On Computing Total Variation Distance Between Mixtures of Product Distributions

Weiming Feng
School of Computing and Data Science, The University of Hong Kong, China

Yucheng Fu
School of Computing and Data Science, The University of Hong Kong, China

Minji Yang
School of Computing and Data Science, The University of Hong Kong, China

Anqi Zhang
The Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Abstract

We study the problem of approximating the total variation distance between two mixtures of product distributions over an n-dimensional discrete domain. Given two mixtures $\mathbb{P}$ and $\mathbb{Q}$ with k_1 and k_2 product distributions over $[q]^n$, respectively, we give a randomized algorithm that approximates $d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q})$ within a multiplicative error of $(1\pm\varepsilon)$ in time $\mathrm{poly}((nq)^{k_1+k_2},1/\varepsilon)$. We also study the special case of mixtures of Boolean subcubes over $\{0,1\}^n$. For this class, we give a deterministic algorithm that exactly computes the total variation distance in time $\mathrm{poly}(n,2^{O(k_1+k_2)})$, and show that exact computation is #P-hard when $k_1+k_2=\Theta(n)$.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases Randomized algorithm, Total variation distance

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2026.51

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2605.03839

Funding *Weiming Feng*: ECS grant 27202725 from Hong Kong RGC.

Acknowledgements We thank Arnab Bhattacharyya and Guy Van den Broeck for bringing the problem to our attention.

1 Introduction

Let $\mathbb{P}$ and $\mathbb{Q}$ be two discrete distributions over a sample space Ω. The *total variation distance (TV-distance)* between $\mathbb{P}$ and $\mathbb{Q}$ is defined as

$$d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q})=\frac{1}{2}\sum_{\omega\in\Omega}|\mathbb{P}(\omega)-\mathbb{Q}(\omega)|=\sum_{\omega\in\Omega}\max\{0,\mathbb{P}(\omega)-\mathbb{Q}(\omega)\}.$$

The TV-distance is one of the most fundamental metrics for measuring the difference between distributions, as it characterizes the optimal distinguishability between $\mathbb{P}$ and $\mathbb{Q}$. Computing the TV-distance is therefore a basic problem arising in learning and testing.

A line of research has investigated the problem of *approximating* the TV-distance between two high-dimensional distributions, which is particularly interesting when the two distributions admit *succinct representations*. Early works [22, 8, 10, 19, 5] studied the algorithms and complexity of approximating the TV-distance with additive error. More recently, [2] established that even for two product distributions, the *exact computation* of the

TV-distance is #P-hard. In the same paper, the authors initiated the study of approximating the TV-distance with *relative error*, a more challenging task than the classical additive-error approximation. Subsequently, polynomial-time approximation algorithms have been developed for product distributions [2, 12, 14], as well as for many high-dimensional distributions defined by *local interactions*, including Markov chains [14], Bayesian networks [3], and undirected graphical models [13].

Beyond local interactions, it is natural to consider more general high-dimensional distributions with inherently *non-local structure*. A fundamental and extensively studied example is given by *mixtures of product distributions*. Let $[q] = \{1, 2, \ldots, q\}$ and let $\Omega = [q]^n$ denote the n-dimensional sample space. Fix two integers $k_1, k_2 \geq 1$. For each $s \in [k_1]$, let $\mathbb{P}^{(s)} = \bigotimes_{i=1}^n \mathbb{P}_i^{(s)}$ be a product distribution over Ω, where $\mathbb{P}_i^{(s)}$ denotes the marginal on the i-th coordinate, so that the coordinates are mutually independent under $\mathbb{P}^{(s)}$. Let $\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k_1)}$ be mixing weights satisfying $\sum_{s=1}^{k_1} \alpha^{(s)} = 1$. The resulting mixture distribution is written as

$$\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}.$$

More explicitly, $\mathbb{P}$ is the distribution over $\Omega = [q]^n$ given by

$$\forall \omega \in \Omega, \quad \mathbb{P}(\omega) = \sum_{s=1}^{k_1} \alpha^{(s)} \mathbb{P}^{(s)}(\omega) = \sum_{s=1}^{k_1} \alpha^{(s)} \prod_{i=1}^{n} \mathbb{P}_i^{(s)}(\omega_i), \tag{1}$$

Mixtures of product distributions form a fundamental class of *latent-variable models* and have been extensively studied in learning theory [11, 17, 9, 15, 16]. To draw a random sample $X \sim \mathbb{P}$, one first samples a *hidden* component index $S \in [k_1]$ according to $\mathbf{Pr}[S = s] = \alpha^{(s)}$, and then samples $X \sim \mathbb{P}^{(S)} = \bigotimes_{i=1}^n \mathbb{P}_i^{(S)}$ from the product distribution $\mathbb{P}^{(S)}$. Although each component $\mathbb{P}^{(s)}$ is itself a product distribution, the hidden index S induces a highly *non-local dependency* that simultaneously couples all coordinates of the sample $X = (X_1, X_2, \ldots, X_n)$, which cannot be captured by any sparse local structure.

In this paper, we study the problem of *approximating* the TV-distance between two mixtures of product distributions to within an arbitrarily small relative error, formally stated as follows.

▶ **Problem 1.** *Approximate the TV-distance between two mixtures of product distributions.*

- ***Input****: An error bound $\varepsilon > 0$ and two mixtures of product distributions over $\Omega = [q]^n$:*

$$\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)} \text{ and } \mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}.$$

  The distribution $\mathbb{P}$ is specified by the k_1 mixing weights $\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k_1)}$ together with the k_1 product distributions $\mathbb{P}^{(1)}, \mathbb{P}^{(2)}, \ldots, \mathbb{P}^{(k_1)}$, where each $\mathbb{P}^{(s)}$ is described by its n marginals $\mathbb{P}_1^{(s)}, \mathbb{P}_2^{(s)}, \ldots, \mathbb{P}_n^{(s)}$ over $[q]$. The distribution $\mathbb{Q}$ is given analogously. The total input size of the two mixtures is $O((k_1 + k_2)nq)$.
- ***Output****: A number $\hat{d}$ such that*

$$(1 - \varepsilon) d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}) \leq \hat{d} \leq (1 + \varepsilon) d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}).$$

Due to the non-local nature of these dependencies among all coordinates, existing techniques for TV-distance approximation cannot be directly applied to mixtures of product distributions. The only prior progress is a polynomial-time algorithm for *deciding* whether

$d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q}) = 0$ or $d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q}) > 0$ [4], leaving the approximation problem open. We partially resolve this open question by giving an FPRAS for the TV-distance between two mixtures of product distributions, provided that the numbers of components k_1 and k_2 are constants.

▶ **Theorem 2.** *Let $k_1, k_2 \geq 1$ be constants. There exists an FPRAS for Problem 1: given any $\varepsilon > 0$ and any pair of mixtures of product distributions $\mathbb{P},\mathbb{Q}$ over $[q]^n$, where $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}$ has k_1 components and $\mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}$ has k_2 components, it solves the problem in time $O(\frac{q^2(nq)^{2(k_1+k_2-1)}}{\varepsilon^2})$ with success probability at least 99%.*

Our algorithm is a Monte Carlo algorithm based on a carefully designed estimator, and we provide an overview of it in Section 2. Since the number of components $k_1 + k_2$ appears as an exponent in the running time, we require $k_1 + k_2$ to be a constant. A similar dependence on the number of components also arises in the running time of certain learning tasks for mixtures of product distributions [11, 9, 15]. We leave it as an open question to understand the computational complexity when the number of components grows with n.

Our second result concerns mixtures of Boolean subcubes, a special case of mixtures of product distributions. A distribution $\mathbb{P} = \sum_{s=1}^{k} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}$ over $\{0,1\}^n$ is called a mixture of Boolean subcubes if $\mathbb{P}_i^{(s)}(1) \in \{0, \frac{1}{2}, 1\}$ for every $s \in [k]$ and $i \in [n]$, where $\mathbb{P}_i^{(s)}$ denotes the marginal distribution of the i-th coordinate in the s-th component. Equivalently, for each component s, the product distribution $\mathbb{P}^{(s)}$ fixes every coordinate i with $\mathbb{P}_i^{(s)}(1) = 1$ to 1, fixes every coordinate i with $\mathbb{P}_i^{(s)}(1) = 0$ to 0, and is uniform on the remaining coordinates. Mixtures of Boolean subcubes have been extensively studied in learning theory [9, 6, 7] and capture many natural distributions; for instance, the uniform distribution over the satisfying assignments of a decision tree is a mixture of Boolean subcubes [11].

In contrast to the general case, where we only obtain an FPRAS when $k_1 + k_2 = O(1)$, we show that the TV-distance between two mixtures of Boolean subcubes $\mathbb{P}$ and $\mathbb{Q}$ can be computed *exactly* in time polynomial in the dimension n, provided that $k_1 + k_2 = O(\log n)$.

▶ **Problem 3.** *Compute exactly the total variation distance between two mixtures of Boolean subcubes $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}$ and $\mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}$.*

- ***Input**: two mixtures of Boolean subcubes, given in the same format as in Problem 1.*
- ***Output**: the* exact *value of $d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q})$*

▶ **Theorem 4.** *There exists a deterministic algorithm for Problem 3: given any pair of mixtures of Boolean subcubes $\mathbb{P},\mathbb{Q}$ over $\{0,1\}^n$, where $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}$ has k_1 components and $\mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}$ has k_2 components, it solves the problem in $O((k_1+k_2)3^{k_1+k_2} \cdot n)$ time.*

The running time in Theorem 4 is polynomial in n whenever $k_1 + k_2 = O(\log n)$, yielding a deterministic polynomial-time algorithm in this regime. This does not contradict the #P-hardness of exact TV-distance computation established in [2]: their hardness already holds for two product distributions with arbitrary marginals, whereas our algorithm crucially exploits the restriction that each marginal of a Boolean subcube takes value in $\{0, \frac{1}{2}, 1\}$.

The idea of the algorithm is as follows. For each component $\mathbb{P}^{(s)}$ (and similarly for each $\mathbb{Q}^{(t)}$), the value of $\mathbb{P}^{(s)}(\omega)$ is either 0 or 2^{-r}, where r is the number of coordinates $i \in [n]$ with $\mathbb{P}_i^{(s)}(1) = \frac{1}{2}$. Indeed, if some coordinate is fixed to be 1 (or 0) but $\omega_i = 0$ (or 1), then $\mathbb{P}^{(s)}(\omega) = 0$; otherwise, each free coordinate contributes a factor of 1/2, so $\mathbb{P}^{(s)}(\omega) = 2^{-r}$. Thus, although there are 2^n configurations $\omega \in \{0,1\}^n$, the possible values of $|\mathbb{P}(\omega) - \mathbb{Q}(\omega)|$ are at most $2^{k_1+k_2}$. Our algorithm enumerates these possible values of $|\mathbb{P}(\omega) - \mathbb{Q}(\omega)|$ and counts the number of configurations $\omega \in \{0,1\}^n$ that achieve each one. The details are given in Section 5.

Finally, we complement this positive result by showing that the dependence on the number of components cannot be removed entirely: when the number of components grows linearly with n, exact computation becomes #P-hard.

▶ **Theorem 5.** *If the number of mixture components is $k_1 + k_2 = \Theta(n)$, then the exact computation of the total variation distance in Problem 3 is #P-hard.*

The theorem is proved by a reduction showing that if we can exactly compute the TV-distance between two mixtures of Boolean subcubes with $k_1 + k_2 = \Theta(n)$ in polynomial time, then we can also exactly compute #3SAT in polynomial time. The detailed proof is in Section 6.

Open problems. As discussed after Theorem 2, the dependence on the number of components in the running time is not yet well understood. [4] gave an algorithm that decides whether $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}) = 0$ in time $\mathrm{poly}(n, k_1 + k_2)$, whereas the running time of our approximation algorithm depends exponentially on $k_1 + k_2$. The analogous regime $k_1 + k_2 = \omega(1)$ is also not well understood for learning tasks on mixtures of product distributions, where obtaining algorithms with polynomial dependence on the number of components remains a central challenge [11, 9, 15]. A natural question is whether one can approximate $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})$ in time polynomial in $k_1 + k_2$, or prove a hardness result ruling this out when the number of components grows with n. As a first step toward the latter, Theorem 5 already shows that exact computation is #P-hard when $k_1 + k_2 = \Theta(n)$, and one direction is to extend its proof to rule out approximation as well.

One can ask analogous TV-distance approximation questions for other structured mixture models. Natural candidates include mixtures of Gaussians [1, 21], mixtures of Markov chains and MDPs [18]. These models also give succinct descriptions of high-dimensional distributions and are widely studied in machine learning, but their dependence structures are more complex than that of mixtures of product distributions. More generally, it would be interesting to understand the complexity of computing or approximating the TV-distance between trajectory distributions induced by reward-mixing Markov decision processes (RMMDPs). RMMDPs use latent mixture structure to model the reward context, and TV-distance bounds between trajectory distributions have been used to learn near-optimal policies [20]. The problem of computing the TV-distance between RMMDPs appears to require new ideas.

At last, our algorithm for mixtures of product distributions is randomized. It remains open to design a deterministic algorithm for approximating the TV-distance between mixtures of product distributions. Currently, deterministic algorithms are only known for approximating the TV-distance between product distributions [14, 2].

2 Technical Overview

We present the main ideas behind the algorithm in Theorem 2. An obstacle for approximating the TV-distance is that the TV-distance itself can be very small. To see the reason, for two distributions $\mathbb{P}$ and $\mathbb{Q}$ over the sample space Ω, where $\mathbb{Q}$ is absolutely continuous w.r.t. $\mathbb{P}$, one can write

$$d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}) = \sum_{\omega \in \Omega} \mathbb{P}(\omega) \max\left\{0, 1 - \frac{\mathbb{Q}(\omega)}{\mathbb{P}(\omega)}\right\} = \mathbf{E}_{\omega \sim \mathbb{P}}\left[g(\omega)\right],$$

where $g(\omega) := \max\left\{0, 1 - \frac{\mathbb{Q}(\omega)}{\mathbb{P}(\omega)}\right\}$. We can efficiently sample ω and compute $g(\omega)$, so that a direct Monte Carlo estimate of $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}) = \mathbf{E}_{\omega \sim \mathbb{P}}\left[g(\omega)\right]$ can only achieve *additive* error in polynomial time. However, since the expectation $\mathbf{E}_{\omega \sim \mathbb{P}}\left[g(\omega)\right]$ itself can be exponentially small, this simple approach is not sufficient to achieve a *relative* error approximation required by Problem 1.

2.1 Approximating the TV-distance via a coarse coupling

A framework for approximating the TV-distance was proposed in [12] for two product distributions (i.e., $k_1 = k_2 = 1$ in our setting). It can be generalized to any two distributions $\mathbb{P}$ and $\mathbb{Q}$. Instead of approximating $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})$ directly, they estimated the normalized quantity $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}) / \tilde{d}$, where $\tilde{d}$ is a polynomial-time computable *coarse* estimator. Moreover, if the ratio $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}) / \tilde{d}$ is lower bounded by $1/\mathrm{poly}(n)$, one can try to achieve a relative-error approximation to the ratio in polynomial time. Hence, we can recover $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})$ by multiplying $\tilde{d}$ by the estimate of the ratio.

The coarse estimator $\tilde{d}$ can be constructed via a *coupling*. For two distributions $\mathbb{P}$ and $\mathbb{Q}$ over Ω, a coupling is a pair of joint random variables (X, Y) such that $X \sim \mathbb{P}$ and $Y \sim \mathbb{Q}$. It is well known that $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}) = \mathbf{Pr}_{\mathcal{O}}[X' \neq Y']$, where $\mathcal{O} = (X', Y')$ is the *optimal* coupling between $\mathbb{P}$ and $\mathbb{Q}$. Suppose we can construct a coupling $\mathcal{C} = (X, Y)$ that is not far from the optimal coupling $\mathcal{O}$. Formally, $\mathbf{Pr}_{\mathcal{O}}[X' \neq Y'] = \gamma \cdot \mathbf{Pr}_{\mathcal{C}}[X \neq Y]$ for some $\gamma = 1/\mathrm{poly}(n)$. Then $\tilde{d} = \mathbf{Pr}_{\mathcal{C}}[X \neq Y]$ is a coarse estimator of $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})$. To estimate the ratio $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}) / \tilde{d}$, define a distribution π:

$$\omega \in \Omega, \quad \pi(\omega) = \mathbf{Pr}_{\mathcal{C}}[X = \omega \mid X \neq Y] \tag{2}$$

and define function f such that

$$\omega \in \Omega, \quad f(\omega) = \frac{\max\{0, \mathbb{P}(\omega) - \mathbb{Q}(\omega)\}}{\mathbf{Pr}_{\mathcal{C}}[X = \omega \wedge X \neq Y]}.$$

We can verify that $\mathbf{E}_{\pi}[f] = \frac{d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})}{\mathbf{Pr}_{\mathcal{C}}[X \neq Y]} = \frac{d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})}{\tilde{d}} = \gamma \geq \frac{1}{\mathrm{poly}(n)}$. Using the approach described above, if we can perform the following operations in polynomial time

(a) sample $\omega \sim \pi$;
(b) compute $f(\omega)$ for any ω;
(c) compute $\mathbf{Pr}_{\mathcal{C}}[X \neq Y]$;

then we can get a relative-error approximation of $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})$. Specifically, we draw $T = \mathrm{poly}(\frac{1}{\gamma}) = \mathrm{poly}(n)$ independent samples $\omega_1, \ldots, \omega_T \sim \pi$ and compute $\bar{f} = \frac{1}{T} \sum_{i=1}^{T} f(\omega_i)$. By Chebyshev's inequality, with high probability, $\bar{f} \cdot \mathbf{Pr}_{\mathcal{C}}[X \neq Y]$ is an estimate of $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})$ within relative error.

2.2 Recursive coupling for mixtures of product distributions

It remains to construct a coupling that supports the three operations above. For product distributions, [12] used a coordinate-wise greedy coupling $\mathcal{C} = (X, Y)$, which couples each coordinate optimally and independently, and showed that $1/n \leq \mathbf{E}_{\pi}[f] \leq 1$. This approach does not directly extend to mixtures of product distributions, where all the coordinates are *correlated*, so one cannot optimally couple all coordinates simultaneously.

Our main technical contribution in this part is a new *explicit recursive coupling* for mixtures of product distributions, together with an algorithmic implementation of the three operations above. At a high level, the recursive structure draws inspiration from [20], who introduced a recursive argument to bound the TV-distance between two mixtures of product distributions. Their analysis, however, is non-constructive: it yields a *fixed-error* bound on the TV-distance, but does not by itself provide the algorithmic interface required here. To obtain a relative-error approximation with *arbitrary* ε, we need a coupling whose failure probability can be computed, and whose failure-conditioned distribution can be sampled and evaluated. We design such a coupling tailored to mixtures of product distributions, adapt the recursive analysis of [20] to this explicit construction, and show that the three operations in Item a, Item b, and Item c can all be implemented in polynomial time.

We construct a coupling between two mixtures of product distributions $\mathbb{P}$ and $\mathbb{Q}$, where

$$\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)} \quad \text{and} \quad \mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}.$$

Although coupling the first coordinate would bias the marginal distribution of the remaining coordinates, we can still extract the common parts of $\mathbb{P}_1^{(s)}$ and $\mathbb{Q}_1^{(t)}$, and reduce the problem to coupling *another* mixture of product distributions on the remaining coordinates. For each $c \in [q]$, we first compute a lower bound for each active mixture component:

$$\ell(c) \triangleq \min \left\{ \min_{s \in [k_1]: \alpha^{(s)} > 0} \mathbb{P}_1^{(s)}(c), \min_{t \in [k_2]: \beta^{(t)} > 0} \mathbb{Q}_1^{(t)}(c) \right\}, \tag{3}$$

where $\mathbb{P}_1^{(s)}$ (resp. $\mathbb{Q}_1^{(t)}$) is the marginal distribution of the first coordinate of the s-th component of $\mathbb{P}^{(s)}$ (resp. $\mathbb{Q}^{(t)}$). For $\omega \in [q]^n$ with $\omega_1 = c$, we can write the probability of ω as

$$\begin{aligned}
\mathbb{P}(\omega) &= \ell(c) \sum_{s=1}^{k_1} \alpha^{(s)} \prod_{i=2}^{n} \mathbb{P}_i^{(s)}(\omega_i) + \sum_{s=1}^{k_1} \alpha^{(s)} \left(\mathbb{P}_1^{(s)}(c) - \ell(c) \right) \prod_{i=2}^{n} \mathbb{P}_i^{(s)}(\omega_i) \\
&= \ell(c) \underbrace{\sum_{s=1}^{k_1} \alpha^{(s)} \prod_{i=2}^{n} \mathbb{P}_i^{(s)}(\omega_i)}_{A_{\mathbb{P}}} + (\mathbb{P}_1(c) - \ell(c)) \underbrace{\sum_{s=1}^{k_1} \alpha_c^{(s)} \prod_{i=2}^{n} \mathbb{P}_i^{(s)}(\omega_i)}_{B_{\mathbb{P},c}},
\end{aligned}$$

where $\alpha_c^{(s)} \triangleq \frac{\alpha^{(s)} \left(\mathbb{P}_1^{(s)}(c) - \ell(c) \right)}{(\mathbb{P}_1(c) - \ell(c))}$ is a valid distribution over $s \in [k_1]$. Note that $\mathbb{P}_1$ in the denominator is the marginal distribution of the mixture $\mathbb{P}$ over the first coordinate. Similarly, let $\beta_c^{(t)} \triangleq \frac{\beta^{(t)} \left(\mathbb{Q}_1^{(t)}(c) - \ell(c) \right)}{(\mathbb{Q}_1(c) - \ell(c))}$ and

$$\mathbb{Q}(\omega) = \ell(c) \underbrace{\sum_{t=1}^{k_2} \beta^{(t)} \prod_{i=2}^{n} \mathbb{Q}_i^{(t)}(\omega_i)}_{A_{\mathbb{Q}}} + (\mathbb{Q}_1(c) - \ell(c)) \underbrace{\sum_{t=1}^{k_2} \beta_c^{(t)} \prod_{i=2}^{n} \mathbb{Q}_i^{(t)}(\omega_i)}_{B_{\mathbb{Q},c}}.$$

Each of $A_{\mathbb{P}}, A_{\mathbb{Q}}, B_{\mathbb{P},c}$, and $B_{\mathbb{Q},c}$ is a mixture of product distributions over the remaining coordinates. The component distributions in $A_{\mathbb{P}}$ and $B_{\mathbb{P},c}$ are obtained from the components of $\mathbb{P}$ by removing the first coordinate; the same relation holds between $A_{\mathbb{Q}}, B_{\mathbb{Q},c}$ and $\mathbb{Q}$. However, the mixing weights may be different from those in $\mathbb{P}$ and $\mathbb{Q}$.

We use the above decomposition to construct a coupling (X, Y) between $\mathbb{P}$ and $\mathbb{Q}$. We first couple the first coordinate of X and Y. For each $c \in [q]$, with probability $\ell(c)$, we set $X_1 = Y_1 = c$ and then couple $A_{\mathbb{P}}$ and $A_{\mathbb{Q}}$ *recursively* for the remaining coordinates. For each $c \in [q]$, with probability $\min\{\mathbb{P}_1(c) - \ell(c), \mathbb{Q}_1(c) - \ell(c)\}$, we set $X_1 = Y_1 = c$ and couple $B_{\mathbb{P},c}$ and $B_{\mathbb{Q},c}$ *recursively* for the remaining coordinates. For the remaining probability mass[1] of the first coordinate, there is no common part with $X_1 = Y_1 = c$ for any $c \in [q]$. We arbitrarily couple this remaining mass; it must hold that $X_1 \neq Y_1$, and hence $X \neq Y$. We call this case the *coupling failure* case. Suppose that after coupling the first coordinate, we have $X_1 = c \neq c' = Y_1$ for some $c, c' \in [q]$. We then independently sample the remaining coordinates according to $B_{\mathbb{P},c}$ and $B_{\mathbb{Q},c'}$, respectively.

[1] Specifically, the remaining probability for $X_1 = c$ is $\max\{0, \mathbb{P}_1(c) - \mathbb{Q}_1(c)\}$ and the remaining probability for $Y_1 = c$ is $\max\{0, \mathbb{Q}_1(c) - \mathbb{P}_1(c)\}$.

The above recursive process terminates either when coupling fails at some coordinate or when it reaches the last coordinate. Denote the resulting recursive coupling by $\mathcal{C}_{\mathrm{RC}}$. Adapting the recursive TV-distance analysis of [20] to this explicit coupling, we show that $\mathcal{C}_{\mathrm{RC}}$ is not far from the optimal coupling:

$$\mathbf{Pr}_{\mathcal{C}_{\mathrm{RC}}}[X \neq Y] \leq (4nq)^{k_1+k_2-1} \cdot d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}).$$

Therefore, we set $\gamma = (4nq)^{-k_1-k_2+1}$, which is inverse-polynomial in n and q when k_1 and k_2 are constants.

2.3 Implementation of three operations

We now give polynomial-time algorithms for the three operations in Item a, Item b, and Item c. We create a state space $\mathcal{S}$ to keep track of the recursive coupling process. A state $S \in \mathcal{S}$ is a tuple $S = (i, \bar{\alpha}, \bar{\beta})$, where $i \in [n]$ means we need to couple the remaining coordinates $i, i+1, \ldots, n$, and $\bar{\alpha} = (\bar{\alpha}^{(s)})_{s\in[k_1]}$ and $\bar{\beta} = (\bar{\beta}^{(t)})_{t\in[k_2]}$ are the *current* mixing weights for each mixture component. Section 2.2 describes the recursive coupling starting from the root state $(1, \alpha, \beta)$, and hence defines the quantities $A_{\mathbb{P}}, A_{\mathbb{Q}}, B_{\mathbb{P},c}, B_{\mathbb{Q},c}$ and $\ell(c)$ in that setting. For the implementation, we use the same decomposition at an arbitrary state $S = (i, \bar{\alpha}, \bar{\beta})$, replacing the first coordinate by the current coordinate i and the original mixing weights by the current weights $\bar{\alpha}, \bar{\beta}$. Thus,

$$\ell(c) = \min\left\{ \min_{s\in[k_1]:\bar{\alpha}^{(s)}>0} \mathbb{P}_i^{(s)}(c), \min_{t\in[k_2]:\bar{\beta}^{(t)}>0} \mathbb{Q}_i^{(t)}(c) \right\}.$$

Similarly, the updated weights $\bar{\alpha}_c = (\bar{\alpha}_c^{(s)})_{s\in[k_1]}$ in the $B_{\mathbb{P},c}$ branch are defined by

$$\forall s \in [k_1], \bar{\alpha}_c^{(s)} = \bar{\alpha}^{(s)} \cdot \frac{\mathbb{P}_i^{(s)}(c) - \ell(c)}{\bar{\mathbb{P}}_i(c) - \ell(c)} \text{ where } \bar{\mathbb{P}}_i(c) = \sum_{s\in[k_1]} \bar{\alpha}^{(s)} \mathbb{P}_i^{(s)}(c).$$

The $\bar{\beta}_c$ for the $B_{\mathbb{Q},c}$ is analogous. There is also a failure state $\perp$. If we couple $X_i \neq Y_i$ (coupling failure case), the state S transitions to the failure state $\perp$. If we couple $X_i = Y_i$ and then recursively couple $A_{\mathbb{P}}$ and $A_{\mathbb{Q}}$ for the remaining coordinates (we call it the good case A), the state S transitions to $S' = (i+1, \bar{\alpha}, \bar{\beta})$. If we couple $X_i = Y_i$ and then recursively couple $B_{\mathbb{P},c}$ and $B_{\mathbb{Q},c}$ for the remaining coordinates (we call it the good case B), the state S transitions to $S'_c = (i+1, \bar{\alpha}_c, \bar{\beta}_c)$.

The whole recursive coupling process is a random walk on the state space $\mathcal{S}$ starting from the root state $S_{\mathrm{root}} = (1, \alpha, \beta)$. Every non-failure state $S = (i, \bar{\alpha}, \bar{\beta}) \neq \perp$ can transition to a state in $\{\perp\} \cup \{S'\} \cup \{S'_c\}_{c\in[q]}$, and it transitions to some S'_c if and only if the good case B occurs. The key observation is that case B can occur at most $k_1 + k_2 - 2$ times in the whole recursive coupling process. By the definition of $\ell(c)$, $\ell(c)$ is equal to some $\mathbb{P}_i^{(s)}(c)$ or $\mathbb{Q}_i^{(t)}(c)$. Suppose $\ell(c) = \mathbb{P}_i^{(s)}(c)$. Then, the new mixing weights in S'_c satisfy $\bar{\alpha}_c^{(s)} = \bar{\alpha}^{(s)} \cdot \frac{\mathbb{P}_i^{(s)}(c) - \ell(c)}{\bar{\mathbb{P}}_i(c) - \ell(c)} = 0$. Thus, whenever case B occurs, the total number of positive mixing weights decreases by at least one. Since the root state has $k_1 + k_2$ positive mixing weights, case B can occur at most $k_1 + k_2 - 2$ times. Using this property, we can show that the total number of states is $|\mathcal{S}| = (nq)^{O(k_1+k_2)}$, which is polynomial in n and q.

Given a state S, it is easy to compute the transition probabilities to the next states. The whole transition graph forms a DAG with polynomial size. To sample $\omega \sim \pi$ in (2), it suffices to sample a trajectory starting from the root state, conditioned on the random walk reaching the failure state $\perp$. We first run a dynamic programming algorithm to compute the

probability of reaching $\perp$ from each state in $\mathcal{S}$. Then, we use these probabilities to sample the desired conditional trajectory. Similarly, using dynamic programming, we can compute $f(\omega)$ and $\mathbf{Pr}_{\mathcal{C}_{\mathrm{RC}}}[X \neq Y]$ in polynomial time.

▶ Remark 6 (Comparison with the coupling in [3]). Finally, it is worth comparing our coupling with the coupling used in [3]. They proposed a simpler coupling and applied it to approximating the TV-distance between two Bayesian networks. Given two n-dimensional distributions $\mathbb{P}$ and $\mathbb{Q}$ over $[q]^n$, they couple the coordinates one by one using the optimal coupling of the conditional marginals. Suppose the first $i-1$ coordinates of X and Y have been generated and coupled consistently such that $X_{1:i-1} = Y_{1:i-1} = \sigma \in [q]^{i-1}$. For the i-th coordinates X_i and Y_i, they use the optimal coupling of the conditional marginals $\mathbb{P}_i$ and $\mathbb{Q}_i$ given σ. The coupling terminates if $X_i \neq Y_i$ at some coordinate or when it reaches the last coordinate.

Their coupling is *different* from ours. Unlike Bayesian networks, for general mixtures of product distributions, the conditional marginals at the i-th coordinate require the *full information* of the prefix condition $\sigma \in [q]^{i-1}$. Since there are $O(q^n)$ possible prefix conditions σ, this coupling needs to consider exponentially many possible distributions for our problem. In our coupling, even when the good event $X_i = Y_i$ occurs, we further split it into two sub-cases, A and B. This splitting allows us to exploit the structure of mixtures of product distributions, so the total number of possible distributions appearing in our coupling is bounded by a polynomial.

3 A General Approach for Approximating the TV-Distance

When $k_1 = k_2 = 1$, the problem reduces to approximating the TV-distance between two product distributions P and Q over $\Omega = [q]^n$. This special case was studied in [12]. Although the algorithm there is tailored to product distributions, its underlying idea extends to a more general framework for approximating TV-distances for general discrete distributions.

Let $\mathbb{P}$ and $\mathbb{Q}$ be arbitrary discrete distributions over Ω. A *coupling* of $\mathbb{P}$ and $\mathbb{Q}$ is a pair of jointly distributed random variables (X, Y) such that $X \sim \mathbb{P}$ and $Y \sim \mathbb{Q}$. For every coupling $\mathcal{C}$ of $\mathbb{P}$ and $\mathbb{Q}$, the coupling inequality states that

$$d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}) \leq \mathbf{Pr}_{\mathcal{C}}[X \neq Y].$$

A coupling $\mathcal{C}$ is called *optimal* if equality holds, namely if $\mathbf{Pr}_{\mathcal{C}}[X \neq Y] = d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})$. An optimal coupling exists for every pair of distributions $\mathbb{P}$ and $\mathbb{Q}$.

The following fact holds for every coupling $\mathcal{C}$ of $\mathbb{P}$ and $\mathbb{Q}$.

▶ **Fact 7.** *For any coupling $\mathcal{C}$ of $\mathbb{P}$ and $\mathbb{Q}$, it holds that*

$$\forall \omega \in \Omega, \quad \mathbf{Pr}_{\mathcal{C}}[X = \omega \wedge X \neq Y] \geq \max\{0, \mathbb{P}(\omega) - \mathbb{Q}(\omega)\}.$$

If $\mathcal{C}$ is an optimal coupling, then for all $\omega \in \Omega$,

$$\mathbf{Pr}_{\mathcal{C}}[X = \omega \wedge X \neq Y] = \max\{0, \mathbb{P}(\omega) - \mathbb{Q}(\omega)\}.$$

The proof can be found in the full version.

We now state an abstract version of the algorithm from [12], formulated for two arbitrary discrete distributions $\mathbb{P}$ and $\mathbb{Q}$. Let $\gamma, T_0, T_1, T_2, T_3 > 0$ be parameters.

▶ **Assumption 8.** *There exists a probability mass oracle that, for any $\omega \in \Omega$, returns $\mathbb{P}(\omega)$ and $\mathbb{Q}(\omega)$ in time T_0.*

▶ **Assumption 9.** *There exists a coupling $\mathcal{C}$ between $\mathbb{P}$ and $\mathbb{Q}$ such that $\frac{d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q})}{\mathbf{Pr}_{\mathcal{C}}[X\neq Y]} \geq \gamma$, together with an oracle that can be preprocessed in time T_1 and supports the following query:*

- *Discrepancy query: return the value of $\mathbf{Pr}_{\mathcal{C}}[X \neq Y]$ in time $O(1)$.*

Furthermore, if $\mathbf{Pr}_{\mathcal{C}}[X \neq Y] > 0$, then the oracle also supports the following two queries:

- *Sampling query: draw an independent sample of X from the conditional distribution of X given $X \neq Y$ under $\mathcal{C}$ in time T_2. Equivalently, it returns $\omega \in \Omega$ with probability $\mathbf{Pr}_{\mathcal{C}}[X = \omega \mid X \neq Y]$.*
- *Evaluation query: given any $\omega \in \Omega$, return the value of $\mathbf{Pr}_{\mathcal{C}}[X = \omega \wedge X \neq Y]$ in time T_3.*

Under Assumption 8 and Assumption 9, the following theorem gives an algorithm for approximating the TV-distance between two *arbitrary* discrete distributions $\mathbb{P}$ and $\mathbb{Q}$.

▶ **Theorem 10** ([12]). *Suppose Assumption 8 and Assumption 9 hold for distributions $\mathbb{P}$ and $\mathbb{Q}$ with parameters $\gamma, T_0, T_1, T_2, T_3$. Then there exists a randomized algorithm such that, given any $\varepsilon > 0$, the parameters γ, and access to the two oracles above, it outputs a random number $\hat{d}$ in time $O(T_1 + \frac{T_0+T_2+T_3}{\gamma\varepsilon^2})$ satisfying $\mathbf{Pr}\left[(1-\varepsilon)d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q}) \leq \hat{d} \leq (1+\varepsilon)d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q})\right] \geq 99\%$.*

The proof can be found in the full version.

4 Algorithm for Mixtures of Product Distributions

To apply Theorem 10 to the problem of approximating the TV-distance between two mixtures of product distributions, we need to verify Assumption 8 and Assumption 9. By the definition of mixture of product distributions in (1), the first assumption holds with $T_0 = O(n(k_1+k_2))$. Our main task is to construct a coupling to verify the second assumption.

▶ **Lemma 11.** *Let $k_1, k_2 \geq 1$ be two constants. For any two mixtures of product distributions $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}$ and $\mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}$, there exists a coupling $\mathcal{C}$ between $\mathbb{P}$ and $\mathbb{Q}$ such that Assumption 9 holds with parameters $\gamma = (4nq)^{-k_1-k_2+1}$, $T_1, T_2, T_3 = O(q^2(nq)^{k_1+k_2-1})$.*

Theorem 2 is a simple corollary of Theorem 10 and Lemma 11. The rest of this section is devoted to the proof of the lemma.

4.1 The recursive coupling for mixtures of product distributions

In this subsection, we construct a recursive coupling between two mixtures of product distributions $\mathbb{P}$ and $\mathbb{Q}$ in two steps. In Section 4.1.1, we first introduce a recursive procedure for drawing a sample from a single mixture $\mathbb{P}$. Building on this procedure, in Section 4.1.2 we then construct a recursive coupling between $\mathbb{P}$ and $\mathbb{Q}$. Our construction is inspired by the technique of [20], who develop a recursive analysis that bounds the total variation distance between two mixtures of product distributions up to a *fixed* error. Our main contribution is a new recursive coupling process, which is crucial for approximating the TV-distance with *arbitrary* relative error. In Section 4.2, we use this process to implement an oracle that answers the queries in Assumption 9 within polynomial time.

4.1.1 The recursive sampling process

We now describe a recursive sampling procedure that generates a sample X from the mixture of product distributions

$$\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}.$$

This sampling procedure will later serve as the basis for our coupling construction. Throughout the recursion, the component product distributions $\mathbb{P}^{(s)}$ remain fixed, while the mixing weights are updated from one recursive step to the next. We use $(\bar{\alpha}^{(s)})_{s=1}^{k_1}$ to denote the mixing weights at the current recursive step, to distinguish them from the original mixing weights $(\alpha^{(s)})_{s=1}^{k_1}$. At recursion depth j, let

$$\bar{\mathbb{P}} = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \bigotimes_{i=j}^{n} \mathbb{P}_i^{(s)}$$

be the current mixture over $[q]^{n-j+1}$. We consider the recursive coordinate-wise procedure

$$\text{RecursiveSample}\,(j, \bar{\alpha})\,,$$

where $j \in [n]$ is the current coordinate. At this point, the coordinates $X_1, \ldots, X_{j-1}$ have already been fixed, and the task is to generate X_j from the current mixture $\bar{\mathbb{P}}$ and then update the mixing weights for the next recursive step. To describe the procedure, we define a lower-bound selector.

▶ **Definition 12** (lower-bound selector). *A lower-bound selector $\ell^{\mathbb{P}}$ is a function $\ell^{\mathbb{P}}(j, \bar{\alpha}, c)$ defined for any coordinate $j \in [n]$, any mixing-weight vector $\bar{\alpha} = (\bar{\alpha}^{(s)})_{s=1}^{k_1}$, and any $c \in [q]$, satisfying:*

$$\min_{s \in [k_1] : \bar{\alpha}^{(s)} > 0} \mathbb{P}_j^{(s)}(c) \geq \ell^{\mathbb{P}}(j, \bar{\alpha}, c) \geq 0. \tag{4}$$

Here the minimum ranges over all components $s \in [k_1]$ with $\bar{\alpha}^{(s)} > 0$, namely over the components that are still active at the current recursive step. Thus $\ell^{\mathbb{P}}(j, \bar{\alpha}, c)$ is required to be a nonnegative lower bound on the marginal probability of seeing value c in coordinate j across all active components. The above definition does *not* specify a unique lower-bound selector $\ell^{\mathbb{P}}$. Any function satisfying (4) is valid. Let us fix an arbitrary lower-bound selector $\ell^{\mathbb{P}}$ satisfying (4). Our recursive sampling process in Section 4.1.1 works for any lower-bound selector $\ell^{\mathbb{P}}$. In Section 4.1.2, when constructing the coupling, we will specify a particular choice of $\ell^{\mathbb{P}}$.

For the current recursive state $(j, (\bar{\alpha}^{(s)})_{s=1}^{k_1})$, we first compute the lower bounds $\ell^{\mathbb{P}}(j, \bar{\alpha}, c)$ for all $c \in [q]$. Fix a value $c \in [q]$ and an assignment $\omega \in [q]^{n-j+1}$ satisfying $\omega_j = c$. The probability of ω under the current mixture $\bar{\mathbb{P}}$ can be written as

$$\bar{\mathbb{P}}(\omega) = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \prod_{i=j}^{n} \mathbb{P}_i^{(s)}(\omega_i) = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \mathbb{P}_j^{(s)}(c) \prod_{i=j+1}^{n} \mathbb{P}_i^{(s)}(\omega_i),$$

where the second equality uses the assumption $\omega_j = c$. By (4), it holds that $\mathbb{P}_j^{(s)}(c) \geq \ell^{\mathbb{P}}(j, \bar{\alpha}, c)$ for all $s \in [k_1]$ with $\bar{\alpha}^{(s)} > 0$. Hence, for each s with $\bar{\alpha}^{(s)} > 0$, we can decompose $\mathbb{P}_j^{(s)}(c)$ as two non-negative terms: $\ell^{\mathbb{P}}(j, \bar{\alpha}, c)$ and $\mathbb{P}_j^{(s)}(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c)$. We have the following decomposition:

$$\bar{\mathbb{P}}(\omega) = \ell^{\mathbb{P}}(j, \bar{\alpha}, c) \underbrace{\sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \prod_{i=j+1}^{n} \mathbb{P}_i^{(s)}(\omega_i)}_{A} + \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \left(\mathbb{P}_j^{(s)}(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c) \right) \prod_{i=j+1}^{n} \mathbb{P}_i^{(s)}(\omega_i). \tag{5}$$

The above summation enumerates all $s \in [k_1]$, which may contain inactive components s with $\bar{\alpha}^{(s)} = 0$. However, those inactive components only contribute 0 to the whole sum. In (5), the first term is the product of $\ell^{\mathbb{P}}(j, \bar{\alpha}, c)$ and A, where A is exactly the probability of the suffix $(\omega_i)_{i=j+1}^{n}$ under the current mixture $\sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \bigotimes_{i=j+1}^{n} \mathbb{P}_i^{(s)}$. To write the second term in a similar form, we introduce a modified mixing weight $\bar{\alpha}_{j,c}^{(s)}$ for all $s \in [k_1]$ as follows:

$$\bar{\alpha}_{j,c}^{(s)} \triangleq \begin{cases} \bar{\alpha}^{(s)}, & \bar{\mathbb{P}}_j(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c) = 0; \\ \bar{\alpha}^{(s)} \frac{\mathbb{P}_j^{(s)}(c) - \ell^{\mathbb{P}}(j,\bar{\alpha},c)}{\bar{\mathbb{P}}_j(c) - \ell^{\mathbb{P}}(j,\bar{\alpha},c)}, & \text{otherwise,} \end{cases} \quad \text{where } \bar{\mathbb{P}}_j(c) = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \mathbb{P}_j^{(s)}(c).$$

We next verify that $\bar{\alpha}_{j,c}$ defines a valid mixing-weight vector. By Definition 12, $\mathbb{P}_j^{(s)}(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c) \geq 0$, and $\bar{\mathbb{P}}_j(c) = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \mathbb{P}_j^{(s)}(c)$ is an average of the active values $\mathbb{P}_j^{(s)}(c)$, therefore $\bar{\mathbb{P}}_j(c) \geq \ell^{\mathbb{P}}(j, \bar{\alpha}, c)$. If $\bar{\mathbb{P}}_j(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c) = 0$, $\bar{\alpha}_{j,c} = \bar{\alpha}$ is a valid mixing-weight vector as long as $\bar{\alpha}$ is valid. Otherwise, when $\bar{\mathbb{P}}_j(c) > \ell^{\mathbb{P}}(j, \bar{\alpha}, c)$,

$$\sum_{s=1}^{k_1} \bar{\alpha}_{j,c}^{(s)} = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \frac{\mathbb{P}_j^{(s)}(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c)}{\bar{\mathbb{P}}_j(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c)} = \frac{\sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \mathbb{P}_j^{(s)}(c)}{\bar{\mathbb{P}}_j(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c)} - \frac{\ell^{\mathbb{P}}(j, \bar{\alpha}, c)}{\bar{\mathbb{P}}_j(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c)} = 1.$$

Hence, for $\omega \in [q]^{n-j+1}$ with $\omega_j = c$, we can rewrite $\bar{\mathbb{P}}(\omega)$ as

$$\bar{\mathbb{P}}(\omega) = \ell^{\mathbb{P}}(j, \bar{\alpha}, c) \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \prod_{i=j+1}^{n} \mathbb{P}_i^{(s)}(\omega_i) + \left(\bar{\mathbb{P}}_j(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c)\right) \sum_{s=1}^{k_1} \bar{\alpha}_{j,c}^{(s)} \prod_{i=j+1}^{n} \mathbb{P}_i^{(s)}(\omega_i). \quad (6)$$

Equation (6) naturally suggests a recursive sampling rule. Given the state $(j, (\bar{\alpha}^{(s)})_{s=1}^{k_1})$, we first sample the current coordinate X_j, and then reduce the task of sampling the suffix $X_{j+1:n}$ to sampling from one of two smaller mixtures. Concretely, once the current coordinate is set to c, the suffix distribution is either $\sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \bigotimes_{i=j+1}^{n} \mathbb{P}_i^{(s)}$ or $\sum_{s=1}^{k_1} \bar{\alpha}_{j,c}^{(s)} \bigotimes_{i=j+1}^{n} \mathbb{P}_i^{(s)}$. The component distributions remain the same (except that their dimension is reduced by one), while the mixing weights may change from $\bar{\alpha}$ to $\bar{\alpha}_{j,c}$. Applying this rule recursively yields the sampling procedure shown in Algorithm 1.

In Algorithm 1, the lower-bound selector function $\ell^{\mathbb{P}}$ is treated as part of the input. The following lemma shows that the algorithm outputs a correct sample from $\mathbb{P}$ as long as $\ell^{\mathbb{P}}$ satisfies (4).

▶ **Lemma 13.** *Given a mixture of product distributions $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}$ and an arbitrary lower-bound selector function $\ell^{\mathbb{P}}$ satisfying (4), the procedure RecursiveSample$(1, \alpha)$ in Algorithm 1 outputs a sample $X \sim \mathbb{P}$.*

Lemma 13 follows by induction on the recursion depth using (6). The proof can be found in the full version.

The same recursive procedure applies to any mixture of product distributions. In particular, we may use it to sample from $\bar{\mathbb{Q}} = \sum_{t=1}^{k_2} \bar{\beta}^{(t)} \bigotimes_{i=j}^{n} \mathbb{Q}_i^{(t)}$ by replacing k_1 with k_2, replacing $(\bar{\alpha}^{(s)})_{s=1}^{k_1}$ with $(\bar{\beta}^{(t)})_{t=1}^{k_2}$, replacing $\mathbb{P}^{(s)}$ with $\mathbb{Q}^{(t)}$, and using an arbitrary lower-bound selector $\ell^{\mathbb{Q}}(j, \bar{\beta}, c)$ satisfying

$$\min_{t \in [k_2]: \bar{\beta}^{(t)} > 0} \mathbb{Q}_j^{(t)}(c) \geq \ell^{\mathbb{Q}}(j, \bar{\beta}, c) \geq 0. \quad (7)$$

Lemma 13 also shows that by substituting P with Q and α with β, the procedure RecursiveSample$(1, \beta)$ outputs a sample $Y \sim \mathbb{Q}$.

Algorithm 1 Recursive sampling process for mixtures of product distributions.

1 RecursiveSample $(j, \bar{\alpha})$;
Parameter : an index $j \in [n+1]$ and mixing weights $\bar{\alpha}^{(1)}, \bar{\alpha}^{(2)}, \ldots, \bar{\alpha}^{(k_1)}$
Static Input : product distributions $\mathbb{P}^{(1)}, \mathbb{P}^{(2)}, \ldots, \mathbb{P}^{(k_1)}$ over $[q]^n$; an arbitrary lower-bound selector function $\ell^{\mathbb{P}}$ satisfying (4)
Output : a random sample $X = (X_j, X_{j+1}, \ldots, X_n) \sim \bar{\mathbb{P}} = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \bigotimes_{i=j}^{n} \mathbb{P}_i^{(s)}$
2 **if** $j > n$ **then return** $\emptyset$;
3 Compute the probability lower bound $\ell^{\mathbb{P}}(j, \bar{\alpha}, c)$ for all $c \in [q]$;
4 Sample a random pair $(c, e) \in [q] \times \{0, 1\}$ such that for each $c \in [q]$, the probability of $(c, 0)$ is $\ell^{\mathbb{P}}(j, \bar{\alpha}, c)$ and the probability of $(c, 1)$ is $\bar{\mathbb{P}}_j(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c)$, where $\bar{\mathbb{P}}_j(c) = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \mathbb{P}_j^{(s)}(c)$ is the marginal probability of coordinate j under $\bar{\mathbb{P}}$;
5 **if** $e = 0$ **then**
6 Sample $(X_{j+1}, X_{j+2}, \ldots, X_n) \leftarrow$ RecursiveSample$(j+1, \bar{\alpha})$;
7 **else**
8 Compute mixing weights $\bar{\alpha}_{j,c}^{(s)} = \bar{\alpha}^{(s)} \frac{\mathbb{P}_j^{(s)}(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c)}{\bar{\mathbb{P}}_j(c) - \ell^{\mathbb{P}}(j, \bar{\alpha}, c)}$ for all $s \in [k_1]$;
9 Sample $(X_{j+1}, X_{j+2}, \ldots, X_n) \leftarrow$ RecursiveSample$(j+1, \bar{\alpha}_{j,c})$;
10 **return** $(X_j = c, X_{j+1}, X_{j+2}, \ldots, X_n)$;

4.1.2 The recursive coupling process

We now define a coupling between two mixtures of product distributions

$$\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)} \qquad \text{and} \qquad \mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}.$$

At a high level, the coupling runs Algorithm 1 on both $\mathbb{P}$ and $\mathbb{Q}$ to generate samples $X \sim \mathbb{P}$ and $Y \sim \mathbb{Q}$, while sharing randomness between the two executions so that the resulting pair (X, Y) is a coupling of $\mathbb{P}$ and $\mathbb{Q}$.

We use RecursiveSample$(1, \alpha)$ to generate a sample $X \sim \mathbb{P}$ and RecursiveSample$(1, \beta)$ to generate a sample $Y \sim \mathbb{Q}$, and couple the two recursive procedures at the same time. Suppose that at some point the execution for $\mathbb{P}$ is at recursion state RecursiveSample$(j, \bar{\alpha})$ and the execution for $\mathbb{Q}$ is at recursion state RecursiveSample$(j, \bar{\beta})$, where $\bar{\alpha} = (\bar{\alpha}^{(s)})_{s=1}^{k_1}$ and $\bar{\beta} = (\bar{\beta}^{(t)})_{t=1}^{k_2}$ denote the current mixing weights and may differ from the original vectors α and β. We define a shared lower-bound selector $\ell(j, \bar{\alpha}, \bar{\beta}, c)$ as follows.

▶ **Definition 14.** *For any $j \in [n]$, $c \in [q]$, and valid mixing-weight vectors $\bar{\alpha} = (\bar{\alpha}^{(s)})_{s=1}^{k_1}$ and $\bar{\beta} = (\bar{\beta}^{(t)})_{t=1}^{k_2}$, define*

$$\ell\left(j, \bar{\alpha}, \bar{\beta}, c\right) \triangleq \min\left\{ \min_{s \in [k_1] : \bar{\alpha}^{(s)} > 0} \mathbb{P}_j^{(s)}(c), \min_{t \in [k_2] : \bar{\beta}^{(t)} > 0} \mathbb{Q}_j^{(t)}(c) \right\}.$$

The one-sided lower-bound selectors used by the two recursive sampling procedures are then defined by

$$\ell^{\mathbb{P}}(j, \bar{\alpha}, c) = \ell^{\mathbb{Q}}(j, \bar{\beta}, c) = \ell(j, \bar{\alpha}, \bar{\beta}, c).$$

By construction, $\ell^{\mathbb{P}}(j, \bar{\alpha}, c)$ satisfies (4) for $\mathbb{P}$, and $\ell^{\mathbb{Q}}(j, \bar{\beta}, c)$ satisfies (7) for $\mathbb{Q}$. In Line 4 of Algorithm 1, each process draws a random pair $(c, e) \in [q] \times \{0, 1\}$. Let $(c_{\bar{\mathbb{P}}}, e_{\bar{\mathbb{P}}})$ and $(c_{\bar{\mathbb{Q}}}, e_{\bar{\mathbb{Q}}})$ denote the pairs drawn by RecursiveSample$(j, \bar{\alpha})$ and RecursiveSample$(j, \bar{\beta})$, respectively. Since both procedures use the same lower bounds $\ell(j, \bar{\alpha}, \bar{\beta}, c)$, we define a coupling $\bar{\mathcal{C}}$ between these two pairs as follows:

- For each $c \in [q]$, $\mathbf{Pr}_{\bar{\mathcal{C}}}\left[c_{\bar{\mathbb{P}}} = c_{\bar{\mathbb{Q}}} = c \wedge e_{\bar{\mathbb{P}}} = e_{\bar{\mathbb{Q}}} = 0\right] = \ell(j, \bar{\alpha}, \bar{\beta}, c)$.
- For each $c \in [q]$, $\mathbf{Pr}_{\bar{\mathcal{C}}}\left[c_{\bar{\mathbb{P}}} = c_{\bar{\mathbb{Q}}} = c \wedge e_{\bar{\mathbb{P}}} = e_{\bar{\mathbb{Q}}} = 1\right] = \min\{\bar{\mathbb{P}}_j(c), \bar{\mathbb{Q}}_j(c)\} - \ell(j, \bar{\alpha}, \bar{\beta}, c)$, where $\bar{\mathbb{P}} = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \bigotimes_{i=j}^{n} \mathbb{P}_i^{(s)}$ and $\bar{\mathbb{Q}} = \sum_{t=1}^{k_2} \bar{\beta}^{(t)} \bigotimes_{i=j}^{n} \mathbb{Q}_i^{(t)}$ are the mixtures of product distributions at the current recursive step.
- Given the above two points, for each $c \in A = \{x \in [q] \mid \bar{\mathbb{P}}_j(x) > \bar{\mathbb{Q}}_j(x)\}$, the random variable $(c_{\bar{\mathbb{P}}}, e_{\bar{\mathbb{P}}})$ has a remaining probability $\bar{\mathbb{P}}_j(c) - \bar{\mathbb{Q}}_j(c)$ of taking value $(c, 1)$; and for each $c \in B = \{x \in [q] \mid \bar{\mathbb{P}}_j(x) < \bar{\mathbb{Q}}_j(x)\}$, the random variable $(c_{\bar{\mathbb{Q}}}, e_{\bar{\mathbb{Q}}})$ has remaining probability $\bar{\mathbb{Q}}_j(c) - \bar{\mathbb{P}}_j(c)$ of taking value $(c, 1)$. The total remaining probabilities on the two sides are equal: $\sum_{c \in A}(\bar{\mathbb{P}}_j(c) - \bar{\mathbb{Q}}_j(c)) = \sum_{c \in B}(\bar{\mathbb{Q}}_j(c) - \bar{\mathbb{P}}_j(c))$. We therefore couple the remaining mass arbitrarily. Since A and B are disjoint, in this case we necessarily have

$$c_{\bar{\mathbb{P}}} \neq c_{\bar{\mathbb{Q}}} \quad \text{and} \quad e_{\bar{\mathbb{P}}} = e_{\bar{\mathbb{Q}}} = 1.$$

Using the coupling $\bar{\mathcal{C}}$ between $(c_{\bar{\mathbb{P}}}, e_{\bar{\mathbb{P}}})$ and $(c_{\bar{\mathbb{Q}}}, e_{\bar{\mathbb{Q}}})$, we now couple two sampling processes RecursiveSample$(j, \bar{\alpha})$ and RecursiveSample$(j, \bar{\beta})$ as follows:

- Sample $(c_{\bar{\mathbb{P}}}, e_{\bar{\mathbb{P}}})$ and $(c_{\bar{\mathbb{Q}}}, e_{\bar{\mathbb{Q}}})$ according to $\bar{\mathcal{C}}$.
- If $c_{\bar{\mathbb{P}}} = c_{\bar{\mathbb{Q}}} = c$ for some $c \in [q]$ and $e_{\bar{\mathbb{P}}} = e_{\bar{\mathbb{Q}}} = 0$, then we set $X_j = Y_j = c$ and recursively couple the suffixes via RecursiveCouple$(j+1, \bar{\alpha}, \bar{\beta})$.
- If $c_{\bar{\mathbb{P}}} = c_{\bar{\mathbb{Q}}} = c$ for some $c \in [q]$ and $e_{\bar{\mathbb{P}}} = e_{\bar{\mathbb{Q}}} = 1$, then we set $X_j = Y_j = c$ and recursively couple the suffixes via RecursiveCouple$(j+1, \bar{\alpha}_{j,c}, \bar{\beta}_{j,c})$.
- In the remaining case, we have $X_j = c_{\bar{\mathbb{P}}} \neq c_{\bar{\mathbb{Q}}} = Y_j$ and $e_{\bar{\mathbb{P}}} = e_{\bar{\mathbb{Q}}} = 1$. Hence the coupling already fails at coordinate j, and we simply sample the suffixes independently using RecursiveSample$(j+1, \bar{\alpha}_{j,c_{\bar{\mathbb{P}}}})$ and RecursiveSample$(j+1, \bar{\beta}_{j,c_{\bar{\mathbb{Q}}}})$.

The pseudocode of the coupling process is given in Algorithm 2.

▶ **Lemma 15** (Validity of recursive coupling). *Given two mixtures of product distributions $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}$ and $\mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}$, fix a shared lower-bound selector ℓ as in Definition 14. Then RecursiveCouple$(1, \alpha, \beta)$ in Algorithm 2 outputs joint random variables (X, Y) such that $X \sim \mathbb{P}$ and $Y \sim \mathbb{Q}$.*

Lemma 15 follows by a straightforward induction. The proof can be found in the full version.

4.1.3 Discrepancy of recursive coupling

We now analyze the discrepancy of the recursive coupling process. Let $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}$ and $\mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}$ be two mixtures of product distributions. Define

$$(X, Y) \leftarrow \text{RecursiveCouple}(1, \alpha, \beta).$$

Let $\mathcal{C}_{\mathrm{RC}}$ denote the joint distribution of (X, Y). By the coupling inequality, the discrepancy $\mathbf{Pr}_{\mathcal{C}_{\mathrm{RC}}}[X \neq Y]$ is always at least $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})$. The following lemma provides an upper bound on the discrepancy of our recursive coupling.

▶ **Lemma 16** (Discrepancy of recursive coupling). *Let $[q]$ be a finite domain. For two mixtures of product distributions $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}$ and $\mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}$ over $[q]^n$, the recursive coupling satisfies*

$$\mathbf{Pr}_{\mathcal{C}_{\mathrm{RC}}}[X \neq Y] \leq (4nq)^{k_1+k_2-1} \cdot d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q}).$$

The proof of Lemma 16 follows the known techniques of [20] and can be found in the full version.

Algorithm 2 Recursive coupling process for mixtures of product distributions.

RecursiveCouple $(j, \bar{\alpha}, \bar{\beta})$;
Parameter : an index $j \in [n+1]$ and mixing weights $\bar{\alpha} = (\bar{\alpha}^{(s)})_{s=1}^{k_1}$, $\bar{\beta} = (\bar{\beta}^{(t)})_{t=1}^{k_2}$
Static Input : product distributions $\mathbb{P}^{(1)}, \mathbb{P}^{(2)}, \ldots, \mathbb{P}^{(k_1)}$ and $\mathbb{Q}^{(1)}, \mathbb{Q}^{(2)}, \ldots, \mathbb{Q}^{(k_2)}$ over $[q]^n$; a fixed shared probability lower-bound selector ℓ as in Definition 14
Output : joint random variables (X, Y) such that $X = (X_j, X_{j+1}, \ldots, X_n) \sim \bar{\mathbb{P}} = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)} \bigotimes_{i=j}^{n} \mathbb{P}_i^{(s)}$ and $Y = (Y_j, Y_{j+1}, \ldots, Y_n) \sim \bar{\mathbb{Q}} = \sum_{t=1}^{k_2} \bar{\beta}^{(t)} \bigotimes_{i=j}^{n} \mathbb{Q}_i^{(t)}$
if $j > n$ **then return** $(\emptyset, \emptyset)$;
Compute $\ell(j, \bar{\alpha}, \bar{\beta}, c)$ for all $c \in [q]$, and then use the coupling $\bar{\mathcal{C}}$ to jointly sample $(c_{\bar{\mathbb{P}}}, e_{\bar{\mathbb{P}}})$ and $(c_{\bar{\mathbb{Q}}}, e_{\bar{\mathbb{Q}}})$;
if $e_{\bar{\mathbb{P}}} = e_{\bar{\mathbb{Q}}} = 0$ **then**
    Sample $(X_{j+1:n}, Y_{j+1:n}) \leftarrow$ RecursiveCouple$(j+1, \bar{\alpha}, \bar{\beta})$;
else
    Let $\bar{\alpha}_{j,c_{\bar{\mathbb{P}}}}^{(s)} = \bar{\alpha}^{(s)} \frac{\mathbb{P}_j^{(s)}(c_{\bar{\mathbb{P}}}) - \ell(j, \bar{\alpha}, \bar{\beta}, c_{\bar{\mathbb{P}}})}{\bar{\mathbb{P}}_j(c_{\bar{\mathbb{P}}}) - \ell(j, \bar{\alpha}, \bar{\beta}, c_{\bar{\mathbb{P}}})}$ for $s \in [k_1]$ and $\bar{\beta}_{j,c_{\bar{\mathbb{Q}}}}^{(t)} = \bar{\beta}^{(t)} \frac{\mathbb{Q}_j^{(t)}(c_{\bar{\mathbb{Q}}}) - \ell(j, \bar{\alpha}, \bar{\beta}, c_{\bar{\mathbb{Q}}})}{\bar{\mathbb{Q}}_j(c_{\bar{\mathbb{Q}}}) - \ell(j, \bar{\alpha}, \bar{\beta}, c_{\bar{\mathbb{Q}}})}$ for $t \in [k_2]$;
    if $c_{\bar{\mathbb{P}}} = c_{\bar{\mathbb{Q}}}$ **then**
        Sample $(X_{j+1:n}, Y_{j+1:n}) \leftarrow$ RecursiveCouple$(j+1, \bar{\alpha}_{j,c}, \bar{\beta}_{j,c})$, where $c = c_{\bar{\mathbb{P}}} = c_{\bar{\mathbb{Q}}}$;
    else
        Sample $X_{j+1:n} \leftarrow$ RecursiveSample$(j+1, \bar{\alpha}_{j,c_{\bar{\mathbb{P}}}})$ with $\ell^{\mathbb{P}} = 0$;
        Sample $Y_{j+1:n} \leftarrow$ RecursiveSample$(j+1, \bar{\beta}_{j,c_{\bar{\mathbb{Q}}}})$ with $\ell^{\mathbb{Q}} = 0$;
Let $X = (X_j = c_{\bar{\mathbb{P}}}, X_{j+1:n})$ and $Y = (Y_j = c_{\bar{\mathbb{Q}}}, Y_{j+1:n})$ and return (X, Y);

4.2 Implementation of oracles

In this section, we implement the oracle queries in Assumption 9 based on Algorithm 2. Consider the execution of Algorithm 2 on two mixtures of product distributions $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}_i^{(s)}$ and $\mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}_i^{(t)}$. Define the following set of states and transitions between them.

For each recursion instance RecursiveCouple$(j, \bar{\alpha}, \bar{\beta})$, we create a state S with label $(j, \bar{\alpha}, \bar{\beta})$, where $1 \le j \le n+1$. A state S is said to be in layer j if j is the first coordinate of the label. We add a special failure state $S = \perp$ to denote that the coupling fails. Let $\mathcal{S}$ denote the set of all possible states that can be reached by the coupling process with a *positive* probability, including the failure state $\perp$.

For each state $S \in \mathcal{S}$ with label $(j, \bar{\alpha}, \bar{\beta})$ for $j \in [n]$, we define the following transitions according to the coupling process. The coupling process samples $(c_{\bar{\mathbb{P}}}, e_{\bar{\mathbb{P}}})$ and $(c_{\bar{\mathbb{Q}}}, e_{\bar{\mathbb{Q}}})$ for $\bar{\mathbb{P}}$ and $\bar{\mathbb{Q}}$, respectively, and then makes the transition.

- **Type-I**: For any $c \in [q]$ with $\ell(j, \bar{\alpha}, \bar{\beta}, c) > 0$, with probability $\ell(j, \bar{\alpha}, \bar{\beta}, c)$, S moves to a new state S' with label $(j+1, \bar{\alpha}, \bar{\beta})$. Define a transition t from S to S' with weight $w(t) = \ell(j, \bar{\alpha}, \bar{\beta}, c)$ and label $L(t) = (c, c)$. This means that S moves to S' through t with probability $\ell(j, \bar{\alpha}, \bar{\beta}, c)$ and sets $X_j = Y_j = c$.
- **Type-II**: For any $c \in [q]$ with $p_c \triangleq \min\{\bar{\mathbb{P}}_j(c), \bar{\mathbb{Q}}_j(c)\} - \ell(j, \bar{\alpha}, \bar{\beta}, c) > 0$, with probability p_c, S moves to a new state S' with label $(j+1, \bar{\alpha}_{j,c}, \bar{\beta}_{j,c})$. Define a transition t from S to S' with weight $w(t) = p_c$ and label $L(t) = (c, c)$. This means that S moves to S' through t with probability p_c and sets $X_j = Y_j = c$.

- **Type-III**: With the remaining probability, the coupling fails ($X_j \neq Y_j$). For any $c, c' \in [q]$ with $c \neq c'$ and $p_{c,c'} \triangleq \mathbf{Pr}\left[c_{\bar{\mathbb{P}}} = c \wedge c_{\bar{\mathbb{Q}}} = c'\right] > 0$, define a transition t from S to the failure state $\perp$ with weight $w(t) = p_{c,c'}$ and label $L(t) = (X_{j:n}, Y_{j:n})$, where $X_j = c$, $Y_j = c'$, and $X_{j+1:n}, Y_{j+1:n}$ are sampled independently in Lines 11 and 12 of Algorithm 2. Hence, the label contains a pair of random configurations, both of length $n - j + 1$. The label $L(t)$ is random because we further sample the pair $(X_{j+1:n}, Y_{j+1:n})$. We remark that there are multiple transitions from S to the failure state $\perp$ with different weights and labels.

▶ **Lemma 17.** *The total number of states is bounded by* $|\mathcal{S}| \leq (nq+1)^{k_1+k_2-1} + 1$.

The proof can be found in the full version.

The oracle queries in Assumption 9 can be implemented by the following algorithm. One execution of the coupling process can be represented as a random walk on the state space $\mathcal{S}$. The random walk starts from the root state $S = S_{\text{root}} = (1, \alpha, \beta)$. At each step, the random walk samples a transition $t = (S \to S')$ from S with probability $w(t)$ and moves to the next state S'. The random walk ends either at a state in layer $n+1$ (coupling succeeds) or at the failure state $\perp$ (coupling fails). The coupling process outputs a pair $(X_{1:n}, Y_{1:n})$, which is determined by the labels of all transitions along the random walk.

Preprocessing step. We build the DAG with all states in $\mathcal{S}$ and all transitions between them. Using Lemma 17, the total number of states and transitions in the DAG is at most $O(q^2(nq+1)^{k_1+k_2-1})$. For each state $S \in \mathcal{S}$, compute $p_{\text{fail}}(S)$, the probability that a random walk starting from S reaches the failure state $\perp$. By definition,

$$p_{\text{fail}}(\perp) = 1 \text{ and } p_{\text{fail}}(S) = 0 \text{ for all } S \text{ in layer } n+1.$$

For each non-failed state $S \in \mathcal{S}$ in layer $j \in [n]$, we have the following recursive relation:

$$p_{\text{fail}}(S) = \sum_{t=(S\to S') \in \text{transitions from } S} w(t) p_{\text{fail}}(S').$$

Using dynamic programming, we can compute $p_{\text{fail}}(S)$ for all $S \in \mathcal{S}$ in time $O(q^2(nq+1)^{k_1+k_2-1})$. By the definition of $p_{\text{fail}}(S)$, the failure probability of the recursive coupling is

$$\mathbf{Pr}_{\mathcal{C}_{\text{RC}}}[X \neq Y] = p_{\text{fail}}(S_{\text{root}}).$$

Sampling query. Next, we give an algorithm that generates a random sample of X conditional on $X \neq Y$ in the recursive coupling. Equivalently, we can sample a random walk on the DAG starting from the root state S_{root}, conditional on the random walk reaching the failure state $\perp$. If $p_{\text{fail}}(S_{\text{root}}) = 0$, then $d_{\text{TV}}(\mathbb{P}, \mathbb{Q}) = 0$ by the coupling inequality, and this sampling query is not needed. Otherwise, the query can be implemented by the following algorithm.

- Initialize $S = S_{\text{root}}$.
- Sample a random transition $t = (S \to S')$ from S with probability $\frac{w(t)p_{\text{fail}}(S')}{p_{\text{fail}}(S)}$ and then update the current state S to S'.
- The process terminates when S reaches the failure state $\perp$.

The above process gives a random sequence of transitions on the DAG. By concatenating the labels in the sequence, we obtain a random pair $(X, Y) \in [q]^n \times [q]^n$, and we output the sample $X \in [q]^n$.

The correctness of the conditional sample X follows directly from the definition of $p_{\text{fail}}(\cdot)$. The running time to generate a random sample X is $O(q^2(nq+1)^{k_1+k_2-1})$.

Evaluation query. Given a fixed configuration $\sigma \in [q]^n$, we want to evaluate the probability

$$\mathbf{Pr}_{\mathcal{C}_{\mathrm{RC}}}[X = \sigma \wedge X \neq Y].$$

For each non-failed state $S \in \mathcal{S}$ in layer $j \in [n]$, let $p^{\sigma}_{\mathrm{fail}}(S)$ be the probability that a random walk starting from S reaches the failure state $\perp$ and that the label $X_{j:n}$ of the random walk is exactly $\sigma_{j:n}$. By definition, since S_{root} is the layer-1 state, it holds that

$$p^{\sigma}_{\mathrm{fail}}(S_{\mathrm{root}}) = \mathbf{Pr}_{\mathcal{C}_{\mathrm{RC}}}[X = \sigma \wedge X \neq Y].$$

For each $S \in \mathcal{S}$ in layer $j = n+1$, by definition, we have $p^{\sigma}_{\mathrm{fail}}(S) = 0$.
Fix any non-failed state $S \in \mathcal{S}$ in layer $j \in [n]$ with $S = (j, \bar{\alpha}, \bar{\beta})$. Define

$$\mathcal{T} = \{t \mid t \text{ is a transition from } S \text{ to } S' \neq \perp \text{ and } L(t) = (\sigma_j, \sigma_j)\}.$$

The above definition considers all transitions from S to non-failed states S'. We still need to consider the type-III transitions from S to the failure state $\perp$. By definition, there are at most $q(q-1)$ such transitions. Let $\mathcal{T}_{\perp}(S)$ denote the set of all type-III transitions from S to the failure state $\perp$. Every such transition can be specified by a pair (c, c') with $c \neq c'$. We use $t_{c,c'}$ to denote the corresponding type-III transition. Note that the weight $w(t_{c,c'})$ is $\mathbf{Pr}\left[c_{\bar{\mathbb{P}}} = c \wedge c_{\bar{\mathbb{Q}}} = c'\right]$ in the coupling at Line 3 of Algorithm 2. The label of $t_{c,c'}$ is a random pair $(X_{j:n}, Y_{j:n})$, where $X_j = c$, $Y_j = c'$, and $X_{j+1:n}$ and $Y_{j+1:n}$ are sampled independently in Lines 11 and 12 of Algorithm 2. We define

$$\tau(t_{c,c'}, \sigma) \triangleq \mathbf{Pr}[X_{j+1:n} = \sigma_{j+1:n}] = \sum_{s=1}^{k_1} \bar{\alpha}^{(s)}_{j,c} \prod_{i=j+1}^{n} \mathbb{P}^{(s)}_i(\sigma_i).$$

Now, we have the following recursive relation:

$$p^{\sigma}_{\mathrm{fail}}(S) = \sum_{t=(S \to S') \in \mathcal{T}} w(t) p^{\sigma}_{\mathrm{fail}}(S') + \sum_{t_{c,c'} \in \mathcal{T}_{\perp}(S): c = \sigma_j} w(t_{c,c'}) \tau(t_{c,c'}, \sigma).$$

Again, we can use dynamic programming on the DAG from layer $n+1$ to layer 1 to compute all values $p^{\sigma}_{\mathrm{fail}}(S)$ for all $S \in \mathcal{S}$ in time $O(q^2(nq+1)^{k_1+k_2-1})$. The final result is $p^{\sigma}_{\mathrm{fail}}(S_{\mathrm{root}})$.

5 Algorithm for Mixtures of Boolean Subcubes

5.1 Reduction to a counting problem

Consider two mixtures of Boolean subcubes $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^{n} \mathbb{P}^{(s)}_i$ and $\mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^{n} \mathbb{Q}^{(t)}_i$. A component $\mathbb{P}^{(s)}$ over $\{0,1\}^n$ is a Boolean subcube if, for any $i \in [n]$, $\mathbb{P}^{(s)}_i(1) \in \{0, \frac{1}{2}, 1\}$. Thus, for any $\omega \in \{0,1\}^n$, $\mathbb{P}^{(s)}(\omega) = \prod_{i=1}^{n} \mathbb{P}^{(s)}_i(\omega_i)$ is a product of values in $\{0, \frac{1}{2}, 1\}$. We define the following partition $\Lambda^{\mathrm{one}}_{\mathbb{P}^{(s)}}$, $\Lambda^{\mathrm{zero}}_{\mathbb{P}^{(s)}}$, and $\Lambda^{\mathrm{half}}_{\mathbb{P}^{(s)}}$ of $[n]$:

$$\Lambda^{\mathrm{one}}_{\mathbb{P}^{(s)}} = \{i \in [n] \mid \mathbb{P}^{(s)}_i(1) = 1\},\ \Lambda^{\mathrm{zero}}_{\mathbb{P}^{(s)}} = \{i \in [n] \mid \mathbb{P}^{(s)}_i(1) = 0\},$$
$$\Lambda^{\mathrm{half}}_{\mathbb{P}^{(s)}} = \left\{i \in [n] \mid \mathbb{P}^{(s)}_i(1) = \frac{1}{2}\right\}.$$

In words, for the product distribution $\mathbb{P}^{(s)}$, the set $\Lambda^{\mathrm{one}}_{\mathbb{P}^{(s)}}$ contains all dimensions whose value is fixed to 1, the set $\Lambda^{\mathrm{zero}}_{\mathbb{P}^{(s)}}$ contains all dimensions whose value is fixed to 0, and the set $\Lambda^{\mathrm{half}}_{\mathbb{P}^{(s)}}$ contains all dimensions whose value is sampled uniformly at random. Similarly, for a component $\mathbb{Q}^{(t)}$, we define the sets $\Lambda^{\mathrm{one}}_{\mathbb{Q}^{(t)}}$, $\Lambda^{\mathrm{zero}}_{\mathbb{Q}^{(t)}}$, and $\Lambda^{\mathrm{half}}_{\mathbb{Q}^{(t)}}$ in the same way.

The probability of each $\omega \in \{0,1\}^n$ can be written as follows:

$$\mathbb{P}^{(s)}(\omega) = \left(\frac{1}{2}\right)^{\left|\Lambda^{\text{half}}_{\mathbb{P}^{(s)}}\right|} \cdot \mathbb{1}\left[\forall i \in \Lambda^{\text{one}}_{\mathbb{P}^{(s)}}, \omega_i = 1\right] \cdot \mathbb{1}\left[\forall i \in \Lambda^{\text{zero}}_{\mathbb{P}^{(s)}}, \omega_i = 0\right],$$

$$\mathbb{Q}^{(t)}(\omega) = \left(\frac{1}{2}\right)^{\left|\Lambda^{\text{half}}_{\mathbb{Q}^{(t)}}\right|} \cdot \mathbb{1}\left[\forall i \in \Lambda^{\text{one}}_{\mathbb{Q}^{(t)}}, \omega_i = 1\right] \cdot \mathbb{1}\left[\forall i \in \Lambda^{\text{zero}}_{\mathbb{Q}^{(t)}}, \omega_i = 0\right],$$

where the indicator function $\mathbb{1}[*]$ returns 1 if statement $*$ holds and 0 otherwise. For the distribution $\mathbb{P}^{(s)}$, we say $\omega \in \{0,1\}^n$ is *feasible* if for all $i \in \Lambda^{\text{one}}_{\mathbb{P}^{(s)}}$, $\omega_i = 1$ and for all $i \in \Lambda^{\text{zero}}_{\mathbb{P}^{(s)}}$, $\omega_i = 0$. Each feasible $\omega \in \{0,1\}^n$ has probability $(1/2)^{|\Lambda^{\text{half}}_{\mathbb{P}^{(s)}}|}$ under the distribution $\mathbb{P}^{(s)}$, while each infeasible $\omega \in \{0,1\}^n$ has probability 0. A similar structural property holds for the distribution $\mathbb{Q}^{(t)}$.

The feasibility of a configuration $\omega \in \{0,1\}^n$ under the distribution $\mathbb{P}^{(s)}$ can be expressed by a Boolean formula $F_{\mathbb{P}^{(s)}} : \{0,1\}^n \to \{0,1\}$ as follows:

$$F_{\mathbb{P}^{(s)}}(\omega) = \bigwedge_{i \in \Lambda^{\text{one}}_{\mathbb{P}^{(s)}}} \omega_i \wedge \bigwedge_{i \in \Lambda^{\text{zero}}_{\mathbb{P}^{(s)}}} \neg\omega_i. \tag{8}$$

Similarly, for the distribution $\mathbb{Q}^{(t)}$, define the Boolean formula $F_{\mathbb{Q}^{(t)}} : \{0,1\}^n \to \{0,1\}$ as follows:

$$F_{\mathbb{Q}^{(t)}}(\omega) = \bigwedge_{i \in \Lambda^{\text{one}}_{\mathbb{Q}^{(t)}}} \omega_i \wedge \bigwedge_{i \in \Lambda^{\text{zero}}_{\mathbb{Q}^{(t)}}} \neg\omega_i. \tag{9}$$

By definition, $\mathbb{P}^{(s)}(\omega) = (1/2)^{|\Lambda^{\text{half}}_{\mathbb{P}^{(s)}}|} \cdot F_{\mathbb{P}^{(s)}}(\omega)$ and $\mathbb{Q}^{(t)}(\omega) = (1/2)^{|\Lambda^{\text{half}}_{\mathbb{Q}^{(t)}}|} \cdot F_{\mathbb{Q}^{(t)}}(\omega)$. The total variation distance between mixtures of Boolean subcubes $\mathbb{P} = \sum_{s=1}^{k_1} \alpha^{(s)} \bigotimes_{i=1}^n \mathbb{P}_i^{(s)}$ and $\mathbb{Q} = \sum_{t=1}^{k_2} \beta^{(t)} \bigotimes_{i=1}^n \mathbb{Q}_i^{(t)}$ can be expressed as follows:

$$\begin{aligned}
d_{\text{TV}}(\mathbb{P}, \mathbb{Q}) &= \frac{1}{2} \sum_{\omega \in \{0,1\}^n} |\mathbb{P}(\omega) - \mathbb{Q}(\omega)| \\
&= \frac{1}{2} \sum_{\omega \in \{0,1\}^n} \left| \sum_{s=1}^{k_1} \alpha^{(s)} \prod_{i=1}^n \mathbb{P}_i^{(s)}(\omega_i) - \sum_{t=1}^{k_2} \beta^{(t)} \prod_{i=1}^n \mathbb{Q}_i^{(t)}(\omega_i) \right| \\
&= \frac{1}{2} \sum_{\omega \in \{0,1\}^n} \left| \sum_{s=1}^{k_1} \alpha^{(s)} \cdot \left(\frac{1}{2}\right)^{\left|\Lambda^{\text{half}}_{\mathbb{P}^{(s)}}\right|} \cdot F_{\mathbb{P}^{(s)}}(\omega) - \sum_{t=1}^{k_2} \beta^{(t)} \cdot \left(\frac{1}{2}\right)^{\left|\Lambda^{\text{half}}_{\mathbb{Q}^{(t)}}\right|} \cdot F_{\mathbb{Q}^{(t)}}(\omega) \right|.
\end{aligned}$$

Define the *characteristic function* $F : \{0,1\}^n \to \{0,1\}^{k_1+k_2}$ as follows:

$$F(\omega) = \left(F_{\mathbb{P}^{(1)}}(\omega), F_{\mathbb{P}^{(2)}}(\omega), \ldots, F_{\mathbb{P}^{(k_1)}}(\omega), F_{\mathbb{Q}^{(1)}}(\omega), F_{\mathbb{Q}^{(2)}}(\omega), \ldots, F_{\mathbb{Q}^{(k_2)}}(\omega)\right).$$

By the above calculation, the value of $|\mathbb{P}(\omega) - \mathbb{Q}(\omega)|$ is fully determined by $F(\omega)$. The key observation is that the image space of F has constant size $2^{k_1+k_2}$, so we can reorganize all $\omega \in \{0,1\}^n$ according to the value of the characteristic function $F(\omega)$. Formally,

$$d_{\text{TV}}(\mathbb{P}, \mathbb{Q}) = \frac{1}{2} \sum_{\chi \in \{0,1\}^{k_1+k_2}} N_\chi \cdot \left| \sum_{s=1}^{k_1} \alpha^{(s)} \cdot \left(\frac{1}{2}\right)^{\left|\Lambda^{\text{half}}_{\mathbb{P}^{(s)}}\right|} \cdot \chi_s - \sum_{t=1}^{k_2} \beta^{(t)} \cdot \left(\frac{1}{2}\right)^{\left|\Lambda^{\text{half}}_{\mathbb{Q}^{(t)}}\right|} \cdot \chi_{t+k_1} \right| \tag{10}$$

where N_χ counts the number of $\omega \in \{0,1\}^n$ such that $F(\omega) = \chi$, i.e.,

$$N_\chi = |\{\omega \in \{0,1\}^n \mid F(\omega) = \chi\}|.$$

By (10), computing the total variation distance between two mixtures of Boolean subcubes $\mathbb{P}$ and $\mathbb{Q}$ reduces to counting N_χ for all $\chi \in \{0,1\}^{k_1+k_2}$.

5.2 The counting algorithm

▶ **Lemma 18.** *Given any $\chi \in \{0,1\}^{k_1+k_2}$, the value of N_χ can be computed in time $O(2^{|S_0|} \cdot n(k_1+k_2))$, where $S_0 = \{\, j \in [k_1+k_2] \mid \chi_j = 0 \,\}$ is the set of indices j with $\chi_j = 0$.*

Fix $\chi \in \{0,1\}^{k_1+k_2}$, and consider the set of $\omega \in \{0,1\}^n$ such that $F(\omega) = \chi$. We need to count the number of such $\omega \in \{0,1\}^n$. To simplify the notation, define a sequence of Boolean functions $F_1, F_2, \ldots, F_{k_1+k_2} : \{0,1\}^n \to \{0,1\}$ as follows:

$$\forall j \in [k_1+k_2], \quad F_j(\omega) = \begin{cases} F_{\mathbb{P}^{(j)}}(\omega) & \text{if } j \in [k_1], \\ F_{\mathbb{Q}^{(j-k_1)}}(\omega) & \text{if } j \in [k_1+1, k_1+k_2], \end{cases}$$

where $F_{\mathbb{P}^{(s)}}$ and $F_{\mathbb{Q}^{(t)}}$ are defined in (8) and (9), respectively. For a configuration $\omega \in \{0,1\}^n$ with $F(\omega) = \chi$, we have $F_j(\omega) = \chi_j$ for all $j \in [k_1+k_2]$.

For a subset $S \subseteq [k_1+k_2]$ of indices, define the following set of configurations:

$$\Phi(S) = \{\omega \in \{0,1\}^n \mid F_j(\omega) = 1 \text{ for all } j \in S\}.$$

Let $S_1 = \{j \in [k_1+k_2] \mid \chi_j = 1\}$ and $S_0 = \{j \in [k_1+k_2] \mid \chi_j = 0\}$. By the definition of N_χ and the inclusion-exclusion principle, we have

$$N_\chi = |\Phi(S_1)| - |\{\omega \in \Phi(S_1) \mid \exists\, j \in S_0, F_j(\omega) = 1\}| = \sum_{S \subseteq S_0} (-1)^{|S|} \cdot |\Phi(S_1 \cup S)|. \tag{11}$$

▶ **Observation 19.** *Given any $S \subseteq [k_1+k_2]$, the size $|\Phi(S)|$ can be computed in time $O(n \cdot |S|)$.*

Proof. By the definition of $F_{\mathbb{P}^{(s)}}$ and $F_{\mathbb{Q}^{(t)}}$, we have $F_{\mathbb{P}^{(s)}}(\omega) = 1$ iff $\omega_i = 1$ for all $i \in \Lambda^{\text{one}}_{\mathbb{P}^{(s)}}$ and $\omega_i = 0$ for all $i \in \Lambda^{\text{zero}}_{\mathbb{P}^{(s)}}$. Similarly, $F_{\mathbb{Q}^{(t)}}(\omega) = 1$ iff $\omega_i = 1$ for all $i \in \Lambda^{\text{one}}_{\mathbb{Q}^{(t)}}$ and $\omega_i = 0$ for all $i \in \Lambda^{\text{zero}}_{\mathbb{Q}^{(t)}}$. We can go through all $j \in S$ and fix the values of the corresponding dimensions $i \in [n]$. If there is a contradiction, then $|\Phi(S)| = 0$. Otherwise, $|\Phi(S)| = 2^r$, where r is the number of dimensions whose values are not fixed. The running time is $O(n \cdot |S|)$. ◀

We now prove Lemma 18. Combining (11) and Observation 19, we can exactly compute the value of N_χ in time

$$\sum_{S \subseteq S_0} O(n|S_1 \cup S|) \le \sum_{0 \le k \le |S_0|} \binom{|S_0|}{k} \cdot O(n(k_1+k_2)) = O(2^{|S_0|} \cdot n(k_1+k_2)).$$

5.3 The total variation distance algorithm

The algorithm for computing the total variation distance between two mixtures of Boolean subcubes $\mathbb{P}$ and $\mathbb{Q}$ is obtained by combining (10) and Lemma 18. Let $K = k_1 + k_2$. The overall running time is

$$\sum_{0 \le m \le K} \binom{K}{m} \cdot O(2^m \cdot nK) = O(3^K K \cdot n) = O(3^{k_1+k_2}(k_1+k_2) \cdot n).$$

Here m enumerates the number of indices j with $\chi_j = 0$. This proves the main result in Theorem 4.

6 Proof of the hardness result

Proof of Theorem 5. We show that if there is a polynomial-time algorithm for computing the TV-distance between two mixtures of Boolean subcubes with $k_1 + k_2 = \Theta(n)$, then there is a polynomial-time algorithm for #3SAT, restricted to 3-CNF formulas. Let

$$\varphi(x_1, x_2, \ldots, x_r) = C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

be a 3-CNF formula over the used variables $x_1, x_2, \ldots, x_r$, and $m \geq 1$. Set $N = \max\{r, m\}$. If $N > r$, we add $N - r$ dummy variables $x_{r+1}, \ldots, x_N$. Let S denote the number of satisfying assignments of φ over $x_1, \ldots, x_N$. As the dummy variables do not appear in any clause, the number of satisfying assignments for φ is given by

$$\#\mathsf{SAT}(\varphi) = \frac{S}{2^{N-r}}.$$

We establish the proof by showing the relation between S and the TV-distance between certain mixtures of Boolean subcubes. Let $n = N + 1$. We construct two mixtures of subcubes $\mathbb{P}$ and $\mathbb{Q}$ over $\{0,1\}^n$. A point in this space can be represented as $(b, \omega_1, \ldots, \omega_N)$, where $b \in \{0,1\}$ is an additional bit, and $(\omega_1, \ldots, \omega_N)$ is an assignment to $(x_1, \ldots, x_N)$. For each $j \in [m]$, let $x_{j,a}, x_{j,b}, x_{j,c}$ be the three variables appearing in the clause C_j. Define $(\omega_{j,a}, \omega_{j,b}, \omega_{j,c}) \in \{0,1\}^3$ to be the unique assignment to $x_{j,a}, x_{j,b}, x_{j,c}$ that falsifies C_j. Let U_j be a subcube such that

$$\mathbf{Pr}_{U_j}\left[(b, x_{j,a}, x_{j,b}, x_{j,c}) = (0, \omega_{j,a}, \omega_{j,b}, \omega_{j,c})\right] = 1,$$

and the other $N - 3$ coordinates are uniform over $\{0,1\}$. Define

$$\mathbb{P} = \frac{1}{m}\sum_{j=1}^{m} U_j,$$

is a mixture of $k_1 = m$ subcubes. Next we define $\mathbb{Q}$. Let V_0 and V_1 be two subcubes such that

$$\mathbf{Pr}_{V_0}[b = 0] = \mathbf{Pr}_{V_1}[b = 1] = 1,$$

and that are uniform over $(x_1, \ldots, x_N)$. Let $\lambda = \frac{1}{2m}$, and define

$$\mathbb{Q} = \lambda V_0 + (1 - \lambda) V_1,$$

which is a mixture of $k_2 = 2$ subcubes.

We now compute $d_{\mathrm{TV}}(\mathbb{P}, \mathbb{Q})$. For any assignment $\omega = (\omega_1, \ldots, \omega_N)$, let

$$F(\omega) = |\{j \in [m] \mid \omega \text{ falsifies } C_j\}|$$

denote the number of clauses falsified by ω. For every $\omega \in \{0,1\}^N$, we have

$$\mathbb{P}(0,\omega) = \frac{1}{m}\sum_{j=1}^{m} U_j(0,\omega) = \frac{1}{m}\sum_{j=1}^{m} \mathbf{1}[\omega \text{ falsifies } C_j]\cdot 2^{-(N-3)} = \frac{F(\omega)}{m}\cdot 2^{-(N-3)} = \frac{8F(\omega)}{m}\cdot 2^{-N},$$

and $\mathbb{P}(1,\omega) = 0$ by definition of U_j. Remark that

$$\mathbb{Q}(0,\omega) = \lambda \cdot 2^{-N} = \frac{1}{2m}\cdot 2^{-N} < \frac{8F(\omega)}{m}\cdot 2^{-N} = \mathbb{P}(0,\omega) \quad \text{iff } F(\omega) \geq 1;$$

$$\mathbb{Q}(1,\omega) = (1-\lambda)\cdot 2^{-N} > 0 = \mathbb{P}(1,\omega).$$

Therefore

$$\begin{aligned} d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q}) &= \sum_{(b,\omega)\in\{0,1\}^{N+1}} \max\{0, \mathbb{P}(b,\omega) - \mathbb{Q}(b,\omega)\} = \sum_{\omega\in\{0,1\}^N : F(\omega)\geq 1} \mathbb{P}(0,\omega) - \mathbb{Q}(0,\omega) \\ &= \sum_{\omega\in\{0,1\}^N : F(\omega)\geq 1} \frac{8F(\omega)}{m}\cdot 2^{-N} - \sum_{\omega\in\{0,1\}^N : F(\omega)\geq 1} \frac{1}{2m}\cdot 2^{-N}. \end{aligned} \tag{12}$$

For the first term, we consider counting all pairs (ω, C_j) such that ω falsifies C_j. On the one hand, for each ω, it falsifies $F(\omega)$ clauses; on the other hand, for each C_j, it is falsified by 2^{N-3} assignments. Thus

$$\sum_{\omega\in\{0,1\}^N : F(\omega)\geq 1} F(\omega) = \sum_{\omega\in\{0,1\}^N} F(\omega) = m\cdot 2^{N-3}.$$

For the second term, there are S assignments ω satisfying all clauses, that is, $F(\omega) = 0$. So there are $2^N - S$ assignments satisfying $F(\omega) \geq 1$. Substituting back to (12), we have

$$\begin{aligned} d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q}) &= \sum_{\omega\in\{0,1\}^N : F(\omega)\geq 1} \frac{8F(\omega)}{m}\cdot 2^{-N} - \sum_{\omega\in\{0,1\}^N : F(\omega)\geq 1} \frac{1}{2m}\cdot 2^{-N} \\ &= \frac{2^{-N+3}}{m}\cdot(m\cdot 2^{N-3}) - \frac{2^{-N}}{2m}\cdot(2^N - S) = 1 - \frac{1}{2m} + \frac{2^{-N}}{2m}S. \end{aligned}$$

Thus the number of satisfying assignments is given by

$$\#\mathsf{SAT}(\varphi) = \frac{S}{2^{N-r}} = 2^r(2m\cdot d_{\mathrm{TV}}(\mathbb{P},\mathbb{Q}) - 2m + 1).$$

The constructed distributions live on the $n = N+1$ dimensional subcube, with $k_1 + k_2 = m + 2$ total components. Since $N = \max\{r, m\}$ and $r \leq 3m$, we have $m \leq N \leq 3m$, hence

$$k_1 + k_2 = m + 2 = \Theta(N+1) = \Theta(n).$$

Therefore, an algorithm for Problem 3 with $k_1 + k_2 = \Theta(n)$ yields an exact algorithm for #3SAT. ◀

References

1 Hassan Ashtiani, Shai Ben-David, Nicholas J. A. Harvey, Christopher Liaw, Abbas Mehrabian, and Yaniv Plan. Nearly tight sample complexity bounds for learning mixtures of Gaussians via sample compression schemes. In *NeurIPS*, pages 3416–3425, 2018. URL: `https://proceedings.neurips.cc/paper/2018/hash/70ece1e1e0931919438fcfc6bd5f199c-Abstract.html`.

2 Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, Dimitrios Myrisiotis, A. Pavan, and N. V. Vinodchandran. On approximating total variation distance. In *IJCAI*, pages 3479–3487. ijcai.org, 2023. `doi:10.24963/IJCAI.2023/387`.

3 Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, Dimitrios Myrisiotis, A. Pavan, and N. V. Vinodchandran. Total variation distance meets probabilistic inference. In *ICML*. OpenReview.net, 2024.

4 Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, Dimitrios Myrisiotis, Aduri Pavan, and N. V. Vinodchandran. Computational explorations of total variation distance. In *ICLR*. OpenReview.net, 2025. URL: `https://openreview.net/forum?id=xak8c9l1nu`.

5 Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, and N. V. Vinodchandran. Efficient distance approximation for structured high-dimensional distributions via learning. In *NeurIPS*, 2020.

6 Guy Blanc, Jane Lange, Ali Malik, and Li-Yang Tan. Lifting uniform learners via distributional decomposition. In *STOC*, pages 1755–1767. ACM, 2023. doi:10.1145/3564246.3585212.

7 Guy Blanc, Jane Lange, Carmen Strassle, and Li-Yang Tan. A distributional-lifting theorem for PAC learning. In Nika Haghtalab and Ankur Moitra, editors, *COLT*, Proceedings of Machine Learning Research, pages 375–379. PMLR, 2025. URL: https://proceedings.mlr.press/v291/blanc25a.html.

8 Clément L. Canonne and Ronitt Rubinfeld. Testing probability distributions underlying aggregated data. In *ICALP*, volume 8572 of *Lecture Notes in Computer Science*, pages 283–295. Springer, 2014. doi:10.1007/978-3-662-43948-7_24.

9 Sitan Chen and Ankur Moitra. Beyond the low-degree algorithm: mixtures of subcubes and their applications. In *STOC*, pages 869–880. ACM, 2019. doi:10.1145/3313276.3316375.

10 Taolue Chen and Stefan Kiefer. On the total variation distance of labelled markov chains. In *LICS*, pages 33:1–33:10. ACM, 2014. doi:10.1145/2603088.2603099.

11 Jon Feldman, Ryan O'Donnell, and Rocco A. Servedio. Learning mixtures of product distributions over discrete domains. *SIAM*, 37(5):1536–1564, 2008. doi:10.1137/060670705.

12 Weiming Feng, Heng Guo, Mark Jerrum, and Jiaheng Wang. A simple polynomial-time approximation algorithm for the total variation distance between two product distributions. *TheoretiCS*, 2, 2023. doi:10.46298/THEORETICS.23.7.

13 Weiming Feng, Hongyang Liu, and Minji Yang. Approximating the total variation distance between spin systems. In *COLT*, volume 291 of *Proceedings of Machine Learning Research*, pages 1974–2025. PMLR, 2025. URL: https://proceedings.mlr.press/v291/feng25a.html.

14 Weiming Feng, Liqiang Liu, and Tianren Liu. On deterministically approximating total variation distance. In *SODA*, pages 1766–1791. SIAM, 2024. doi:10.1137/1.9781611977912.70.

15 Spencer Gordon, Bijan H. Mazaheri, Yuval Rabani, and Leonard J. Schulman. Source identification for mixtures of product distributions. In *COLT*, Proceedings of Machine Learning Research, pages 2193–2216. PMLR, 2021. URL: http://proceedings.mlr.press/v134/gordon21a.html.

16 Spencer L. Gordon, Erik Jahn, Bijan Mazaheri, Yuval Rabani, and Leonard J. Schulman. Identification of mixtures of discrete product distributions in near-optimal sample and time complexity. In *COLT*, Proceedings of Machine Learning Research, pages 2071–2091. PMLR, 2024. URL: https://proceedings.mlr.press/v247/gordon24a.html.

17 Prateek Jain and Sewoong Oh. Learning mixtures of discrete product distributions using spectral decompositions. In *COLT*, JMLR Workshop and Conference Proceedings, pages 824–856. JMLR.org, 2014. URL: http://proceedings.mlr.press/v35/jain14.html.

18 Chinmaya Kausik, Kevin Tan, and Ambuj Tewari. Learning mixtures of Markov Chains and MDPs. In *ICML*, Proceedings of Machine Learning Research, pages 15970–16017. PMLR, 2023. URL: https://proceedings.mlr.press/v202/kausik23a.html.

19 Stefan Kiefer. On computing the total variation distance of hidden Markov models. In *ICALP*, volume 107 of *LIPIcs*, pages 130:1–130:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.130.

20 Jeongyeol Kwon, Yonathan Efroni, Constantine Caramanis, and Shie Mannor. Reward-Mixing MDPs with few latent contexts are learnable. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 18057–18082. PMLR, 2023. URL: https://proceedings.mlr.press/v202/kwon23b.html.

21 Jerry Li and Ludwig Schmidt. Robust and proper learning for mixtures of Gaussians via systems of polynomial inequalities. In *COLT*, Proceedings of Machine Learning Research, pages 1302–1382. PMLR, 2017. URL: http://proceedings.mlr.press/v65/li17a.html.

22 Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003. doi:10.1145/636865.636868.