On Optimal Learned Bloom Filter Design

Anonymous Authors[1]

Abstract

Bloom filter (BF) is a well-known data structure for efficient set membership testing. Recently, learned Bloom Filter (LBF) has emerged as a variant of canonical BF that is augmented with learning capabilities by incorporating a pre-trained learning model. While LBF demonstrates promising performance enhancement across various applications, the integration of learning components opens up a vast and complex design space. Currently, most existing LBF designs rely on heuristic methods and hence fail to provide any theoretical guarantee. Driven by this gap, we develop a combinatorial algorithmic framework on optimal LBF design that, by exploring the entire design space, computes the parameters with the objective of minimizing the false positive rate under given space budget. We demonstrate theoretically that our algorithm achieves 2-optimality. Armed with our theoretical framework, we prove that sharing a single backup filter for all the score regions of the learning model suffices to achieve optimal performance, therefore leading to a neat and theoretically backed-up architecture for LBF.

1. Introduction

In the domain of data processing and storage, Bloom filter (BF) (Bloom, 1970) has long held a significant position. Introduced as a means to address the challenge of efficiently determining set membership with limited memory resources, it has been widely adopted in various applications where quick approximations of whether an element belongs to a particular set are essential, such as network security (Geravand & Ahmadi, 2013), databases (Xiong et al., 2014), biometric issues (Gomez-Barrero et al., 2014), etc. Since its creation, there have been continuous research efforts in enhancing its performance while overcoming its inherent limitations, among which learned Bloom Filters (LBF) were recently proposed (Kraska et al., 2018; Mitzenmacher, 2018) as a learning-augmented variant of the canonical BF. By orchestrating an learning oracle with a backup BF (cf. Figure 1 for LBF architecture), LBF is able to achieve lower false positive rate under the same memory footprint.

Compared to BF, the integration of machine learning has opened up a vast and complex design space for LBF, by incorporating different learning models, thresholding strategies, and backup filter configurations. This large design space offers both opportunities and challenges. On one hand, it allows for the customization of LBF to suit a wide variety of application scenarios and data characteristics. On the other hand, this complexity also poses non-trivial technical challenges in optimizing LBF.

Since the introduction of LBF, a handful of propositions are developed to optimize its performance by exploring the corresponding design space. Notably, (Vaidya et al., 2021) divided the output of the learning oracle into distinct regions and dedicated a separate backup BF for each region. The current state-of-the-art LBF design, called Adaptive LBF (Ada-BF), improved upon (Kraska et al., 2018; Mitzenmacher, 2018) by attributing different parameters for each backup BF. However, despite the performance gain of Ada-BF demonstrated empirically, the optimization approach developed in (Dai & Shrivastava, 2020) is mainly heuristic-based and hence fails to provide any theoretical guarantee. In summary, the current state of the art still lacks of theoretical framework on how to optimize LBF.

Driven by the above theoretical gap, we embark in this paper on a systematic analysis of the optimal LBF design in its generic form. By generic we mean to explore the entire design space to achieve the overall optimality. Our major contribution consists of a combinatorial algorithmic framework to set the LBF parameters achieving $(2+\epsilon)$-optimality, where ϵ can be tuned to trade off optimality against complexity. Armed with our theoretical framework, we prove that sharing a single backup filter for all the regions is optimal, therefore leading to a neat and theoretically backed-up architecture for LBF.

The paper is articulated as follows. Section 2 introduces technical preliminaries and related work. Section 3 presents our formulation and optimization of LBF design for the case

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

of single backup filter. Section 4 presents our formulation and optimization of LBF design for the case of multiple backup filters. Section 5 presents empirical experiments to evaluate our propositions. Section 6 concludes the paper.

2. Technical Preliminaries and Related Work

2.1. Standard Bloom Filter

BF is a probabilistic data structure that offers an efficient solution for set membership testing with low memory overhead (Bloom, 1970).

Essentially, a BF operates by compressing a set $\mathcal{S}$ of n items into a bit array of m bits. This is achieved through the utilization of a set of k independent hash functions $\mathcal{H} \triangleq \{h_i\}_{i=1}^{k}$. In this paper, we adopt the common assumption that, for every hash function, each element in the universe is mapped independently and uniformly to a integer within the range $[0, \cdots, m]$. At the initialization stage, all bits in the bit array are set to 0. Upon inserting an item x, the bits at positions $h_i(x)$ for $i = 1, \cdots, k$ are flipped to 1. This process effectively marks the presence of the item in the filter. When an item x is queried, the filter checks if all the bits at positions $h_i(x)$ are 1 for $i = 1, \cdots, k$. If this condition is met, x is considered to be potentially in $\mathcal{S}$. Due to potential hash collisions, BF has false positives (FP) by incorrectly reporting an element not actually in $\mathcal{S}$ as being in.

Mathematically, the probability of a specific bit remaining 0 after inserting all the n items can be approximated as $(1 - \frac{1}{m})^{n \cdot k}$, which for large m is approximately $e^{-\frac{n \cdot k}{m}}$. The false positive rate (FPR) is then given by $(1 - e^{-\frac{n \cdot k}{m}})^k$. Extensive research in this area proved that to minimize the FPR, the optimal number of hash functions is $\frac{m}{n} \cdot \ln 2$, under which the FPR can be minimized to $\alpha^{\frac{m}{n}}$, where $\alpha = (\frac{1}{2})^{\ln 2} \approx 0.6185$, cf. e.g. (Broder & Mitzenmacher, 2004). This mathematical foundation has guided the application and further development of BF in a variety of domains.

2.2. Learned Bloom Filter

LBF builds upon standard BF by integrating machine learning techniques to boost performance (Kraska et al., 2018; Mitzenmacher, 2018).

In LBF, a learning model called learning oracle, or more concisely oracle, is introduced, which assigns a score $s(x)$ to each item x. The score serves as a new dimension for determining set membership. Instead of solely relying on the bit array and hash functions as in traditional BF, LBF leverages the score in membership testing process, as illustrated in the left subfigure of Figure 1. If the score of x exceeds a predefined threshold, x is considered a member of the set. This initial assessment by the learning model can potentially reduce the need for further checks, thus speeding up the query process. On the other hand, if the score does not exceed the threshold, the item is then passed to a backup filter, a traditional BF of smaller size, for further evaluation.

The motivation behind incorporating a learning model is to leverage the power of machine learning in understanding data patterns and distributions. By doing so, LBF aims to improve the accuracy of set membership testing, especially in reducing FPR. The learning model can learn from the data's inherent structure and characteristics, allowing it to make more informed decisions about an item's membership compared to the more simplistic approach of traditional BF.

However, this integration also brings new challenges. Particularly, there could be cases where the oracle assigns a score greater than or equal to the threshold for an item that is not actually in the set, resulting in a false positive. Additionally, the backup filter can also contribute to false positives, similar to traditional BF. Therefore, LBF needs to be carefully engineered. Compared to BF, the integration of machine learning has opened up a vast and complex design space for LBF, by incorporating different learning models, thresholding strategies, and backup filter configurations, rendering optimizing LBF a non-trivial task.

Despite the above challenges, LBF offers a promising direction for more intelligent and efficient set membership testing, and research efforts have been focused on optimizing their performance and minimizing the FPR, which we summarize in next subsection.

Table 1. Major notations.

Major notations in Section 3	
n	number of items
m	space size of backup filter
τ	threshold
p	the membership score distribution
q	the nonmembership score distribution
r	number of regions
k_j	number of hash functions of items in region j
β	prob. of an arbitrary bit of the filter set to 1
$D_{p\|\|q}$	KL divergence between p and q
ϵ	approximation parameter
C	$m \cdot \ln 2/n$
n_s	number of distribution segments
Δ_a^b	KL divergence of p, q between segments a, b
Additional notations in Section 4	
g	number of region groups
$\mathcal{G}_i$	set of regions of group i

2.3. Prior Arts in Optimizing Learned Bloom Filter

Since the introduction of LBF, a palette of propositions are developed in the literature to optimize its performance by exploring the corresponding design space.

One of the first efforts in optimizing LBF is the solution developed in (Vaidya et al., 2021) called partitioned LBF (PLBF). As indicated by its name, PLBF divides the score of the learning oracle into distinct regions and dedicates a separate backup filter for each region. The main technical part in their work consists of optimizing the thresholds separating the regions. Sato and Matsui further improved partitioned LBF by developing a fast construction method without sacrificing the query accuracy (Sato & Matsui, 2023). Liu *et al.* developed stable LBF for data streams where member updates are frequent (Liu et al., 2020).

Another notable optimization approach of LBF, arguably the closest to our work, is the adaptive LBF (Ada-BF) developed in (Dai & Shrivastava, 2020). In Ada-BF, the score space is partitioned into multiple regions as in PLBF. Meanwhile, (Dai & Shrivastava, 2020) explores two architectures: Canonical Ada-BF let all the regions share a common backup filter, as illustrated in the middle subfigure of Figure 1; Disjoint Ada-BF attributes a separate backup filter to each region, as illustrated in the right subfigure of Figure 1. In both cases, (Dai & Shrivastava, 2020) focuses on optimizing the parameters associated to each backup filter by letting items of different regions use different number of hash functions. However, both Ada-BF and disjoint Ada-BF rely on heuristic-based methods for parameter tuning. These methods are designed based on empirical observations and do not have any theoretical guarantee of achieving the best performance across all possible data distributions and application scenarios.

In summary, the current state of the art still lacks of theoretical framework on how to optimize LBF, which is precisely what we address in this paper.

3. Optimal Learned Bloom Filter Design: the Case of Single Backup Filter

3.1. System Model and Problem Formulation

We consider a generic LBF setting where we dispose an oracle that, given an input item x, outputs a score $s(x) \in [0,1]$ representing with higher score indicating higher likelihood of the item being in $\mathcal{S}$. The score space $[0,1]$ is divided into r regions. An item x is in region j if $s(x) \in [\tau_{j-1}, \tau_j), j = 1, \cdots, r$, where $0 = \tau_0 < \cdots < \tau_r = 1$ denote the thresholds separating the regions. Let $\mathcal{D}$ denote the distribution of items over the universe. Let p and q denote the membership score distribution and nonmembership score distribution, respectively, denoted as follows.

$$p_j \triangleq \Pr_{x\sim\mathcal{D}}\left(\tau_{j-1} \le s(x) < \tau_j \middle| x \in \mathcal{S}\right),$$
$$q_j \triangleq \Pr_{x\sim\mathcal{D}}\left(\tau_{j-1} \le s(x) < \tau_j \middle| x \notin \mathcal{S}\right).$$

Let k_j denote the number of hash functions of the items in region j. We can derive the probability that an arbitrary bit of the backup filter is set to 1, denoted by β, as follows.

$$\beta = 1 - \left(1 - \frac{1}{m}\right)^{\sum_{j=1}^{r} p_j \cdot n \cdot k_j}.$$

In the above formula, m denotes the size of the backup filter; $1 - \frac{1}{m}$ is thus the probability that a bit is not hit by a hash. In each region j, a nonmember item is wrongly classified as member if all its k_j hashprints are 1, whose probability is β^{k_j}. The overall FPR, denoted by Φ, can be derived as

$$\Phi = \sum_{j=1}^{r} q_j \cdot \beta^{k_j}.$$

Our goal is to minimize Φ by finding the optimal LBF configuration including the region borders $\boldsymbol{\tau} \triangleq [\tau_0, \cdots, \tau_r]$ and the number of hash functions for each region $\mathbf{k} \triangleq [k_1, \cdots, k_r]$. Mathematically, our problem is formulated as:

$$\mathbf{P}: \min_{\boldsymbol{\tau}} \min_{\mathbf{k}} \quad \sum_{j=1}^{r} q_j \cdot \beta^{k_j}$$
$$\text{subject to:} \quad \sum_{j=1}^{r} p_j \cdot k_j = \frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})}$$
$$0 = \tau_0 < \cdots < \tau_r = 1$$
$$k_j \in \mathbb{N},\ j = 1, \cdots, r.$$

The first constraint of $\mathbf{P}$ follows from the definition of β through straightforward algebraic operations. This formulation sets the foundation for our subsequent exploration of the optimal LBF design with a single backup filter.

3.2. Solving the Inner Optimization Problem

We start by solving the inner minimization problem under fixed τ. Technically, we first give a 2-approximate solution by solving the LP relaxation of the problem and then perform rounding. Analyzing the LP relaxation not only leads to a 2-approximate solution, but also reveals structural insights into LBF design. We then develop a randomized approximation algorithm providing additive performance guarantee not exceeding $\Phi^* + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}$. To conclude, we develop a deterministic algorithm based on dynamic programming based ensuring essentially the same additive performance guarantee. By combining the multiplicative and additive approximation algorithms, we develop an algorithmic framework with both multiplicative and additive performance guarantee.

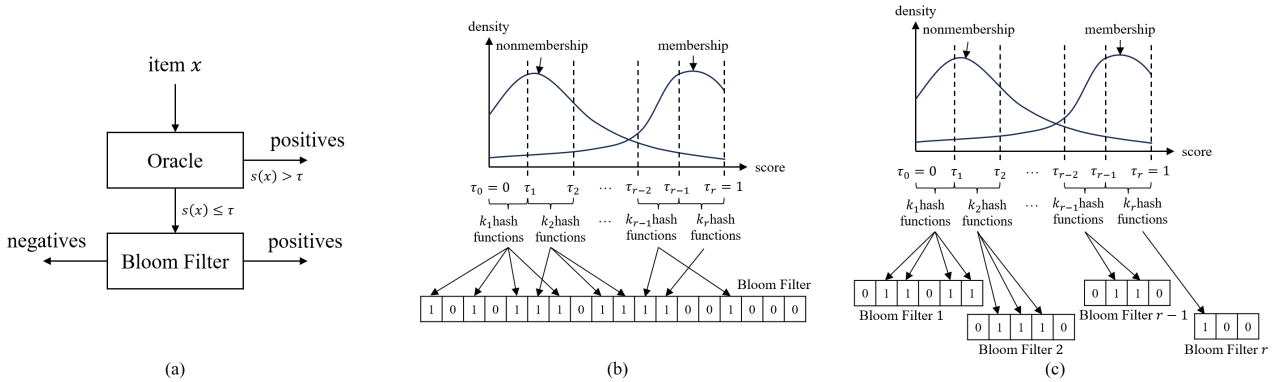

Figure 1. Illustration of (a) original LBF architecture; (b) Ada-BF: all regions share a common backup filter; (c) disjoint Ada-BF: each region is attributed a dedicated backup filter.

3.2.1. A 2-APPROXIMATE SOLUTION

We examine the LP relaxation of the inner minimization problem stated below:

$$\mathbf{P_1}: \min_{k_j} \quad \Phi(\mathbf{k}) \triangleq \sum_{j=1}^{r} q_j \cdot \beta^{k_j}$$
$$\text{subject to:} \quad \sum_{j=1}^{r} p_j \cdot k_j = \frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})}.$$

We note that the constraint $k_j \geq 0$ is not necessary in that any negative k_j would not contribute to minimize the objective function. We derive the solution of the above problem in Theorem 3.1. To make the presentation streamlined, we assemble all the proofs in the Appendix.

Theorem 3.1. *Let $D_{p||q} \triangleq \sum_{j=1}^{r} p_j \log_2 \frac{p_j}{q_j}$ denote the Kullback-Leibler (KL) divergence between p and q. The solution of $\mathbf{P_1}$ is given by*

$$k_j^* = \frac{m}{n} \cdot \ln 2 + D_{p||q} + \log_{\frac{1}{2}} \frac{p_j}{q_j}.$$

The minimal objective value, achieved when $\beta^* = \frac{1}{2}$, is given by

$$\Phi(\mathbf{k}^*) = \left(\frac{1}{2}\right)^{\frac{m}{n} \cdot \ln 2} \cdot \left(\frac{1}{2}\right)^{D_{p||q}}.$$

Theorem 3.2 implies that, without considering the size of the oracle, the FPR lower-bound of LBF degenerates to standard BF when $p = q$, i.e, when the membership score distribution and the nonmembership score distribution are fully overlapped with each other. When taking into consideration the oracle size denoted by l_o, the following inequality must be hold to ensure that LBF outperforms standard BF:

$$\left(\frac{1}{2}\right)^{\frac{m}{n} \cdot \ln 2} \cdot \left(\frac{1}{2}\right)^{D_{p||q}} < \left(\frac{1}{2}\right)^{\frac{m+l_o}{n} \cdot \ln 2}.$$

The right-hand side represents the FPR of standard BF. The condition can be more conveniently written as

$$l_o < \frac{n \cdot D_{p||q}}{\ln 2}.$$

We now demonstrate that rounding down k_j^* derived in Theorem 3.1 to its leftmost integer, which we term the rounding-down strategy, yields a 2-approximate solution.

Theorem 3.2. *The rounding-down strategy outputs a 2-approximate solution of $\mathbf{P_1}$.*

3.2.2. A RANDOMIZED APPROXIMATE SOLUTION WITH ADDITIVE PERFORMANCE GUARANTEE

Noting that $\Phi \in (0, 1)$, more strictly $(0, 0.5)$, as beyond 0.5 there are more false positives than true positives, the rounding-down strategy may fail to provide satisfactory performance when Φ^* is relatively large. Generically, any strategy providing multiplicative guarantee exhibits this limitation. In this subsection, we develop an approximation algorithm providing additive performance guarantee not exceeding $\Phi^* + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}$. We demonstrate that, under typical parameter settings, when Φ^* is relatively large, $\phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}} \ll \Phi^*$. Therefore, our algorithm offers a complementary solution to the rounding-down strategy.

Technically, we formulate a tighter version of $\mathbf{P_1}$ by introducing a discount factor $\gamma \in (0, 1)$, usually slightly smaller than 1, to the main constraint.

$$\mathbf{P_2}: \min_{k_j} \quad \sum_{j=1}^{r} q_j \cdot \left(1 - \frac{1}{2^\gamma}\right)^{k_j}$$
$$\text{subject to:} \quad \sum_{j=1}^{r} p_j \cdot k_j = \gamma \cdot \frac{m}{n} \cdot \ln 2 \triangleq \gamma \cdot C$$

To gain better insight of $\mathbf{P_2}$, we perform the following

algebraic manipulation:

$$1-\frac{1}{2^{\gamma}}=1-e^{-\gamma\cdot\ln 2}=1-\left(1-\frac{1}{m}\right)^{\sum_{j=1}^{r}p_j\cdot n\cdot k_j}.$$

The second equality follows from the constraint in $\mathbf{P_2}$. It follows from the analysis in Section 3.1 that the right-hand side $1-\left(1-\frac{1}{m}\right)^{\sum_{j=1}^{r}p_j\cdot n\cdot k_j}$ represents the probability of an arbitrary bit of the backup filter being set to 1. Hence, the objective function of $\mathbf{P_2}$ precisely represents the overall FPR. In other words, $\mathbf{P_2}$ seeks to minimize the overall FPR under a slightly tighter constraint.

We then solve $\mathbf{P_2}$ and denote the solution as $\mathbf{k^0}\triangleq\{k_j^0\}$. We further apply the following randomized rounding mechanism to obtain an integer solution $\hat{\mathbf{k}}\triangleq\{\hat{k}_j\}$:

$$\hat{k}_j=\begin{cases}\lfloor k_j^0\rfloor+1 & \text{with probability } k_j^0-\lfloor k_j^0\rfloor,\\ \lfloor k_j^0\rfloor & \text{with probability } 1+\lfloor k_j^0\rfloor-k_j^0.\end{cases}$$

Given a real number $\phi>0$, we define

$$\xi(\gamma)\triangleq e^{(-(1-p_{max})\cdot(1-\gamma)\cdot C)}+\frac{\left(\frac{2^{\gamma}}{2^{\gamma}-1}\right)^{2^{\gamma}}\cdot\left(1-\frac{1}{2^{\gamma}}\right)^{\gamma\cdot C}}{e\cdot 2^{\gamma-C}\cdot\left(\gamma\cdot\ln 2-\ln(2^{\gamma}-1)\right)\cdot(2^{C}\cdot\phi+1)}$$

where $p_{max}\triangleq\max_{j=1}^{r}p_j$. The following theorem establishes the optimality of $\hat{\mathbf{k}}$.

Theorem 3.3. *For any ϵ and γ satisfying $\xi(\gamma)\le 1-\epsilon$, we can guarantee, with probability ϵ, $\hat{\mathbf{k}}$ is a feasible solution of $\mathbf{P}$ and $\Phi(\hat{\mathbf{k}})\le\Phi(\mathbf{k}^*)+\phi\cdot\left(\frac{1}{2}\right)^{D_{p||q}}$.*

In practice, we often encounter the following two scenarios. First, we may have a target ϕ that we want to achieve. In this case, Theorem 3.3 can be applied to find γ meeting the performance requirement. Specifically, noticing $\xi(0)<0$, we distinguish two cases. If $\xi(\gamma)<0$ for all $\gamma\in(0,1)$, the target ϕ cannot be achieved; Otherwise, we can choose a sufficiently small ϵ and search γ^* satisfying $\xi(\gamma^*)\ge\epsilon$; setting $\gamma=\gamma^*$ ensures the target ϕ can be achieved with probability at least ϵ. Second, we may want to achieve minimal ϕ. In this case, we can begin by setting ϕ to a small value and gradually increase ϕ; for each ϕ we try to find γ leading to $\xi(\gamma)\ge\epsilon$; we stop as soon as such γ is found. In both scenarios, repeating the process $\Theta(\epsilon^{1+a})$ times with different seeds allows to obtain at least one $\hat{\mathbf{k}}$ achieving $\Phi(\hat{\mathbf{k}})\le\Phi(\mathbf{k}^*)+\phi\cdot\left(\frac{1}{2}\right)^{D_{p||q}}$ with probability $\Omega(\epsilon^{-a})$.

We conclude by remarking that the approximation algorithms developed in Section 3.2.1 and 3.2.2 provide multiplicative and additive performance guarantee, respectively. They are mutually complementary in the sense that, when $\Phi^*\le\phi\cdot\left(\frac{1}{2}\right)^{D_{p||q}}$, the 2-approximate algorithm has better theoretical guarantee; otherwise, the randomized algorithm is more theoretically backed-up. By invoking both algorithms, we can ensure an FPR upper-bound $\min\left\{2\Phi^*,\Phi^*+\phi\cdot\left(\frac{1}{2}\right)^{D_{p||q}}\right\}$.

3.2.3. A Deterministic approximate solution by Dynamic Programming

In this subsection, we further develop a deterministic $(1+\epsilon)$-approximate solution by leveraging dynamic programming techniques tailored to our problem.

Recalling Theorem 3.1 that $\beta^*=\frac{1}{2}$, we reformulate $\mathbf{P_1}$ as the following integer programming problem:

$$\begin{aligned}\mathbf{P_3}:\ \min_{k_j}\quad & \sum_{j=1}^{r}q_j\cdot\left(\frac{1}{2}\right)^{k_j}\\ \text{subject to:}\quad & \sum_{j=1}^{r}p_j\cdot k_j\le\frac{m}{n}\cdot\ln 2=C\\ & k_j\in\mathbb{N},\quad j=1,\cdots,r.\end{aligned}$$

We define the value function $v(z,c)$ as follows:

$$v(z,c)\triangleq\min_{\mathbf{k}\in\mathbb{N}^z}\left\{\sum_{j=1}^{z}q_j\cdot\left(\frac{1}{2}\right)^{k_j}:\ \sum_{j=1}^{z}p_j\cdot k_j\le c\right\}.$$

The original inner minimization problem $\mathbf{P_1}$ can be mapped to computing $v(r,C)$. To solve this problem, we develop a dynamic programming mechanism by leveraging the following recursion on the value of k_z:

$$v(z,c)=\min_{t\in[0,c/p_z]\cap\mathbb{Z}}q_z\cdot\left(\frac{1}{2}\right)^{t}+v(z-1,c-p_z\cdot t).$$

We discretize c as $c_i\triangleq i\delta_1$ with stepsize δ_1 to run the above dynamic program. Theorem 3.4 demonstrates that, under mild condition on δ_1, our dynamic program algorithm approaches the performance achieved by the previous randomized algorithm with deterministic guarantee.

Theorem 3.4. *Under the condition $\delta_1\le p_{min}\cdot\log(1+\epsilon)$ where $p_{min}\triangleq\min_{j=1}^{r}p_j$, i.e., $\delta_1=O(\epsilon)$, it holds that $v(r,\lfloor\frac{C}{\delta_1}\rfloor)\le(1+\epsilon)\cdot\left(\Phi(\mathbf{k}^*)+\phi\cdot\left(\frac{1}{2}\right)^{D_{p||q}}\right)$.*[1]

We conclude by analyzing the complexity of our algorithm. In the dynamic program, we have two dimensions z and c, ranging from 1 to r and 0 to $\left\lfloor\frac{C}{\delta_1}\right\rfloor$, respectively. For each pair (z,c), when calculating $v(z,c)$, we need to traverse $t\in\left[1,\frac{c}{p(z)}\right]\cap\mathbb{Z}$. Therefore, the total time complexity sums up

[1] Throughout our analysis log is taken with base 2.

to $O\left(r \cdot \frac{C}{\delta_1} \cdot \frac{C}{\delta_1 \cdot p_{min}}\right)$, which approximates to $O\left(\frac{r \cdot C^2}{\epsilon^2}\right)$ under the condition of Theorem 3.4. Similarly, we can derive the overall space complexity as $O\left(\frac{r \cdot C}{\epsilon}\right)$.

3.3. Solving the Outer Optimization Problem

Building upon the solution obtained previously for the inner optimization problem, we now direct our attention to the outer optimization problem. By eliminating the constant term, the outer optimization problem is transformed into the following KL divergence maximization problem.

$$\mathbf{Q}: \max_{\boldsymbol{\tau}} \quad \sum_{j=1}^{r} p_j \log \frac{p_j}{q_j}$$
$$\text{subject to:} \quad 0 = \tau_0 < \cdots < \tau_j = 1$$

To solve $\mathbf{Q}$, we discretize the threshold range $[0, 1]$ with stepsize δ_2 to obtain $n_s \triangleq \lceil \delta_2^{-1} \rceil$ segments. Let $D(i, j)$ denote the maximal KL divergence that we can achieve by dividing the first to j segments into i regions, where $j \leq n_s$ and $i \leq r$. Furthermore, let Δ_a^b denote the KL divergence of p and q between segment a and b, i.e.,

$$\Delta_a^b \triangleq \left(\sum_{z=a+1}^{b} p_z\right) \log \frac{\sum_{z=a+1}^{b} p_z}{\sum_{z=a+1}^{b} q_z}.$$

We can establish the following recursive formula:

$$D(i, j) = \max\Big\{ \max_{1 \leq z \leq i} D(z-1, j-1) + \Delta_{j-1}^{j}, \; D(i, j-1) + \Delta_{s_i}^{j} \Big\}.$$

In the above formula, s_i represents the start point of region i. The two terms within the right-hand side brace correspond to two different strategies that may update $D(i, j)$: (1) starting a new region and (2) extending the rightmost region by a single segment.

The standard approach employed in the literature is to compute the above DP table $D(i, j)$ by recursion. However, this method leads to time and space complexity that is quadratic in terms of n_s, more precisely $O\left(r \cdot n_s^2\right)$ in the most generic case[2]. In our work, we develop a novel algorithm exhibiting pseudo-linear complexity in r and linear in n_s without relying on any specific assumption.

Data structure: We maintain a list $\mathcal{L}$ of tuples (i, d, Γ), where each tuple corresponds to a region division strategy. Here, i denotes the number of regions from segment 1 to the current segment being scanned by our algorithm under the corresponding strategy; d denotes the corresponding KL divergence; Γ denotes the corresponding division strategy. The tuples are indexed by the pair (i, d). For two tuples $l_1 \triangleq (i_1, d_1, \Gamma_1)$ and $l_2 \triangleq (i_2, d_2, \Gamma_2)$, we define that l_1 dominates l_2 if $d_1 \geq d_2$ and $i_1 \leq i_2$. The dominance is considered strict if at least one inequality holds strictly.

Algorithm: Our algorithm proceeds by sequentially scanning each segment. When scanning segment j, we update the list $\mathcal{L}$ as follows.

- For each entry $l \triangleq (i, d, \Gamma) \in \mathcal{L}$, we build two new entries. The first corresponds to extending the last region of Γ by one segment, denoted as Γ_1; The second corresponds to starting a new region from Γ, denoted as Γ_2. Subsequently, we calculate the KL divergence for Γ_1 and Γ_2, denoted as d_1 and d_2, respectively. We then add (i_1, d_1, Γ_1) and (i_1+1, d_2, Γ_2) into $\mathcal{L}$. Finally, we remove the original entry l from $\mathcal{L}$.
- Once the processing of all the old entries in $\mathcal{L}$ is completed, we update $\mathcal{L}$ by removing dominated entries and those entries with more than r regions.

After all the segments have been processed, we output the entry in $\mathcal{L}$ possessing the maximal KL divergence along with the corresponding region division strategy.

Our algorithm yields a $(1-\epsilon)$-approximate solution of $\mathbf{Q}$, as stated in Theorem 3.5.

Theorem 3.5. *Let* $p_{max} \triangleq \max_{1 \leq j \leq r}\{p_j\}$, $q_{max} \triangleq \max_{1 \leq j \leq r}\{q_j\}$, $q_{min} \triangleq \min_{1 \leq j \leq r}\{q_j\}$ *and* $\mu \triangleq \min_{j: p_j \neq q_j} |p_j - q_j|$. *Under the condition* $\delta_2 \leq \frac{\epsilon \cdot \mu^2}{r(p_{max}+q_{max})\left(\left|\log \frac{p_{max}}{q_{min}}+1\right|+\frac{p_{max}}{q_{min}}\right)}$, *asymptotically* $\delta_2 = O(\epsilon)$, *the above dynamic program outputs a* $(1-\epsilon)$*-approximate solution of* $\mathbf{Q}$.

Algorithm 1 Solving the outer optimization problem

Input: p, q, δ_2
Output: a segment division strategy

Initialization: $\mathcal{L} \leftarrow \{(0, 0, \emptyset)\}$
for $j = 1$ **to** $\lceil \delta_2^{-1} \rceil$ **do**
  for each $l \in \mathcal{L}$ **do**
    build a new segment division strategy by extending the last region of Γ by one segment, termed as Γ_1
    compute the KL divergence of Γ_1, denoted by d_1
    add (i, d_1, Γ_1) into $\mathcal{L}$
    build a new segment division strategy by starting a new region to Γ, termed as Γ_2
    compute the KL divergence of Γ_2, denoted by d_2
    add $(i+1, d_2, \Gamma_2)$ into $\mathcal{L}$
    remove l from $\mathcal{L}$
  end for
  remove all strictly dominated entries in $\mathcal{L}$
end for
return the entry in $\mathcal{L}$ with maximal KL divergence

[2]The approach developed in (Sato & Matsui, 2023) achieves $O\left(r \cdot n_s \cdot \log n_s + r^2 \cdot n_s\right)$ by requiring specific structural properties on p and q, otherwise the complexity is $O\left(r \cdot n_s^2\right)$.

We proceed to analyze the complexity of our algorithm. The list $\mathcal{L}$ contains r entries, excluding Γ which takes $O(r)$ space. Considering that Γ maintains up to r thresholds, the overall space complexity amounts to $O(r^2)$. For each segment, the process of building new strategies for each entry requires $O(r)$ operations; the task of removing the dominated entries necessitates $O(r \cdot \log r)$ operations through merge sorting. As there are $\lceil \delta_2^{-1} \rceil$ segments, the overall time complexity of our algorithm sums up to $O\left(r \cdot \log r \cdot \delta_2^{-1}\right)$.

3.4. Solving the Overall Optimization Problem

After analyzing both the inner and outer optimization problem, it is time for us to solve the overall optimization problem. To this end, we run Algorithm 3.5 to set τ and the algorithms developed in Section 3.2. In the following theorem, we establish the approximation ratio of our comprehensive algorithmic framework.

Theorem 3.6. *For small $\epsilon > 0$, by setting $\delta_2 = O(\epsilon)$, the approximation ratio is $2 + \epsilon$ if we run the rounding-down strategy for the inner problem and it holds that $\hat{OPT} \leq (1+\epsilon)\cdot\left(OPT^* + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q,max}}\right)$ if we run the randomized and deterministic strategies.*

As a concluding remark, the only parameter within the entire LBF design space that remains unoptimized in our algorithmic framework is the number of regions r. It can be easily observed that increasing r leads to more fine-grained exploration of the design space that in turn can contribute to the reduction of FPR. However, this comes at the expense of increased time and space complexity. One potential approach for determining the value of r is to incrementally increase r until the resulting FPR satisfies the specific requirements of the application in question.

4. Optimal Learned Bloom Filter Design: the Case of Multiple Backup Filters

In Section 3, we delved into the design where a single backup filter is shared among all regions. In this section, we explore a more generic design question:

Can we further reduce FPR by employing multiple backup filters in LBF design?

An extreme example of multiple backup filter design is disjoint Ada-BF proposed in (Dai & Shrivastava, 2020), where each region is equipped with its own dedicated backup filter. A more generalized design, which we will analyze in detail hereinafter, involves assigning a backup filter to a subset of regions. Through our analysis in this section, we will demonstrate that the single backup filter design indeed achieves the optimal FPR, thereby showing that the deployment of multiple backup filters is not advantageous.

We start by formalizing the multiple backup filter LBF optimal design problem. Similar to the previous model, we divide the scores returned by the oracle into multiple regions. However, in the multiple backup filter case, we further cluster these regions into g groups. For each group i, we allocate a backup filter of m_i bits. It should be noted that the regions within a group are not necessarily contiguous. This multiple backup filter LBF design encompasses two degenerate cases. When $g = 1$, it reverts back to the single backup filter scenario that we have previously analyzed. On the other hand, when $g = r$, i.e., each group of regions has a dedicated backup filter, it corresponds to disjoint Ada-BF studied in (Dai & Shrivastava, 2020).

Let $\mathcal{G}$ denote a partition of regions representing a grouping strategy and $\mathcal{G}_i$ denote the set of regions belonging to group i in partition $\mathcal{G}$. Let β_i denote the probability that an arbitrary bit of the backup filter $\mathcal{G}_i$ is set to 1. The probability β_i can be computed as follows:

$$\beta_i = 1 - \left(1 - \frac{1}{m_i}\right)^{\sum_{j\in\mathcal{G}_i} p_j \cdot n \cdot k_j}.$$

We can derive the overall FPR as $\sum_{i=1}^{g} \sum_{j\in\mathcal{G}_i} q_j \cdot \beta_i^{k_j}$. Let $\mathcal{P}$ denote the set of all possible region partitions. We can then formulate the optimal LBF design as the following optimization problem:

$$\begin{aligned}
\mathbf{P_g}: \min_{\boldsymbol{\tau}} \min_{\mathcal{C}\in\mathcal{P}} \min_{\mathbf{k}} \quad & \sum_{i=1}^{g} \sum_{j\in\mathcal{G}_i} q_j \cdot \beta_i^{k_j} \\
\text{subject to:} \quad & \sum_{j\in\mathcal{G}_i} p_j \cdot k_j = \frac{\ln(1-\beta_i)}{n \ln\left(1-\frac{1}{m_i}\right)}, \\
& \qquad\qquad i = 1, \cdots, g \\
& \sum_{i=1}^{g} m_i = m \\
& 0 = \tau_0 < \cdots < \tau_j = 1 \\
& k_j \in \mathbb{N}, \quad j = 1, \cdots, r.
\end{aligned}$$

The first constraint of $\mathbf{P_g}$ follows from the definition of β_i through straightforward algebraic operations.

We now examine the LP relaxation of the inner minimization problem of $\mathbf{P_g}$. The following theorem states the main result of this section.

Theorem 4.1. *The solution of the LP relaxation of the inner minimization problem of $\mathbf{P_g}$ is given by*

$$k_j^* = \frac{m}{n} \cdot \ln 2 + D_{p||q} + \log_{\frac{1}{2}} \frac{p_j}{q_j}.$$

The minimal objective value, denoted by Φ_g^*, achieved when $\beta_i^* = \frac{1}{2}$ for each group i, is given by

$$\Phi_g^* = \left(\frac{1}{2}\right)^{\frac{m}{n}\cdot\ln 2} \cdot \left(\frac{1}{2}\right)^{D_{p||q}}.$$

Theorem 4.1 reveals that the FPR lower-bound in the case of multiple backup filters is the same as the case of single backup filter. Since this lower-bound is achievable with only ϵ performance loss using our algorithms developed in the previous section, we have conclusively shown that deploying multiple backup filters does not offer any benefits. On the contrary, our analysis indicates that sharing a single backup filter for all the regions suffices to achieve optimal performance, leading to a neat and theoretically backed-up architecture for LBF.

5. Experiments

We conduct empirical experiments on a standard off-the-shelf desktop computer equipped with an Intel(R) Core i7–12700F CPU running at 2.10 GHz and 16 GB of RAM. The aim is to evaluate our proposed algorithmic framework against state-of-the-art solutions on LBF design.

5.1. Experiment Setting

We focus on two practical data analysis tasks, each utilizing real-world datasets.

Malicious URLs Detection. BFs offer an efficient probabilistic solution to quickly check whether a URL belongs to those flagged as malicious with limited memory overhead. We downloaded a real-world dataset from Kaggle (Pilarpieiro, 2024), containing 1,682,213 URLs, 21.5% of which are malicious, others are benign. A total of 54 numerical features are extracted from these URLs, including URL length, URL unique character ratio and so on.

Virus Signature Scan. BFs are widely used in virus signature scan, which, given a suspect file, checks whether the file's signature is present in the virus signature database. We use a real-world dataset EMBER (Anderson & Roth, 2018), an open-source collection of 1M sha256 file hashes. The dataset includes 400K malicious, 400K benign and 200K unlabeled files, all scanned by VirusTotal in 2018. We ignore unlabeled files. The 2381 features of the files are included in the dataset, from which we randomly select 23 features.

Following the literature, we choose the random forest classifier from sklearn (Pedregosa et al., 2011) as oracles [3]. We run our combinatorial algorithm architecture with different strategies solving the inner optimization problem. We compare these three corresponding configurations of LBF parameters against major state-of-the-art propositions, i.e., the standard BF (Bloom, 1970), Ada-BF (Dai & Shrivastava, 2020), Fast PLBF and Fast PLBF++ (Sato & Matsui, 2023).

[3] In both tasks, we randomly sample 30% data to train learned oracle, which consists 15 decision trees and at most 5 leaf nodes for each tree as in the previous paper (Dai & Shrivastava, 2020); the oracle size is 140Kbits and 128Kbits, respectively.

We ignore comparing against PLBF (Vaidya et al., 2021) as it is completely replaced by Fast PLBF (Sato & Matsui, 2023). We set $\epsilon = 0.01$, $r = 5$ and $N = 5000$.

5.2. Experiment Results

5.2.1. Time Overhead

We compare the time overhead of the evaluated solutions under 2000Kbits memory budget, by time overhead we mean the time to compute the optimal configuration of the entire data structure including the threshold partition, the number of hash functions, etc. We find the optimal threshold of LBF at $[0.5, 0.9]$ with a stepsize of 0.1, and the optimal hyper-parameter c of Ada-BF at $[1, 3]$ with a stepsize of 0.1, as they rely on heuristics. Figure 2 shows the time overhead of outstanding evaluated solutions including Fast PLBF++ and our methods, which excludes the others due to their much higher time overhead, i.e., 14.5 and 6.4 seconds for LBF, 327.9 and 175.9 seconds for Ada-BF, 34.0 and 32.9 seconds for Fast PLBF on the two tasks, respectively. We report that the time overhead of the standard BF is negligible, which only has an insertion time of 0.416 and 0.475 seconds on the two tasks, respectively. We make the following observations from the results. (1) Our methods are 2.8-3.4x faster than Fast PLBF++, which gives the credit to our algorithm for the outer optimization problem. (2) Our rounding-down strategy and randomized strategy have the same time overhead, while our deterministic strategy is slower, as the inner dynamic program is operated to find the optimal number of hashprints for each region.

5.2.2. Memory Overhead and FPR

We compare FPR of the evaluated solutions under different memory overhead, as shown in Figure 3. We make the following observations from the results. (1) Our propositions almost perform the same as PLBFs and outperforms BF, LBF and Ada-BF for all the memory overhead. (2) Our deterministic algorithm have better Pareto curves than our rounding-down strategy and randomized strategy, which confirms their theocratically guaranteed optimality. (3) As the memory overhead increases, the performance improvement of Ada-BF gradually slows down and becomes worse than that of BF, which is attributed to its heuristics.

5.2.3. Ablation Study for Hyper-parameters

We also perform ablation studies for hyper-parameters of our propositions, including approximation factor ϵ, number of regions r and number of segments N, as presented in Appendix H. The results demonstrates the trade-off between the optimality and complexity of our algorithms.

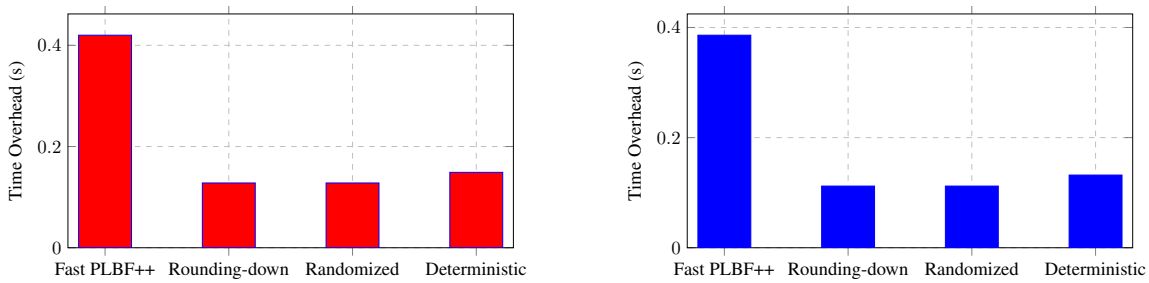

Figure 2. Time overhead: Malicious URLs Detection (left), Virus Signature Scan (right).

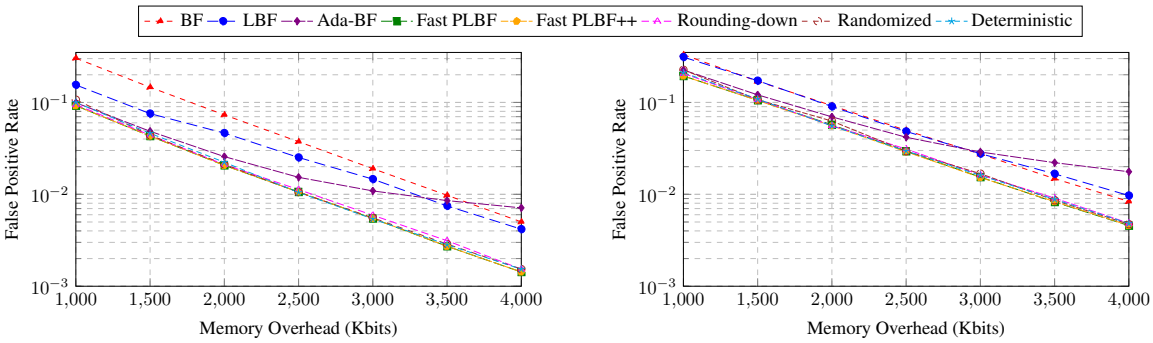

Figure 3. Trade-off between memory overhead and FPR: Malicious URLs Detection (left), Virus Signature Scan (right).

6. Conclusion

In this paper, we have presented a combinatorial algorithmic framework on optimal LBF design that computes the optimal parameter to minimize the false positive rate under given space budget. Armed with our theoretical framework, we prove that sharing a single backup filter for all the score regions of the learning model suffices to achieve optimal performance, therefore leading to a neat and theoretically backed-up architecture for LBF. In our future work, we plan to investigate whether our algorithmic framework can be extended to other learning-augmented data structure design and optimization.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Anderson, H. S. and Roth, P. EMBER: an open dataset for training static PE malware machine learning models. *arXiv:1804.04637*, 2018.

Bloom, B. H. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7): 422–426, 1970.

Broder, A. and Mitzenmacher, M. Network applications of Bloom filters: A survey. *Internet Mathematics*, 1(4): 485–509, 2004.

Dai, Z. and Shrivastava, A. Adaptive learned Bloom filter (Ada-BF): Efficient utilization of the classifier with application to real-time information filtering on the web. *Advances in Neural Information Processing Systems*, 33: 11700–11710, 2020.

Geravand, S. and Ahmadi, M. Bloom filter applications in network security: A state-of-the-art survey. *Computer Networks*, 57(18):4047–4064, 2013.

Gomez-Barrero, M., Rathgeb, C., Galbally, J., Fierrez, J., and Busch, C. Protected facial biometric templates based on local gabor patterns and adaptive bloom filters. In *2014 22nd International Conference on Pattern Recognition*, pp. 4483–4488, 2014.

Kraska, T., Beutel, A., Chi, E. H., Dean, J., and Polyzotis, N. The case for learned index structures. In *Proc. SIGMOD*, pp. 489–504, 2018.

Liu, Q., Zheng, L., Shen, Y., and Chen, L. Stable learned Bloom filters for data streams. *Proc. VLDB Endow.*, 13 (12):2355–2367, July 2020.

Mitzenmacher, M. A model for learned Bloom filters and optimizing by sandwiching. *Advances in Neural Information Processing Systems*, 31, 2018.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P.,

Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pilarpieiro. Tabular dataset ready for malicious url detection, 2024. URL https://www.kaggle.com/datasets/pilarpieiro/tabular-dataset-ready-for-malicious-url-detection.

Sato, A. and Matsui, Y. Fast partitioned learned Bloom filter. In *Proc. NeurIPS*, 2023.

Vaidya, K., Knorr, E., Mitzenmacher, M., and Kraska, T. Partitioned learned Bloom filters. In *Proc. ICLR*, 2021.

Xiong, S., Yao, Y., Cao, Q., and He, T. kbf: A bloom filter for key-value storage with an application on approximate state machines. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pp. 1150–1158, 2014.

A. Proof of Theorem 3.1

Using augmented Lagrangian approach, we write the following Lagrangian function for optimization:

$$\mathcal{L}(k_1, k_2, \ldots, k_g, \lambda) = \sum_{j=1}^{r} q_j \cdot \beta^{k_j} + \lambda \left(\sum_{j=1}^{r} p_j \cdot k_j - \frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})} \right),$$

where λ is the Lagrange multiplier.

The partial derivatives with respect to k_j and λ can be derived as below.

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial k_j} = q_j \cdot \beta^{k_j} \ln \beta - \lambda p_j \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} = \displaystyle\sum_{j=1}^{r} p_j \cdot k_j - \frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})} \end{cases}$$

The partial derivative for k_j equals 0 when

$$\beta^{k_j} = \frac{\lambda}{\ln \beta} \cdot \frac{p_j}{q_j}$$

We can obtain $k_j = \log_\beta \frac{\lambda}{\ln \beta} + \log_\beta \frac{p_j}{q_j}$.

Concerning the partial derivative on λ equaling 0, we have the computation as follows:

$$\begin{aligned} \sum_{j=1}^{r} p_j \cdot k_j = \frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})} &\Longrightarrow \sum_{j=1}^{r} p_j \left(\log_\beta \frac{\lambda}{\ln \beta} + \log_\beta \frac{p_j}{q_j} \right) = \frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})} \\ &\Longrightarrow \log_\beta \frac{\lambda}{\ln \beta} + \sum_{j=1}^{r} p_j \log_\beta \frac{p_j}{q_j} = \frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})} \\ &\Longrightarrow \log_\beta \frac{\lambda}{\ln \beta} + D_{p||q} \cdot \log_\beta 2 = \frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})} \\ &\Longrightarrow \log_\beta \frac{\lambda}{\ln \beta} = \frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})} + D_{p||q} \cdot \log_\beta \frac{1}{2}. \end{aligned}$$

Thus, we can derive the solution of $\mathbf{P_1}$ as

$$k_j^* = \frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})} + D_{p||q} \cdot \log_\beta \frac{1}{2} + \log_\beta \frac{p_j}{q_j}.$$

Armed with k_j^*, we can compute the objective as follows:

$$\begin{aligned} \sum_{j=1}^{r} q_j \cdot \beta^{k_j^*} &= \sum_{j=1}^{r} \beta^{\frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})}} \cdot \left(\frac{1}{2} \right)^{D_{p||q}} \cdot p_j \\ &= \beta^{\frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})}} \cdot \left(\frac{1}{2} \right)^{D_{p||q}}. \end{aligned}$$

It follows algebraically that the above is minimized when $\beta = \frac{1}{2}$. Noticing that in practice m is usually large, we have

$$\frac{\ln(1-\beta)}{n \ln(1-\frac{1}{m})} \simeq \frac{m}{n} \cdot \ln 2.$$

Therefore, the solution k_j^* can be written as

$$\begin{aligned} k_j^* &= \frac{m}{n} \cdot \ln 2 + D_{p||q} \cdot \log_\beta \frac{1}{2} + \log_\beta \frac{p_j}{q_j} \\ &= \frac{m}{n} \cdot \ln 2 + D_{p||q} + \log_{\frac{1}{2}} \frac{p_j}{q_j}. \end{aligned}$$

The corresponding objective value can be written as

$$\left(\frac{1}{2}\right)^{\frac{m}{n}\cdot \ln 2} \cdot \left(\frac{1}{2}\right)^{D_{p||q}}.$$

B. Proof of Theorem 3.2

It follows from Theorem 3.1 that

$$\left(1 - \frac{1}{m}\right)^{\sum_{j=1}^r p_j \cdot n \cdot k_j^*} = 1 - \beta^* = \frac{1}{2}.$$

Let $\hat{\Phi}$ denote the FPR obtained after rounding down all k_j. We have

$$\begin{aligned} \hat{\Phi}_1 &= \sum_{j=1}^r q_j \cdot \left(1 - \left(1 - \frac{1}{m}\right)^{\sum_{j=1}^r p_j \cdot n \cdot \lfloor k_j^* \rfloor}\right)^{\lfloor k_j^* \rfloor} \\ &< \sum_{j=1}^r q_j \cdot \left(1 - \left(1 - \frac{1}{m}\right)^{\sum_{j=1}^r p_j \cdot n \cdot k_j^*}\right)^{k_j^* - 1} \\ &= \sum_{j=1}^r q_j \cdot \left(\frac{1}{2}\right)^{k_j^* - 1} \\ &= 2\Phi^*. \end{aligned}$$

As Φ^* is the solution of the LP relaxation of the inner minimization problem and hence a lower-bound of the minimal FPR, we can conclude that $\hat{\Phi}$ is at most twice the minimal FPR, which completes our proof of Theorem 3.2.

C. Proof of Theorem 3.3

We first state an auxiliary lemma.

Lemma C.1. *$\forall \delta \in (0, 1)$, it holds that*

$$\Pr\left[\sum_{j=1}^r p_j \cdot \hat{k}_j > (1+\delta) \cdot \gamma \cdot C\right] < \left(\frac{e^{p_{max}\delta}}{(1+\delta)^{1+\delta}}\right)^{\gamma \cdot C},$$

where $p_{max} \triangleq \max_{1 \le j \le r}\{p_j\}$.

Proof. It follows from Markov inequality that

$$\begin{aligned} \Pr\left[\sum_{j=1}^r p_j \hat{k}_j > (1+\delta)\gamma C\right] &= \Pr\left[\exp\left(t \sum_{j=1}^r p_j \hat{k}_j\right) > \exp[t(1+\delta)\gamma C]\right] \\ &< \exp[-t(1+\delta)\gamma C] \cdot \mathbb{E}\left[\exp\left(t \sum_{j=1}^r p_j \hat{k}_j\right)\right]. \end{aligned}$$

Denote $y_j \triangleq k_j^0 - \lfloor k_j^0 \rfloor$. It follows from the definition of $\hat{k}_j$ that

$$\begin{aligned}
\mathbb{E}\left[\exp\left(t\sum_{j=1}^{r} p_j \hat{k}_j\right)\right] &= \prod_{j=1}^{r} y_j \exp\left[p_j t\left(\lfloor k_j^0 \rfloor + 1\right)\right] + (1-y_j)\exp\left[p_j t \lfloor k_j^0 \rfloor\right] \\
&= \prod_{j=1}^{r} \exp\left(p_j t \lfloor k_j^0 \rfloor\right)\left[y_j \exp(p_j t) + 1 - y_j\right] \\
&< \prod_{j=1}^{r} \exp\left(p_j t \lfloor k_j^0 \rfloor\right) \cdot y_j \cdot \left[\exp(p_j t) - 1\right].
\end{aligned}$$

Imposing $t = \ln(1+\delta)$, we have

$$\begin{aligned}
(1+\delta)^{-(1+\delta)\gamma C} \prod_{j=1}^{r} \exp\left(p_j t \lfloor k_j^0 \rfloor\right) \cdot y_j \cdot \left[\exp(p_j t) - 1\right] &= (1+\delta)^{-(1+\delta)\gamma C} \prod_{j=1}^{r} \exp\left(p_j t \lfloor k_j^0 \rfloor\right) \cdot y_j \cdot \left[(1+\delta)^{p_j} - 1\right] \\
&\le (1+\delta)^{-(1+\delta)\gamma C} \prod_{j=1}^{r} \exp\left(p_j t \lfloor k_j^0 \rfloor\right) \cdot y_j \cdot \delta p_j \\
&\le (1+\delta)^{-(1+\delta)\gamma C} \prod_{j=1}^{r} \exp\left(p_j t \lfloor k_j^0 \rfloor\right) \cdot \delta p_{max} \\
&\le (1+\delta)^{-(1+\delta)\gamma C} \prod_{j=1}^{r} \exp(p_j t k_j^0 \delta p_{max}) \\
&= (1+\delta)^{-(1+\delta)\gamma C} \cdot (1+\delta)^{\sum_j p_j k_j^0 \delta p_{max}} \\
&= \left[\frac{e^{p_{max}\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\gamma C}.
\end{aligned}$$

The lemma is thus proved. □

Lemma C.1 leads to the following corollary.

Corollary C.2. *The probability that* $\hat{\mathbf{k}}$ *is a feasible solution of* $\mathbf{P}$ *is at least* $1 - \exp\big(-(1-p_{max})\cdot(1-\gamma)\cdot C\big)$.

Proof. Imposing $(1+\delta)\cdot\gamma = 1$ and applying lemma C.1, we have

$$\begin{aligned}
\Pr\left[\sum_{j=1}^{r} p_j \hat{k}_j > C\right] &= \Pr\left[\sum_{j=1}^{r} p_j \hat{k}_j > (1+\delta)\gamma C\right] \\
&< \left[\frac{e^{p_{max}\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\gamma C} \\
&= \exp\Big(\big(p_{max}\cdot\delta - (1+\delta)\ln(1+\delta)\big)\cdot\gamma\cdot C\Big) \\
&= \exp\Big(\big(p_{max}\cdot\delta - \delta + o(\delta)\big)\cdot\gamma\cdot C\Big) \\
&= \exp\Big(-(1-p_{max})\cdot\delta\cdot\gamma\cdot C\Big) \\
&= \exp\Big(-(1-p_{max})\cdot(1-\gamma)\cdot C\Big).
\end{aligned}$$

The corollary is proved. □

We proceed to prove the main theorem.

Recall the proof of Theorem 3.1. We can solve $\mathbf{k^0}$ similarly as follows.

$$k_j^0 = \gamma \cdot C + D_{p||q} \cdot \log_{1-\frac{1}{2^\gamma}} \frac{1}{2} + \log_{1-\frac{1}{2^\gamma}} \frac{p_j}{q_j}.$$

We then have

$$\sum_{j=1}^{r} q_j \cdot \left(1-\frac{1}{2^\gamma}\right)^{k_j^0} = \left(1-\frac{1}{2^\gamma}\right)^{\gamma \cdot C} \cdot \left(\frac{1}{2}\right)^{D_{p||q}}.$$

We further get

$$\begin{aligned}
\mathbb{E}\left[\sum_{j=1}^{r} q_j \cdot \left(1-\frac{1}{2^\gamma}\right)^{\hat{k}_j}\right] &= \sum_{j=1}^{r} q_j \cdot \mathbb{E}\left[\left(1-\frac{1}{2^\gamma}\right)^{\hat{k}_j}\right] \\
&= \sum_{j=1}^{r} q_j \cdot \left(\left(1-\frac{1}{2^\gamma}\right)^{\lfloor k_j^0 \rfloor + 1} \cdot (k_j^0 - \lfloor k_j^0 \rfloor) + \left(1-\frac{1}{2^\gamma}\right)^{\lfloor k_j^0 \rfloor} \cdot (1 - (k_j^0 - \lfloor k_j^0 \rfloor))\right) \\
&= \sum_{j=1}^{r} q_j \cdot \left(1-\frac{1}{2^\gamma}\right)^{\lfloor k_j^0 \rfloor} \cdot \left(1 - \frac{1}{2^a}(k_j^0 - \lfloor k_j^0 \rfloor)\right) \\
&= \sum_{j=1}^{r} q_j \cdot \left(1-\frac{1}{2^\gamma}\right)^{k_j^0} \cdot \left(1-\frac{1}{2^\gamma}\right)^{-(k_j^0 - \lfloor k_j^0 \rfloor)} \cdot \left(1 - \frac{1}{2^\gamma}(k_j^0 - \lfloor k_j^0 \rfloor)\right) \\
&\overset{x \triangleq k_j^0 - \lfloor k_j^0 \rfloor}{=} \sum_{j=1}^{r} q_j \cdot \left(1-\frac{1}{2^\gamma}\right)^{k_j^0} \cdot \left(1-\frac{1}{2^\gamma}\right)^{-x} \cdot \left(1-\frac{x}{2^\gamma}\right) \\
&\leq A_{max} \cdot \left(1-\frac{1}{2^\gamma}\right)^{\gamma \cdot C} \cdot \left(\frac{1}{2}\right)^{D_{p||q}},
\end{aligned}$$

where

$$A_{max} = \max_{x \in [0,1)} \left(1-\frac{1}{2^\gamma}\right)^{-x} \cdot \left(1-\frac{x}{2^\gamma}\right) = \frac{\left(\frac{2^\gamma}{2^\gamma - 1}\right)^{2^\gamma}}{e \cdot 2^\gamma \cdot \left(\gamma \cdot \ln 2 - \ln(2^\gamma - 1)\right)}.$$

We study the probability that $\Phi(\hat{\mathbf{k}}) > \Phi(\mathbf{k}^*) + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}$ by applying Markov inequality as follows.

$$\begin{aligned}
&\Pr\left[\Phi(\hat{\mathbf{k}}) > \Phi(\mathbf{k}^*) + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}\right] \\
&= \Pr\left[\sum_{j=1}^{r} q_j \cdot \left(1-\frac{1}{2^\gamma}\right)^{\hat{k}_j} > \sum_{j=1}^{r} q_j \cdot \left(\frac{1}{2}\right)^{k_j^*} + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}\right] \\
&< \frac{\mathbb{E}\left[\sum_{j=1}^{r} q_j \cdot \left(1-\frac{1}{2^\gamma}\right)^{\hat{k}_j}\right]}{\sum_{j=1}^{r} q_j \cdot \left(\frac{1}{2}\right)^{k_j^*} + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}} \\
&\leq \frac{A_{max} \cdot \left(1-\frac{1}{2^\gamma}\right)^{\gamma \cdot C} \cdot \left(\frac{1}{2}\right)^{D_{p||q}}}{\left(\frac{1}{2}\right)^{C} \cdot \left(\frac{1}{2}\right)^{D_{p||q}} + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}} \\
&= \frac{A_{max} \cdot \left(1-\frac{1}{2^\gamma}\right)^{\gamma \cdot C} \cdot 2^C}{2^C \cdot \phi + 1} \\
&= \frac{\left(\frac{2^\gamma}{2^\gamma - 1}\right)^{2^\gamma} \cdot \left(1-\frac{1}{2^\gamma}\right)^{\gamma \cdot C}}{e \cdot 2^{\gamma - C} \cdot \left(\gamma \cdot \ln 2 - \ln(2^\gamma - 1)\right) \cdot (2^C \cdot \phi + 1)}
\end{aligned}$$

Therefore, the probability that $\hat{\mathbf{k}}$ is a feasible solution of $\mathbf{P}$ and $\Phi(\hat{\mathbf{k}}) > \Phi(\mathbf{k}^*) + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}$ can be lower-bounded as follows.

$$
\begin{aligned}
&1 - \Pr\left[\sum_{j=1}^{r} p_j \hat{k}_j > C\right] - \Pr\left[\sum_{j=1}^{r} q_j \cdot \left(1 - \frac{1}{2^\gamma}\right)^{\hat{k}_j} > \sum_{j=1}^{r} q_j \cdot \left(\frac{1}{2}\right)^{k_j^*} + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}\right] \\
&> 1 - e^{(-(1-p_{max})\cdot(1-\gamma)\cdot C)} - \frac{\left(\frac{2^\gamma}{2^\gamma - 1}\right)^{2^\gamma} \cdot \left(1 - \frac{1}{2^\gamma}\right)^{\gamma \cdot C}}{e \cdot 2^{\gamma - C} \cdot \Big(\gamma \cdot \ln 2 - \ln(2^\gamma - 1)\Big) \cdot (2^C \cdot \phi + 1)} \\
&= 1 - \xi(\gamma) \geq 0.
\end{aligned}
$$

Theorem 3.3 is thus proved.

D. Proof of Theorem 3.4

Let $\hat{\mathbf{k}} = \{\hat{k}_j\}$ denote the output of our dynamic program and $\hat{\Phi}$ denote the corresponding FPR value. Denoting $\delta \triangleq \frac{\sum_{j=1}^{r} p_j \cdot \hat{k}_j}{C}$, we have

$$
\hat{\Phi} = \sum_{j=1}^{r} q_j \cdot (1 - \frac{1}{2^\delta})^{\hat{k}_j},
$$

It follows from the constraint in $\mathbf{P_3}$ that $\delta \leq 1$.

Let $\mathbf{k}' = \{k'_j\}$ denote the optimal integer solution of the inner optimization problem of $\mathbf{P}$ and Φ' denote the corresponding FPR value.

For each j, the error of $p_j \cdot k_j$ is within the stepsize Δ. That is, $p_j \cdot |\hat{k}_j - k'_j| \leq \Delta$, i.e., $|\hat{k}_j - k'_j| \leq \frac{\Delta}{p_j}$.

Now consider the change of the objective function value. Known that $\sum_{j=1}^{r} p_j \cdot k'_j = C$, we have

$$
\begin{aligned}
\hat{\Phi} - \Phi' &= \sum_{j=1}^{r} q_j \left((1 - \frac{1}{2^\delta})^{\hat{k}_j} - (\frac{1}{2})^{k'_j}\right) \\
&\leq \sum_{j=1}^{r} q_j \left((\frac{1}{2})^{\hat{k}_j} - (\frac{1}{2})^{k'_j}\right) \\
&\leq \sum_{j=1}^{r} q_j \left|(\frac{1}{2})^{\hat{k}_j} - (\frac{1}{2})^{k'_j}\right| \\
&\leq \sum_{j=1}^{r} q_j \left|(\frac{1}{2})^{k'_j - \frac{\Delta}{p_j}} - (\frac{1}{2})^{k'_j}\right| \\
&\leq \sum_{j=1}^{r} q_j \cdot (\frac{1}{2})^{k'_j} \cdot (2^{\frac{\Delta}{p_j}} - 1) \\
&\leq \epsilon \cdot \sum_{j=1}^{r} q_j \cdot (\frac{1}{2})^{k'_j} = \epsilon \cdot \Phi'
\end{aligned}
$$

where the last inequality follows from the condition $\Delta \leq p_{min} \cdot \log_2(1 + \epsilon)$, where $p_{min} = \min_{j=1}^{r} p_j$, asymptotically $\Delta = O(\epsilon)$. It then follows straightforwardly that the solution of our dynamic program is $(1 + \epsilon)$-approximation of the inner integer problem of $\mathbf{P}$. According to Theorem 3.3, we can derive that $\Phi' \leq \left(\Phi(\mathbf{k}^*) + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}\right)$. Thus, it holds that $\hat{\Phi} \leq (1 + \epsilon) \cdot \left(\Phi(\mathbf{k}^*) + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}\right)$

E. Proof of Theorem 3.5

Let $f(\tau_1, \cdots, \tau_j) = \sum_{j=1}^{g} p_j \log \frac{p_j}{q_j}$ be the original objective function, which is a function of the thresholds τ_j. Let $(\tau_1^*, \cdots, \tau_j^*)$ be the optimal threshold configuration in the continuous space, corresponding to the optimal value $V^* = f(\tau_1^*, \cdots, \tau_j^*)$. For the discretized case, let $(\hat{\tau}_1, \cdots, \hat{\tau}_j)$ be the optimal threshold configuration found by dynamic programming at the discrete positions, corresponding to the value $\hat{V} = f(\hat{\tau}_1, \cdots, \hat{\tau}_j)$.

Let $p_{max} \triangleq \max_{1 \le j \le r}\{p_j\}$, $q_{max} \triangleq \max_{1 \le j \le r}\{q_j\}$ and $q_{min} \triangleq \min_{1 \le j \le r}\{q_j\}$.

Consider the change of $f(\tau)$ in the threshold interval $[\tau, \tau + \delta_2]$.

When we change τ_j to $\tau_j + \delta_2$, only the values of p_j, p_{j+1}, q_j and q_{j+1} update, i.e. $p_j \log \frac{p_j}{q_j}$ and $p_{j+1} \log \frac{p_{j+1}}{q_{j+1}}$ is affected. We use the fact that the first order Taylor expansion of the function $y = p \log \frac{p}{q}$ with respect to p and q is:

$$d\left(p \log \frac{p}{q}\right) = \left(\log \frac{p}{q} + 1\right) dp - \frac{p}{q} dq$$

For any small change δ_2, the derivative of $p_j \log \frac{p_j}{q_j}$ with respect to threshold τ_j that affects p_j and q_j can be bounded as follows:

$$\left| \frac{d\left(p_j \log \frac{p_j}{q_j}\right)}{d\tau_j} \right| \le \left| \log \frac{p_{max}}{q_{min}} + 1 \right| + \frac{p_{max}}{q_{min}}$$

Since two such terms of $f(\tau)$ are updated, we can take $2\left(\left|\log \frac{p_{max}}{q_{min}} + 1\right| + \frac{p_{max}}{q_{min}}\right)$ as an upper bound for $f'(\tau)$.

By the Mean Value Theorem, for any τ and $\tau + \delta_2$, there exists $\xi \in [\tau, \tau + \delta_2]$ such that $f(\tau + \delta_2) - f(\tau) = f'(\xi) \cdot \delta_2$, which indicates

$$|f(\tau + \delta_2) - f(\tau)| \le 2\left(\left|\log \frac{p_{max}}{q_{min}} + 1\right| + \frac{p_{max}}{q_{min}}\right) \cdot \delta_2.$$

On the other hand, as each threshold can deviate by at most δ_2, we have

$$\left|V^* - \hat{V}\right| \le r \cdot 2\left(\left|\log \frac{p_{max}}{q_{min}} + 1\right| + \frac{p_{max}}{q_{min}}\right) \cdot \delta_2.$$

Denoting $\mu \triangleq \min_{j: p_j \ne q_j} |p_j - q_j|$, we have

$$\begin{aligned} V^* &= \sum_{j=1}^{r} p_j \log \frac{p_j}{q_j} \\ &\ge \sum_{j=1}^{r} \frac{(p_j - q_j)^2}{p_j + q_j} \\ &\ge \sum_{j: p_j \ne q_j} \frac{\mu^2}{p_{max} + q_{max}} \\ &\ge \frac{2 \cdot \mu^2}{p_{max} + q_{max}} \end{aligned}$$

where the first inequality follows from $a \log \frac{a}{b} \ge \frac{(a-b)^2}{a+b}, \forall a, b > 0$, and the last inequality follows from the assumption that $p \ne q$.

Hence, we have

$$\begin{aligned} V^* - \hat{V} &\leq \left| V^* - \hat{V} \right| \\ &\leq r \cdot 2 \left(\left| \log \frac{p_{max}}{q_{min}} + 1 \right| + \frac{p_{max}}{q_{min}} \right) \cdot \delta_2 \\ &\leq \epsilon \cdot \frac{2 \cdot \mu^2}{p_{max} + q_{max}} \\ &\leq \epsilon \cdot V^* \end{aligned}$$

where the third inequality follows from the condition $\delta_2 \leq \frac{\epsilon \cdot \mu^2}{r(p_{max}+q_{max})\left(\left|\log \frac{p_{max}}{q_{min}}+1\right|+\frac{p_{max}}{q_{min}}\right)}$, asymptotically $\delta_2 = O(\epsilon)$.

That is, $\frac{\hat{V}}{V^*} \geq 1 - \epsilon$, so the solution of the algorithm is $(1-\epsilon)$-approximation.

F. Proof of Theorem 3.6

We first prove the case with rounding-down strategy. Recall Theorem 3.2 that the rounding-down strategy outputs a 2-approximate solution of $\mathbf{P_1}$, which is the relaxation problem. Thus, denoting the objective value of the rounding-down strategy as Π_1, for any small ϵ' we have

$$\begin{aligned} \Pi_1 &\leq 2 \cdot \Phi^* \\ &= 2 \cdot \left(\frac{1}{2}\right)^{\frac{m}{n} \cdot \ln 2} \cdot \left(\frac{1}{2}\right)^{(1-\epsilon') \cdot D_{p||q,max}} \\ &= 2 \cdot 2^{\epsilon' \cdot D_{p||q,max}} \cdot \left(\frac{1}{2}\right)^{\frac{m}{n} \cdot \ln 2} \cdot \left(\frac{1}{2}\right)^{D_{p||q,max}} \\ &= 2 \cdot (1 + \epsilon' \cdot D_{p||q,max} \cdot \ln 2) \cdot \left(\frac{1}{2}\right)^{\frac{m}{n} \cdot \ln 2} \cdot \left(\frac{1}{2}\right)^{D_{p||q,max}}, \end{aligned}$$

where the penultimate equality can be derived by applying $2^{\epsilon' \cdot D_{p||q,max}} \sim 1 + \epsilon' \cdot D_{p||q,max} \cdot \ln 2$ while the last equality holds as the KL divergence is non-negative. By setting $\epsilon = \frac{\epsilon' \cdot D_{p||q,max} \cdot \ln 2}{2}$, we get $(2+\epsilon)-$approximation.

Similarly, we can prove the case with our randomized strategy. Recall the randomized strategy guarantees $\Phi(\hat{\mathbf{k}}) \leq \Phi(\mathbf{k}^*) + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}}$ with probability ϵ by setting γ such that $\xi(\gamma) \leq 1 - \epsilon$. Thus, denoting the objective value of the randomized strategy as Π_2, for any small ϵ' we have

$$\begin{aligned} \Pi_2 &\leq \Phi^* + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q}} \\ &= \left(\left(\frac{1}{2}\right)^{\frac{m}{n} \cdot \ln 2} + \phi \right) \cdot \left(\frac{1}{2}\right)^{(1-\epsilon') \cdot D_{p||q,max}} \\ &= 2^{\epsilon' \cdot D_{p||q,max}} \cdot \left(\left(\frac{1}{2}\right)^{\frac{m}{n} \cdot \ln 2} + \phi \right) \cdot \left(\frac{1}{2}\right)^{D_{p||q,max}} \\ &= (1 + \epsilon' \cdot D_{p||q,max} \cdot \ln 2) \cdot \left(\left(\frac{1}{2}\right)^{\frac{m}{n} \cdot \ln 2} + \phi \right) \cdot \left(\frac{1}{2}\right)^{D_{p||q,max}} \\ &= (1 + \epsilon' \cdot D_{p||q,max} \cdot \ln 2) \cdot \left(\left(\frac{1}{2}\right)^{\frac{m}{n} \cdot \ln 2} \cdot \left(\frac{1}{2}\right)^{D_{p||q,max}} + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q,max}} \right) \end{aligned}$$

By setting $\epsilon = \epsilon' \cdot D_{p||q,max} \cdot \ln 2$, it holds that $O\hat{P}T \leq (1+\epsilon) \cdot \left(OPT^* + \phi \cdot \left(\frac{1}{2}\right)^{D_{p||q,max}} \right)$.

Next, we prove the case with our deterministic strategy. Recall the deterministic strategy holds that $v(r, \lfloor \frac{C}{\delta_1} \rfloor) \leq (1 +$

$\epsilon)\left(\Phi(\mathbf{k}^*)+\phi\cdot\left(\frac{1}{2}\right)^{D_{p||q}}\right)$. Thus, denoting the objective value of the deterministic strategy as Π_3, for any small ϵ' we have

$$\begin{aligned}
\Pi_3 &\leq (1+\epsilon')\cdot\left(\Phi^*+\phi\cdot\left(\frac{1}{2}\right)^{D_{p||q}}\right)\\
&= (1+\epsilon')\cdot\left(\left(\frac{1}{2}\right)^{\frac{m}{n}\cdot\ln 2}+\phi\right)\cdot\left(\frac{1}{2}\right)^{(1-\epsilon')\cdot D_{p||q,max}}\\
&= (1+\epsilon')\cdot 2^{\epsilon'\cdot D_{p||q,max}}\cdot\left(\left(\frac{1}{2}\right)^{\frac{m}{n}\cdot\ln 2}+\phi\right)\cdot\left(\frac{1}{2}\right)^{D_{p||q,max}}\\
&= (1+\epsilon')\cdot(1+\epsilon'\cdot D_{p||q,max}\cdot\ln 2)\cdot\left(\left(\frac{1}{2}\right)^{\frac{m}{n}\cdot\ln 2}+\phi\right)\cdot\left(\frac{1}{2}\right)^{D_{p||q,max}}\\
&= \left(1+\epsilon'\cdot(1+D_{p||q,max}\cdot\ln 2)+o(\epsilon')\right)\cdot\left(\left(\frac{1}{2}\right)^{\frac{m}{n}\cdot\ln 2}\cdot\left(\frac{1}{2}\right)^{D_{p||q,max}}+\phi\cdot\left(\frac{1}{2}\right)^{D_{p||q,max}}\right)
\end{aligned}$$

By setting $\epsilon=\epsilon'\cdot(1+D_{p||q,max}\cdot\ln 2)$, it holds that $O\hat{P}T\leq(1+\epsilon)\cdot\left(OPT^*+\phi\cdot\left(\frac{1}{2}\right)^{D_{p||q,max}}\right)$.

G. Proof of Theorem 4.1

We start by considering the following optimization problem for group i under given backup filter size m_i.

$$\begin{aligned}
\min_{k_j}\quad & \sum_{j\in\mathcal{G}_i} q_j\cdot\beta_i^{k_j}\\
\text{subject to:}\quad & \sum_{j\in\mathcal{G}_i} p_j\cdot k_j=\frac{\ln(1-\beta_i)}{n\ln(1-\frac{1}{m_i})}
\end{aligned}$$

By using augmented Lagrangian approach, we can obtain

$$k_j^*=\log_{\beta_i}\frac{\lambda_i}{\ln\beta_i}+\log_{\beta_i}\frac{p_j}{q_j},$$

where λ_i is the Lagrange multiplier.

By imposing the partial derivative w.r.t. λ_i to 0, we get

$$\sum_{j\in\mathcal{G}_i} p_j\cdot k_j=\frac{\ln(1-\beta_i)}{n\ln(1-\frac{1}{m_i})},$$

which further leads to

$$\log_{\beta_i}\frac{\lambda_i}{\ln\beta_i}=\frac{\ln(1-\beta_i)}{P_i\cdot n\ln(1-\frac{1}{m_i})}+\frac{D_{p||q}^{(i)}}{P_i}\cdot\log_{\beta_i}\frac{1}{2},$$

where $P_i\triangleq\sum_{j\in\mathcal{G}_i}p_j$ denotes the cumulative probability of membership for group i, $D_{p||q}^{(i)}\triangleq\sum_{j\in\mathcal{C}_i}p_j\log_2\frac{p_j}{q_j}$ denotes part of the KL divergence between p and q.

We can then solve $\{k_j\}$ as

$$k_j^*=\frac{\ln(1-\beta_i)}{P_i\cdot n\ln(1-\frac{1}{m_i})}+\frac{D_{p||q}^{(i)}}{P_i}\cdot\log_{\beta_i}\frac{1}{2}+\log_{\beta_i}\frac{p_j}{q_j}$$

The corresponding objective value under $\mathbf{k}^*$ can be derived as

$$\sum_{j\in\mathcal{G}_i} q_j\cdot\beta_i^{k_j} = \sum_{j\in\mathcal{G}_i} \beta_i^{\frac{\ln(1-\beta_i)}{P_i\cdot n\ln(1-\frac{1}{m_i})}}\cdot\left(\frac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}}\cdot p_j$$
$$= P_i\cdot\beta_i^{\frac{\ln(1-\beta_i)}{P_i\cdot n\ln(1-\frac{1}{m_i})}}\cdot\left(\frac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}}.$$

When $\beta_i=\frac{1}{2}$ and $\frac{\ln(1-\beta_i)}{n\ln(1-\frac{1}{m_i})}=\frac{m_i}{n}\cdot\ln 2$, for large m_i, $\mathbf{k}^*$ can be written as

$$k_j^* = \frac{m_i}{P_i\cdot n}\cdot\ln 2 + \frac{D^{(i)}_{p||q}}{P_i}\cdot\log_{\beta_i}\frac{1}{2} + \log_{\beta_i}\frac{p_j}{q_j}.$$

The corresponding FPR of group i can be written as

$$P_i\cdot\left(\frac{1}{2}\right)^{\frac{m_i}{P_i\cdot n}\cdot\ln 2}\cdot\left(\frac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}},\ \text{or}\ P_i\cdot\alpha^{\frac{m_i}{P_i\cdot n}}\cdot\left(\frac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}}.$$

Armed with the above analysis, we next optimize the following problem of minimizing the overall FPR for given thresholds and groups:

$$\min_{m_i}\quad \sum_{i=1}^{g} P_i\cdot\alpha^{\frac{m_i}{P_i\cdot n}}\cdot\left(\frac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}}$$
$$\text{subject to:}\quad \sum_{i=1}^{g} m_i = m.$$

Using augmented Lagrangian approach again, we write the following Lagrangian function for optimization:

$$\mathcal{L}(r_1,r_2,\ldots,r_m,\lambda) = \sum_{i=1}^{g} P_i\cdot\alpha^{\frac{m_i}{P_i\cdot n}}\cdot\left(\frac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}} - \lambda\left(\sum_{i=1}^{g} m_i - m\right),$$

where $\lambda\le 0$ is the Lagrange multiplier.

The partial derivatives with respect to m_i and λ can be derived as below:

$$\begin{cases}\dfrac{\partial\mathcal{L}}{\partial m_i} = \dfrac{1}{n}\cdot\alpha^{\frac{m_i}{P_i\cdot n}}\cdot\ln\alpha\cdot\left(\dfrac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}} - \lambda\\ \dfrac{\partial\mathcal{L}}{\partial\lambda} = \sum_{i=1}^{g} m_i - m\end{cases}$$

The partial derivative for m_i equals 0 when

$$\frac{1}{n}\cdot\alpha^{\frac{m_i}{P_i\cdot n}}\cdot\ln\alpha\cdot\left(\frac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}} - \lambda = 0,$$

or more specifically, when

$$m_i = P_i\cdot n\left(\log_\alpha\frac{\lambda\cdot n}{\ln\alpha} + \log_\alpha 2^{\frac{D^{(i)}_{p||q}}{P_i}}\right).$$

By imposing the partial derivative w.r.t. m_i to 0, we get

$$\begin{aligned}\sum_{i=1}^{g} P_i \cdot n \left(\log_\alpha \frac{\lambda \cdot n}{\ln \alpha} + \log_\alpha 2^{\frac{D^{(i)}_{p||q}}{P_i}} \right) = m \Longrightarrow & n \cdot \log_\alpha \frac{\lambda \cdot n}{\ln \alpha} + n \cdot \sum_{i=1}^{g} P_i \cdot \log_2 2^{\frac{D^{(i)}_{p||q}}{P_i}} \cdot \log_\alpha 2 = m \\ \Longrightarrow & n \cdot \log_\alpha \frac{\lambda \cdot n}{\ln \alpha} - n \cdot \sum_{i=1}^{g} D^{(i)}_{p||q} \cdot \log_\alpha \frac{1}{2} = m \\ \Longrightarrow & \log_\alpha \frac{\lambda \cdot n}{\ln \alpha} = \frac{m}{n} + D_{p||q} \cdot \log_\alpha \frac{1}{2}\end{aligned}$$

Thus, we obtain

$$m_i = P_i \cdot n \left(\frac{m}{n} + D_{p||q} \cdot \log_\alpha \frac{1}{2} + \log_\alpha 2^{\frac{D^{(i)}_{p||q}}{P_i}} \right),$$

which indicates that, for each group i, k_j^* can be written as

$$\begin{aligned}k_j^* &= \left(\frac{m}{n} + D_{p||q} \cdot \log_\alpha \frac{1}{2} + \log_\alpha 2^{\frac{D^{(i)}_{p||q}}{P_i}} \right) \cdot \ln 2 + \frac{D^{(i)}_{p||q}}{P_i} \cdot \log_{\beta_i} \frac{1}{2} + \log_{\beta_i} \frac{p_j}{q_j} \\ &= \left(\frac{m}{n} + D_{p||q} \cdot \log_\alpha \frac{1}{2} + \frac{D^{(i)}_{p||q}}{P_i} \cdot \log_\alpha 2 \right) \cdot \ln 2 + \frac{D^{(i)}_{p||q}}{P_i} \cdot \log_{\beta_i} \frac{1}{2} + \log_{\beta_i} \frac{p_j}{q_j} \\ &= \frac{m}{n} \cdot \ln 2 + D_{p||q} + (\log_{\beta_i} \frac{1}{2} - 1) \cdot \frac{D^{(i)}_{p||q}}{P_i} + \log_{\beta_i} \frac{p_j}{q_j}\end{aligned}$$

Noting that $\beta_i = \frac{1}{2}$ when the FPR of each group i is minimized, we further have

$$k_j^* = \frac{m}{n} \cdot \ln 2 + D_{p||q} + \log_{\frac{1}{2}} \frac{p_j}{q_j}.$$

Hence, we can derive the optimal overall FPR as

$$\begin{aligned}\sum_{i=1}^{g} P_i \cdot \alpha^{\frac{m_i}{P_i \cdot n}} \cdot \left(\frac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}} &= \sum_{i=1}^{g} P_i \cdot \alpha^{\frac{m}{n} + D_{p||q} \cdot \log_\alpha \frac{1}{2} + \log_\alpha 2^{\frac{D^{(i)}_{p||q}}{P_i}}} \cdot \left(\frac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}} \\ &= \sum_{i=1}^{g} P_i \cdot \alpha^{\frac{m}{n}} \cdot \left(\frac{1}{2}\right)^{D_{p||q}} \cdot 2^{\frac{D^{(i)}_{p||q}}{P_i}} \cdot \left(\frac{1}{2}\right)^{\frac{D^{(i)}_{p||q}}{P_i}} \\ &= \alpha^{\frac{m}{n}} \cdot \left(\frac{1}{2}\right)^{D_{p||q}} \\ &= \left(\frac{1}{2}\right)^{\frac{m}{n} \cdot \ln 2} \cdot \left(\frac{1}{2}\right)^{D_{p||q}}.\end{aligned}$$

Theorem 4.1 is thus proved.

H. Ablation Study for Hyper-parameters

We perform the ablation study for hyper-parameters of our propositions under 4000Kbits memory budget.

Figure 4 shows that FPR tends to decrease as ϵ decreases, since smaller ϵ indicates more nearly optimality (note that $\gamma = 1 - \epsilon$ for the randomized strategy), excluding our rounding-down strategy which doesn't require the approximation factor. On the other hand, the time overhead of our deterministic algorithm tends to increase as ϵ decreases, since smaller

ϵ indicates larger state space of our deterministic algorithm, while ϵ doesn't act on the other two strategies, as shown in Figure 4 as well.

Figure 5 and 6 show that FPR tends to decrease as r and N increase, since larger numbers are closer to continuous space, while the time overhead tends to pseudo-linearly increase as r and linearly increase as N increase (note that the horizontal axis is in log mode), which confirms the time complexity of our algorithm solving the outer optimization problem. Our deterministic algorithm is more time-consuming as it runs an inner dynamic programming.

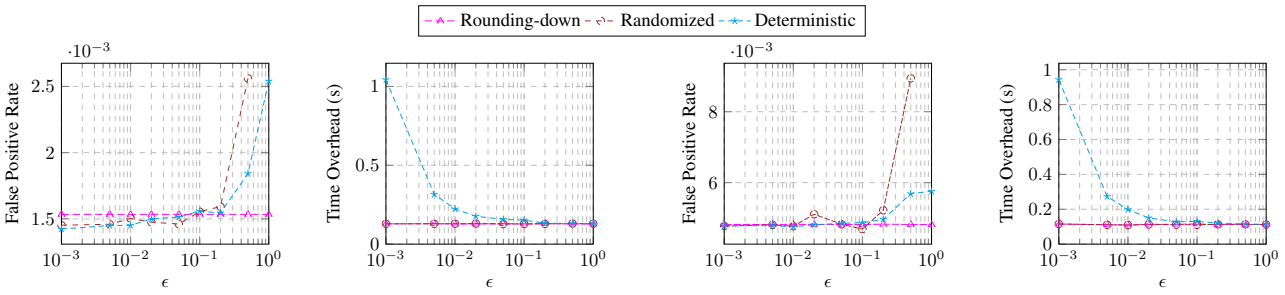

Figure 4. Ablation study for ϵ: Malicious URLs Detection (left), Virus Signature Scan (right).

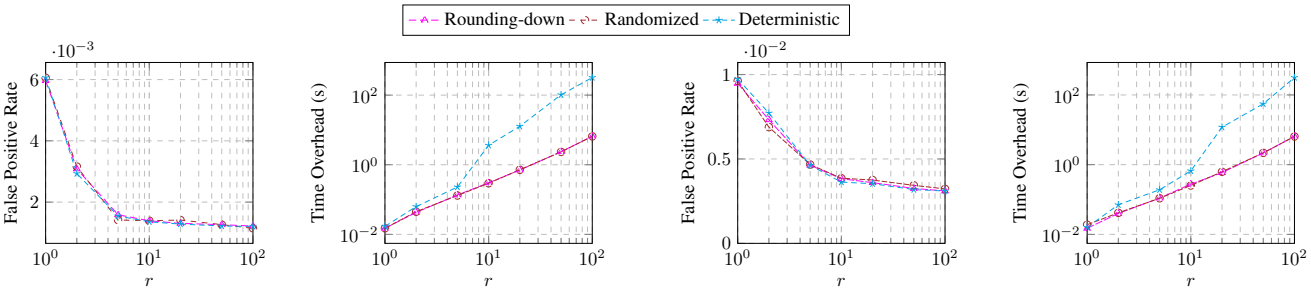

Figure 5. Ablation study for r: Malicious URLs Detection (left), Virus Signature Scan (right).

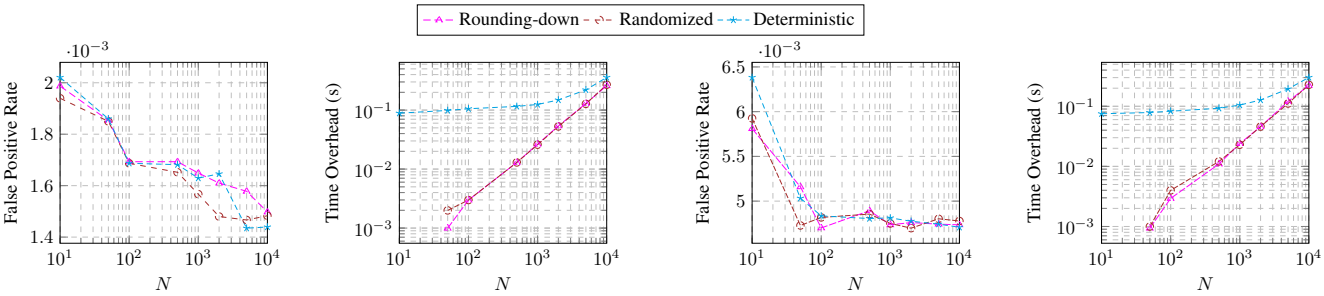

Figure 6. Ablation study for N: Malicious URLs Detection (left), Virus Signature Scan (right).