as possible, while still minimizing the impact on the cache updates. In our algorithm, we use an accumulation-based strategy (line 21) to balance this trade-off. $T_t^{(r)}(q)$ represents the number of cache misses for query q at the time of the most recent `Oracle` call before time t. Accumulation means that an `Oracle` call occurs when the number of cache misses $T_t^{(c)}(q)$ in current for a query exceeds $(1+\alpha)T_t^{(r)}(q)$, where $\alpha > 0$ is a predetermined constant. This is equivalent to calling `Oracle` only when the query has accumulated enough information about its cost. Since our algorithm also needs to estimate the sampling probability of queries, our update strategy maintains a timer T_t, which records the time of the most recent `Oracle` call prior to round t. When the rounds t reaches $(1+\alpha)T_t$, we also perform an `Oracle` call. The role of the timer T_t is to accumulate information about the query's sampling probability. This accumulation-based `Oracle` invoking policy ensures that `Oracle` is called in a timely manner, while also keeping the number of `Oracle` calls logarithmic in scale.

4.2 Regret Bound

We always aim to fill the cache to capacity, i.e., no additional query can be added to the cache. Under this requirement, similar to CUCB (Chen et al., 2016), we define the valid set of caches.

Definition 4.1. *(Valid set): We define the valid set as*

$$\mathfrak{V}=\left\{\mathcal{M}|\mathcal{M}\subseteq\mathcal{Q}, M-\min_{q'\notin\mathcal{M}} S(q')<\sum_{q\in\mathcal{M}} S(q)\leq M\right\}.$$

We define $l_{max} = \max_{\mathcal{M}\in\mathfrak{V}} |\mathcal{M}|$, $l_{min} = \min_{\mathcal{M}\in\mathfrak{V}} |\mathcal{M}|$ *to represent the maximum and minimum number of queries that can be included in the valid set, respectively.*

Definition 4.2. *(Complementary Gap): For each* $\mathcal{M} \in \mathfrak{V}$*, we define the gap:*

$$\Delta_{\mathcal{M}} = \sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\mathcal{M}} P(q)C^*(q).$$

For each query q, we define:

$$\Delta_q = \min_{\mathcal{M}\in\mathfrak{V}:\mathcal{M}\neq\mathcal{M}^*, q\notin\mathcal{M}} \Delta_{\mathcal{M}}.$$

Similar to the definition 2 in Wang and Chen (2017), if there is no cache selection $\mathcal{M}$ *such that* $q \in \mathcal{M}$ *and* $\mathcal{M} \neq \mathcal{M}^*$*, we define* $\Delta_q = +\infty$*. And* $\Delta_{min} = \min_{q\in\mathcal{Q}} \Delta_q = \min_{\mathcal{M}\in\mathfrak{V}} \Delta_{\mathcal{M}}$.

Algorithm 1 Variable Size Online Cache Bandits (VSOCB)

Require: $\mathcal{Q}_0=\phi$. $T_0=0$. $\mathcal{M}_0=\widehat{\mathcal{M}}_0=\phi$, $\alpha>0$. For all $q\in\mathcal{Q}$: $T_0^{(r)}(q)=T_0^{(q)}(q)=T_0^{(c)}(q)=0$, $\widehat{P}_0(q)=V_0(q)=0$, $\widetilde{C}_0(q)=\widehat{C}_0(q)=0$, $S(q)=0$.

Ensure:

for round $t=1,...,T$ **do**
  q_t was received with size $L(q_t)$, $\mathcal{Q}_t=\mathcal{Q}_{t-1}\cup\{q_t\}$
  $T_t^{(q)}(q_t)=T_{t-1}^{(q)}(q_t)+1$
  if $q_t\in\mathcal{M}_{t-1}$ **then**
    Output the response from cache .
    $\widetilde{C}_t(q_t)=\widetilde{C}_{t-1}(q_t), T_t^{(c)}(q_t)=T_{t-1}^{(c)}(q_t)$
  else
    Input q_t to LLM and receive $r(q_t), A(q_t), C_t(q_t)$.
    $S(q_t)=L(q_t)+A(q_t)$ //record query size
    $\widetilde{C}_t(q_t)=\widetilde{C}_{t-1}(q_t)+C_t(q_t), T_t^{(c)}(q_t)=T_{t-1}^{(c)}(q_t)+1$
  end if
  for all $q\neq q_t$ **do**
    $T_t^{(q)}(q)=T_{t-1}^{(q)}(q), T_t^{(r)}(q)=T_{t-1}^{(r)}(q), T_t^{(c)}(q)=T_{t-1}^{(c)}(q), \widetilde{C}_t(q)=\widetilde{C}_{t-1}(q)$
  end for
  compute $\widehat{P}_t(q), \widehat{C}_t(q)$ with Eq. (2), (3), (4), $\forall q\in\mathcal{Q}_t$
  if $q_t\in\widehat{\mathcal{M}}_{t-1}$ or $S(q_t)\leq M-\sum_{q\in\widehat{\mathcal{M}}_{t-1}}S(q)$ **then**
    $\mathcal{M}_t=\mathcal{M}_{t-1}\cup\{q_t\}, \widehat{\mathcal{M}}_t=\widehat{\mathcal{M}}_{t-1}\cup\{q_t\}$
  else
    $\mathcal{M}_t=\mathcal{M}_{t-1}, \widehat{\mathcal{M}}_t=\widehat{\mathcal{M}}_{t-1}$
  end if
  if $T_t^{(c)}(q_t)\geq(1+\alpha)T_{t-1}^{(r)}(q_t)$ or $t\geq(1+\alpha)T_{t-1}$ **then**
    $T_t^{(r)}(q_t)=T_t^{(c)}(q_t), T_t=t$
    $\widehat{\mathcal{M}}_t=Oracle(\mathcal{Q}_t,\widehat{C}_t,\widehat{P}_t,S,M), \mathcal{M}_t=\widehat{\mathcal{M}}_t\cap\mathcal{M}_{t-1}$
  else
    $T_t^{(r)}(q_t)=T_{t-1}^{(r)}(q_t), T_t=T_{t-1}, \widehat{\mathcal{M}}_t=\widehat{\mathcal{M}}_{t-1}$
  end if
end for

Remark 4.1. *We define the gap of a query as the difference between the non-optimal cache that does not include this query and the optimal cache. This gap of query measures the impact when a cache does not select this query.*

Algorithm 2 Exact `Oracle`

Input $\mathcal{Q}'$, $\{C'(q)\}_{q\in\mathcal{Q}'}$, $\{P'(q)\}_{q\in\mathcal{Q}'}$, $\{S'(q)\}_{q\in\mathcal{Q}'}, M$
$w(q)=P'(q)C'(q), \forall q\in\mathcal{Q}'$
$\mathcal{M}\leftarrow Knap(\{w(q)\}_{q\in\mathcal{Q}'},\{S'(q)\}_{q\in\mathcal{Q}'},M)$
Output $\mathcal{M}$

Theorem 4.1. *Setting $\delta=1/T$, (1) if $\Delta_{min}>0$, the problem-dependent regret bound for VSOCB (Alg.1) with exact* `Oracle`*(Alg.2) holds:*

$$Reg(T)\leq\widetilde{O}\left(\sum_{q\in\mathcal{M}^*}\frac{P(q)Nl(M,\mathcal{Q})}{\Delta_q}\right),$$

(2) and problem-independent regret bound:

$$Reg(T)\leq\widetilde{O}\left(\sqrt{Nl(M,\mathcal{Q})T}\right),$$

where $l(M,\mathcal{Q})=\min\{l_{max},N-l_{min}\}$. (3) and the total number of `Oracle` *calls is $O(N\log T)$.*

This proof is deferred to the Appendix. In the proof, we adopt an analysis method similar to Wang and Chen (2017) and combine it with the `Oracle`-invoking policy proposed in our algorithm. We derive separate confidence intervals for $C^*(q)$ and $P(q)$; this decoupling yields tighter bounds. We show that the accumulation-based `Oracle`-invoking policy governs the duration for which the cache remains unchanged. Our conclusion suggests that both very small and very large caches result in relatively low regret. This is intuitive because, if the cache is very small and can only hold one query, we have $l(M,\mathcal{Q})=1$, so the optimal cache can be easily identified. If the cache is large enough to hold all queries, then $l(M,\mathcal{Q})=0$, and the cache can store all queries, which eliminates regret altogether.

We demonstrate that our Online Variable Size Cache Bandits achieves the regret of $\widetilde{O}(\sqrt{MNT})$ (becasue $l(M,\mathcal{Q})\leq M$), which represents an improvement over Caching in Online Learning in Zhu et al. (2024) with the coefficient of $\sqrt{MN}$, where the conclusion of Zhu et al. (2024) was $\widetilde{O}(MN\sqrt{T})$. On the other hand, our model is an extension of (Zhu et al., 2024), which can be regarded as a special case of our model. In this case, (Zhu et al., 2024) proposed a lower bound of $\Omega(\sqrt{T})$. The problem-independent regret established in Theorem 4.1 conforms to $\widetilde{O}(\sqrt{T})$, indicating that our work provides an optimal solution in relation to T.

Although our algorithm requires an additional `Oracle` to assist in computation, our proof shows that the number of calling `Oracle` is $O(N \log T)$, which is logarithmic. This indicates that the additional computational cost caused by using `Oracle` is not significant, and it well balances the trade-off between computational cost and regret.

Since an exact `Oracle` may be impractical in real-world applications, approximate solutions are commonly employed. Accordingly, we present an approximate version of our method, VSOCB-APX, and establish the corresponding theoretical guarantees. Specifically, when the solver only provides approximate solutions, we transform the problem into a minimum knapsack problem (MKP). The detailed algorithm design and theoretical analysis are deferred to Appendix.

5 Experiment

5.1 Simulation Dataset

We compare performance of our algorithms and the baseline using cumulative regret with a simulated dataset. For the baseline, (Zhu et al., 2024) suggest in the appendix combining the classic GDSF (Jin and Bestavros, 2000; Arlitt et al., 2000) algorithm with their COL algorithm to handle heterogeneous query sizes. As we have discussed its limitations above, we select it as our baseline, referred to as COL-GDSF. We consider 100 distinct queries and set the cache size to be 60. The expected cost of each query is within the range of $[1, 2]$. The cost for each query in each round is sampled from a truncated normal distribution with a mean equal to the expected value generated above. We set the frequency distribution as a power-law distribution and its parameter is 0.5. We employ a dynamic programming algorithm as our solver in `Oracle` and set $\alpha = 1$ in VSOCB. We repeat the simulation 10 times and plot the mean in the figure.

As shown in Fig.1, our algorithm outperforms the baseline. This improvement can be attributed to the fact that COL-GDSF provides an approximate solution to the knapsack problem, which poses a risk of falling into sub-optimal solutions. In contrast, our algorithm circumvents this issue by utilizing `Oracle` for the knapsack problem and avoiding significant additional costs to some extent.

5.2 Real Dataset

In line with Zhu et al. (2024), we evaluate our algorithms using the OpenAssistant (Köpf et al., 2023) dataset for the chat assistant task. For this task, we employ the FastChat-T5-3B (Zheng et al., 2023) model to implement our online algorithm, utilizing inference latency as the cost metric. We run algorithms with 100 distinct queries in the online setting over a total of 10K and 20K rounds, cache size is 100. The other parameters are the same as above. As shown in Tab.2, our method reduced the cost by 12% compared to the baseline, with the result averaged over 10 repetitions. After a sufficient number of online learning steps, `Oracle` accurately gets the costs and frequencies of each query, then VSOCB outperform the baseline.

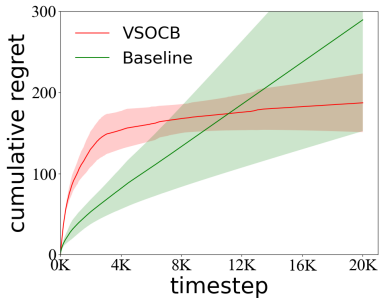

Figure 1: Synthetic dataset with 20000 rounds

Algorithm	Cost (10K)	Cost (20K)
COL-GDSF	2073.1 ± 40.9	4134.3 ± 75.3
VSOCB	1813.5 ± 19.7	3615.7 ± 32.3

Table 2: Average cumulative cost on real dataset

6 Conclusion

In this paper, we study heterogeneous query size caching in LLM systems. To address this, we propose the VSOCB algorithm with rigorous theoretical analysis. We show that our algorithm achieves a tighter regret bound than existing approaches and, notably, we establish a problem-dependent regret bound not provided in prior literature.

7 Limitations of the study

We acknowledge that there is still room for improvement. Our lower bound is directly adapted from (Zhu et al., 2024), which only analyzes regret in terms of T and does not account for other important parameters such as M and N. Additionally, due to technical complexity, the approximate `Oracle` relies on a minimum knapsack problem solver, leading to a looser regret bound than the exact version and posing certain limitations for practical deployment.

References

Martin Arlitt, Ludmila Cherkasova, John Dilley, Rich Friedrich, and Tai Jin. 2000. Evaluating content management techniques for web proxy caches. 27(4):3–11.

Fu Bang. 2023. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 212–218, Singapore. Association for Computational Linguistics.

Dimitri P. Bertsekas. 2005. Dynamic programming and optimal control, 3rd edition.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *Preprint*, arXiv:2305.05176.

Wei Chen, Yajun Wang, Yang Yuan, and Qinshi Wang. 2016. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *Journal of Machine Learning Research*, 17(50):1–33.

J. Csirik, J. B. G. Frenk, M. Labbé, and S. Zhang. 1991. Heuristics for the 0–1 min-knapsack problem. *Acta Cybern.*, 10(1–2):15–20.

DeepSeek-AI, Daya Guo, and et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Shudong Jin and A. Bestavros. 2000. Popularity-aware greedy dual-size web proxy caching algorithms. In *Proceedings 20th IEEE International Conference on Distributed Computing Systems*, pages 254–261.

Hans Kellerer, Ulrich Pferschy, and David Pisinger. 2004. *Knapsack Problems*, 1th edition. Springer Berlin, Heidelberg.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations – democratizing large language model alignment. *Preprint*, arXiv:2304.07327.

Md Tahmid Rahman Laskar, Elena Khasanova, Xue-Yong Fu, Cheng Chen, and Shashi Bhushan Tn. 2024. Query-OPT: Optimizing inference of large language models via multi-query instructions in meeting summarization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1140–1151, Miami, Florida, US. Association for Computational Linguistics.

Jiaxing Li, Chi Xu, Feng Wang, Isaac M von Riedemann, Cong Zhang, and Jiangchuan Liu. 2024. Scalm: Towards semantic caching for automated chat services with large language models. *Preprint*, arXiv:2406.00025.

Xutong Liu, Jinhang Zuo, Siwei Wang, Carlee Joe-Wong, John C.S. Lui, and Wei Chen. 2024. Batch-size independent regret bounds for combinatorial semi-bandits with probabilistically triggered arms or independent arms. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alberto Santini and Enrico Malaguti. 2024. The min-knapsack problem with compactness constraints and applications in statistics. *European Journal of Operational Research*, 312(1):385–397.

Ilias Stogiannidis, Stavros Vassos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2023. Cache me if you can: an online cost-aware teacher-student framework to reduce the calls to large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14999–15008, Singapore. Association for Computational Linguistics.

Gemini Team, Petko Georgiev, and et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Rs, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32.

Qinshi Wang and Wei Chen. 2017. Improving regret bounds for combinatorial semi-bandits with probabilistically triggered arms and its applications. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1161–1171, Red Hook, NY, USA. Curran Associates Inc.

Minrui Xu, Dusit Niyato, Hongliang Zhang, Jiawen Kang, Zehui Xiong, Shiwen Mao, and Zhu Han. 2023. Sparks of gpts in edge intelligence for metaverse: Caching and inference for mobile aigc services. *Preprint*, arXiv:2304.08782.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. *Preprint*, arXiv:2306.14048.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang,

Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Qian Zhou, XiaoFang Zhang, Jin Xu, and Bin Liang. 2017. Large-scale bandit approaches for recommender systems. In *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part I*, page 811–821, Berlin, Heidelberg. Springer-Verlag.

Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. 2008. Budget constrained bidding in keyword auctions and online knapsack problems. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, page 1243–1244, New York, NY, USA. Association for Computing Machinery.

Banghua Zhu, Ying Sheng, Lianmin Zheng, Clark Barrett, Michael I. Jordan, and Jiantao Jiao. 2024. On optimal caching and model multiplexing for large model inference. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

A VSOCB with Approximate `Oracle`

In this section, we examine the use of approximation algorithms for solving the problem. We present a feasible transformation that allows our algorithm to handle approximate solutions, and compare the analytical differences between using approximate and exact solutions.

A.1 Approximate `Oracle`

When the solver can only output an approximate solution, we require `Oracle` to output the complement of the cache. In this case, the optimization problem for solver transforms into min-knapsack problem (MKP) (Csirik et al., 1991):

$$\min_{\mathcal{M}} \sum_{q\in\mathcal{M}^c} P(q)C^*(q),$$
$$s.t. \sum_{q\in\mathcal{M}^c} S(q) \geq \sum_{q\in\mathcal{Q}} S(q) - M.$$

where $q \in \mathcal{M}^c$ means that $q \notin \mathcal{M}$. We use $knap_{min}$ to denote MKP solver. Clearly, the optimal solution set for this MKP is the complement of the optimal solution set for the original knapsack problem. We use $\mathcal{M}^{*c}{=}\mathcal{Q}\setminus\mathcal{M}^*$ to denote the complementary of $\mathcal{M}^*$.

Definition A.1. *(β-approximation solver): A minimum knapsack problem solver is β-approximation solver mean that: the output $\mathcal{M}^c$ of solver under input set $\mathcal{Q}'$ and input value $\{P'(q)C'(q)\}_{q\in\mathcal{Q}'}$ satisfy $\sum_{q\in\mathcal{M}^c} P'(q)C'(q) < (1+\beta)\sum_{q\in\mathcal{M}^{*c}} P'(q)C'(q)$*

Since the MKP solver outputs the complement of the knapsack problem solution, we take the complement of the solver's output to obtain the recommended cache, Algorithm 3 is our approximate `Oracle`. Replacing the $Oracle$ in Alg.1 with Alg.3 is our approximate version.

Remarking that we define the approximate algorithm on the MKP instead of directly defining an approximate algorithm on the original knapsack problem. This mainly due to our theoretical analysis, when `Oracle` outputs an approximate solution, it cannot directly compare the optimal cache with the recommended cache in the same way as exact version. We relax the regret analysis to focus on the complement, using this relaxation to address this challenge.

A.2 Approximate Solution Bound

Since the output in the case of an approximate solver is the complement of the cache, we need to

Algorithm 3 Approximate `Oracle`

1: Input $\mathcal{Q}'$, $\{C'(q)\}_{q\in\mathcal{Q}'}$, $\{P'(q)\}_{q\in\mathcal{Q}'}$, $\{S'(q)\}_{q\in\mathcal{Q}'}$, M
2: $w(q) = P'(q)C'(q), \forall q \in \mathcal{Q}'$
3: $\mathcal{M}^c \leftarrow Knap_{min}(\{w(q)\}_{q\in\mathcal{Q}'}, \{S'(q)\}_{q\in\mathcal{Q}'}, M)$
4: Output $\mathcal{Q}' \setminus \mathcal{M}^c$

redefine some of the previous definitions. Following by CMAB (Chen et al., 2016),we redefine the regret and gap.

Definition A.2. *(β-approximation regret): For a β-approximation solver, the regret is:*

$$Reg_\beta(T) = \sum_{t=1}^{T} \mathbb{E}\left[\sum_{q\in\mathcal{M}_t^c} P(q)C^*(q) - (1+\beta)\sum_{q\in\mathcal{M}^{*c}} P(q)C^*(q)\right].$$

Definition A.3. *(Approximation gap): For each $\mathcal{M} \in \mathfrak{V}$ and $\mathcal{M}^c = \mathcal{Q} \setminus \mathcal{M}$, we define the gap $\Delta_{\mathcal{M}}^c = \sum_{q\in\mathcal{M}_t^c} P(q)C^*(q) - (1+\beta)\sum_{q\in\mathcal{M}^{*c}} P(q)C^*(q)$. For each query q, we define $\Delta_q^c = \min_{\mathcal{M}\in\mathfrak{V}:\mathcal{M}\neq\mathcal{M}^*, q\notin\mathcal{M}} \Delta_{\mathcal{M}}^c$. If there is no $\mathcal{M}^c$ such that $q \in \mathcal{M}^c$ and $\mathcal{M}^c \neq \mathcal{M}^{*c}$, we define $\Delta_q^c = +\infty$. We define $\Delta_{min}^c = \min_{q\in\mathcal{Q}} \Delta_q^c = \min_{\mathcal{M}\in\mathfrak{V}} \Delta_{\mathcal{M}}^c$.*

Theorem A.1. *Setting $\delta = 1/T$, (1) if $\Delta_{min} > 0$, the problem-dependent regret bound for VSOCB (Alg.1) with approximate `Oracle` (Alg.3) holds:*

$$Reg_\beta(T) \leq \widetilde{O}\left(\sum_{q\in\mathcal{Q}} \frac{P(q)N(N-l_{min})}{\Delta_q^c}\right),$$

(2) our problem-independent regret bound:

$$Reg_\beta(T) \leq \widetilde{O}\left(\sqrt{N(N-l_{min})T}\right),$$

(3) and the total number of `Oracle` calls is $O(N\log T)$.

Our result under the approximate `Oracle` is less favorable than that under the exact `Oracle`. This is because, in the case of an approximate solution, it is difficult to directly apply the properties of the exact solution. Consequently, we relax the regret analysis to account for the gap between the optimal cache under the approximate solution and the cache selected by the algorithm.

A.3 Discussion of Gap Definition

To complete our theoretical analysis, we have made some modifications to the traditional definition of the gap. Table 3 shows the differences between our definition and the definition of the gap in previous works. We use $\mathcal{M}^c$ denote the complementary of $\mathcal{M}$ under $\mathcal{Q}$, i.e. $\mathcal{M}^c = \mathcal{Q} \setminus \mathcal{M}$.

For Exact `Oracle`, we define our gap as:

Definition A.4. ***(Complementary gap)**: For each $\mathcal{M} \in \mathfrak{V}$, we define the gap as*

$$\Delta_{\mathcal{M}} = \sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\mathcal{M}} P(q)C^*(q)$$

For each query q, we define $\Delta_q = \min_{\mathcal{M}\in\mathfrak{V}:\mathcal{M}\neq\mathcal{M}^*, q\notin\mathcal{M}} \Delta_{\mathcal{M}}$. Similar to the definition 2 in (Wang and Chen, 2017), if there is no cache selection $\mathcal{M}$ such that $q \in \mathcal{M}$ and $\mathcal{M} \neq \mathcal{M}^*$, we define $\Delta_q = +\infty$. Then we define $\Delta_{min} = \min_{q\in\mathcal{Q}} \Delta_q$.

Remark A.1. *(1)When solver is exact,*

$$\begin{aligned}\Delta_{\mathcal{M}} &= \sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\mathcal{M}} P(q)C^*(q) \\ &= \sum_{q\in\mathcal{M}^c} P(q)C^*(q) - \sum_{q\in\mathcal{M}^{*c}} P(q)C^*(q)\end{aligned}$$

(2) Different from classical gap definition, our gap measures the impact when a cache does not select a certain query. Our definition of gap is $\Delta_q = \min_{\mathcal{M}\in\mathfrak{V}:\mathcal{M}\neq\mathcal{M}^*, q\notin\mathcal{M}} \Delta_{\mathcal{M}}$, and classic gap definition in (Wang and Chen, 2017) is: $\Delta_q = \min_{\mathcal{M}\in\mathfrak{V}:\mathcal{M}\neq\mathcal{M}^*, q\in\mathcal{M}} \Delta_{\mathcal{M}}$.

For Approximate `Oracle` with a β-approximate solver, we define our approximate gap as:

Definition A.5. ***(Approximation gap)**: For each $\mathcal{M} \in \mathfrak{V}$ and $\mathcal{M}^c = \mathcal{Q} \setminus \mathcal{M}$, we define the gap as*

$$\Delta_{\mathcal{M}}^c = \sum_{q\in\mathcal{M}_t^c} P(q)C^*(q) - (1+\beta)\sum_{q\in\mathcal{M}^{*c}} P(q)C^*(q)$$

For each query q, we define $\Delta_q^c = \min_{\mathcal{M}\in\mathfrak{V}:\mathcal{M}\neq\mathcal{M}^*, q\notin\mathcal{M}} \Delta_{\mathcal{M}}^c$. If there is no $\mathcal{M}^c$ such that $q \in \mathcal{M}^c$ and $\mathcal{M}^c \neq \mathcal{M}^{*c}$, we define $\Delta_q^c = +\infty$. We define $\Delta_{min}^c = \min_{q\in\mathcal{Q}} \Delta_q^c = \min_{\mathcal{M}\in\mathfrak{V}} \Delta_{\mathcal{M}}^c$.

We define the gap in this way mainly because the reception of feedback in our model is different from that in the traditional bandit model. As mentioned in Remark 3.2, if we consider queries as arms, then feedback is only received for the arms that are not selected, as if we are actually choosing the complement of the cache. Therefore, we define the gap on the complement of the cache. Our definition of the gap on the complement of the cache corresponds to the definition of the cost for each cache in (Zhu

Table 3: Gap compare

Classical gap definition	**Cache gap**	**Query gap**
(Wang and Chen, 2017)	$\Delta_{\mathcal{M}} = \sum_{q \in \mathcal{M}^*} P(q)C^*(q) - \sum_{q \in \mathcal{M}} P(q)C^*(q)$	$\Delta_q = \min_{\mathcal{M} \in \mathfrak{V}: \mathcal{M} \neq \mathcal{M}^*, q \in \mathcal{M}} \Delta_{\mathcal{M}}$
Our gap definition	**Cache gap**	**Query gap**
Complementary gap (Definition A.4)	$\Delta_{\mathcal{M}} = \sum_{q \in \mathcal{M}^*} P(q)C^*(q) - \sum_{q \in \mathcal{M}} P(q)C^*(q)$	$\Delta_q = \min_{\mathcal{M} \in \mathfrak{V}: \mathcal{M} \neq \mathcal{M}^*, q \notin \mathcal{M}} \Delta_{\mathcal{M}}$
Approximate gap (Definition A.5)	$\Delta^c_{\mathcal{M}} = \sum_{q \in \mathcal{M}^c_t} P(q)C^*(q) - (1+\beta) \sum_{q \in \mathcal{M}^{*c}} P(q)C^*(q)$	$\Delta^c_q = \min_{\mathcal{M} \in \mathfrak{V}: \mathcal{M} \neq \mathcal{M}^*, q \notin \mathcal{M}} \Delta^c_{\mathcal{M}}$

where $\mathcal{M}$ is the cache that belongs to valid set, and $\mathcal{M}^*$ is the optimal cache. We use $\mathcal{M}^c$ denote the complementary of $\mathcal{M}$ under $\mathcal{Q}$, i.e. $\mathcal{M}^c = \mathcal{Q} \setminus \mathcal{M}$.

et al., 2024). Our definition of the gap on the cache measures the difference between the cost reduction achieved by other cache selections and the cost reduction achieved by the optimal cache. We define the gap for each query as the difference between the non-optimal cache that does not include this query and the optimal cache. This gap indicates the potential loss to the system if the cache is selected without considering a particular query.

The next two appendix sections present proofs of the theorems in the preceding sections, and are therefore formatted in single-column.

B Proof of Theorem 1

Before we start the proof, we first give some definitions.

Definition B.1. ***(Waiting time)****: For any sequence of query sample $\{q_t\}_{t=1}^{T}$, we define the waiting time of query q as $g(t,q) = \min_{s>t:q_s=q} s - t$. The waiting time for query q refers to the duration during which q has not been sampled since time t, or in other words, the time interval from t until q is sampled again.*

We define the following events:

$$E_1 = \{\forall t, \forall q \in \mathcal{Q}, 0 \le C^*(q) - \hat{C}_t(q) \le \min\{c_2, 2(c_2 - c_1)\sqrt{\frac{2\ln(8TN/\delta)}{T_t^{(c)}(q)}}\}\} \tag{5}$$

$$E_{2,1} = \{\forall t, \forall q \in \mathcal{Q}, 0 \le P(q) - \hat{P}_t(q) \le 2\min\{1, \sqrt{\frac{3V_t(q)\ln(16TN/\delta)}{t}} + \frac{5\ln(16TN/\delta)}{t}\}\} \tag{6}$$

$$E_{2,2} = \{\forall t, \forall q \in \mathcal{Q}, V_t(q) \le 2P(q)(1-P(q)) + \frac{2\ln(16TN/\delta)}{t}\} \tag{7}$$

$$E_2 = E_{2,1} \cap E_{2,2} \tag{8}$$

$$E_3 = \{\forall t, \forall q \in \mathcal{Q}, t < \max\{\Delta t(q,\delta), \frac{2\sum_{n=1}^{s}\mathbb{I}\{q_n = q\}}{P(q)}\}\} \tag{9}$$

$$E_4 = \{\forall t, \forall q \in \mathcal{Q}, g(t,q) \le \frac{\log(4NT/\delta)}{P(q)}\} \tag{10}$$

where $\Delta t(q,\delta) = \frac{8\log(4TN/\delta)}{P(q)}$ and $V_t(q) = \frac{1}{t}\sum_{s=1}^{t}\left(\mathbb{I}\{q_s = q\} - \frac{\sum_{i=1}^{t}\mathbb{I}\{q_i=q\}}{t}\right)^2$.
We define the good event as $E = E_1 \cap E_2 \cap E_3 \cap E_4$.

We assume that the total number of calls to `Oracle` is N_1. Let the moments when `Oracle` is called are $1 = t_1 < t_2, ..., < t_{N_1} \le t_{N_1+1} = T$, where t_{N_1+1} are the boundaries, and it does not imply that an `Oracle` call must occur at this time. According to the algorithm, an `Oracle` call definitely occurs at time $t = 1$.

B.1 Technique lemma

Lemma B.1. ***(Cache number)****:For any $\mathcal{Q}, N, M$, For any two full caches $\mathcal{M}_1, \mathcal{M}_1 \in \mathfrak{V}$, we have $|\mathcal{M}_1 \setminus \mathcal{M}_2| \le \min\{l_{max}, N - l_{min}\}$*

Proof. It is obviously that $|\mathcal{M}_1 \setminus \mathcal{M}_2| \le |\mathcal{M}_1| \le l_{max}$. On the other hand, $|\mathcal{M}_1 \setminus \mathcal{M}_2| + |\mathcal{M}_2| \le N$, then we have $|\mathcal{M}_1 \setminus \mathcal{M}_2| + \le N - |\mathcal{M}_2| \le N - l_{min}$. □

Lemma B.2. ***(Cost confidence bound)****: The event E_1 holds with probability at least $1 - \frac{\delta}{4}$.*

Proof. According to Hoeffding inequality, for a given query q and time t, we have

$$Pr\left(\left|C^*(q) - \frac{\widetilde{C}_t(q)}{T_t^{(c)}(q)}\right| \ge (c_2 - c_1)\sqrt{\frac{\ln(8NT/\delta)}{2T_t^{(c)}(q)}}\right) \le \frac{\delta}{4NT} \tag{11}$$

Then

$$Pr\left(0 \le C^*(q) - \left(\frac{\widetilde{C}_t(q)}{T_t^{(c)}(q)} - (c_2 - c_1)\sqrt{\frac{\ln(8NT/\delta)}{2T_t^{(c)}(q)}}\right) \le 2(c_2 - c_1)\sqrt{\frac{\ln(8NT/\delta)}{2T_t^{(c)}(q)}}\right) \ge 1 - \frac{\delta}{4NT} \tag{12}$$

On the other hand, $Pr\left(0 \le C^*(q) - 0 \le c_2\right) = 1$. So

$$Pr\left(0 \le C^*(q) - \hat{C}_t(q) \le \min\{c_2, 2(c_2 - c_1)\sqrt{\frac{2\ln(8TN/\delta)}{T_t^{(c)}(q)}}\}\right) \ge 1 - \frac{\delta}{4NT} \tag{13}$$

Take the union of all queries and all times, we can get $Pr(E_1) \geq 1 - \frac{\delta}{4}$. □

Lemma B.3. ***(Variance confidence bound)****: The event E_2 holds with probability at least $1-\frac{\delta}{4}$. If E_2 holds, for all queries q and all times t, there is $0 \leq P(q) - \hat{P}_t(q) \leq 2\min\{1, 2\sqrt{\frac{P(q)\ln(16TN/\delta)}{t}} + \frac{8\ln(16TN/\delta)}{t}\}$*

The proof of this lemma can be obtained from Lemma 7 in (Liu et al., 2024).

Lemma B.4. ***(Time of duration)****: The event E_3 holds with probablity $1 - \frac{\delta}{4}$, i.e. for $\forall 1 \leq t \leq T$, $\forall q \in \mathcal{Q}$, with probability of $1 - \frac{\delta}{4}$, we have*

$$t < \max\{\Delta t(q,\delta), \frac{2\sum_{n=1}^{t}\mathbb{I}\{q_n = q\}}{P(q)}\}$$

where $\Delta t(q,\delta) = \frac{8\log(4TN/\delta)}{P(q)}$.

Proof.

$$Pr\left(\exists t = s, 1 \leq s \leq T, \exists q', s \geq \max\{\Delta t(q',\delta), \frac{2\sum_{n=1}^{s}\mathbb{I}\{q_n = q\}}{P(q)}\}\right) \tag{14}$$

$$= Pr\left(\cup_{s=1}^{T}\cup_{q'\in\mathcal{Q}}\{t = s, s \geq \max\{\Delta t(q',\delta), \frac{2\sum_{n=1}^{s}\mathbb{I}\{q_n = q\}}{P(q)}\}\}\right) \tag{15}$$

$$\leq \sum_{s=1}^{T}\sum_{q'\in\mathcal{Q}} Pr\left(t = s, s \geq \max\{\Delta t(q',\delta), \frac{2\sum_{n=1}^{s}\mathbb{I}\{q_n = q\}}{P(q)}\}\right) \tag{16}$$

$$\leq \sum_{s=1}^{T}\sum_{q'\in\mathcal{Q}} Pr\left(s \geq \max\{\Delta t(q',\delta), \frac{2\sum_{n=1}^{s}\mathbb{I}\{q_n = q\}}{P(q)}\}\right) \tag{17}$$

$$= \sum_{q'\in\mathcal{Q}}\sum_{s=1}^{\Delta t(q',\delta)-1} Pr\left(s \geq \max\{\Delta t(q,\delta), \frac{2\sum_{n=1}^{s}\mathbb{I}\{q_n = q\}}{P(q)}\}\right) \tag{18}$$

$$+ \sum_{q'\in\mathcal{Q}}\sum_{s=\Delta t(q',\delta)}^{T} Pr\left(s \geq \max\{\Delta t(q,\delta), \frac{2\sum_{n=1}^{s}\mathbb{I}\{q_n = q\}}{P(q)}\}\right) \tag{19}$$

$$= 0 + \sum_{q'\in\mathcal{Q}}\sum_{s=\Delta t(q',\delta)}^{T} Pr\left(s \geq \max\{\Delta t(q,\delta), \frac{2\sum_{n=1}^{s}\mathbb{I}\{q_n = q\}}{P(q)}\}\right) \tag{20}$$

According to Chernoff bound, for any q', if $s \geq \Delta t(q',\delta)$, we have

$$Pr\left(\sum_{n=1}^{s}\mathbb{I}\{q_n = q'\} \leq \frac{s \cdot P(q')}{2}\right) \leq \frac{\delta}{4NT}$$

Then

$$\sum_{q'\in\mathcal{Q}}\sum_{s=\Delta t(q',\delta)}^{T} Pr\left(s \geq \max\{\Delta t(q,\delta), \frac{2\sum_{n=1}^{s}\mathbb{I}\{q_n = q\}}{P(q)}\}\right) \tag{21}$$

$$\leq \sum_{q'\in\mathcal{Q}}\sum_{s=\Delta t(q',\delta)}^{T} \frac{\delta}{4NT} \leq \frac{\delta}{4} \tag{22}$$

□

Lemma B.5. ***(Max wait time)****: The event E_4 holds with probability at least $1 - \frac{\delta}{4}$.*

The proof of Lemma B.5 can be obtained from Lemma 3 in (Zhu et al., 2024).

Lemma B.6. ***(Good event)****: The good event E holds with probability at least $1-\delta$.*

Combining the above lemmas, Lemma B.6 can be directly derived.

Lemma B.7. ***(Number of `Oracle` call)****: For any $\alpha > 0$, there exists a sufficiently large constant R such that $N_1 \leq RN \ln T$.*

Proof. Let $N_1(q)$ denote the number of calling `Oracle` triggered by condition $T_t^{(c)}(q) \geq (1+\alpha)T_{t-1}^{(r)}(q)$ with query q, and $N_1(T)$ denote the number of calling `Oracle` triggered by condition $t \geq (1+\alpha)T_{t-1}$. We use $t_1(q), ..., t_{N_1(q)}(q)$ to denote the moments when `Oracle` is called due to q satisfying the condition. We use $t_1(T), ..., t_{N_1(T)}(T)$ to denote the moments when `Oracle` is called due to time satisfying the condition.

For query: According to our accumulation-based `Oracle` invoking policy, when `Oracle` is called, we have

$$T_{t_{i+1}(q)}^{(c)}(q) \geq (1+\alpha)T_{t_{i+1}(q)-1}^{(r)}(q) \geq (1+\alpha)T_{t_i(q)}^{(r)}(q) = (1+\alpha)T_{t_i(q)}^{(c)}(q) \tag{23}$$

Then

$$T \geq T_{t_{N_1(q)}(q)}^{(c)}(q) \geq (1+\alpha)^{N_1(q)-1}T_{t_1(q)}^{(c)}(q) \geq (1+\alpha)^{N_1(q)-1}$$

So we have $N_1(q) \leq R_q \ln T$, where R_q is a large enough constant.

For time: Let t_i' denote the most recent moment when `Oracle` was called before time $t_i(T)$. According to the `Oracle` invoking policy.

$$t_{i+1}(T) \geq (1+\alpha)t_{i+1}' \geq (1+\alpha)t_i(T) \tag{24}$$

So

$$T \geq t_{N_1(T)}(T) \geq (1+\alpha)^{N_1(T)-1}t_1(T) \geq (1+\alpha)^{N_1(T)-1} \tag{25}$$

Thus, we can get $N_1(T) \leq R' \ln T$, where R' is a large enough constant.

Above all, set $R = max_{q\in\mathcal{Q}}R_q + 1$, we can get

$$N_1 = \sum_{q\in\mathcal{Q}} N_1(q) + N_1(T) \leq RN \ln T$$

□

B.2 Full cache in online setting

Remarking that our valid set was defined on the gobal query set $\mathcal{Q}$, but in online setting, the record set $\mathcal{Q}_t$ may not encounter all of querys when `Oracle` is called. At this time, $\mathcal{Q}_t \neq \mathcal{Q}$, which may cause the recommend cache not full, i.e. $\widehat{\mathcal{M}}_t \notin \mathfrak{V}$ in a global view. Because when $\widehat{\mathcal{M}}_t$ is not full, there may have may choices to make it full, according to the algorithm, the query that first satisfy the condition full will be this choices. We assume that there is a seer know which query will come first to make recommend cache full, from this a full version of one recommend cache is fixed.

Definition B.2. ***(Valid cache)****: We assume that each recommend cache have a full version of it, denoted by $\widehat{\mathcal{M}}_t^{(full)}$. Assuming that `Oracle` is called at time t, and it is not called again afterwards, meaning that the recommendation cache remains unchanged. If $\widehat{\mathcal{M}}_{t_i}$ originally belongs to the valid set, then $\widehat{\mathcal{M}}_t = \widehat{\mathcal{M}}_{t_i}^{(full)}$. Otherwise, according to our algorithm, if the recommendation cache does not belong to the valid set, it will be filled with some queries that are not in the recommendation cache, so that it becomes a valid set. Suppose that after time t, some queries not in the recommendation cache fill up it. Then the union of $\widehat{\mathcal{M}}_t$ and these queries is defined as the valid cache $\widehat{\mathcal{M}}_t^{(full)}$ of it at time t.*

Because the recommendation cache is full within the scope of $\mathcal{Q}_t$, when $\mathcal{Q}_t = \mathcal{Q}$, we have $\widehat{\mathcal{M}}_t = \widehat{\mathcal{M}}_t^{(full)}$. Clearly, $\mathcal{M}_t \subset \widehat{\mathcal{M}}_t \subset \widehat{\mathcal{M}}_t^{(full)}$.

Lemma B.8. ***(Full sum equal)****: If* `Oracle` *is not called again after* t*:*

$$\sum_{q\in\widehat{\mathcal{M}}_t^{(full)}} \hat{P}_t(q)\hat{C}_t(q) = \sum_{q\in\widehat{\mathcal{M}}_t} \hat{P}_{t_i}(q)\hat{C}_{t_i}(q) \tag{26}$$

Proof. If $\widehat{\mathcal{M}}_t \neq \widehat{\mathcal{M}}_t^{(full)}$, according to the definition, it implies that there are some queries that the algorithm has never seen before, which can fill up the recommendation cache. Since the estimated values of these unseen queries are all 0, the process of filling the recommendation cache by these unseen queries is equivalent to these queries being selected into the recommendation cache with an estimated value of 0 at time t. □

Lemma B.9. ***(Full cache compare)****: If good event* E *holds, assuming that* `Oracle` *is called at* t*, and is not called again afterwards, then:*

$$\sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\widehat{\mathcal{M}}_t^{(full)}} P(q)C^*(q) \tag{27}$$

$$\leq \sum_{q\in\mathcal{M}^*\setminus\widehat{\mathcal{M}}_t^{(full)}} \Big(P(q)C^*(q) - \hat{P}_t(q)\hat{C}_t(q)\Big) \tag{28}$$

Proof.

$$\sum_{q\in\widehat{\mathcal{M}}_t^{(full)}} \hat{P}_t(q)\hat{C}_t(q) = \sum_{q\in\widehat{\mathcal{M}}_t} \hat{P}_t(q)\hat{C}_t(q) \geq \sum_{q\in\mathcal{M}^*} \hat{P}_t(q)\hat{C}_t(q) \tag{29}$$

Then, with E holds:

$$\sum_{q\in\widehat{\mathcal{M}}_t^{(full)}\setminus\mathcal{M}^*} P(q)C^*(q) \geq \sum_{q\in\widehat{\mathcal{M}}_t^{(full)}\setminus\mathcal{M}^*} \hat{P}_t(q)\hat{C}_t(q) \geq \sum_{q\in\mathcal{M}^*\setminus\widehat{\mathcal{M}}_t^{(full)}} \hat{P}_t(q)\hat{C}_t(q) \tag{30}$$

We have

$$\sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\widehat{\mathcal{M}}_t^{(full)}} P(q)C^*(q) \tag{31}$$

$$= \sum_{q\in\mathcal{M}^*\setminus\widehat{\mathcal{M}}_t^{(full)}} P(q)C^*(q) - \sum_{q\in\widehat{\mathcal{M}}_t^{(full)}\setminus\mathcal{M}^*} P(q)C^*(q) \tag{32}$$

$$\leq \sum_{q\in\mathcal{M}^*\setminus\widehat{\mathcal{M}}_t^{(full)}} P(q)C^*(q) - \sum_{q\in\mathcal{M}^*\setminus\widehat{\mathcal{M}}_t^{(full)}} \hat{P}_t(q)\hat{C}_t(q) \tag{33}$$

$$= \sum_{q\in\mathcal{M}^*\setminus\widehat{\mathcal{M}}_t^{(full)}} \Big(P(q)C^*(q) - \hat{P}_t(q)\hat{C}_t(q)\Big) \tag{34}$$

□

Lemma B.10. ***(Full regret equal)****: Assuming that* `Oracle` *is called at* t_i*, and is not called during* $t_i+1, t_{i+1}-1$*, if the event* E_4 *holds:*

$$\sum_{t=t_i+1}^{t_{i+1}-1} \mathbb{E}[\sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\mathcal{M}_t} P(q)C^*(q)] \tag{35}$$

$$\leq (t_{i+1}-t_i-1)\mathbb{E}[\sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\widehat{\mathcal{M}}_{t_i}^{(full)}} P(q)C^*(q)] + 2c_2 N\ln(4NT/\delta) \tag{36}$$

Proof. According to our algorithm, $\mathcal{M}_{t_i} = \widehat{\mathcal{M}}_{t_i} \cap \mathcal{M}_{t_i-1} \subset \widehat{\mathcal{M}}_{t_i}^{(full)} \cap \mathcal{M}_{t_i-1}$

$$\sum_{t=t_i+1}^{t_{i+1}-1} \mathbb{E}\left[\sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\mathcal{M}_t} P(q)C^*(q)\right] \tag{37}$$

$$= \sum_{t=t_i+1}^{t_{i+1}-1} \mathbb{E}\left[\sum_{q\in\mathcal{Q}} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\mathcal{M}_t\}\right)\right] \tag{38}$$

$$= \sum_{q\in\mathcal{Q}} \mathbb{E}\left[\sum_{t=t_i+1}^{t_{i+1}-1} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\mathcal{M}_t\}\right)\right] \tag{39}$$

$$\tag{40}$$

For any q, if $t_{i+1} - 1 < g(t_i, q)$, we have $\sum_{t=t_i+1}^{t_{i+1}-1} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\mathcal{M}_t\}\right) \leq g(t_i,q)P(q)C^*(q)$, otherwise:

$$\sum_{t=t_i+1}^{t_{i+1}-1} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\mathcal{M}_t\}\right) \tag{41}$$

$$= \underbrace{\sum_{t=t_i+1}^{t_i+g(t_i,q)} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\mathcal{M}_t\}\right)}_{A} + \underbrace{\sum_{t=t_i+g(t_i,q)+1}^{t_{i+1}-1} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\mathcal{M}_t\}\right)}_{B} \tag{42}$$

where $g(t_i, q)$ is the waiting time that denotes the time interval from time t until the first moment when query q is received starting from time t.

For term A:

(1) If $q \in \widehat{\mathcal{M}}_{t_i}^{(full)} \setminus \mathcal{M}_{t_i-1}$, then $\mathbb{I}\{q\in\mathcal{M}_t\} = 0$ during $t_i \sim g(t_i,q) + t_i$, we have

$$\sum_{t=t_i+1}^{t_i+g(t_i,q)} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\mathcal{M}_t\}\right) = \sum_{t=t_i+1}^{t_i+g(t_i,q)} P(q)C^*(q)\mathbb{I}\{q\in\mathcal{M}^*\} \tag{43}$$

$$\leq g(t_i,q)P(q)C^*(q) \tag{44}$$

(2) If $q \in \widehat{\mathcal{M}}_{t_i}^{(full)} \cap \mathcal{M}_{t_i-1}$, then $\mathbb{I}\{q\in\mathcal{M}_t\} = \mathbb{I}\{q\in\widehat{\mathcal{M}}_{t_i}^{(full)}\} = 1$.

(3) If $q \notin \widehat{\mathcal{M}}_{t_i}^{(full)}$, then $\mathbb{I}\{q\in\mathcal{M}_t\} = \mathbb{I}\{q\in\widehat{\mathcal{M}}_{t_i}^{(full)}\} = 0$.

For term B: Because $\mathcal{M}_t \subset \widehat{\mathcal{M}}_t \subset \widehat{\mathcal{M}}_t^{(full)}$, algorithm will place the query $q \in \widehat{\mathcal{M}}_{t_i}^{(full)}$ into the current cache upon first receiving it. Therefore, from the perspective of query q after it is received, the current cache $\mathcal{M}_t$ is equivalent to the valid cache $\widehat{\mathcal{M}}_t^{(full)}$. In the same time, `Oracle` is not call, so $\widehat{\mathcal{M}}_t^{(full)} = \widehat{\mathcal{M}}_{t_i}^{(full)}$. As a reslut, for any q, we have

$$\sum_{t=t_i+g(t_i,q)+1}^{t_{i+1}-1} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\mathcal{M}_t\}\right) = \sum_{t=t_i+g(t_i,q)+1}^{t_{i+1}-1} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\widehat{\mathcal{M}}_{t_i}^{(full)}\}\right)$$

Above all, we can get

$$\sum_{t=t_i+1}^{t_{i+1}-1} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\mathcal{M}_t\}\right) \tag{45}$$

$$\leq g(t_i,q)P(q)C^*(q) + \sum_{t=t_i+g(t_i,q)+1}^{t_{i+1}-1} P(q)C^*(q)\left(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\widehat{\mathcal{M}}_{t_i}^{(full)}\}\right) \tag{46}$$

One the other hand, for any $q, \widehat{\mathcal{M}}_{t_i}^{(full)}$ it holds that:

$$0 \leq \sum_{t=t_i+1}^{t_i+g(t_i,q)} P(q)C^*(q) \left(\mathbb{I}\{q \in \mathcal{M}^*\} - \mathbb{I}\{q \in \widehat{\mathcal{M}}_{t_i}^{(full)}\}\right) + \sum_{t=t_i+1}^{t_i+g(t_i,q)} P(q)C^*(q) \tag{47}$$

So

$$\sum_{t=t_i+1}^{t_{i+1}-1} P(q)C^*(q) \left(\mathbb{I}\{q \in \mathcal{M}^*\} - \mathbb{I}\{q \in \mathcal{M}_t\}\right) \tag{48}$$

$$\leq g(t_i,q)P(q)C^*(q) + \sum_{t=t_i+g(t_i,q)+1}^{t_{i+1}-1} P(q)C^*(q) \left(\mathbb{I}\{q \in \mathcal{M}^*\} - \mathbb{I}\{q \in \widehat{\mathcal{M}}_{t_i}^{(full)}\}\right) \tag{49}$$

$$\leq 2g(t_i,q)P(q)C^*(q) + (t_{i+1} - t_i - 1)\, P(q)C^*(q) \left(\mathbb{I}\{q \in \mathcal{M}^*\} - \mathbb{I}\{q \in \widehat{\mathcal{M}}_{t_i}^{(full)}\}\right) \tag{50}$$

If E_4 holds, $2g(t_i,q)P(q)C^*(q) \leq 2c_2 \ln(4NT/\delta)$.

$$\sum_{q \in \mathcal{Q}} \mathbb{E}\left[\sum_{t=t_i+1}^{t_{i+1}-1} P(q)C^*(q) \left(\mathbb{I}\{q \in \mathcal{M}^*\} - \mathbb{I}\{q \in \mathcal{M}_t\}\right)\right] \tag{51}$$

$$\leq \sum_{q \in \mathcal{Q}} \mathbb{E}\left[2c_2 \ln(4NT/\delta) + (t_{i+1} - t_i - 1)\, P(q)C^*(q) \left(\mathbb{I}\{q \in \mathcal{M}^*\} - \mathbb{I}\{q \in \widehat{\mathcal{M}}_{t_i}^{(full)}\}\right)\right] \tag{52}$$

$$= (t_{i+1} - t_i - 1)\mathbb{E}[\sum_{q \in \mathcal{M}^*} P(q)C^*(q) - \sum_{q \in \widehat{\mathcal{M}}_{t_i}^{(full)}} P(q)C^*(q)] + 2c_2 N \ln(4NT/\delta) \tag{53}$$

□

B.3 Main regret

For Theorem 1, the solver in `Oracle` is exact. In this case, the output of `Oracle` is the optimal solution. First we assume that for each query q there is the constant $Z_q, q \in \mathcal{Q}$ and for each valid set $\mathcal{M} \in \mathfrak{V}$ we define $Z_{\mathcal{M}} = \max_{q \notin \mathcal{M}} Z_q$. This assumption and method for problem-dependent bound is inspired by [CMAB-T], but we define on the complementary.

As a convention, we define $l(M, \mathcal{Q}) = \min\{l_{max}, N - l_{min}\}$. We use $\neg E$ to denote that event E does not occur.

According to the definition [valid cache], we known that $\widehat{\mathcal{M}}_t^{(full)} \in \mathfrak{V}$ for any $\widehat{\mathcal{M}}_t$. Then we define $\Delta_{\widehat{\mathcal{M}}_t^{(full)}} = \sum_{q \in \mathcal{M}^*} P(q)C^*(q) - \sum_{q \in \widehat{\mathcal{M}}_t^{(full)}} P(q)C^*(q)$.

We first bound the gap.

Lemma B.11. ***(Gap bound)****: At any moment t_i when an* `Oracle` *call occurs, if E holds and the gap satisfies $\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}$, then*

$$\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \tag{54}$$

$$\leq 4c_2\sqrt{2\ln(8TN/\delta)} \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} P(q)\sqrt{\frac{1}{T_{t_i}^{(c)}(q)}} \cdot \mathbb{I}\{T_{t_i}^{(c)}(q) \leq l_T^{(1)}(q, Z_q)\} \tag{55}$$

$$+ 8c_2\sqrt{\ln(16TN/\delta)} \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} \sqrt{\frac{P(q)}{t_i}} \cdot \mathbb{I}\{t_i \leq l_T^{(2)}(q, Z_q)\} + \frac{16 l(M,\mathcal{Q}) \ln(16TN/\delta)}{t_i} \tag{56}$$

where $l_T^{(1)}(q, Z_q) = \frac{128P^2(q)l^2(M,\mathcal{Q})c_2^2 \ln(8TN/\delta)}{Z_q^2}$, *and* $l_T^{(2)}(q, Z_q) = \frac{256c_2^2 P(q) l^2(M,\mathcal{Q}) \ln(16TN/\delta)}{Z_q^2}$.

Proof. According to Lemma B.9, we have

$$\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \leq \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} \Big(P(q)C^*(q) - \hat{P}_{t_i}(q)\hat{C}_{t_i}(q) \Big) \tag{57}$$

On the other hand

$$\sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} \Big(P(q)C^*(q) - \hat{P}_{t_i}(q)\hat{C}_{t_i}(q) \Big) \tag{58}$$

$$= \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} \Big(P(q)(C^*(q) - \hat{C}_{t_i}(q)) + \hat{C}_{t_i}(q)(P(q) - \hat{P}_{t_i}(q)) \Big) \tag{59}$$

When good event E holds and $\mathbb{I}\left\{ \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \right\}$, then at time t_i:

$$\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \tag{60}$$

$$\leq 2 \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} \Big(P(q)(C^*(q) - \hat{C}_{t_i}(q)) + \hat{C}_{t_i}(q)(P(q) - \hat{P}_{t_i}(q)) \Big) - Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \tag{61}$$

$$\leq 2 \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} \Big(P(q)(C^*(q) - \hat{C}_{t_i}(q)) + c_2(P(q) - \hat{P}_{t_i}(q)) \Big) - Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \tag{62}$$

$$\leq 2 \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} (P(q)|C^*(q) - \hat{C}_{t_i}(q)| - \frac{Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}}{4|\mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}|}) \tag{63}$$

$$+ 2c_2 \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} (|P(q) - \hat{P}_{t_i}(q)| - \frac{Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}}{4c_2|\mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}|}) \tag{64}$$

$$\leq 2 \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} (P(q)|C^*(q) - \hat{C}_{t_i}(q)| - \frac{Z_q}{4l(M, \mathcal{Q})}) \tag{65}$$

$$+ 2c_2 \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} (|P(q) - \hat{P}_{t_i}(q)| - \frac{Z_q}{4c_2 l(M, \mathcal{Q})}) \tag{66}$$

Eq. (62) is because when E holds, $\hat{C}_{t_i}(q) \leq C^*(q) \leq c_2$. Eqs. (65) and (66) is because $Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \geq Z_q, \forall q \notin \widehat{\mathcal{M}}_{t_i}^{(full)}$ and Lemma B.1.

For the term in Eq. (65): By good event E, we have

$$P(q)|C^*(q) - \hat{C}_{t_i}(q)| - \frac{Z_q}{4l(M, \mathcal{Q})} \tag{67}$$

$$\leq P(q) \min\{c_2, 2(c_2 - c_1)\sqrt{\frac{2\ln(8TN/\delta)}{T_{t_i}^{(c)}(q)}}\} - \frac{Z_q}{4l(M, \mathcal{Q})} \tag{68}$$

$$\leq 2c_2 P(q)\sqrt{\frac{2\ln(8TN/\delta)}{T_{t_i}^{(c)}(q)}} - \frac{Z_q}{4l(M, \mathcal{Q})} \tag{69}$$

$$= 2c_2 P(q)\sqrt{2\ln(8TN/\delta)} \left(\frac{1}{\sqrt{T_{t_i}^{(c)}(q)}} - \frac{Z_q}{8c_2 l(M, \mathcal{Q}) P(q)\sqrt{2\ln(8TN/\delta)}} \right) \tag{70}$$

When $T_{t_i}^{(c)}(q) > \frac{128P^2(q)l^2(M,\mathcal{Q})c_2^2\ln(8TN/\delta)}{Z_q^2} \doteq l_T^{(1)}(q, Z_q)$,

$$\frac{1}{\sqrt{T_{t_i}^{(c)}(q)}} - \frac{Z_q}{8c_2 l(M,\mathcal{Q})P(q)\sqrt{2\ln(8TN/\delta)}} < 0 \tag{71}$$

When $T_{t_i}^{(c)}(q) \leq l_T^{(1)}(q, Z_q)$,

$$0 \leq \frac{1}{\sqrt{T_{t_i}^{(c)}(q)}} - \frac{Z_q}{8c_2 l(M,\mathcal{Q})P(q)\sqrt{2\ln(8TN/\delta)}} < \frac{1}{\sqrt{T_{t_i}^{(c)}(q)}} \tag{72}$$

Plugging these into above,

$$P(q)|C^*(q) - \hat{C}_{t_i}(q)| - \frac{Z_q}{4l(M,\mathcal{Q})} \tag{73}$$

$$\leq 2c_2 P(q)\sqrt{\frac{2\ln(8TN/\delta)}{T_{t_i}^{(c)}(q)}} \cdot \mathbb{I}\{T_{t_i}^{(c)}(q) \leq l_T^{(1)}(q, Z_q)\} \tag{74}$$

For the term in Eq. (66): By E and Lemma [variance confidence bound],

$$|P(q) - \hat{P}_{t_i}(q)| - \frac{Z_q}{4c_2 l(M,\mathcal{Q})} \tag{75}$$

$$\leq 2\min\{1, 2\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} + \frac{8\ln(16TN/\delta)}{t_i}\} - \frac{Z_q}{4c_2 l(M,\mathcal{Q})} \tag{76}$$

$$\leq 4\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} - \frac{Z_q}{4c_2 l(M,\mathcal{Q})} + \frac{16\ln(16TN/\delta)}{t_i} \tag{77}$$

When $t_i > \frac{256c_2^2 P(q)l^2(M,\mathcal{Q})\ln(16TN/\delta)}{Z_q^2} \doteq l_T^{(2)}(q, Z_q)$,

$$4\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} - \frac{Z_q}{4c_2 l(M,\mathcal{Q})} < 0 \tag{78}$$

When $t_i \leq l_T^{(2)}(q, Z_q)$,

$$0 \leq 4\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} - \frac{Z_q}{4c_2 l(M,\mathcal{Q})} < 4\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} \tag{79}$$

Combining these, we have

$$|P(q) - \hat{P}_{t_i}(q)| - \frac{Z_q}{4c_2 l(M,\mathcal{Q})} \tag{80}$$

$$\leq 4\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} \cdot \mathbb{I}\{t_i \leq l_T^{(2)}(q, Z_q)\} + \frac{16\ln(16TN/\delta)}{t_i} \tag{81}$$

Thus

$$\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \tag{82}$$

$$\leq 4c_2\sqrt{2\ln(8TN/\delta)} \sum_{q\in\mathcal{M}^*\setminus\widehat{\mathcal{M}}_{t_i}^{(full)}} P(q)\sqrt{\frac{1}{T_{t_i}^{(c)}(q)}} \cdot \mathbb{I}\{T_{t_i}^{(c)}(q) \leq l_T^{(1)}(q, Z_q)\} \tag{83}$$

$$+ 8c_2\sqrt{\ln(16TN/\delta)} \sum_{q\in\mathcal{M}^*\setminus\widehat{\mathcal{M}}_{t_i}^{(full)}} \sqrt{\frac{P(q)}{t_i}} \cdot \mathbb{I}\{t_i \leq l_T^{(2)}(q, Z_q)\} + \frac{16l(M,\mathcal{Q})\ln(16TN/\delta)}{t_i} \tag{84}$$

□

We can write the regret as follows.

$$Reg(T) = \mathbb{E}\left[\sum_{t=1}^{T} C(q_t)\left(\mathbb{I}\{q_t \in \mathcal{M}^*\} - \mathbb{I}\{q_t \in \mathcal{M}_t\}\right)\right] \tag{85}$$

$$= \mathbb{E}\left[\sum_{t=1}^{T} C(q_t)\left(\mathbb{I}\{q_t \in \mathcal{M}^*\} - \mathbb{I}\{q_t \in \mathcal{M}_t\}\right)|E\right] Pr(E) \tag{86}$$

$$+ \mathbb{E}\left[\sum_{t=1}^{T} C(q_t)\left(\mathbb{I}\{q_t \in \mathcal{M}^*\} - \mathbb{I}\{q_t \in \mathcal{M}_t\}\right)|\neg E\right] Pr(\neg E) \tag{87}$$

$$\leq \mathbb{E}\left[\sum_{t=1}^{T} C(q_t)\left(\mathbb{I}\{q_t \in \mathcal{M}^*\} - \mathbb{I}\{q_t \in \mathcal{M}_t\}\right)|E\right] + Tc_2\delta \tag{88}$$

For convenience in writing, we omit the conditioning event E subsequently. Taking expection on q_t, we have

$$\mathbb{E}\left[\sum_{t=1}^{T} C(q_t)\left(\mathbb{I}\{q_t \in \mathcal{M}^*\} - \mathbb{I}\{q_t \in \mathcal{M}_t\}\right)|E\right] \tag{89}$$

$$\leq \sum_{t=1}^{T} \mathbb{E}\left[\sum_{q\in Q} P(q)C^*(q)\left(\mathbb{I}\{q \in \mathcal{M}^*\} - \mathbb{I}\{q \in \mathcal{M}_t\}\right)\right] \tag{90}$$

Partitioning based on the moments of `Oracle` calls, we obtain

$$\sum_{t=1}^{T} \mathbb{E}\left[\sum_{q\in Q} P(q)C^*(q)\left(\mathbb{I}\{q \in \mathcal{M}^*\} - \mathbb{I}\{q \in \mathcal{M}_t\}\right)\right] \tag{91}$$

$$= \sum_{i=1}^{N_1} \sum_{t=t_i+1}^{t_{i+1}-1} \mathbb{E}[\sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\mathcal{M}_t} P(q)C^*(q)] \tag{92}$$

$$+ \sum_{i=1}^{N_1+1} \mathbb{E}[\sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\mathcal{M}_{t_i}} P(q)C^*(q)] \tag{93}$$

According to Lemma B.10, for any t_i:

$$\sum_{t=t_i+1}^{t_{i+1}-1} \mathbb{E}[\sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\mathcal{M}_t} P(q)C^*(q)] \tag{94}$$

$$\leq (t_{i+1} - t_i - 1)\mathbb{E}[\sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\widehat{\mathcal{M}}_{t_i}^{(full)}} P(q)C^*(q)] + 2c_2 N \ln(4NT/\delta) \tag{95}$$

And

$$\mathbb{E}\left[\sum_{i=1}^{N_1+1}\left(\sum_{q\in\mathcal{M}^*} P(q)C^*(q) - \sum_{q\in\mathcal{M}_{t_i}} P(q)C^*(q)\right)\right] \tag{96}$$

$$\leq (N_1+1)\sum_{q\in\mathcal{M}^*} P(q)C^*(q) \leq (N_1+1)c_2 \sum_{q\in\mathcal{M}^*} P(q) \leq (N_1+1)c_2 \tag{97}$$

The last inequality is because $C^*(q) \leq c_2$ and $\sum_{q\in\mathcal{M}^*} P(q) \leq \sum_{q\in\mathcal{Q}} P(q) = 1$.

Combining these inequality, we can get

$$\sum_{i=1}^{N_1}\sum_{t=t_i+1}^{t_{i+1}-1}\mathbb{E}[\sum_{q\in\mathcal{M}^*}P(q)C^*(q)-\sum_{q\in\mathcal{M}_t}P(q)C^*(q)] \tag{98}$$

$$+\sum_{i=1}^{N_1+1}\mathbb{E}[\sum_{q\in\mathcal{M}^*}P(q)C^*(q)-\sum_{q\in\mathcal{M}_{t_i}}P(q)C^*(q)] \tag{99}$$

$$\leq \mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\left(\sum_{q\in\mathcal{M}^*}P(q)C^*(q)-\sum_{q\in\widehat{\mathcal{M}}_{t_i}^{(full)}}P(q)C^*(q)\right)\right]+2c_2N_1N\ln(4NT/\delta) \tag{100}$$

$$+(N_1+1)c_2 \tag{101}$$

$$\leq \mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right]+4c_2N_1N\ln(4NT/\delta) \tag{102}$$

According to Lemma B.7, we have $N_1 \leq RN\ln T$ for a large enough constant. So

$$4c_2N_1N\ln(4NT/\delta)\leq 4c_2N^2\ln^2(4NT/\delta)$$

Considering $\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\leq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}$ and $\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}$, we have

$$\mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right] \tag{103}$$

$$=\mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\}\right] \tag{104}$$

$$+\mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\leq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\}\right] \tag{105}$$

$$\leq\mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\}\right] \tag{106}$$

$$+T\max_{t_i}\{Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\}\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}<Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\} \tag{107}$$

Combining above inequality, we have

$$Reg(T)\leq\mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}|E,\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\}\right] \tag{108}$$

$$+Tc_2\delta+4c_2N^2\ln^2(4NT/\delta)+T\max_{t_i}\{Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\}\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}<Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\} \tag{109}$$

By Lemma B.11,

$$\mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}|E,\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\}\right] \tag{110}$$

$$\leq 4c_2\sqrt{2\ln(8TN/\delta)}\sum_{i=1}^{N_1}\sum_{q\in\mathcal{M}^*\setminus\widehat{\mathcal{M}}_{t_i}^{(full)}}(t_{i+1}-t_i-1)P(q)\sqrt{\frac{1}{T_{t_i}^{(c)}(q)}}\cdot\mathbb{I}\{T_{t_i}^{(c)}(q)\leq l_T^{(1)}(q,Z_q)\} \tag{111}$$

$$+8c_2\sqrt{\ln(16TN/\delta)}\sum_{i=1}^{N_1}\sum_{q\in\mathcal{M}^*\setminus\widehat{\mathcal{M}}_{t_i}^{(full)}}(t_{i+1}-t_i-1)\sqrt{\frac{P(q)}{t_i}}\cdot\mathbb{I}\{t_i\leq l_T^{(2)}(q,Z_q)\} \tag{112}$$

$$+16l(M,\mathcal{Q})\ln(16TN/\delta)\sum_{i=1}^{N_1}\frac{t_{i+1}-t_i-1}{t_i} \tag{113}$$

$$=\underbrace{4c_2\sqrt{2\ln(8TN/\delta)}\sum_{q\in\mathcal{M}^*}\sum_{i=1}^{N_1}\mathbb{I}\{q\notin\widehat{\mathcal{M}}_{t_i}^{(full)}\}(t_{i+1}-t_i-1)P(q)\sqrt{\frac{1}{T_{t_i}^{(c)}(q)}}\cdot\mathbb{I}\{T_{t_i}^{(c)}(q)\leq l_T^{(1)}(q,Z_q)\}}_{A_1}$$

$$+\underbrace{8c_2\sqrt{\ln(16TN/\delta)}\sum_{q\in\mathcal{M}^*}\sum_{i=1}^{N_1}\mathbb{I}\{q\notin\widehat{\mathcal{M}}_{t_i}^{(full)}\}(t_{i+1}-t_i-1)\sqrt{\frac{P(q)}{t_i}}\cdot\mathbb{I}\{t_i\leq l_T^{(2)}(q,Z_q)\}}_{A_2}$$

$$+\underbrace{16l(M,\mathcal{Q})\ln(16TN/\delta)\sum_{i=1}^{N_1}\frac{t_{i+1}-t_i-1}{t_i}}_{A_3}$$

For term A_1: According to Lemma B.4

$$\mathbb{I}\{q\notin\widehat{\mathcal{M}}_{t_i}^{(full)}\}(t_{i+1}-t_i-1)P(q)\sqrt{\frac{1}{T_{t_i}^{(c)}(q)}} \tag{114}$$

$$\leq\mathbb{I}\{q\notin\widehat{\mathcal{M}}_{t_i}^{(full)}\}\Delta t(q,\delta)P(q)\sqrt{\frac{1}{T_{t_i}^{(c)}(q)}} \tag{115}$$

$$+2\mathbb{I}\{q\notin\widehat{\mathcal{M}}_{t_i}^{(full)}\}\sum_{n=1}^{t_{i+1}-t_i-1}\mathbb{I}\{q_n=q\}\sqrt{\frac{1}{T_{t_i}^{(c)}(q)}} \tag{116}$$

$$\leq 8\ln(4TN/\delta)+2\mathbb{I}\{q\notin\widehat{\mathcal{M}}_{t_i}^{(full)}\}\sum_{n=1}^{t_{i+1}-t_i-1}\mathbb{I}\{q_n=q\}\sqrt{\frac{1}{T_{t_i}^{(c)}(q)}} \tag{117}$$

$\sum_{n=1}^{t_{i+1}-t_i-1}\mathbb{I}\{q_n=q\}$ denotes the number of arrivals of the query q within the interval from t_i+1 to $t_{i+1}-1$. If $\mathbb{I}\{q\notin\widehat{\mathcal{M}}_{t_i}^{(full)}\}=1$, it indicates that the query is not in the cache during this period, so the number of arrivals of the query within this period is the same as its number of misses. Then we have

$$\mathbb{I}\{q\notin\widehat{\mathcal{M}}_{t_i}^{(full)}\}\sum_{n=1}^{t_{i+1}-t_i-1}\mathbb{I}\{q_n=q\}=T_{t_{i+1}-1}^{(c)}(q)-T_{t_i}^{(c)}(q) \tag{118}$$

In our algorithm, $T_t^{(r)}(q)$ denotes the number of misses of the query q at the most recent `Oracle` call triggered by q prior to time t. Moreover, $T_t^{(r)}(q)$ is only updated to the current number of misses of the

query when `Oracle` is called, so $T_t^{(r)}(q)$ is monotonically non-decreasing and $T_t^{(r)}(q) \leq T_t^{(c)}(q)$. Since `Oracle` is not called during the interval from $t_i + 1$ to $t_{i+1} - 1$, according to the cumulative update policy, we have

$$T_{t_{i+1}-1}^{(c)}(q) < (1+\alpha)T_{t_{i+1}-1}^{(r)}(q) = (1+\alpha)T_{t_i}^{(r)}(q) \leq (1+\alpha)T_{t_i}^{(c)}(q) \tag{119}$$

Then

$$\mathbb{I}\{q \notin \widehat{\mathcal{M}}_{t_i}^{(full)}\} \sum_{n=1}^{t_{i+1}-t_i-1} \mathbb{I}\{q_n = q\} \leq \alpha T_{t_i}^{(c)}(q)$$

Plugging this into above,

$$\mathbb{I}\{q \notin \widehat{\mathcal{M}}_{t_i}^{(full)}\}(t_{i+1} - t_i - 1)P(q)\sqrt{\frac{1}{T_{t_i}^{(c)}(q)}} \tag{120}$$

$$\leq 8\ln(4TN/\delta) + 2\alpha\sqrt{T_{t_i}^{(c)}(q)} \tag{121}$$

So

$$A_1 \leq 4c_2\sqrt{2\ln(8TN/\delta)} \sum_{q\in\mathcal{M}^*}\sum_{i=1}^{N_1}\left(8\ln(4TN/\delta) + 2\alpha\sqrt{T_{t_i}^{(c)}(q)}\right) \cdot \mathbb{I}\{T_{t_i}^{(c)}(q) \leq l_T^{(1)}(q, Z_q)\} \tag{122}$$

$$\leq 4N_1c_2\sqrt{2\ln(8TN/\delta)} \sum_{q\in\mathcal{M}^*}\left(8\ln(4TN/\delta) + 2\alpha\sqrt{l_T^{(1)}(q, Z_q)}\right) \tag{123}$$

$$\leq 64c_2N_1l_{max}\ln^{3/2}(8TN/\delta) + 256c_2^2\alpha N_1\ln(8TN/\delta)\sum_{q\in\mathcal{M}^*}\frac{P(q)l(M,\mathcal{Q})}{Z_q} \tag{124}$$

$$\leq 64c_2Nl_{max}\ln^{5/2}(8TN/\delta) + 256c_2^2\alpha\ln^2(8TN/\delta)\sum_{q\in\mathcal{M}^*}\frac{P(q)Nl(M,\mathcal{Q})}{Z_q} \tag{125}$$

For term A_2: In our algorithm, T_t denotes the moment of the most recent `Oracle` call triggered by time accumulation prior to time t. Like $T_t^{(r)}(q)$, T_t is also only updated to the current time when `Oracle` is called, T_t is monotonically non-decreasing and $T_t < t$. During $t_i + 1 \sim t_{i+1} - 1$, according to our `Oracle` invoking policy:

$$t_{i+1} - 1 < (1+\alpha)T_{t_{i+1}-1} = (1+\alpha)T_{t_i} \leq (1+\alpha)t_i \tag{126}$$

So,

$$(t_{i+1} - t_i - 1)\sqrt{\frac{P(q)}{t_i}} \leq \alpha t_i\sqrt{\frac{P(q)}{t_i}} \leq \alpha\sqrt{P(q)t_i} \tag{127}$$

Thus,

$$A_2 \leq 8c_2\alpha\sqrt{\ln(16TN/\delta)}\sum_{q\in\mathcal{M}^*}\sum_{i=1}^{N_1}\sqrt{P(q)t_i} \cdot \mathbb{I}\{t_i \leq l_T^{(2)}(q, Z_q)\} \tag{128}$$

$$\leq 8c_2\alpha N_1\sqrt{\ln(16TN/\delta)}\sum_{q\in\mathcal{M}^*}\sqrt{P(q)l_T^{(2)}(q, Z_q)} \tag{129}$$

$$\leq 128c_2^2\alpha N_1\ln(16TN/\delta)\sum_{q\in\mathcal{M}^*}\frac{P(q)l(M,\mathcal{Q})}{Z_q} \tag{130}$$

$$\leq 128c_2^2\alpha\ln^2(16TN/\delta)\sum_{q\in\mathcal{M}^*}\frac{P(q)Nl(M,\mathcal{Q})}{Z_q} \tag{131}$$

For term A_3: Similar to the analysis for term A_2,

$$A_3 \leq 16l(M,\mathcal{Q})\ln(16TN/\delta)\sum_{i=1}^{N_1}\alpha \tag{132}$$
$$\leq 16\alpha N_1 l(M,\mathcal{Q})\ln(16TN/\delta) \tag{133}$$
$$\leq 16\alpha N l(M,\mathcal{Q})\ln^2(16TN/\delta) \tag{134}$$

Combining these together,

$$\mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}|E,\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\}\right] \tag{135}$$
$$\leq A_1+A_2+A_3 \tag{136}$$
$$\leq 64c_2Nl_{max}\ln^{5/2}(8TN/\delta)+256c_2^2\alpha\ln^2(8TN/\delta)\sum_{q\in\mathcal{M}^*}\frac{P(q)Nl(M,\mathcal{Q})}{Z_q} \tag{137}$$
$$+128c_2^2\alpha\ln^2(16TN/\delta)\sum_{q\in\mathcal{M}^*}\frac{P(q)Nl(M,\mathcal{Q})}{Z_q} \tag{138}$$
$$+16\alpha Nl(M,\mathcal{Q})\ln^2(16TN/\delta) \tag{139}$$
$$\leq 64(c_2+\alpha)Nl_{max}\ln^{5/2}(16TN/\delta)+256c_2^2\alpha\ln^2(8TN/\delta)\sum_{q\in\mathcal{M}^*}\frac{P(q)Nl(M,\mathcal{Q})}{Z_q} \tag{140}$$

For problem-dependent bound, take $Z_q=\Delta_q,\forall q\in\mathcal{Q}$. For $\forall q\notin\widehat{\mathcal{M}}_{t_i}^{(full)}$, according to the gap definition, we have $\Delta_q\leq\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}$, then $\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}<Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\}=0$. Plugging this into regret, we have

$$Reg(T)\leq\mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}|E,\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\}\right] \tag{141}$$
$$+T\delta+4c_2N^2\ln^2(4NT/\delta)+T\max_{t_i}\{Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\}\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}<Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\} \tag{142}$$
$$\leq 256c_2^2\alpha\ln^2(8TN/\delta)\sum_{q\in\mathcal{M}^*}\frac{P(q)Nl(M,\mathcal{Q})}{\Delta_q}+64(c_2+\alpha)N^2\ln^{5/2}(16TN/\delta)+Tc_2\delta \tag{143}$$

By taking $\delta=1/T$, we have our problem-dependent bound:

$$Reg(T)\leq\widetilde{O}\left(\sum_{q\in\mathcal{M}^*}\frac{P(q)Nl(M,\mathcal{Q})}{\Delta_q}\right) \tag{144}$$

For problem-independent bound, take $Z_q=\sqrt{\frac{16c_2Nl(M,\mathcal{Q})}{T}}\ln(8TN/\delta),\forall q\in\mathcal{Q}$, then $Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}=$

$\sqrt{\frac{16c_2Nl(M,\mathcal{Q})}{T}}\ln(8TN/\delta), \forall \widehat{\mathcal{M}}_{t_i}^{(full)} \in \mathfrak{V}$, and regret:

$$Reg(T) \leq \mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1}-t_i-1)\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}|E, \mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\}\right] \tag{145}$$

$$+ T\delta + 4c_2N^2\ln^2(4NT/\delta) + T\max_{t_i}\{Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\}\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} < Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\} \tag{146}$$

$$\leq 16c_2\alpha\ln(8TN/\delta)\sqrt{Nl(M,\mathcal{Q})T}\sum_{q\in\mathcal{M}^*}P(q) + 64(c_2+\alpha)N^2\ln^{5/2}(16TN/\delta) + Tc_2\delta \tag{147}$$

$$+ TZ_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \tag{148}$$

$$\leq 32c_2\alpha\ln(8TN/\delta)\sqrt{Nl(M,\mathcal{Q})T} + 64(c_2+\alpha)N^2\ln^{5/2}(16TN/\delta) + Tc_2\delta \tag{149}$$

The last inequality is because $\sum_{q\in\mathcal{M}^*}P(q) \leq 1$. Thus, take $\delta = 1/T$, we have our problem-independent bound:

$$Reg(T) \leq \widetilde{O}\left(\sqrt{Nl(M,\mathcal{Q})T}\right) \tag{150}$$

C Proof of Theorem A.1

Definition C.1. ***(Approximate regret)****: If the solver in* `Oracle` *is β-approximation, then we define the β-approximation regret as:*

$$Reg_\beta(T) = \sum_{t=1}^{T}\mathbb{E}[\sum_{q\in\mathcal{M}_t^c}P(q)C^*(q) - (1+\beta)\sum_{q\in\mathcal{M}^{*c}}P(q)C^*(q)]$$

where $q \in \mathcal{M}^c$ means that $q \in \mathcal{Q}\setminus\mathcal{M}$, or say $q \notin \mathcal{M}$.

The proof of Theorem 2 is essentially the same as that of Theorem 1, with the only difference being the relaxation applied when summing over the queries. We still adopt the lemmas and definitions used in the proof of Theorem 1, nn this basis, we supplement some lemmas.

Lemma C.1. ***(Complementary cache number)****:For any two full caches $\mathcal{M}_1$, we have $|\mathcal{M}_1^c| \leq N - l_{min}$.*

The proof of Lemma C.1 is obviously.

Lemma C.2. ***(Complementary full regret equal)****: Assuming that* `Oracle` *is called at t_i, and is not called during $t_i+1, t_{i+1}-1$, if the event E_4 holds:*

$$\sum_{t=t_i+1}^{t_{i+1}-1}\mathbb{E}[\sum_{q\in\mathcal{M}_t^c}P(q)C^*(q) - (1+\beta)\sum_{q\in\mathcal{M}^{*c}}P(q)C^*(q)]$$
$$\leq (t_{i+1}-t_i-1)\mathbb{E}[\sum_{q\in\widehat{\mathcal{M}}_{t_i}^{(full)c}}P(q)C^*(q) - (1+\beta)\sum_{q\in\mathcal{M}^{*c}}P(q)C^*(q)] + 2c_2N\ln(4NT/\delta)$$

Proof.

$$\sum_{t=t_i+1}^{t_{i+1}-1}\mathbb{E}[\sum_{q\in\mathcal{M}_t^c}P(q)C^*(q) - (1+\beta)\sum_{q\in\mathcal{M}^{*c}}P(q)C^*(q)] \tag{151}$$

$$= \sum_{q\in\mathcal{Q}}\sum_{t=t_i+1}^{t_{i+1}-1}\mathbb{E}[P(q)C^*(q)(\mathbb{I}\{q\notin\mathcal{M}_t\} - (1+\beta)\mathbb{I}\{q\notin\mathcal{M}^*\})] \tag{152}$$

$$= \sum_{q\in\mathcal{Q}}\sum_{t=t_i+1}^{t_{i+1}-1}\mathbb{E}[P(q)C^*(q)(\mathbb{I}\{q\in\mathcal{M}^*\} - \mathbb{I}\{q\in\mathcal{M}_t\})] - \beta\sum_{q\in\mathcal{Q}}\sum_{t=t_i+1}^{t_{i+1}-1}P(q)C^*(q)\mathbb{I}\{q\notin\mathcal{M}^*\} \tag{153}$$

According to Lemma B.10,

$$\sum_{q\in\mathcal{Q}}\sum_{t=t_i+1}^{t_{i+1}-1}\mathbb{E}[P(q)C^*(q)(\mathbb{I}\{q\in\mathcal{M}^*\}-\mathbb{I}\{q\in\mathcal{M}_t\})] \tag{154}$$

$$\leq (t_{i+1}-t_i-1)\mathbb{E}[\sum_{q\in\mathcal{M}^*}P(q)C^*(q)-\sum_{q\in\widehat{\mathcal{M}}_{t_i}^{(full)}}P(q)C^*(q)]+2c_2N\ln(4NT/\delta) \tag{155}$$

$$=(t_{i+1}-t_i-1)\mathbb{E}[\sum_{q\in\widehat{\mathcal{M}}_{t_i}^{(full)c}}P(q)C^*(q)-\sum_{q\in\mathcal{M}^{*c}}P(q)C^*(q)]+2c_2N\ln(4NT/\delta) \tag{156}$$

Then

$$\sum_{t=t_i+1}^{t_{i+1}-1}\mathbb{E}[\sum_{q\in\mathcal{M}_t^c}P(q)C^*(q)-(1+\beta)\sum_{q\in\mathcal{M}^{*c}}P(q)C^*(q)] \tag{157}$$

$$\leq (t_{i+1}-t_i-1)\mathbb{E}[\sum_{q\in\widehat{\mathcal{M}}_{t_i}^{(full)c}}P(q)C^*(q)-\sum_{q\in\mathcal{M}^{*c}}P(q)C^*(q)]-\beta\sum_{q\in\mathcal{M}^{*c}}\sum_{t=t_i+1}^{t_{i+1}-1}P(q)C^*(q)+2c_2N\ln(4NT/\delta) \tag{158}$$

$$=(t_{i+1}-t_i-1)\mathbb{E}[\sum_{q\in\widehat{\mathcal{M}}_{t_i}^{(full)c}}P(q)C^*(q)-(1+\beta)\sum_{q\in\mathcal{M}^{*c}}P(q)C^*(q)]+2c_2N\ln(4NT/\delta) \tag{159}$$

□

Lemma C.3. ***(Complementary full cache compare)****: If good event E holds, assuming that* `Oracle` *with a β-approximation solver is called at t, and is not called again afterwards, then:*

$$\sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}_t^{(full)}}P(q)C^*(q)-(1+\beta)\sum_{q\in\mathcal{M}^{*c}}P(q)C^*(q) \tag{160}$$

$$\leq \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}_t^{(full)}}\Big(P(q)C^*(q)-\hat{P}_t(q)\hat{C}_t(q)\Big) \tag{161}$$

Proof. Because solver is β-approximation for min-knapsack problem, we have

$$\sum_{q\in\widehat{\mathcal{M}}_t^{(full)c}}\hat{P}_t(q)\hat{C}_t(q)\leq(1+\beta)\sum_{q\in\widehat{\mathcal{M}}_t^{(ground)c}}\hat{P}_t(q)\hat{C}_t(q) \tag{162}$$

where $\mathcal{M}_t^{(ground)c}$ is the optimal solution for min knapsack problem with value of $\hat{P}_t(q)\hat{C}_t(q)$, then

$$\sum_{q\in\widehat{\mathcal{M}}_t^{(ground)c}}\hat{P}_t(q)\hat{C}_t(q)\leq\sum_{q\in\widehat{\mathcal{M}}^{*c}}\hat{P}_t(q)\hat{C}_t(q) \tag{163}$$

So,

$$\sum_{q\in\widehat{\mathcal{M}}_t^{(full)c}}\hat{P}_t(q)\hat{C}_t(q)\leq(1+\beta)\sum_{q\in\widehat{\mathcal{M}}^{*c}}\hat{P}_t(q)\hat{C}_t(q) \tag{164}$$

Thus,

$$\sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}_t^{(full)}}P(q)C^*(q)-(1+\beta)\sum_{q\in\mathcal{M}^{*c}}P(q)C^*(q) \tag{165}$$

$$\leq \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}_t^{(full)}}\Big(P(q)C^*(q)-\hat{P}_t(q)\hat{C}_t(q)\Big) \tag{166}$$

□

Lemma C.4. ***(Approximate gap bound)****: At any moment t_i when an* `Oracle` *call occurs, if E holds and the approximate gap satisfies $\Delta^c_{\widehat{\mathcal{M}}^{(full)}_{t_i}} \geq Z_{\widehat{\mathcal{M}}^{(full)}_{t_i}}$, then*

$$\Delta^c_{\widehat{\mathcal{M}}^{(full)}_{t_i}} \tag{167}$$

$$\leq 4c_2\sqrt{2\ln(8TN/\delta)} \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} P(q)\sqrt{\frac{1}{T^{(c)}_{t_i}(q)}} \cdot \mathbb{I}\{T^{(c)}_{t_i}(q) \leq l^{(3)}_T(q, Z_q)\} \tag{168}$$

$$+ 8c_2\sqrt{\ln(16TN/\delta)} \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} \sqrt{\frac{P(q)}{t_i}} \cdot \mathbb{I}\{t_i \leq l^{(4)}_T(q, Z_q)\} + \frac{16(N-l_{min})\ln(16TN/\delta)}{t_i} \tag{169}$$

where $l^{(3)}_T(q, Z_q) = \frac{128P^2(q)(N-l_{min})^2c_2^2\ln(8TN/\delta)}{Z_q^2}$, $l^{(4)}_T(q, Z_q) = \frac{256c_2^2P(q)(N-l_{min})^2\ln(16TN/\delta)}{Z_q^2}$.

Proof. According to Lemma C.3, we have

$$\Delta_{\widehat{\mathcal{M}}^{(full)}_{t_i}} \leq \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} \Big(P(q)C^*(q) - \hat{P}_{t_i}(q)\hat{C}_{t_i}(q)\Big) \tag{170}$$

On the other hand

$$\sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} \Big(P(q)C^*(q) - \hat{P}_{t_i}(q)\hat{C}_{t_i}(q)\Big) \tag{171}$$

$$= \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} \Big(P(q)(C^*(q) - \hat{C}_{t_i}(q)) + \hat{C}_{t_i}(q)(P(q) - \hat{P}_{t_i}(q))\Big) \tag{172}$$

When good event E holds and $\mathbb{I}\left\{\Delta_{\widehat{\mathcal{M}}^{(full)}_{t_i}} \geq Z_{\widehat{\mathcal{M}}^{(full)}_{t_i}}\right\}$, then at time t_i:

$$\Delta_{\widehat{\mathcal{M}}^{(full)}_{t_i}} \tag{173}$$

$$\leq 2 \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} \Big(P(q)(C^*(q) - \hat{C}_{t_i}(q)) + \hat{C}_{t_i}(q)(P(q) - \hat{P}_{t_i}(q))\Big) - Z_{\widehat{\mathcal{M}}^{(full)}_{t_i}} \tag{174}$$

$$\leq 2 \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} \Big(P(q)(C^*(q) - \hat{C}_{t_i}(q)) + c_2(P(q) - \hat{P}_{t_i}(q))\Big) - Z_{\widehat{\mathcal{M}}^{(full)}_{t_i}} \tag{175}$$

$$\leq 2 \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} (P(q)|C^*(q) - \hat{C}_{t_i}(q)| - \frac{Z_{\widehat{\mathcal{M}}^{(full)}_{t_i}}}{4|\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}|}) \tag{176}$$

$$+ 2c_2 \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} (|P(q) - \hat{P}_{t_i}(q)| - \frac{Z_{\widehat{\mathcal{M}}^{(full)}_{t_i}}}{4c_2|\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}|}) \tag{177}$$

$$\leq 2 \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} (P(q)|C^*(q) - \hat{C}_{t_i}(q)| - \frac{Z_q}{4(N-l_{min})}) \tag{178}$$

$$+ 2c_2 \sum_{q\in\mathcal{Q}\setminus\widehat{\mathcal{M}}^{(full)}_{t_i}} (|P(q) - \hat{P}_{t_i}(q)| - \frac{Z_q}{4c_2(N-l_{min})}) \tag{179}$$

Eq. (175) is because when E holds, $\hat{C}_{t_i}(q) \leq C^*(q) \leq c_2$. The last inequality is because $Z_{\widehat{\mathcal{M}}^{(full)}_{t_i}} \geq Z_q, \forall q \notin \widehat{\mathcal{M}}^{(full)}_{t_i}$ and Lemma C.1.

For the term in Eq. (178): By good event E, we have

$$P(q)|C^*(q) - \hat{C}_{t_i}(q)| - \frac{Z_q}{4(N - l_{min})} \tag{180}$$

$$\leq P(q)\min\{c_2, 2(c_2 - c_1)\sqrt{\frac{2\ln(8TN/\delta)}{T_{t_i}^{(c)}(q)}}\} - \frac{Z_q}{4(N - l_{min})} \tag{181}$$

$$\leq 2c_2 P(q)\sqrt{\frac{2\ln(8TN/\delta)}{T_{t_i}^{(c)}(q)}} - \frac{Z_q}{4l(M, \mathcal{Q})} \tag{182}$$

$$= 2c_2 P(q)\sqrt{2\ln(8TN/\delta)}\left(\frac{1}{\sqrt{T_{t_i}^{(c)}(q)}} - \frac{Z_q}{8c_2(N - l_{min})P(q)\sqrt{2\ln(8TN/\delta)}}\right) \tag{183}$$

When $T_{t_i}^{(c)}(q) > \frac{128P^2(q)(N-l_{min})^2 c_2^2 \ln(8TN/\delta)}{Z_q^2} \doteq l_T^{(3)}(q, Z_q)$,

$$\frac{1}{\sqrt{T_{t_i}^{(c)}(q)}} - \frac{Z_q}{8c_2(N - l_{min})P(q)\sqrt{2\ln(8TN/\delta)}} < 0 \tag{184}$$

When $T_{t_i}^{(c)}(q) \leq l_T^{(3)}(q, Z_q)$,

$$0 \leq \frac{1}{\sqrt{T_{t_i}^{(c)}(q)}} - \frac{Z_q}{8c_2(N - l_{min})P(q)\sqrt{2\ln(8TN/\delta)}} < \frac{1}{\sqrt{T_{t_i}^{(c)}(q)}} \tag{185}$$

Plugging these into above,

$$P(q)|C^*(q) - \hat{C}_{t_i}(q)| - \frac{Z_q}{4(N - l_{min})} \tag{186}$$

$$\leq 2c_2 P(q)\sqrt{\frac{2\ln(8TN/\delta)}{T_{t_i}^{(c)}(q)}} \cdot \mathbb{I}\{T_{t_i}^{(c)}(q) \leq l_T^{(3)}(q, Z_q)\} \tag{187}$$

For the term in Eq. (179): By E and Lemma B.3,

$$|P(q) - \hat{P}_{t_i}(q)| - \frac{Z_q}{4c_2(N - l_{min})} \tag{188}$$

$$\leq 2\min\{1, 2\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} + \frac{8\ln(16TN/\delta)}{t_i}\} - \frac{Z_q}{4c_2(N - l_{min})} \tag{189}$$

$$\leq 4\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} - \frac{Z_q}{4c_2(N - l_{min})} + \frac{16\ln(16TN/\delta)}{t_i} \tag{190}$$

When $t_i > \frac{256c_2^2 P(q)(N-l_{min})^2 \ln(16TN/\delta)}{Z_q^2} \doteq l_T^{(4)}(q, Z_q)$,

$$4\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} - \frac{Z_q}{4c_2(N - l_{min})} < 0 \tag{191}$$

When $t_i \leq l_T^{(4)}(q, Z_q)$,

$$0 \leq 4\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} - \frac{Z_q}{4c_2(N - l_{min})} < 4\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} \tag{192}$$

Combining these, we have

$$|P(q) - \hat{P}_{t_i}(q)| - \frac{Z_q}{4c_2(N - l_{min})} \tag{193}$$

$$\leq 4\sqrt{\frac{P(q)\ln(16TN/\delta)}{t_i}} \cdot \mathbb{I}\{t_i \leq l_T^{(4)}(q, Z_q)\} + \frac{16\ln(16TN/\delta)}{t_i} \tag{194}$$

Combining the inequalities above together, we finish our proof.

□

C.1 Approximate Regret

Then we define approximation gap of $\widehat{\mathcal{M}}_t^{(full)}$ as $\Delta^c_{\widehat{\mathcal{M}}_t^{(full)}} = \sum_{q \in \widehat{\mathcal{M}}_t^{(full)c}} P(q)C^*(q) - (1 + \beta)\sum_{q \in \mathcal{M}^{*c}} P(q)C^*(q)$. We assume that for each query q there is the constant $Z_q, q \in \mathcal{Q}$ and for each valid set $\mathcal{M} \in \mathfrak{V}$ we define $Z_{\mathcal{M}} = \max_{q \notin \mathcal{M}} Z_q$.

For approximate `Oracle`, if E holds:

$$Reg_\beta(T) \leq \sum_{i=1}^{N_1} \sum_{t=t_i+1}^{t_{i+1}-1} \mathbb{E}[\sum_{q \in \mathcal{M}_t^c} P(q)C^*(q) - (1+\beta)\sum_{q \in \mathcal{M}^{*c}} P(q)C^*(q)] \tag{195}$$

$$+ \sum_{i=1}^{N_1+1} \mathbb{E}[\sum_{q \in \mathcal{M}_{t_i}^c} P(q)C^*(q) - (1+\beta)\sum_{q \in \mathcal{M}^{*c}} P(q)C^*(q)] + Tc_2\delta \tag{196}$$

$$\leq \mathbb{E}[\sum_{i=1}^{N_1+1} (t_{i+1} - t_i - 1)\Delta^c_{\widehat{\mathcal{M}}_t^{(full)}}] + 4c_2 N N_1 \ln(4NT/\delta) + Tc_2\delta \tag{197}$$

The last inequality is because Lemma C.2.

Inserting the conclusion of Lemma C.4, the subsequent proof process analysis is the same as in Theorem 1. As a result,

$$\mathbb{E}\left[\sum_{i=1}^{N_1}(t_{i+1} - t_i - 1)\Delta^c_{\widehat{\mathcal{M}}_{t_i}^{(full)}} | E, \mathbb{I}\left\{\Delta^c_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\}\right] \tag{198}$$

$$= 4c_2\sqrt{2\ln(8TN/\delta)}\sum_{q \in \mathcal{Q}}\sum_{i=1}^{N_1}\mathbb{I}\{q \notin \widehat{\mathcal{M}}_{t_i}^{(full)}\}(t_{i+1} - t_i - 1)P(q)\sqrt{\frac{1}{T_{t_i}^{(c)}(q)}} \cdot \mathbb{I}\{T_{t_i}^{(c)}(q) \leq l_T^{(3)}(q, Z_q)\}$$

$$+ 8c_2\sqrt{\ln(16TN/\delta)}\sum_{q \in \mathcal{Q}}\sum_{i=1}^{N_1}\mathbb{I}\{q \notin \widehat{\mathcal{M}}_{t_i}^{(full)}\}(t_{i+1} - t_i - 1)\sqrt{\frac{P(q)}{t_i}} \cdot \mathbb{I}\{t_i \leq l_T^{(4)}(q, Z_q)\}$$

$$+ 16(N - l_{min})\ln(16TN/\delta)\sum_{i=1}^{N_1}\frac{t_{i+1} - t_i - 1}{t_i}$$

$$\leq 64c_2N^2\ln^{5/2}(8TN/\delta) + 256c_2^2\alpha\ln^2(8TN/\delta)\sum_{q \in \mathcal{Q}}\frac{P(q)N(N - l_{min})}{Z_q} \tag{199}$$

$$+ 128c_2^2\alpha\ln^2(16TN/\delta)\sum_{q \in \mathcal{Q}}\frac{P(q)N(N - l_{min})}{Z_q} \tag{200}$$

$$+ 16\alpha N(N - l_{min})\ln^2(16TN/\delta) \tag{201}$$

$$\leq 384c_2^2\alpha\ln^2(16TN/\delta)\sum_{q \in \mathcal{Q}}\frac{P(q)N(N - l_{min})}{Z_q} + 64(c_2 + \alpha)N^2\ln^{5/2}(16TN/\delta) \tag{202}$$

Therefore,

$$Reg_\beta(T) \leq 384c_2^2\alpha \ln^2(16TN/\delta) \sum_{q\in\mathcal{Q}} \frac{P(q)N(N-l_{min})}{Z_q} + 128(c_2+\alpha)N^2 \ln^{5/2}(16TN/\delta) \quad (203)$$

$$+ Tc_2\delta + T \max_{t_i}\{Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\}\mathbb{I}\left\{\Delta^c_{\widehat{\mathcal{M}}_{t_i}^{(full)}} < Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\} \quad (204)$$

For problem-dependent bound, take $Z_q = \Delta_q^c, \forall q \in \mathcal{Q}$. For $\forall q \notin \widehat{\mathcal{M}}_{t_i}^{(full)}$, according to the gap definition, we have $\Delta_q^c \leq \Delta^c_{\widehat{\mathcal{M}}_{t_i}^{(full)}}$, then $\mathbb{I}\left\{\Delta^c_{\widehat{\mathcal{M}}_{t_i}^{(full)}} < Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\right\} = 0$. Plugging this into regret, we have

$$Reg(T) \leq 384c_2^2\alpha \ln^2(16TN/\delta) \sum_{q\in\mathcal{Q}} \frac{P(q)N(N-l_{min})}{\Delta_q^c} + 128(c_2+\alpha)N^2 \ln^{5/2}(16TN/\delta) \quad (205)$$

$$+ Tc_2\delta \quad (206)$$

By taking $\delta = 1/T$, we have our problem-dependent bound:

$$Reg(T) \leq \widetilde{O}\left(\sum_{q\in\mathcal{Q}} \frac{P(q)N(N-l_{min})}{\Delta_q^c}\right) \quad (207)$$

For problem-independent bound, take $Z_q = \sqrt{\frac{16c_2N(N-l_{min})}{T}} \ln(16TN/\delta), \forall q \in \mathcal{Q}$, then $Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} = \sqrt{\frac{16c_2N(N-l_{min})}{T}} \ln(16TN/\delta), \forall \widehat{\mathcal{M}}_{t_i}^{(full)} \in \mathfrak{V}$, and regret:

$$Reg(T) \leq 20c_2\alpha \ln(16TN/\delta)\sqrt{N(N-l_{min})T} \sum_{q\in\mathcal{M}^*} P(q) + 128(c_2+\alpha)N^2 \ln^{5/2}(16TN/\delta) + Tc_2\delta \quad (208)$$

$$+ TZ_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \quad (209)$$

$$\leq 40c_2\alpha \ln(8TN/\delta)\sqrt{N(N-l_{min})T} + 128(c_2+\alpha)N^2 \ln^{5/2}(16TN/\delta) + Tc_2\delta \quad (210)$$

The last inequality is because $\sum_{q\in\mathcal{Q}} P(q) = 1$. Thus, taking $\delta = 1/T$, we have our problem-independent bound:

$$Reg(T) \leq \widetilde{O}\left(\sqrt{N(N-l_{min})T}\right) \quad (211)$$