

LLM Cache Bandits with Query Heterogeneity for Cost-Efficient Inference

Hantao Yang¹, Hong Xie^{1*}, Xutong Liu², Defu Lian¹

¹University of Science and Technology of China, State Key Laboratory of Cognitive Intelligence

²University of Washington, Tacoma

yanghantao@mail.ustc.edu.cn, hongx87@ustc.edu.cn, xutongl@uw.edu, liandefu@ustc.edu.cn

Abstract

This paper revisits the LLM cache bandit problem, with a special focus on addressing query heterogeneity for cost-effective LLM inference. Previous works often assume uniform query sizes. Heterogeneous query sizes introduce a combinatorial structure into 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 our 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. \(2023\)](#), 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. Experiments 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) points out that repetition implies computation is redundantly applied to similar outputs. This lack of diversity results in the 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 [Stogianidis 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 costs 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](#)) optimizes internal data flow in LLMs to reduce delays and resource consumption during inference. GPTcache ([Bang, 2023](#)) is known for its simplicity and flexibility, and SCALM ([Li et al., 2024](#)) improves cache efficiency by leveraging the semantic understanding of queries. [Zhu et al. \(2023\)](#) 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 techniques into LLMs poses challenges due to the variability and unpredictability of query costs. To address this issue, [Zhu et al.](#)

*Corresponding Author

Table 1: Summary of the main results

Previous work	Heterogeneous query size	Problem-dependent regret	Problem-independent regret
Zhu et al. (2023)	$\times$	$\times$	$\tilde{O}(MN\sqrt{T})$
Our Work	Heterogeneous query size	Problem-dependent regret	Problem-independent regret
VSOCB (Algorithm 1)	$\checkmark$	$\tilde{O}\left(\sum_{q \in \mathcal{M}^*} \frac{P(q)MN}{\Delta_q}\right)$ (Theorem 4.1)	$\tilde{O}(\sqrt{MNT})$ (Theorem 4.1)
VSOCB-APX(Appendix A)	$\checkmark$	$\tilde{O}\left(\sum_{q \in \mathcal{Q}} \frac{P(q)N(N-l_{\min})}{\Delta_q^c}\right)$	$\tilde{O}(\sqrt{N(N-l_{\min})T})$

$\mathcal{Q}$ 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., 2023) 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., 2023) the coefficient of $\sqrt{MNT}$.

(2023) suggests using both expected cost and popularity as criteria in cache selection in their on-line 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 a 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. (2023) briefly discusses but leaves open for further improvement.

To tackle this open problem, we revisit the framework of Zhu et al. (2023) 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 an incoming query.

In this paper, we propose an online algorithm—**Variable Size Online Cache Bandits** (VSOCB), an extension variant of Zhu et al. (2023). 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 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 this 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 a 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 a 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 analytical challenges. Our main theoretical contributions are as follows:

1) *Regret bound and Oracle calls:* We prove that our algorithm achieves $\tilde{O}(\sqrt{MNT})$ 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., 2023), is shown to be a special case of our approach. In particular, Zhu et al. (2023) provided a regret bound of $\tilde{O}(MN\sqrt{T})$ and established 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 with respect to T .

3) *Problem-dependent regret bound:* We provide a problem-dependent regret bound of $\tilde{O}\left(\sum_{q \in \mathcal{M}^*} \frac{P(q)MN}{\Delta_q}\right)$, 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 results 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 LLMs, 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, aiming 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) leveraged semantic understanding of queries to improve cache hit rates through a semantic-oriented caching approach. Closer to our online-learning perspective, Liu et al. (2026) formulated semantic cache eviction with an explicit mismatch cost between incoming queries and cached responses, and derived offline and online algorithms with provable guarantees when query arrival probabilities and serving costs are unknown; Atalar et al. (2026) extended this framework to a continuous query embedding space via ϵ -net discretization and kernel ridge regression. Both works, however, assume a cache holding a fixed number of equally sized entries, so the heterogeneous query sizes studied here remain unaddressed. Zhu et al. (2023) proposed 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 in 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 $\mathcal{Q} = \{q_1, \dots, q_N\}$ 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 of 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 a fixed sampling distribution. We use $P(q) \in (0, 1]$ to denote the sampling probability that query q is selected such that $\sum_{q \in \mathcal{Q}} P(q) = 1$. Each time the LLM processes q , it incurs 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 uncertainty 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 returned by LLM in response to query q . We maintain a cache $\mathcal{M}$ for the LLM system, storing a small subset of queries along with their corresponding results, that is, we store a question-answer pair $(q, r(q))$ in the 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 the cache. The size of cache has a maximum size M , and 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

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

To reduce overhead, we should place queries with high expected costs and high sampling frequencies into the cache. Following [Zhu et al. \(2023\)](#), we can define the expected cost of a given cache $\mathcal{M}$ is

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

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

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

Previous works such as [Zhu et al. \(2023\)](#) 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 works.

Oracle. From the formula for the optimal cache, we formulate the optimization 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 M , and the optimal cache is the optimal solution of this knapsack problem. Inspired by the CMAB model, we assume that the agent has access to an offline Oracle to guide cache selection decisions. The Oracle employs 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. Further details are provided in Appendix A.

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 , the 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 historical feedback information from previous rounds. After q_t arrives, the agent checks 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 the 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 saves a potential cost $C(q_t)$, which is unobserved by the agent. If query q_t *misses* 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 obtains the query’s size $S(q_t)$ at the same time.

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

Remark 3.2. *As noted in [Zhu et al. \(2023\)](#), 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 over the time horizon T . By applying the indicator function to the regret in [Zhu et al. \(2023\)](#) for an equivalent transformation, we get:

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

where $\mathcal{M}^* = \underset{\mathcal{M} \in \mathcal{J}}{\text{argmax}} \sum_{q \in \mathcal{M}} C^*(q)P(q)$.

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 does it have information about that query and can estimate its cost and sampling probability. Subsequently, when invoking *Oracle* or updating the cache, the agent takes 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 for this query (excluding $r(q)$) have been recorded by the 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, the *Oracle* is denoted by $\text{Oracle}(\mathcal{Q}', C^*, P, S, M)$, 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 outputs a set $\mathcal{M}$ that yields the

solution to knapsack problem with a volume constraint of cache size M . We assume that the *Oracle* has a knapsack problem solver (e.g. dynamic programming) that can output the exact optimal solution. We denote this solver as $\text{Knap}(\mathbf{w}, \mathbf{v}, m)$. By inputting the item values $\mathbf{w}$, volumes $\mathbf{v}$, and the knapsack capacity m into Knap , it will return the optimal knapsack solution. We encapsulate this solver as our *Oracle* in [Alg.2](#). The *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 . The *Oracle* can see global information, allowing it to provide global guidance on cache replacement.

Lower Confidence Bound. Similar to [Zhu et al. \(2023\)](#), 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 method 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 our 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 $\mathbb{I}\{q_s \notin \mathcal{M}_s, q_s = q\}$ denotes query q arrives in round s and misses the cache. Let $\tilde{C}_t(q) = \sum_{s=1}^t \mathbb{I}\{q_s \notin \mathcal{M}_s, q_s = q\} C_t(q)$ denote the accumulated cost information of query q up to the t -th round. The lower-confidence-bound based estimator for cost is:

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

Remark 4.1. We use the LCB method to prevent our cache policy from getting trapped in a suboptimal solution. For example, consider a query set $\mathcal{Q} = \{q_1, q_2\}$ with a cache size of 1. If the optimal cache is $\{q_1\}$, and the algorithm inserts q_2 into the cache, the algorithm will always replace the cached query with the one having the higher estimated value. In this case, only queries not in

the cache have their cost estimates updated due to a miss, while those in the cache retain their estimates. Under LCB, when q_1 experiences a miss, its estimate increases as the confidence interval narrows, eventually surpassing q_2 and leading to its replacement. In contrast, with traditional UCB (the optimism method), q_1 's estimate would decrease with each miss, always remaining lower than that of q_2 , and thus never being replaced. This situation is referred to as the algorithm getting trapped in a suboptimal solution.

Variance Confidence Bound. Different from Zhu et al. (2023), for the estimation of the sampling probability $P(q)$, we also use an LCB-type estimate rather than an empirical estimate. Additionally, the confidence bound for $\hat{P}_t(q)$ is derived from the variance bound in Liu et al. (2022). According to the work in Liu et al. (2022), 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:

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

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

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

Recommend and Wait Strategy. In the 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 is cleared. Because the cache needs to store $(q, r(q))$, cache selection can only access the queries currently in the cache and any newly arriving queries. As mentioned in Remark 3.3, the queries chosen for cache updates are not arbitrary. Although the 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 the Oracle due to the online constraint. For example, if a query in the 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 $r(q)$ is re-obtained before it can be added to the cache. Under this setting, we introduce a *recommended cache* denoted by $\hat{\mathcal{M}}_t$ to temporarily store the results output by our Oracle at the current time. When the Oracle is invoked, the policy does not update the cache immediately. Instead, it first receives a recommended cache $\hat{\mathcal{M}}_t$, and the cache retains only the portion that matches the recommended cache, while 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 the 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 (lines 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.

Calling the 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 the Oracle is substantial. Therefore, we need to minimize the number of Oracle calls as much as possible, while still minimizing the impact on 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 current number of cache misses $T_t^{(c)}(q)$ for a query exceeds $(1 + \alpha)T_t^{(r)}(q)$, where $\alpha > 0$ is a predetermined constant. This is equivalent to calling the Oracle only when the query has accu-

mulated 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 number of 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 the 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 that in CUCB (Chen et al., 2016), we define the valid set of caches.

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

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

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 $\Delta_{min} = \min_{q \in \mathcal{Q}} \Delta_q = \min_{\mathcal{M} \in \mathfrak{V}} \Delta_{\mathcal{M}}$.

Remark 4.2. 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 quantifies the performance degradation when the cache does not select this query.

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:

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

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:

```

1: for round  $t = 1, \dots, T$  do
2:    $q_t$  was received with size  $L(q_t)$ ,
    $\mathcal{Q}_t = \mathcal{Q}_{t-1} \cup \{q_t\}$ 
3:    $T_t^{(q)}(q_t) = T_{t-1}^{(q)}(q_t) + 1$ 
4:   if  $q_t \in \mathcal{M}_{t-1}$  then
5:     Output the response from cache .
6:      $\widetilde{C}_t(q_t) = \widetilde{C}_{t-1}(q_t)$ ,  $T_t^{(c)}(q_t) = T_{t-1}^{(c)}(q_t)$ 
7:   else
8:     Input  $q_t$  to LLM and receive
        $r(q_t)$ ,  $A(q_t)$ ,  $C_t(q_t)$ .
9:      $S(q_t) = L(q_t) + A(q_t)$  //record query size
10:     $\widetilde{C}_t(q_t) = \widetilde{C}_{t-1}(q_t) + C_t(q_t)$ ,  $T_t^{(c)}(q_t) =$ 
       $T_{t-1}^{(c)}(q_t) + 1$ 
11:   end if
12:   for all  $q \neq q_t$  do
13:      $T_t^{(q)}(q) = T_{t-1}^{(q)}(q)$ ,  $T_t^{(r)}(q) = T_{t-1}^{(r)}(q)$ ,
       $T_t^{(c)}(q) = T_{t-1}^{(c)}(q)$ ,  $\widetilde{C}_t(q) = \widetilde{C}_{t-1}(q)$ 
14:   end for
15:   compute  $\widehat{P}_t(q)$ ,  $\widehat{C}_t(q)$  with Eq. (2), (3), (4),
      $\forall q \in \mathcal{Q}_t$ 
16:   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
17:      $\mathcal{M}_t = \mathcal{M}_{t-1} \cup \{q_t\}$ ,  $\widehat{\mathcal{M}}_t = \widehat{\mathcal{M}}_{t-1} \cup \{q_t\}$ 
18:   else
19:      $\mathcal{M}_t = \mathcal{M}_{t-1}$ ,  $\widehat{\mathcal{M}}_t = \widehat{\mathcal{M}}_{t-1}$ 
20:   end if
21:   if  $T_t^{(c)}(q_t) \geq (1 + \alpha)T_{t-1}^{(r)}(q_t)$  or  $t \geq (1 + \alpha)T_{t-1}$ 
     then
22:      $T_t^{(r)}(q_t) = T_t^{(c)}(q_t)$ ,  $T_t = t$ 
23:      $\widehat{\mathcal{M}}_t = \text{Oracle}(\mathcal{Q}_t, \widehat{C}_t, \widehat{P}_t, S, M)$ ,  $\mathcal{M}_t =$ 
       $\widehat{\mathcal{M}}_t \cap \mathcal{M}_{t-1}$ 
24:   else
25:      $T_t^{(r)}(q_t) = T_{t-1}^{(r)}(q_t)$ ,  $T_t = T_{t-1}$ ,  $\widehat{\mathcal{M}}_t =$ 
       $\widehat{\mathcal{M}}_{t-1}$ 
26:   end if
27: end for
```

(2) and problem-independent regret bound:

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

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

Algorithm 2 Exact 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: $\mathcal{M} \leftarrow \text{Knapsack}(\{w(q)\}_{q \in \mathcal{Q}'}, \{S'(q)\}_{q \in \mathcal{Q}'}, M)$
 - 4: Output $\mathcal{M}$
-

This proof is deferred to the Appendix. In the proof, we adopt an analysis method similar to that of 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 $\tilde{O}(\sqrt{MNT})$ (since $l(M, \mathcal{Q}) \leq M$), which represents an improvement over Caching in Online Learning in Zhu et al. (2023) with the coefficient of $\sqrt{MN}$, where the conclusion of Zhu et al. (2023) was $\tilde{O}(MN\sqrt{T})$. On the other hand, our model extends that of Zhu et al. (2023), which can be regarded as a special case of our model. In that case, Zhu et al. (2023) proposed a lower bound of $\Omega(\sqrt{T})$. The problem-independent regret established in Theorem 4.1 is $\tilde{O}(\sqrt{T})$, indicating that our work provides an optimal solution with respect to T .

Although our algorithm requires an Oracle to assist in computation, our proof shows that the number of calls to the Oracle is $O(N \log T)$, which is logarithmic. This indicates that the additional computational cost caused by using the Oracle is not significant, and it effectively 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 guaran-

tees. 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 the Appendix A.

5 Experiment

5.1 Simulation Dataset

We compare the performance of our algorithms and the baseline using cumulative regret on a simulated dataset. For the baseline, Zhu et al. (2023) 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 60. The expected cost of each query is within the range $[1, 2]$. 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 with parameter 0.5. We employ a dynamic programming algorithm as our solver in the 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 suboptimal solutions. In contrast, our algorithm circumvents this issue by utilizing the Oracle for the knapsack problem.

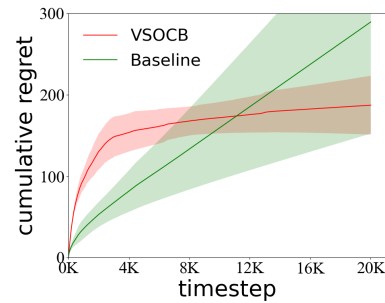


Figure 1: Synthetic dataset with 20000 rounds

5.2 Real Dataset

In line with Zhu et al. (2023), we evaluate our algorithms using the OpenAssistant (Köpf et al., 2023) dataset for the chat assistant task. For this

	VSOCB	VSOCB-apx	COL-GDSF	Trivial method
Regret	$479.3 \pm 97.6^*$	845.4 ± 136.5	2030.9 ± 940.5	$472.1 \pm 45.7^*$
Total Cost	$51.2K \pm 655.7$	$51.6K \pm 546.1$	$52.6K \pm 761.1$	$51.1K \pm 386.5^*$
Running Times(s)	247.6 ± 4.1	80 ± 3.9	$58.1 \pm 2.4^*$	12871.3 ± 275

Table 2: Comparison with Trivial method

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 the algorithms with 100 distinct queries in the online setting over a total of 10K and 20K rounds, with a cache size of 100. The other parameters are the same as above. As shown in Tab.3, our method reduces 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 obtains the costs and frequencies of each query, and then VSOCB outperform the baseline.

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

Table 3: Average cumulative cost on real dataset

5.3 Computational Overhead

In this section, we include a comparison of computational overhead to demonstrate the trade-off between computational cost and cache selection optimization in our algorithm. We introduce the following baselines for comparison:

1) VSOCB-apx: VSOCB using the approximate oracle (Appendix A), which is a greedy method for the MKP.

2) Trivial method: This is a simple method for handling heterogeneous queries: it invokes the exact Oracle to solve the knapsack problem at each round, which is clearly convergent. However, this naive approach incurs substantial additional computational overhead. The accumulation-based Oracle invoking policy in our algorithm is designed precisely to mitigate such extra overhead, thereby achieving a balance between computational cost and cache accuracy.

We compare the algorithm running time, regret, and total cost. Here, we compare with the Trivial method to demonstrate the advantages of our algorithm. We set $N = 100$, $M = 60$, $\alpha = 1$ and sample the query size from a uniform distribution over $[2, 20)$. The total number of query rounds is

set to $T = 40K$, and the results are averaged over 5 independent runs. Experimental results are shown in Tab.2.

As shown by Tab.2, the Trivial method achieves the most accurate cache selection and thus the lowest regret and total cost, but its computational overhead is enormous—its running time is nearly 50 times that of our algorithm. COL-GDSF runs the fastest, as it incurs no additional computational overhead; however, its cache accuracy is the lowest. Our method strikes a balance between regret and running time, achieving a relatively short computation time while ensuring regret convergence.

Additional experimental comparisons and results are provided in the Appendix. We have supplemented the baselines with LRU and LEC, and have also included experimental results for the approximate Oracle. Furthermore, we have added comparisons across multiple dimensions, including changes in M , N , α and query-size distributions.

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 the prior literature.

7 Limitations

We acknowledge that there is still room for further improvement. Our lower bound is directly adapted from Zhu et al. (2023), which only analyzes regret in terms of T and does not incorporate other important parameters, such as M and N . Moreover, due to technical complexity, our approximate Oracle relies on a minimum knapsack problem solver, which yields a looser regret bound compared to its exact counterpart and imposes certain limitations on practical deployment. In addition, given that our work primarily centers on algorithm design and theoretical analysis, the experimental results are intended mainly as a val-

idation of the rationality and effectiveness of our proposed algorithm, rather than as a comprehensive empirical study. As a result, the experiments conducted under real-world deployment scenarios remain somewhat preliminary and warrant further investigation. We plan to address these limitations and refine the relevant aspects in our future work.

8 Acknowledgement

The work of Hong Xie was supported by the National Natural Science Foundation of China under Grant 62476261.

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.
- Baran Atalar, Xutong Liu, Jinhang Zuo, Siwei Wang, Wei Chen, and Carlee Joe-Wong. 2026. Continuous semantic caching for low-cost llm serving. *arXiv preprint arXiv:2604.20021*.
- 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, Baran Atalar, Xiangxiang Dai, Jinhang Zuo, Siwei Wang, John CS Lui, Wei Chen, and Carlee Joe-Wong. 2026. Semantic caching for low-cost llm serving: From offline learning to online adaptation. In *IEEE INFOCOM 2026-IEEE Conference on Computer Communications*, pages 1–10. IEEE.
- Xutong Liu, Jinhang Zuo, Siwei Wang, Carlee Joe-Wong, John C.S. Lui, and Wei Chen. 2022. 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 Malakiotis, 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 meta-verse: 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, Yuan-dong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. [H₂O: 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. 2023. 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):

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

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 complementarity 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.

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: $\mathcal{M}^c \leftarrow Knapsack_{min}(\{w(q)\}_{q \in \mathcal{Q}'}, \{S'(q)\}_{q \in \mathcal{Q}'}, M)$
 - 4: Output $\mathcal{Q}' \setminus \mathcal{M}^c$
-

A.2 Approximate Solution Bound

Since the output in the case of an approximate solver is the complement of the cache, we need to 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 VSO CB (Alg.1) with approximate Oracle (Alg.3) holds:

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

(2) our problem-independent regret bound:

$$\text{Reg}_\beta(T) \leq \tilde{O} \left(\sqrt{N(N-l_{\min})T} \right),$$

(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

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)	$\Delta_{\mathcal{M}}^c = \sum_{q \in \mathcal{M}^c} P(q)C^*(q) - (1+\beta) \sum_{q \in \mathcal{M}^{*c}} P(q)C^*(q)$	$\Delta_q^c = \min_{\mathcal{M} \in \mathfrak{V}: \mathcal{M} \neq \mathcal{M}^*, q \notin \mathcal{M}} \Delta_{\mathcal{M}}^c$

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. $\mathcal{M}^c = \mathcal{Q} \setminus \mathcal{M}$.

Table 4: Gap compare

To complete our theoretical analysis, we have made some modifications to the traditional definition of the gap. Table 4 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

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

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 et al., 2023). 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.

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:

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

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

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

$$E_2 = E_{2,1} \cap E_{2,2} \quad (8)$$

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

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

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 $1 = t_1 < t_2, \dots, < t_{N_1} \leq t_{N_1+1} = T$, 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 $\mathcal{M}_1, \mathcal{M}_2 \in \mathfrak{V}$, we have $|\mathcal{M}_1 \setminus \mathcal{M}_2| \leq \min\{l_{\max}, N - l_{\min}\}$

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

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

$$\Pr \left(\left| C^*(q) - \frac{\tilde{C}_t(q)}{T_t^{(c)}(q)} \right| \geq (c_2 - c_1) \sqrt{\frac{\ln(8NT/\delta)}{2T_t^{(c)}(q)}} \right) \leq \frac{\delta}{4NT} \quad (11)$$

Then

$$Pr \left(0 \leq C^*(q) - \left(\frac{\tilde{C}_t(q)}{T_t^{(c)}(q)} - (c_2 - c_1) \sqrt{\frac{\ln(8NT/\delta)}{2T_t^{(c)}(q)}} \right) \leq 2(c_2 - c_1) \sqrt{\frac{\ln(8NT/\delta)}{2T_t^{(c)}(q)}} \right) \geq 1 - \frac{\delta}{4NT} \quad (12)$$

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

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

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

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., 2022).

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

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

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

Proof.

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

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

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

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

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

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

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

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

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

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

□

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., 2023).

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 $t_1(q), \dots, t_{N_1(q)}(q)$ to denote the moments when Oracle is called due to q satisfying the condition. We use $t_1(T), \dots, t_{N_1(T)}(T)$ 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

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

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,

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

So

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

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

Remark that our valid set was defined on the global query set $\mathcal{Q}$, but in the online setting, the record set $\mathcal{Q}_t$ may not have encountered all of queries when the Oracle is called. At this time, $\mathcal{Q}_t \neq \mathcal{Q}$, which may cause the recommended cache not to be 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 be multiple choices to make it full. According to the algorithm, the query that first fills the cache is selected as the one to be included in the full cache. We assume that there exists a seer that knows which query will come first to make the recommended cache full, thus fixing a full version of the recommended cache.

Definition B.2. (Full 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 :

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

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 . $\square$

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:

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

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

Proof.

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

Then, with E holds:

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

We have

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

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

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

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

$\square$

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:

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

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

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

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

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

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

$$(40)$$

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

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

$$= \underbrace{\sum_{t=t_i+1}^{t_i+g(t_i, q)} P(q)C^*(q) (\mathbb{I}\{q \in \mathcal{M}^*\} - \mathbb{I}\{q \in \mathcal{M}_t\})}_A + \underbrace{\sum_{t=t_i+g(t_i, q)+1}^{t_{i+1}-1} P(q)C^*(q) (\mathbb{I}\{q \in \mathcal{M}^*\} - \mathbb{I}\{q \in \mathcal{M}_t\})}_B \quad (42)$$

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

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

$$\leq g(t_i, q)P(q)C^*(q) \quad (44)$$

(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

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

Above all, we can get

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

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

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

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

So

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

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

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

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

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

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

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

□

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*

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

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

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

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

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

On the other hand

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

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

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 :

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

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

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

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

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

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

$$+ 2c_2 \sum_{q \in \mathcal{M}^* \setminus \widehat{\mathcal{M}}_{t_i}^{(full)}} \left(|P(q) - \hat{P}_{t_i}(q)| - \frac{Z_q}{4c_2l(M, \mathcal{Q})} \right) \tag{66}$$

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

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

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

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

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

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

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

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

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

Plugging these into above,

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

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

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

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

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

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

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

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

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

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

Combining these, we have

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

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

Thus

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

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

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

□

We can write the regret as follows.

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

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

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

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

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

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

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

Partitioning based on the moments of Oracle calls, we obtain

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

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

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

According to Lemma B.10, for any t_i :

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

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

And

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

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

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

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

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

$$\leq \mathbb{E} \left[\sum_{i=1}^{N_1} (t_{i+1} - t_i - 1) \left(\sum_{q \in \mathcal{M}^*} P(q) C^*(q) - \sum_{q \in \widehat{\mathcal{M}}_{t_i}^{(full)}} P(q) C^*(q) \right) \right] + 2c_2 N_1 N \ln(4NT/\delta) \quad (100)$$

$$+ (N_1 + 1) c_2 \quad (101)$$

$$\leq \mathbb{E} \left[\sum_{i=1}^{N_1} (t_{i+1} - t_i - 1) \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \right] + 4c_2 N_1 N \ln(4NT/\delta) \quad (102)$$

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

$$4c_2 N_1 N \ln(4NT/\delta) \leq 4c_2 N^2 \ln^2(4NT/\delta)$$

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

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

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

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

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

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

Combining above inequality, we have

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

$$+ T c_2 \delta + 4c_2 N^2 \ln^2(4NT/\delta) + T \max_{t_i} \{Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\} \mathbb{I} \left\{ \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} < Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \right\} \quad (109)$$

By Lemma B.11,

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

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

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

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

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

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

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

For term A_1 : According to Lemma B.4

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

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

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

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

$\sum_{n=1}^{t_{i+1} - t_i - 1} \mathbb{I} \{q_n = q\}$ 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

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

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

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

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,

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

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

So

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

$$\leq 4N_1 c_2 \sqrt{2 \ln(8TN/\delta)} \sum_{q \in \mathcal{M}^*} \left(8 \ln(4TN/\delta) + 2\alpha \sqrt{l_T^{(1)}(q, Z_q)} \right) \quad (123)$$

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

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

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:

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

So,

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

Thus,

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

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

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

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

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

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

$$\leq 16\alpha N_1 l(M, \mathcal{Q}) \ln(16TN/\delta) \quad (133)$$

$$\leq 16\alpha Nl(M, \mathcal{Q}) \ln^2(16TN/\delta) \quad (134)$$

Combining these together,

$$\mathbb{E} \left[\sum_{i=1}^{N_1} (t_{i+1} - t_i - 1) \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} | E, \mathbb{I} \left\{ \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \right\} \right] \quad (135)$$

$$\leq A_1 + A_2 + A_3 \quad (136)$$

$$\leq 64c_2 Nl_{max} \ln^{5/2}(8TN/\delta) + 256c_2^2\alpha \ln^2(8TN/\delta) \sum_{q \in \mathcal{M}^*} \frac{P(q)Nl(M, \mathcal{Q})}{Z_q} \quad (137)$$

$$+ 128c_2^2\alpha \ln^2(16TN/\delta) \sum_{q \in \mathcal{M}^*} \frac{P(q)Nl(M, \mathcal{Q})}{Z_q} \quad (138)$$

$$+ 16\alpha Nl(M, \mathcal{Q}) \ln^2(16TN/\delta) \quad (139)$$

$$\leq 64(c_2 + \alpha)Nl_{max} \ln^{5/2}(16TN/\delta) + 256c_2^2\alpha \ln^2(8TN/\delta) \sum_{q \in \mathcal{M}^*} \frac{P(q)Nl(M, \mathcal{Q})}{Z_q} \quad (140)$$

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

$$Reg(T) \leq \mathbb{E} \left[\sum_{i=1}^{N_1} (t_{i+1} - t_i - 1) \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} | E, \mathbb{I} \left\{ \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \geq Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \right\} \right] \quad (141)$$

$$+ T\delta + 4c_2 N^2 \ln^2(4NT/\delta) + T \max_{t_i} \{Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\} \mathbb{I} \left\{ \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} < Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \right\} \quad (142)$$

$$\leq 256c_2^2\alpha \ln^2(8TN/\delta) \sum_{q \in \mathcal{M}^*} \frac{P(q)Nl(M, \mathcal{Q})}{\Delta_q} + 64(c_2 + \alpha)N^2 \ln^{5/2}(16TN/\delta) + Tc_2\delta \quad (143)$$

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

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

For problem-independent bound, take $Z_q = \sqrt{\frac{16c_2 Nl(M, \mathcal{Q})}{T}} \ln(8TN/\delta), \forall q \in \mathcal{Q}$, then $Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} = \sqrt{\frac{16c_2 Nl(M, \mathcal{Q})}{T}} \ln(8TN/\delta), \forall \widehat{\mathcal{M}}_{t_i}^{(full)} \in \mathfrak{V}$, and regret:

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

$$+ T\delta + 4c_2 N^2 \ln^2(4NT/\delta) + T \max_{t_i} \{Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}}\} \mathbb{I} \left\{ \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}} < Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \right\} \quad (146)$$

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

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

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

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:

$$\text{Reg}(T) \leq \tilde{O} \left(\sqrt{Nl(M, \mathcal{Q})T} \right) \quad (150)$$

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:

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

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, on 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:

$$\begin{aligned} & \sum_{t=t_i+1}^{t_{i+1}-1} \mathbb{E} \left[\sum_{q \in \mathcal{M}_t^c} P(q) C^*(q) - (1 + \beta) \sum_{q \in \mathcal{M}^{*c}} P(q) C^*(q) \right] \\ & \leq (t_{i+1} - t_i - 1) \mathbb{E} \left[\sum_{q \in \widehat{\mathcal{M}}_{t_i}^{(full)c} } P(q) C^*(q) - (1 + \beta) \sum_{q \in \mathcal{M}^{*c}} P(q) C^*(q) \right] + 2c_2 N \ln(4NT/\delta) \end{aligned}$$

Proof.

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

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

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

According to Lemma B.10,

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

$$\leq (t_{i+1} - t_i - 1) \mathbb{E} \left[\sum_{q \in \mathcal{M}^*} P(q)C^*(q) - \sum_{q \in \widehat{\mathcal{M}}_{t_i}^{(full)}} P(q)C^*(q) \right] + 2c_2 N \ln(4NT/\delta) \quad (155)$$

$$= (t_{i+1} - t_i - 1) \mathbb{E} \left[\sum_{q \in \widehat{\mathcal{M}}_{t_i}^{(full)c} } P(q)C^*(q) - \sum_{q \in \mathcal{M}^{*c}} P(q)C^*(q) \right] + 2c_2 N \ln(4NT/\delta) \quad (156)$$

Then

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

$$\leq (t_{i+1} - t_i - 1) \mathbb{E} \left[\sum_{q \in \widehat{\mathcal{M}}_{t_i}^{(full)c} } P(q)C^*(q) - \sum_{q \in \mathcal{M}^{*c}} P(q)C^*(q) \right] - \beta \sum_{q \in \mathcal{M}^{*c}} \sum_{t=t_i+1}^{t_{i+1}-1} P(q)C^*(q) + 2c_2 N \ln(4NT/\delta) \quad (158)$$

$$= (t_{i+1} - t_i - 1) \mathbb{E} \left[\sum_{q \in \widehat{\mathcal{M}}_{t_i}^{(full)c} } P(q)C^*(q) - (1 + \beta) \sum_{q \in \mathcal{M}^{*c}} P(q)C^*(q) \right] + 2c_2 N \ln(4NT/\delta) \quad (159)$$

□

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:

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

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

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

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

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

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

So,

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

Thus,

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

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

□

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

$$\Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}^c \quad (167)$$

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

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

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

Proof. According to Lemma C.3, we have

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

On the other hand

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

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

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 :

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Plugging these into above,

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

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

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

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

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

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

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

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

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

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

Combining these, we have

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

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

Combining the inequalities above together, we finish our proof. $\square$

C.1 Approximate Regret

Then we define approximation gap of $\widehat{\mathcal{M}}_t^{(full)}$ as $\Delta_{\widehat{\mathcal{M}}_t^{(full)}}^c = \sum_{q \in \widehat{\mathcal{M}}_t^{(full)c}} P(q)C^*(q) - (1 + \beta) \sum_{q \in \mathcal{M}^{*c}} P(q)C^*(q)$. 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:

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

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

$$\leq \mathbb{E} \left[\sum_{i=1}^{N_1+1} (t_{i+1} - t_i - 1) \Delta_{\widehat{\mathcal{M}}_t^{(full)}}^c \right] + 4c_2NN_1 \ln(4NT/\delta) + Tc_2\delta \quad (197)$$

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,

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

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

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

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

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

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

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

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

Therefore,

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

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

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 $\Delta_q^c \leq \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}^c$, then $\mathbb{I} \left\{ \Delta_{\widehat{\mathcal{M}}_{t_i}^{(full)}}^c < Z_{\widehat{\mathcal{M}}_{t_i}^{(full)}} \right\} = 0$. 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:

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

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

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

D Additional Experiments

The various algorithms compared in our experiments and the additional baselines are as follows:

- 1) VSOCB: The algorithms in our paper.
- 2) VSOCB-apx: VSOCB using the approximate oracle, which is a greedy method for the MKP.
- 3) COL-GDSF: baseline in our Section 5.
- 4) Least Recent Used (LRU)/Least Frequently Used (LFU): Prioritize selecting the query with the largest estimated sampling probability value $\hat{P}_t(q)$ to place into the cache.
- 5) Least Expected Cost (LEC): Prioritize selecting the query with the largest estimated cost $\hat{C}_t(q)$ to place into the cache.
- 6) Trivial method: This is a simple method for handling heterogeneous queries: it invokes the exact Oracle to solve the knapsack problem at each round, which is clearly convergent. However, this naive approach incurs substantial additional computational overhead. The accumulation-based Oracle invoking policy in our algorithm is designed precisely to mitigate such extra overhead, thereby achieving a balance between additional computational cost and cache accuracy.

Similar to [Zhu et al. \(2023\)](#), we set the frequency distribution as a power-law distribution with parameter 0.5. The expected cost of each query is drawn from $[1, 2]$, i.e., $c_1 = 1$ and $c_2 = 2$. The size of each query is sampled from a uniform distribution over $[2, 20]$.

In each of our experiments, the total number of query rounds is set to $T = 40K$, and the results are averaged over 5 independent runs.

Comparison with N . We fix all other parameters to compare the effect of varying the number of queries N . We set $M = 60$, $\alpha = 1$ and the query size is sampled from a uniform distribution over $[2, 20]$. The experimental results are presented in Table 5, 6.

Regret	VSOCB	VSOCB-apx	COL-GDSF	LEC	LFU
$N = 50$	$289.5 \pm 74.9^*$	542.2 ± 100.7	3026.2 ± 455.2	4848.9 ± 1832.4	2494.9 ± 1737.7
$N = 100$	$479.3 \pm 97.6^*$	845.4 ± 136.5	2030.9 ± 940.5	3744.7 ± 624.7	2313.8 ± 994.5
$N = 150$	$611.5 \pm 49.1^*$	730.6 ± 80.3	1016.8 ± 652.9	2577.3 ± 961.9	1550.8 ± 348.1

Table 5: Regret comparison with N

Total Cost	VSOCB	VSOCB-apx	COL-GDSF	LEC	LFU
$N = 50$	$46.4K \pm 413.2^*$	$46.7K \pm 390.2$	$49.3K \pm 678.4$	$51.1K \pm 1631$	$48.6K \pm 1707$
$N = 100$	$51.2K \pm 655.7^*$	$51.6K \pm 546.1$	$52.6K \pm 761.1$	$54.7K \pm 711.7$	$53.2K \pm 941.8$
$N = 150$	$56.7K \pm 390.5^*$	$56.8K \pm 512.8$	$56.9K \pm 527.2$	$58.5K \pm 1069$	$57.4K \pm 473$

Table 6: Total Cost comparison with N

In the above results, although the baseline regret gradually decreases as N increases, a plot of these values reveals that their regret remains linear, whereas our VSOCB algorithm consistently converges.

The reason for the small gap in total cost is that we selected a relatively small upper bound for the cost, i.e., the cost savings achieved by caching are not pronounced. Therefore, we primarily adopt regret as the evaluation metric here.

Comparison with M . We fix all other parameters to compare the effect of varying the cache size M . We set $N = 100$, $\alpha = 1$ and the query size is sampled from a uniform distribution over $[2, 20]$. The experimental results are presented in Table 7, 8.

Regret	VSOCB	VSOCB-apx	COL-GDSF	LEC	LFU
$M = 40$	$289.8 \pm 61.6^*$	520.1 ± 158.8	397 ± 65	2617.7 ± 676.3	797.8 ± 468.6
$M = 60$	$479.3 \pm 97.6^*$	845.4 ± 136.5	2030.9 ± 940.5	3744.7 ± 624.7	2313.8 ± 994.5
$M = 100$	$831 \pm 142.6^*$	1021.3 ± 169.1	1250.6 ± 513.6	5944.1 ± 946	2034.2 ± 691.9

Table 7: Regret comparison with M

Comparison with α . We fix all other parameters to compare the effect of varying α , which is a parameter that controls the number of oracle calls. We set $N = 100$, $M = 60$ and the query size is sampled from a uniform distribution over $[2, 20]$. We measure the number of oracle calls under different values of α . According to the theoretical results, the larger the value of α , the fewer the Oracle calls, but the lower the accuracy. The experimental results are presented in Table 9, 10, 11.

Comparison with Query-Size Distributions. We fix all other parameters to compare the effect of varying the query-size distributions. We set $N = 100$, $M = 60$, $\alpha = 1$ and the query size is sampled from a uniform distribution over $[2, 10]$, $[2, 15]$, $[2, 20]$ respectively. The experimental results are presented in Table 12, 13.

Total Cost	VSOCB	VSOCB-apx	COL-GDSF	LEC	LFU
$M = 40$	$52.3K \pm 454.2^*$	$52.7K \pm 743.6$	$52.8K \pm 240.9$	$55.1K \pm 662.6$	$53.3K \pm 419.7$
$M = 60$	$51.2K \pm 655.7^*$	$51.6K \pm 546.1$	$52.6K \pm 761.1$	$54.7K \pm 711.7$	$53.2K \pm 941.8$
$M = 100$	$49.7K \pm 437.4^*$	$50.2K \pm 308$	$50.5K \pm 589.1$	$55K \pm 1093.8$	$51K \pm 899$

Table 8: Total Cost comparison with M

Regret	VSOCB	VSOCB-apx	COL-GDSF	LEC	LFU
$\alpha = 0.5$	$454 \pm 135.7^*$	611 ± 95.5	899.1 ± 141.6	3136.9 ± 842.5	1000.8 ± 284.9
$\alpha = 1$	$479.3 \pm 97.6^*$	845.4 ± 136.5	2030.9 ± 940.5	3744.7 ± 624.7	2313.8 ± 994.5
$\alpha = 1.5$	712.1 ± 71.7	$706.6 \pm 73.9^*$	736.6 ± 578.9	4561 ± 1150.4	1476.6 ± 682.7

Table 9: Regret comparison with α

Total Cost	VSOCB	VSOCB-apx	COL-GDSF	LEC	LFU
$\alpha = 0.5$	$53K \pm 343^*$	$53.8K \pm 242.8$	$53.8K \pm 385.1$	$55.9K \pm 1199.5$	$54.1K \pm 479$
$\alpha = 1$	$51.2K \pm 655.7^*$	$51.6K \pm 546.1$	$52.6K \pm 761.1$	$54.7K \pm 711.7$	$53.2K \pm 941$
$\alpha = 1.5$	$49.7K \pm 496.7^*$	$49.9K \pm 434.3$	$50.2K \pm 678$	$54.1K \pm 835.1$	$50.7K \pm 704$

Table 10: Total Cost comparison with α

The number of Oracle calls	VSOCB	VSOCB-apx
$\alpha = 0.5$	1141 ± 9	1142 ± 8
$\alpha = 1$	686 ± 6	686 ± 7
$\alpha = 1.5$	557 ± 6	554 ± 7

Table 11: Oracle calls comparison with α

Regret	VSOCB	VSOCB-apx	COL-GDSF	LEC	LFU
$[2, 10)$	$669.8 \pm 103.8^*$	906.7 ± 102.7	1472.9 ± 573.5	6347.2 ± 2258.7	2082.7 ± 523.4
$[2, 15)$	$709.6 \pm 121.9^*$	1165.5 ± 36.6	1551.9 ± 344	4621 ± 967	1771.5 ± 482.8
$[2, 20)$	$479.3 \pm 97.6^*$	845.4 ± 136.5	2030.9 ± 940.5	3744.7 ± 624.7	2313.8 ± 994.5

Table 12: Regret comparison with query size

Total Cost	VSOCB	VSOCB-apx	COL-GDSF	LEC	LFU
$[2, 10)$	$48.8K \pm 381.8^*$	$49.5K \pm 451.8$	$49.9K \pm 504.4$	$54.9K \pm 2096.7$	$50.5K \pm 877$
$[2, 15)$	$52.3K \pm 283.1^*$	$52.8K \pm 307.6$	$53.2K \pm 749$	$56.3K \pm 1418.3$	$53.1K \pm 793$
$[2, 20)$	$51.2K \pm 655.7^*$	$51.6K \pm 546.1$	$52.6K \pm 761.1$	$54.7K \pm 711.7$	$53.2K \pm 942$

Table 13: Total Cost comparison with query size