Budget-constrained Result Integrity Verification of Outsourced Data Mining Computations

Bo Zhang[1], Boxiang Dong[2], Wendy Wang[1]

Stevens Institute of Technology[1], Montclair State University[2]
Hoboken, NJ, USA[1], Montclair, NJ, USA[2]
{bzhang41,Hui.Wang}@stevens.edu[1], dongb@montclair.edu[2]

Abstract. When outsourcing data mining needs to an untrusted service provider in the Data-Mining-as-a-Service (DMaS) paradigm, it is important to verify whether the service provider (server) returns correct mining results (in the format of data mining objects). We consider the setting in which each data mining object is associated with a weight for its importance. Given a client who is equipped with limited verification budget, the server selects a subset of mining results whose total verification cost does not exceed the given budget, while the total weight of the selected results is maximized. This maps to the well-known *budgeted maximum coverage* (BMC) problem, which is NP-hard. Therefore, the server may execute a heuristic algorithm to select a subset of mining results for verification. The server has financial incentives to cheat on the heuristic output, so that the client has to pay more for verification of the mining results that are less important. Our aim is to verify that the mining results selected by the server indeed satisfy the budgeted maximization requirement. It is challenging to verify the result integrity of the heuristic algorithms as the results are non-deterministic. We design a probabilistic verification method by including *negative candidates* (NCs) that are guaranteed to be excluded from the budgeted maximization result of the ratio-based BMC solutions. We perform extensive experiments on real-world datasets, and show that the NC-based verification approach can achieve high guarantee with small overhead.

Keywords: Data-Mining-as-a-Service (DMaS); cloud computing; result integrity; budgeted maximization.

1 Introduction

Due to the fast increase of data volume, many organizations (clients) with limited computational resources and/or data mining expertise have outsourced their data and data mining needs to a third-party service provider (server) that is computationally powerful. This emerges the Data-Mining-as-a-Service (DMaS) paradigm [8, 21]. Although DMaS offers a cost-effective solution for the client, it raises several security issues. One of the issues is *result integrity verification*, i.e., how to verify that the untrusted server returns the correct mining results [18, 9]. There are many incentives for the server to cheat on the mining results. As data

mining is critical for decision making and knowledge discovery, it is essential for the client to verify the result integrity of the outsourced mining computations.

The key idea of the existing result integrity verification solutions (e.g., [7]) is to associate the mining results with a *proof*, which enables the client to verify the result correctness with a deterministic guarantee. It has been shown that the construction of integrity proofs can be very costly [7]. We assume that the client has to pay for the cost of proof generation. If the client has a tight budget for proof construction, she only can afford the proof construction of a subset of mining results. We consider that the mining results that involve different data items bring different benefits to the client (e.g., the shopping patterns of luxury goods are more valuable than that of bread and milk). Therefore, it is desirable that the mining results that bring the maximum benefits to the client are picked for the proof construction. This problem can be mapped to the well-known budgeted maximum coverage problem (BMC) [13].

Formally, given a set of objects $O = \{o_1, \ldots, o_n\}$, each of which associated with a weight (value) w_i and a cost c_i, the *budgeted maximization problem* (BMC) is to find a subset of objects $O' \subseteq O$ whose total cost does not exceed the given budget value B, while the total weight is maximized. A variant of BMC problem considers the case when the cost of overlapping objects is amortized, i.e., $c(\{o_i, o_j\}) < c_i + c_j$. This motivates the *budgeted maximization with overlapping costs (BMOC)* problem [6]. It has been proven that both BMC and BMOC problems are NP-hard [13].

Given the complexity of BMC/BMOC problems, the server may execute a heuristic BMC/BMOC algorithm (e.g., [19, 6]) to pick a subset of mining results for proof construction. The server is incentivized to cheat on proof selection by picking the results with cheaper computations (e.g., by randomly picking a subset of mining results), and claims that those picked results are returned by the BMC/BMOC heuristic algorithm. To catch such cheating of proof construction, it is important to design efficient methods for verification of the result integrity of heuristic BMC/BMOC computations.

In general, it is difficult to verify the result correctness of heuristic algorithms, as the output of these algorithms has much uncertainty. Prior work (e.g., metamorphic testing [4, 5]) mainly use software testing techniques that execute the algorithm multiple times with different inputs, and verify if multiple outputs satisfy some required conditions. These techniques cannot be applied to the DMaS paradigm, as the client may not be able to afford to execute the (expensive) data mining computations multiple times. Existing works [7, 21] on integrity verification of outsourced data mining computations mainly focus on the authentication of *soundness* and *completeness* of the data mining results, but not the result correctness of budgeted maximization algorithm.

We aim to design efficient verification methods that can catch any incorrect result of heuristic BMC/BMOC algorithms with high *probabilistic* integrity guarantee and small computational overhead. To our best knowledge, this is the first work that focuses on verifying the result integrity of the budgeted maximization algorithm. We focus on the type of *ratio-based* BMC/BMOC heuristic algorithms

that rely on the weight/cost ratio to pick the objects, and make the following main contributions. First, we design an efficient result correctness verification method for the ratio-based BMC/BMOC problem. Following the intuition of [7, 15, 14, 16] to construct evidence patterns for the purpose of verification, our key idea is to construct a set of *negative candidates (NCs)* that will *not* be picked by any ratio-based heuristic BMC/BMOC algorithm. A nice property is that NCs do not impact the original output of BMC/BMOC algorithm (i.e., all original mining results selected by the BMC/BMOC algorithm are still picked under the presence of NCs). The verification mainly checks if the server picks any NC for proof construction. If there is, then the server's cheating on proof construction is caught with 100% certainty. Otherwise, the mining results that the server picked for proof construction are trusted with a probabilistic guarantee. We formally analyze the probabilistic guarantee of the NC-based method. Second, the basic NC-based approach is weak against the attacks that utilize the knowledge of how NCs are constructed. Therefore, we improve the basic NC-based approach to be robust against these attacks. Last but not least, we take *frequent itemset mining* [11] as a case study, and launch a rich set of experiments on three real-world datasets to evaluate the efficiency and robustness of the NC-based verification. The experiment results demonstrate that the NC-based verification method can achieve high guarantee with small overhead (e.g.,it takes at most 0.001 second for proof verification of frequent itemset results from 1 million transactions).

The remaining of the paper is organized as following. Section 2 introduces the preliminaries. Section 3 presents the NC-based verification method. Section 4 reports the experiment results. Section 5 discusses the related work. Section 6 concludes the paper.

2 Preliminaries

2.1 Budgeted Maximization Coverage (BMC) Problem

Given a set of objects $O = \{o_1, \ldots, o_n\}$, each associated with a cost c_i and a weight (i.e., value) w_i, the budgeted maximization coverage (BMC) problem is to find the set $O' \subseteq O$ s.t. (1) $\sum_{o_i \in O'} c_i \leq B$, for any specified budget value B, and (2) $\sum_{o_i \in O'} w_i$ is maximized. When the overlapping objects share the cost, i.e., $c(o_i, o_j) < c_i + c_j$, the problem becomes the budgeted maximization coverage with overlapping cost (BMOC) problem. Both BMC and BMOC problems are NP-hard [13, 6]. Various heuristic algorithms have been proposed [19, 6]. Most of the heuristic BMC/BMOC algorithms [13, 6, 3, 19] follow the same principle: pick the objects repeatedly by their $\frac{weight}{cost}$ ratio (denoted as *WC-ratio*) in descending order, until the total cost exceeds the given budget. The major difference between these algorithms is how the cost is computed for a given set of objects; BMC algorithms simply sum up the cost of these objects, while BMOC algorithms compute the total cost of these objects *with sharing*. We call these *ratio-based* heuristic algorithms the GREEDY algorithms. We assume that the server uses a ratio-based greedy algorithm to pick the subset of mining results for proof construction.

2.2 Budget-constrained Verification

In this paper, we consider the scenario where the data owner (client) outsources her data D as well as her mining needs to a third-party service provider (server).

The server returns the mining results R of D to the client. We represent R as a set of *mining objects* $\{o_1, \ldots, o_n\}$, where each mining object o_i is a pattern that the outsourced data mining computations aim to find. Examples of the mining objects include outliers, clusters, and association rules. Different mining objects can be either *non-overlapping* (e.g., outliers) or *overlapping* (e.g., association rules that share common items).

Since the server is potentially untrusted, it has to provide an *integrity proof* of its mining results, where the proof can be used to verify the correctness of the mining results [7]. In general, each mining object is associated with a single proof [2, 17]. We use c_i to denote the cost of constructing the proof of the mining object o_i. According to [7], the overlapping mining objects share the proofs (and its construction cost) of the common data items. The total proof cost of R is $c_R = \sum_{\forall o_i \in (\cup_{\forall o_j \in R} o_j)} c_i$.

We assume that the client has to pay for the cost of proof construction. The proof construction cost is decided by the number of mining objects that are associated with the proofs. Intuitively, the more the client pays for the proof construction, the more mining objects that she can verify the correctness. We assume that different mining objects bring different benefits to the client. We use w_i to denote the *weight* of the mining object o_i. We follow the same assumption of the BMC/BMOC problem that the weights of different mining objects never share, regardless of the overlaps between the mining objects. Thus given a set of mining objects R, the total weight of R is computed as: $w_R = \sum_{\forall o_i \in R} w_i$.

In this paper, we consider the client who has the limited budget for proof construction. Given the budget B and the mining results R from the outsourced data D, the client asks the server to pick a subset of mining results $R' \subseteq R$ for proof construction where $c_{R'} \leq B$, while $w_{R'}$ is maximized. Apparently, this problem can be mapped to either BMC (for share-free proof construction scenario) or BMOC (for shared proof construction scenario). Given the complexity of BMC/BMOC algorithms, the server runs the GREEDY algorithm to pick a subset of mining objects for proof construction.

2.3 Verification Goal

Due to various reasons, the server may cheat on proof construction. For instance, in order to save the computational cost, it may randomly select a set of data mining objects, instead of executing the GREEDY algorithm. Therefore, after the client receives the mining objects R from the server, in which $R' \subseteq R$ is associated with the result integrity proof, she would like to verify whether R' indeed satisfies *budgeted maximization*. Formally,

Definition 21 [Verification Goal] *Given a set of mining objects $R = \{o_1, \ldots, o_n\}$, let R' denote the mining objects picked by the server for proof construction. A verification algorithm should verify whether R' satisfies the following two budgeted maximization requirements: (1) The total cost of R' does not exceed the budget B; and (2) The total weight of R' is maximized.*

The verification of Goal 1 is trivial, as the client can simply calculate the total cost of R' and compare it with B. The verification of Goal 2 is extremely

challenging due to two reasons: (1) due to the NP-hardness complexity of the BMC/BMOC problem, it is impossible that the client re-computes the budgeted maximization solutions; and (2) as the server uses a heuristic method to pick R', it is expected that R' may not be the optimal solution. A naive solution is to pick a set of objects $R'' \subseteq R$ (either randomly or deliberately), and compare the total weight of R'' with that of R'. However, this naive solution cannot deliver any verification guarantee, even if the total weight of R' is smaller than that of R'', as the output of the heuristic algorithms is not expected to be optimal. Our goal is to design efficient verification method that can deliver *probabilistic* guarantee of the results of heuristic BMC/BMOC algorithms. *We must note that the verification of correctness and completeness of the mining results is not the focus of our paper.*

3 NC-based Verification Approach

3.1 Basic Approach

The key idea of our verification method is to use the *negative candidates* (NCs) of budgeted maximization. The NCs are guaranteed to be *excluded* from the output of any GREEDY algorithm. Therefore, if the server's results include any NC, the client is 100% sure that the server fails the verification of budgeted maximization. Otherwise, the client has a probabilistic guarantee of budgeted maximization.

Intuitively, if the WC-ratio of any NC is smaller than the WC-ratio of any real object in D, the GREEDY algorithm never picks any NC if it is executed faithfully. Besides this, we also require that the presence of NCs should not impact the original results (i.e., all original mining objects selected by GREEDY should still be selected under the presence of NCs). Following this, we define the two requirements of NCs:

- Requirement 1: There is no overlap between any NC and any real object or NC;
- Requirement 2: The WC-ratio of any NC o_i is lower than the ratio of any real object $o_j \in R$, i.e., $\frac{w_i}{c_i} < \frac{w_j}{c_j}$, where R is the mining results of the original dataset D.

Based on this, we have the following theorem:

Theorem 1. *Given a set of objects R and the GREEDY algorithm F, for any NC $o_i \in R$ that satisfies the two requirements above, it is guaranteed that $o_i \notin F(R)$.*

The correctness of Theorem 1 is straightforward. Requirement (1) ensures that NCs do not share cost with any other object, thus WC-ratio is its exact ratio used in the GREEDY algorithm. It also ensures that the presence of NCs do not change the WC-ratio of any other objects, and thus do not change the output of the GREEDY algorithm. Requirement (2) ensures that NCs always have the smallest WC-ratio, and thus are never picked by the GREEDY algorithm.

we formally define (ϵ, θ)-budgeted maximization as our verification goal.

Definition 31 [**(ϵ, θ)-Budgeted Maximization**] *Given a set of objects R, let $R' \subseteq R$ be the set of objects picked by any GREEDY algorithm. Assume the server cheats on θ percent of R'. We say a verification method $\mathcal{M}$ can verify* (ϵ, θ)-budgeted maximization *if the probability p to catch the cheating on R' satisfies that $p \geq \epsilon$.*

Suppose we insert K NCs into the original n objects. Suppose the GREEDY algorithm picks $m < n$ objects. Also suppose that the attacker picks $\ell \geq 1$ budget-maximized objects from the m objects (i.e., $\ell = m\theta$, where θ is the parameter for (ϵ, θ)-budgeted maximization) output by the GREEDY algorithm, and replaces them with other objects with lower weight/cost ratios. Note that if such replacement involves any NC, then the verification method can catch the cheating. Now let us compute the probability that the attacker can escape by picking no NCs when replacing budget maximized objects. If an attacker replaces an object o in the original budgeted-maximization result with another object o', the probability that o' is not a NC is $\frac{n-m}{n-m+K}$. Thus, with probability $\frac{n-m}{n-m+K}$, the attacker's wrong result is not caught. Now, given $\ell \geq 1$ budgeted-maximized objects that are not returned by the server, the probability p that the server can be caught (i.e., it picks at least one NC):

$$p = 1 - \prod_{i=0}^{\ell-1} \frac{n-m-i}{n-m+K-\ell}. \tag{1}$$

Since $\frac{n-i}{n+K-i} \geq \frac{n-i-1}{n+K-i-1}$, it follows that:

$$1 - (\frac{n-m}{n-m+K})^{\ell} \leq p \leq 1 - (\frac{n-m-\ell}{n-m+K-\ell})^{\ell}. \tag{2}$$

Based on this reasoning, we have the following theorem:

Theorem 2. *Given n original objects, among which $m < n$ itemsets are picked by GREEDY, the number K of NCs to verify (ϵ, θ)-budgeted maximization satisfies that: $K \geq (\frac{1}{\sqrt[m\theta]{1-\epsilon}} - 1)(n-m)$.*

We calculate the value of K with regard to various θ and ϵ values, and observe that our NC-based verification method does not require large number NCs when the budget is large, even though θ is a small fraction. For example, when $\theta = 0.1$ and $m = 400$, our method only requires 16 and 25 NCs in order to achieve p of at least 95% and 99%, respectively. A detailed analysis of the relationship between K and the verification parameters (i.e., θ and ϵ) can be found in the full paper [22].

A challenge to calculate the exact K value is that the client may not possess the knowledge of the real m value (i.e., the number of original objects picked by GREEDY). Next, we discuss how to estimate m. Intuitively, for any $m' \leq m$, $(\frac{1}{\sqrt[m'\theta]{1-\epsilon}} - 1)n) \geq (\frac{1}{\sqrt[m\theta]{1-\epsilon}} - 1)n)$. Therefore, we can simply calculate the lowerbound $\hat{m}$ of m, and estimate $K = (\frac{1}{\sqrt[\hat{m}\theta]{1-\epsilon}} - 1)\hat{n}$. To calculate $\hat{m}$, we can calculate the upperbound cost c_{max} of any object o_j in O, then we compute $\hat{m} = [\frac{B}{c_{max}}]$.

3.2 A More Robust Approach

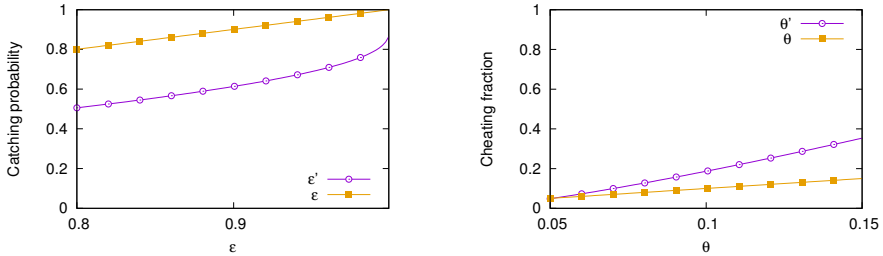
If the server possesses the distribution information of the original dataset D, as well as the details of the NC-based verification, it may try to escape from the verification by distinguishing NCs from the original objects based on their characteristics. In particular, there are two possible attacks that can be launched: (1) the *overlap-based* attack: the server may identify those objects that are disjoint with other objects as NCs, since NCs do not overlap with any other object; and (2) the *ratio-based* attack: the server can sort all objects (including real ones and NCs) by their WC-ratio, and pick K' objects of the *lowest* ratio as NCs. Next, we discuss the strategy to mitigate these attacks.

In order to defend against the overlap-based attack, we introduce the overlap between NCs. In particular, we require the overlap between NCs should be similar to the overlap between the original objects. We define the *overlapping degree* of the object o_i as $od_i = \frac{d_i}{n-1}$, where d_i is the number of objects in D that o_i overlaps with. We assume the overlapping degree follows a normal distribution, i.e., $od \sim \mathcal{N}(u_o, \sigma_o^2)$. We estimate $\hat{u_o}$ and $\hat{\sigma_o}$ from the overlapping degrees of original objects. We require that the overlapping degree of NCs should follow the same distribution $\mathcal{N}(\hat{u_o}, \hat{\sigma_o}^2)$. To make sure that ratio of NCs remains smaller than that of any original object, we introduce the overlap by increasing the cost of NCs, while keeping the weight unchanged. Note that we only introduce overlapping between NCs; NCs and real mining objects do not overlap. Thus NCs still satisfy Requirement 1 of NCs.

Regarding to the ratio-based attack, identifying the NCs of the lowest WC-ratio leads to insufficient number of NCs to satisfy (ϵ, θ)-budgeted maximization. In particular, suppose that the server uniformly picks $K' \in [0, K]$ by random guess, and takes K' objects with the lowest WC-ratio as the NCs. Then $K - K'$ NCs remain to be unidentified by the server. Next, we analyze the the probabilistic integrity guarantee that can be provided if the server launches the ratio-based attack. We have the following theorem:

Theorem 3. *Given a set of K NCs that satisfies (ϵ, θ)-budgeted maximization, our NC-based approach is able to verify (ϵ', θ')-budgeted maximization, where $\epsilon' = 1 - \frac{\epsilon(n-m)-K(1-\epsilon)}{K(m\theta-1)}$, and $\theta' = \frac{(n-m)\ln(1-\epsilon)}{Km}(\ln\frac{K}{n-m} + \frac{K}{n-m} + \frac{K^2}{8(n-m)^2})$, where n is the number of original objects, and m is the number of objects picked by GREEDY.*

Due to the space limit, we omit the proof of Theorem 3.



(a) Comparison between ϵ and ϵ' ($\theta = 0.1$) (b) Comparison between θ and θ' ($\epsilon = 0.99$)

Fig. 1. Evaluation of ϵ' and θ'

We plot ϵ and ϵ' (defined in Theorem 3) in Figure 1 (a) (θ and θ' in Figure 1 (b) respectively). The results show that the ratio-based attack only degrades (ϵ, θ)-budgeted maximization slightly. For example, consider $n = 600$, and $m = 400$. When $\epsilon = 0.99$, and $\theta = 0.1$, it requires $K = 25$ NCs to verify $(0.99, 0.1)$-budgeted maximization (Theorem 2). If the server exploits the ratio-based attack, the NC-based approach needs 495 NCs to provide the same security guarantee.

4 Experiments

4.1 Setup

Hardware. We execute the experiments on a testbed with Intel i5 CPU and 8GB RAM, running Mac OS X 10. We implement all the algorithms in C++.

Datasets. We take *frequent itemset mining* [11] as the mining task. Given a dataset D and a threshold value min_{sup}, it aims at finding all the itemsets whose number of occurrences is at least min_{sup}. We use three datasets, namely *Retail*, *T10* and *Kosarak* datasets, available from the *Frequent Itemset Mining Data Repository*[1]. We use various support threshold values min_{sup} for mining. The data description and the mining setting can be found in our full paper [22].

Budget setting. We vary the budget to observe its impact on the performance of the NC-based approach. We define *budget ratio* $br = \frac{B}{Tb}$, where B is the given budget, and Tb is the total proof cost for all the frequent itemsets. Intuitively, a higher budget ratio leads to the higher budget for proof construction.

Cost and weight model. For each item x, its cost is computed as $c_x = \frac{supp_D(\{x\})}{max_{\forall x \in \mathcal{X}} supp_D(\{x\})}$, where $supp_D(\{x\})$ denotes the frequency of $\{x\}$ in D. We use four different strategies to assign the item weights. We consider that all items have the same weight. Given any itemset $X = \{x_1, \ldots, x_t\}$, the cost and weight of X is the sum of the individual items. In other words, $c_X = \sum c_{x_i}$, while $w_X = \sum w_{x_i}$.

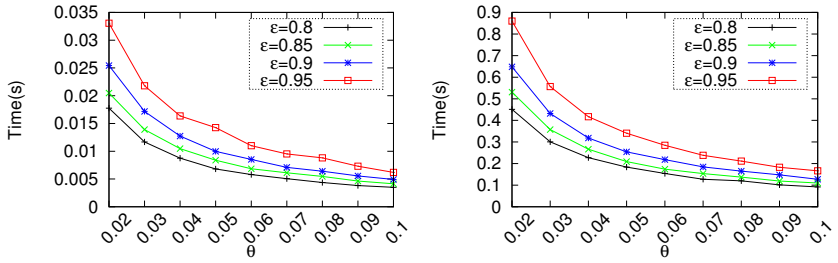
4.2 Robustness of Probabilistic Verification

We measure the robustness of our NC-based approach. In particular, we simulate the cheating behavior by the attacker on budgeted maximization, and measure the probability p that our approach can catch the attacker. We compare p with the pre-defined desired catching probability ϵ. The experiment results on the datasets demonstrate that the detection probability is always larger than ϵ, which verifies the robustness of our verification approach. The detailed result can be found in our full paper [22].

4.3 Verification Performance

Verification preparation time. We measure the time of creating artificial transactions that produce NCs for verification of budgeted maximization. Figure 2 shows the NC creation time for the datasets. In general, the NC creation procedure is very fast. In all the experiments, it takes at most 0.9 second to generate the NCs, even for small $\theta = 0.02$ and large $\epsilon = 0.95$. Furthermore, we observe that larger ϵ and smaller θ lead to longer time for generating NCs.

[1] http://fimi.ua.ac.be/data/.



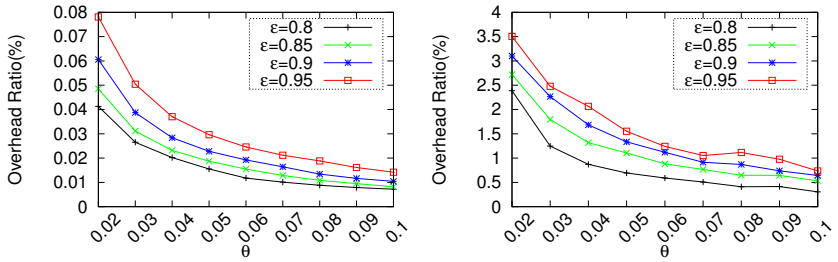
(a) *T10* ($n = 4054$, $m = 2289$, $min_{sup} = 100$)

(b) *Kosarak* ($n = 3023$, $m = 1896$, $min_{sup} = 1981$)

Fig. 2. NC construction time

This is because the number of NCs is positively correlated to ϵ but negatively correlated to θ. In other words, we need more NCs to verify smaller errors of the budgeted maximization results with higher guarantee.

Verification time. We measure the verification time on the datasets for various settings of θ and ϵ. We observe that the verification time is negligible; it never exceeds 0.039 second. The minimal verification time is 0.001 second, even for the *Kosarak* dataset, which consists of nearly 1 million transactions. We omit the results due to the space limits.



(a) *T10* ($n = 4054$, $m = 2289$, $min_{sup} = 100$)

(b) *Kosarak* ($n = 3023$, $m = 2429$, $min_{sup} = 1981$)

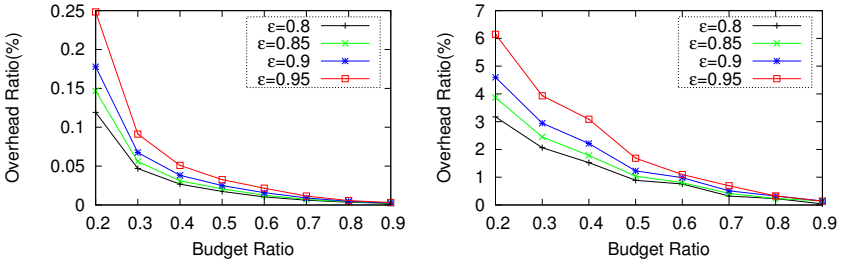
Fig. 3. Mining overhead (various θ setting for (ϵ, θ)-budget maximization)

Mining overhead by verification. We measure the mining overhead by adding artificial transactions for NC creation. We measure the *overhead ratio* as $\frac{(T_{D'} - T_D)}{T_D}$, where T_D and $T_{D'}$ are the mining time of the original dataset D and D' after adding artificial transactions. Intuitively the lower overhead ratio is, the smaller additional mining cost incurred by our NC-based verification.

Various θ. Figure 3 reports the overhead ratio for the datasets when changing θ setting for (ϵ, θ)-budgeted maximization. First, similar to the observation of NC creation time, the mining overhead ratio increases when θ becomes smaller and ϵ rises, as it requires more NCs, which results in larger mining overhead. Second, for all datasets, the overhead ratio is small (no more than 4%). The mining overhead is especially small for the *T10* dataset. It never exceeds 0.08% even though $\epsilon = 0.95$. The reason is that the mining of the original *T10* dataset consumes a long time, which leads to the small mining overhead ratio.

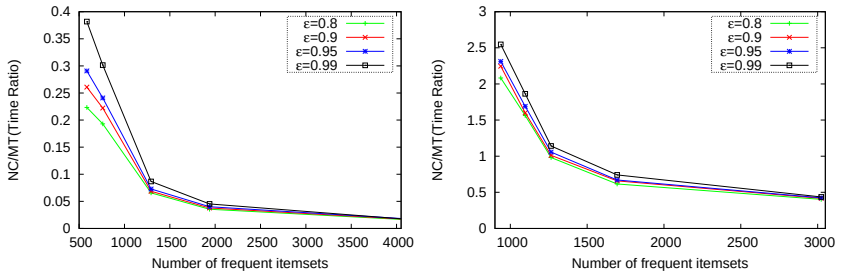
Various budgets. In Figure 4, we display the effect of the budgets (in the format of the budget ratio) on the mining overhead. On both datasets, we

observe that the mining overhead drops with the increase of budgets. Given a large budget, the GREEDY algorithm picks more frequent itemsets for proof construction. This results in the decreased number of NCs. Therefore, the mining overhead on the artificial transactions reduces, as the number of artificial transactions drops. In particular, the mining overhead can be very small, especially on the *T10* dataset (no larger than 0.25%).



(b) *T10* ($min_{sup} = 100$, $\theta = 0.05$) (c) *Kosarak* ($min_{sup} = 1981$, $\theta = 0.1$)

Fig. 4. Mining overhead (various budgets)



(a) *T10* ($\theta = 0.1$ $br = 0.5$) (b) *Kosarak* ($\theta = 0.1$ $br = 0.5$)

Fig. 5. NC-based approach vs. Metamorphic Testing (MT)

4.4 Comparison with Metamorphic Testing (MT)

We compare our NC-based approach with the metamorphic testing (MT) approach [4, 5]. For the implementation of the MT approach, we treat the GREEDY algorithm as a black box M, and run it on the discovered frequent itemsets $\mathcal{I}$ *twice*. The output of the first execution $M(\mathcal{I})$ is used as the input of the second execution of M. Apparently, if the server is honest, the output of the second execution of M should be the same as the output of the first execution, i.e., $M(\mathcal{I}) = M(M(\mathcal{I}))$. The overhead of MT is measured as the time of the second execution of the GREEDY algorithm. We measure the total time of our NC-based approach for verification, which includes the time of both verification preparation and verification. In Figure 5, we show the ratio of the time performance of our NC-based approach to that of MT approach. Overall, our NC-based approach shows great advantage over MT approach, especially when the number of frequent itemsets is large. For example, on the *T10* dataset, our verification approach is at least 10 times faster than MT when the number of frequent itemset is larger than 3,000. We also observe that when the number of

frequent itemsets increases, the ratio decreases (i.e., the NC-based approach is much faster than MT). This is because MT takes more time to pick the budget-maximized objects from the larger set of candidates. It is worth noting that MT does not provide any formal guarantee that the result satisfies budgeted maximization requirement.

5 Related Work

In this section, we discuss the related work, including: (1) the verifiable computation techniques for general-purpose computations; (2) result integrity verification of outsourced data mining computations; and (3) software testing for heuristic algorithms.

Gennaro et al. [9] define the notation of *verifiable computation* (VC) that allows a computationally limited client to be able to verify the result correctness of some expensive general-purpose computations that are outsourced to a computationally powerful server. Babai and Goldwasser et al. [1, 10] design the probabilistic checkable proofs. However, the incurred proofs might be too long for the verifier to process. This is not ideal for the DMaS paradigm where commonly the client outsources the data mining computations with a single input dataset.

Verification of result integrity of outsourced data mining computations have caught much attention recently. The existing methods have covered various types of data mining computations, including clustering [15], outlier mining [14], frequent itemset mining [7, 8], Bayesian networks [16], and collaborative filtering [20]. All these works focus on correctness and completeness verification of mining results, while ours aims at the problem of verification of budgeted maximization.

In software testing, it has been shown that the verification of heuristic algorithms is very challenging, as the heuristic methods may not give exact solutions for the computed problems. This makes it difficult to verify outputs of the corresponding software by a *test oracle* - a mechanism to construct the test cases. This is known as the test oracle problem [12] in software testing. A popular technique that is used to test programs without oracles is metamorphic testing (MT) [4, 5], which generates test cases by making reference to metamorphic relations, that is, relations among multiple executions of the target program. Though effective, MT is not suitable to the budget-constrained DMaS paradigm, as it involves multiple executions of the same function.

6 Conclusion

In this paper, we investigate the problem of result verification of budgeted maximization problem in the setting of DMaS paradigm. We present our probabilistic verification method that authenticates whether the server's results indeed reach budgeted maximization. We analyze the formal guarantee of the budgeted maximization of our method. Our experiments demonstrate the efficiency and effectiveness of our methods. For the future work, one interesting direction is to investigate the deterministic verification methods that can authenticate budgeted maximization with 100% certainty.

References

1. László Babai. Trading group theory for randomness. In *Symposium on Theory of Computing*, 1985.
2. Siavosh Benabbas et al. Verifiable delegation of computation over large datasets. In *Advances in Cryptology–CRYPTO*. 2011.
3. Jens Bleiholder et al. Query planning in the presence of overlapping sources. In *EDBT*. 2006.
4. Tsong Y Chen et al. Metamorphic testing: a new approach for generating next test cases. Technical report, Hong Kong Univ. of Science and Technology, 1998.
5. Tsong Yueh Chen et al. Fault-based testing in the absence of an oracle. In *International Conference on Computer Software and Applications*, 2001.
6. Donald E Curtis et al. Budgeted maximum coverage with overlapping costs: monitoring the emerging infections network. In *Algorithm Engineering & Expermiments*, 2010.
7. Boxiang Dong et al. Integrity verification of outsourced frequent itemset mining with deterministic guarantee. In *ICDM*, 2013.
8. Boxiang Dong et al. Result integrity verification of outsourced frequent itemset mining. In *Data and Applications Security and Privacy XXVII*. 2013.
9. Rosario Gennaro et al. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology*. 2010.
10. Shafi Goldwasser et al. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 1989.
11. Jiawei Han et al. Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, 2000.
12. Upulee Kanewala et al. Techniques for testing scientific programs without an oracle. In *International Workshop on Software Engineering for Computational Science and Engineering*, 2013.
13. Samir Khuller et al. The budgeted maximum coverage problem. *Information Processing Letters*, 1999.
14. Ruilin Liu et al. Audio: An integrity auditing framework of outlier-mining-as-a-service systems. In *Machine Learning and Knowledge Discovery in Databases*. 2012.
15. Ruilin Liu et al. Integrity verification of k-means clustering outsourced to infrastructure as a service (iaas) providers. In *SDM*, 2013.
16. Ruilin Liu et al. Result integrity verification of outsourced bayesian network structure learning. In *SDM*, 2014.
17. Charalampos Papamanthou et al. Optimal verification of operations on dynamic sets. In *Advances in Cryptology*. 2011.
18. Bryan Parno et al. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography*. 2012.
19. Michael Sindelar et al. Sharing-aware algorithms for virtual machine colocation. In *Symposium on Parallelism in Algorithms and Architectures*, 2011.
20. Jaideep Vaidya et al. Efficient integrity verification for outsourced collaborative filtering. In *ICDM*, 2014.
21. Wai Kit Wong et al. Security in outsourcing of association rule mining. In *VLDB*, 2007.
22. Bo Zhang et al. Budget-constrained result integrity verification of outsourced data mining computations. `http://www.cs.stevens.edu/~hwang4/papers/dbsec2017full.pdf`, 2017.