Sampling from a k-DPP without looking at all items

Daniele Calandriello*
DeepMind Paris
dcalandriello@google.com

Michał Dereziński*
University of California, Berkeley
mderezin@berkeley.edu

Michal Valko
DeepMind Paris
valkom@deepmind.com

Abstract

Determinantal point processes (DPPs) are a useful probabilistic model for selecting a small diverse subset out of a large collection of items, with applications in summarization, stochastic optimization, active learning and more. Given a kernel function and a subset size k, our goal is to sample k out of n items with probability proportional to the determinant of the kernel matrix induced by the subset (a.k.a. k-DPP). Existing k-DPP sampling algorithms require an expensive preprocessing step which involves multiple passes over all n items, making it infeasible for large datasets. A naïve heuristic addressing this problem is to uniformly subsample a fraction of the data and perform k-DPP sampling only on those items, however this method offers no guarantee that the produced sample will even approximately resemble the target distribution over the original dataset. In this paper, we develop α-DPP, an algorithm which adaptively builds a sufficiently large uniform sample of data that is then used to efficiently generate a smaller set of k items, while ensuring that this set is drawn exactly from the target distribution defined on all n items. We show empirically that our algorithm produces a k-DPP sample after observing only a small fraction of all elements, leading to several orders of magnitude faster performance compared to the state-of-the-art. Our implementation of α-DPP is provided at https://github.com/guilgautier/DPPy/.

1 Introduction

Selecting k diverse items out of a larger collection of n items is a classical problem in computer science which naturally emerges in many tasks such as summarization (select k phrases) and recommendation (select k articles/ads to show to the user). An increasingly popular approach to model and quantify diversity in this subset selection problem is that of determinantal point processes (DPPs). Given a set $[n] \stackrel{def}{=} \{1, \dots, n\}$ of n items and a target size k, one can define a DPP of size k (known as a k-DPP) through an $n \times n$ posivite semi-definite (PSD) similarity matrix $\mathbf{L}$ (also known as the kernel matrix). The matrix $\mathbf{L}$ encodes the similarities between items, and the user must choose it so that $[\mathbf{L}]_{ij}$ is larger the more the i-th and j-th items are similar. Given k and $\mathbf{L}$, we define $S \sim k\text{-DPP}(\mathbf{L})$ as a distribution over all $\binom{n}{k}$ index subsets $S \subseteq [n]$ of size k, such that $\Pr(S) \propto \det(\mathbf{L}_S)$ is proportional to the determinant of the sub-matrix $\mathbf{L}_S$ induced by the subset. DPPs have found numerous applications in machine learning, not only for summarization [32, 23, 21, 7] and recommendation [19, 8], but also in experimental design [17, 34], stochastic optimization [39, 35, 15], Gaussian Process optimization [26], low-rank approximation [18, 24, 16], and more (recent surveys include [29, 4, 11]). Note that early work on DPPs focused on a *random-size* variant, which we denote $S \sim \text{DPP}(\mathbf{L})$, where the

*Equal contribution.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

subset size is allowed to take any value between 0 and n, and the role of parameter k is replaced by the expected size $\mathbb{E}[|S|] = d_{\text{eff}}(\mathbf{L}) \stackrel{def}{=} \operatorname{tr}\left(\mathbf{L}(\mathbf{L}+\mathbf{I})^{-1}\right)$. The quantity $d_{\text{eff}}(\mathbf{L})$ is known in randomized linear algebra [2, 11] and learning theory [6] as the effective dimension. While random-size DPPs exhibit deep connections to many scientific domains [25], the *fixed-size* k-DPPs are typically more practical from a machine learning stand-point [28].

Sampling from a k-DPP. The first k-DPP samplers scaled poorly, as they all relied on an eigendecomposition [28] of $\mathbf{L}$ taking $\mathcal{O}(n^3)$ time. Replacing the eigendecomposition with a Cholesky factorization can increase numerical stability [30], and empirical performance [36] thanks to dynamically-scheduled, shared-memory parallelizations, but still ultimately require $\mathcal{O}(n^3)$ time. A number of methods have been proposed which use approximate eigendecomposition [31, 1] to reduce the computational cost, however these approaches provide limited guarantees on the accuracy of sampling.

To improve scalability, several approaches based on Monte-Carlo sampling were introduced, using rejection or Gibbs sampling. The fastest MCMC sampler for k-DPPs, to the best of our knowledge, is by [3] and has $n \cdot \operatorname{poly}(k)$ complexity, i.e., asymptotically much faster than the cost of eigendecomposition. However these MCMC methods do *not* sample exactly from the k-DPP distribution, and can only guarantee that the final sample will be close in distribution to a k-DPP. Moreover these guarantees only hold *after mixing*, which is difficult to verify and requires at least $\mathcal{O}(nk^2)$ time, making MCMC methods not applicable when n is large.

	Complexity
[28, 30, 36, 24]	n^3
DPP-VFX [13]	$n \cdot k^{10} + k^{15}$
α-DPP (this paper)	$(\beta n \cdot k^6 + k^9)\sqrt{k}$

Table 1: Runtime comparison of exact k-DPP sampling algorithms. Here, $\beta \leq 1$ is the fraction of items observed by α-DPP (see Theorem 1).

A recent line of works [13, 10], using the ideas from [12, 14], developed sampling algorithms specially designed for a *random-size* DPP (as opposed to a k-DPP), which avoid expensive decomposition of the kernel while sampling exactly from $S \sim \operatorname{DPP}(\mathbf{L})$. In particular, they showed that it is sufficient to first choose an intermediate subset $\sigma \subseteq [n]$ sampled i.i.d. from the *marginal* distribution of the DPP, i.e., $\mathbf{P}(i \in \sigma) \approx \mathbf{P}(i \in S)$, and then sample from a DPP restricted to the items indexed by σ. Since the size of σ is typically much less than n, this leads to a more efficient algorithm. Note that rescaling $\operatorname{DPP}(\mathbf{L})$ into $\operatorname{DPP}(\alpha\mathbf{L})$ using some constant α only changes the expected size of S from $d_{\text{eff}}(\mathbf{L})$ to $d_{\text{eff}}(\alpha\mathbf{L})$. By accurately choosing an appropriate $\alpha_\star$, one can boost the probability that the random size of S is exactly k, and convert a DPP sampler into a k-DPP sampler by repeatedly sampling $S \sim \operatorname{DPP}(\alpha_\star\mathbf{L})$ until S has size k. Based on this reduction, Dereziński et al. [13] gave the first algorithm (DPP-VFX) which is capable of exact sampling from a k-DPP in time $n \cdot \operatorname{poly}(k)$. However, when sampling from k-DPPs, the approach of [13] has *two major limitations*:

1. DPP-VFX has an $\Omega(n)$ runtime bottleneck, since it requires computing all n marginals, one for each item, in order to define the i.i.d. distribution of σ, which may be infeasible for very large n.

2. The reduction used by [13] to convert a DPP sampler into a k-DPP sampler increases the time complexity by a factor of at least k^4, resulting in a $\widetilde{\mathcal{O}}(n \cdot k^{10} + k^{15})$ runtime.

In this paper, we address both of these limitations by introducing a new algorithm called α-DPP, which 1) does not need to compute all of the marginals, and 2) uses a new efficient reduction to convert from a random-size DPP to a fixed-size k-DPP (see Table 1 for comparison).

Main contribution: uniform intermediate sampling for k-DPPs. To resolve the $\Omega(n)$ runtime bottleneck, we use an additional intermediate sample ρ based on *uniform* sub-sampling. Since uniform sampling can be implemented without looking at the actual items in the collection, this means that we do not even have to look at any item outside of ρ. The only necessary assumption required by our approach is that the maximum entry (i.e., similarity) of $\mathbf{L}$ is bounded by a constant κ^2. However, to simplify exposition we also assume w.l.o.g. that $d_{\text{eff}}(\mathbf{L}) \geq k$ (see Section 3).

In particular, we 1) sample ρ uniformly out of $[n]$, then 2) only approximate the marginal probabilities of items in ρ to compute σ, and finally 3) downsample σ into a DPP sample S. To guarantee that S is distributed exactly according to the DPP it is crucial that ρ is diverse enough. We show that sampling a $k^2/d_{\text{eff}}(\mathbf{L})$ fraction of $[n]$ into ρ (i.e., $|\rho| \approx k^2/d_{\text{eff}}(\mathbf{L}) \cdot n$) is enough. Since all the expensive computation is performed only on ρ, this gives us a $d_{\text{eff}}(\mathbf{L})/k^2$ speedup over existing methods.

Theorem 1. *Given any* $\mathbf{L} \succeq \mathbf{0}$ *with* $\max_{ij} \mathbf{L}_{ij} \leq \kappa^2$ *and* $1 \leq k \leq d_{\text{eff}}(\mathbf{L})$*, there exists an algorithm that returns* $S \sim k\text{-DPP}(\mathbf{L})$*, and with probability* $1-\delta$ *runs in time*

$$\widetilde{\mathcal{O}}\big((\beta n \cdot k^6 + k^9)\sqrt{k}\log(1/\delta)\big),$$

where $\beta \leq \min\big\{k^2\kappa^2/d_{\text{eff}}(\mathbf{L}), 1\big\}$ *is the fraction of items observed by the algorithm.*

In the derivation of Theorem 1 we make several novel contributions. First, we provide a DPP sampler that given $\mathbf{L}$ and a rescaling $\alpha \leq 1$ leverages a mixture of uniform and rejection sampling to sample from $\text{DPP}(\alpha\mathbf{L})$ observing only an $\alpha\kappa^2 k$ fraction of the items. We then show that the optimal rescaling $\alpha_\star$ required by the reduction from k-DPP to DPP can be bounded with $\alpha_\star \leq \mathcal{O}(k/d_{\text{eff}}(\mathbf{L}))$, and thus our rescaling-aware sampler can sample from k-DPPs looking only at a $k^2/d_{\text{eff}}(\mathbf{L})$ fraction of the items. Finally, we provide an efficient search algorithm to find a close approximation $\hat{\alpha}$ of $\alpha_\star$.

Model misspecification and computational free lunch. Our result can be also interpreted from a perspective of model misspecification. Note that every time the users define a k-DPP they also implicitly define a random size $\text{DPP}(\mathbf{L})$. Moreover, the natural expected sample size (i.e., implicit number of unique items in $[n]$) of $\text{DPP}(\mathbf{L})$ is $d_{\text{eff}}(\mathbf{L})$, which does not depend on the desired size k. Therefore, if $\mathbf{L}$ is not chosen appropriately $d_{\text{eff}}(\mathbf{L})$ might be much larger than k, and the k-DPP is selecting k unique items out of a much larger implicit pool of $d_{\text{eff}}(\mathbf{L}) \gg k$ unique items. In this case, it is possible to consider only a small $k^2/d_{\text{eff}}(\mathbf{L})$ fraction of the items selected uniformly at random and still have enough unique items to sample a diverse k-subset. Our result shows for the first time that it is possible to take advantage of this modeling disagreement between k and $d_{\text{eff}}(\mathbf{L})$ to gain computational savings while still sampling *exactly* from the DPP, i.e., a computational free lunch.

Binary search reduction from k-DPP to DPP. Both our approach and the one of Dereziński et al. [13] rely on first implementing an efficient random-size DPP sampler, followed by the usage of a black-box construction based on rejection sampling to transform the DPP sampler into a k-DPP sampler. However the reduction of Dereziński et al. [13] requires access to a high-precision estimate of $d_{\text{eff}}(\alpha\mathbf{L})$ in order to appropriately tune α. This makes optimizing α the bottleneck in the reduction from k-DPP to DPP, and therefore there is a large computational gap between the two problems. We close this gap thanks to a novel approach to find a suitable rescaling α based not on optimization but rather on binary search. Crucially, to find a suitable α this approach does not require an estimate of $d_{\text{eff}}(\alpha\mathbf{L})$, but only $\mathcal{O}(\sqrt{k}\log(n))$ black-box calls to a DPP sampler. Therefore, it can transform any random size DPP sampler into a k-DPP sampler with only a $\sqrt{k}$ overhead, and could be applied to any future improved sampler beyond this paper.

2 Sampling from a rescaled DPP with intermediate uniform subsampling

In this section we focus on a specific class of DPPs, $S \sim \text{DPP}(\alpha\mathbf{L})$, specified using a rescaling $\alpha \leq 1$ and a similarity matrix $\mathbf{L}$, which we refer to as rescaled DPPs. The main result of the section is showing that a sufficiently large subset selected uniformly at random can be used as an intermediate sample to sample from a rescaled DPP without looking at all of the items. The main reason to focus on rescaled DPPs is because they naturally appear when reducing k-DPP sampling to DPP sampling, where rescaling is used to align the random size of the DPP and k. This is going to be the focus of the next section. However the approach proposed in this section is not limited to rescaled DPPs, but under the right assumptions can be extended to accelerate sampling from generic DPPs. We will discuss these extensions at the end of the section.

Notation We use $[n]$ to denote the set $\{1, \dots, n\}$. For a matrix $\mathbf{B} \in \mathbb{R}^{n\times m}$ and index sets C, D, we use $\mathbf{B}_{C,D}$ to denote the submatrix of $\mathbf{B}$ consisting of the intersection of rows indexed by C with columns indexed by D. If $C = D$, we use a shorthand $\mathbf{B}_C$ and if $D = [m]$, we may write $\mathbf{B}_{C,[m]}$. Finally, we also allow C, D to be multisets or sequences, in which case each row/column is duplicated in the matrix according to its multiplicity (and in the case of sequences, we order the rows/columns as they appear in the sequence). Note that with this notation if $\mathbf{L} = \mathbf{B}\mathbf{B}^\top$ then $\mathbf{L}_{C,D} = \mathbf{B}_{C,[n]}\mathbf{B}_{D,[n]}^\top$.

2.1 Background: distortion-free intermediate sampling.

Rather than sampling directly from the target DPP, intermediate sampling [10, 12] first selects an intermediate subset σ from $[n]$, and then refines it by extracting S from σ. Crucially, if σ is selected according to a so-called Regularized DPP (R-DPP), this is equivalent to sampling S from a DPP.

Definition 1. *For any psd matrix* $\mathbf{L} \in \mathbb{R}^{n\times n}$*, distribution* $p \overset{\text{def}}{=} \{p_i\}_{i=1}^n$ *and* $r > 0$*, define* $\widetilde{\mathbf{L}} \in \mathbb{R}^{n\times n}$ *with* $\widetilde{\mathbf{L}}_{i,j} \overset{\text{def}}{=} \frac{\mathbf{L}_{i,j}}{r\sqrt{p_i p_j}}$*. We define an* $\text{R-DPP}_p^r(\mathbf{L})$ *as distribution over events* $A \subseteq \bigcup_{k=0}^{\infty}[n]^k$ *such that*

$$\Pr(A) \overset{\text{def}}{=} \mathbb{E}_\sigma\big[\mathbf{1}_{[\sigma\in A]}\det(\mathbf{I}+\widetilde{\mathbf{L}}_\sigma)\big]/\det(\mathbf{I}+\mathbf{L}), \quad \textit{for} \quad \sigma = (\sigma_1,\dots,\sigma_t) \overset{\text{i.i.d.}}{\sim} p, \quad t \sim \text{Poisson}(r).$$

Proposition 1 (10, Theorem 8)**.** *For any* $\mathbf{L}$*,* p*,* r*, and* $\widetilde{\mathbf{L}}$ *defined as in Definition 1,*

$$\textit{if} \quad \sigma \sim \text{R-DPP}_p^r(\mathbf{L}) \quad \textit{and} \quad S \sim \text{DPP}(\widetilde{\mathbf{L}}_\sigma) \quad \textit{then} \quad \{\sigma_i : i \in S\} \sim \text{DPP}(\mathbf{L}).$$

A computationally inefficient but conceptually simple approach to rejection sample σ is the following: 1) compute all marginals $\mathbf{P}(i \in S) = \ell_i(\mathbf{L}) \overset{\text{def}}{=} [\mathbf{L}(\mathbf{I}+\mathbf{L})^{-1}]_i$ and sum to $\sum_{i=1}^n \ell_i(\mathbf{L}) = d_{\text{eff}}(\mathbf{L})$ [2]; 2) sample $t \sim \text{Poisson}(c)$ and $\sigma \sim \text{Multinomial}\left(t, \frac{\ell_1(\mathbf{L})}{d_{\text{eff}}(\mathbf{L})}, \dots, \frac{\ell_n(\mathbf{L})}{d_{\text{eff}}(\mathbf{L})}\right)$ for an appropriate constant c; 3) accept σ w.p. $\frac{\det(\mathbf{I}+\widetilde{\mathbf{L}}_\sigma)}{C\det(\mathbf{I}+\mathbf{L})}$, where C is an appropriate constant used to make the rejection step valid.

All existing intermediate sampling algorithms [10, 13, 12, 14] rely on this approach, refining it to make use of efficient approximations of the marginals $\ell_i(\mathbf{L})$ and adapting the constants c and C to the data. However they all share a common bottleneck: to sample σ i.i.d. they need to approximate all marginals $\ell_i(\mathbf{L})$ and the normalization constant $d_{\text{eff}}(\mathbf{L})$, and therefore the final runtime scales as $n \cdot \text{poly}(k)$. While this is much smaller than the $\mathcal{O}(n^3)$ required by an exact sampler, it still becomes quickly unfeasible when n is very large. In what follows we will introduce another approach to sample from an R-DPP that does not require to approximate the marginals of all items, but only the items selected in a preliminary *uniform* intermediate sample.

2.2 Faster DPP sampling with uniform intermediate sampling

We now introduce our novel α-rescaled DPP sampler, called α-DPP (see Algorithm 1). It requires as input a rescaling α, a similarity matrix $\mathbf{L}$ and a parameter r that will be used to tune the Poisson sampling step of Proposition 1 approach. It also requires as input a *dictionary* $\mathcal{D}$ containing m elements, and set of weights stored in a diagonal matrix $\mathbf{W} \in \mathbb{R}^{m\times m}$. A dictionary is a subset of items $\mathcal{D} \subseteq [n]$ such that reweighting the items in $\mathcal{D}$ by $\mathbf{W}$ provides a good approximation of $\mathbf{L}$, so that the approximate marginals

$$l_i \overset{\text{def}}{=} \alpha[\mathbf{L} - \alpha\mathbf{L}_{[n],\mathcal{D}}^\top(\alpha\mathbf{L}_\mathcal{D} + \mathbf{W}^{-1})^{-1}\mathbf{L}_{[n],\mathcal{D}}]_i \quad (1)$$

computed using $\mathcal{D}$ and $\mathbf{W}$ are close to the true marginals ℓ_i (see Appendix E). Compared to the meta-approach of Proposition 1, the main technical difference is that rather than sampling directly t from an appropriate Poisson, and then σ from a Multinomial, we introduce an intermediate uniform sampling step. In particular, we first sample a Poisson u, and then uniformly sample a subset ρ containing u items. We then compute an approximation l_i of the marginal ℓ_i only for the items in ρ, and downsample ρ into σ using rejection sampling (Line 7). Finally, we accept or reject σ (Line 11) and then downsample σ into S using a standard DPP sampler on the smaller $\widetilde{\mathbf{L}}_\sigma$.

Algorithm 1 α-DPP sampler

Input: α, $\mathbf{L}$, $\mathcal{D}$, $\mathbf{W}$, $r \geq 1$

1: Set $\widehat{\mathbf{L}} = \mathbf{W}^{1/2}\mathbf{L}_{\mathcal{D},\mathcal{D}}\mathbf{W}^{1/2} \in \mathbb{R}^{m\times m}$
2: **repeat**
3: Sample $u \sim \text{Poisson}(re^{1/r}\alpha n\kappa^2)$
4: Sample $\rho = \text{Uniform}(u, [n])$
5: **for** $j = \{1,\dots,u\}$ **do**
6: Compute l_{ρ_j} using Eq. (1)
7: Sample $z_j \sim \text{Bernoulli}(l_{\rho_j}/(\alpha\kappa^2))$
8: **end for**
9: Set $\sigma = \{\rho_j : z_j = 1\}$, $t = |\sigma|$
10: Set $[\widetilde{\mathbf{L}}_\sigma]_{ij} = \frac{1}{r\sqrt{l_{\sigma_i} l_{\sigma_j}}}[\mathbf{L}]_{\sigma_i\sigma_j}$
11: $Acc \sim \text{Bernoulli}\left(\frac{e^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})}\det(\mathbf{I}+\alpha\widetilde{\mathbf{L}}_\sigma)}{e^{t/r}\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})}\right)$
12: **until** $Acc =$ true
13: Sample $\widetilde{S} \sim \text{DPP}\big(\alpha\widetilde{\mathbf{L}}_\sigma\big)$
14: **return** $S = \{\sigma_i : i \in \widetilde{S}\}$

Algorithm 1 is not simply a different implementation of the approach of Proposition 1, since even if Multinomial sampling is implemented with lazy evaluations of l_i, we would still need to compute the *normalization* constant of the Multinomial, which strictly requires computing all l_i. Similarly, the rejection test of Line 11 is also designed to accept as many candidates as possible without requiring the computation of the normalization constant as in [13]. Rather our approach is a novel method to sample from an R-DPP using Poisson rejection sampling. In particular, we prove not only that it always returns an S sampled according to the exact DPP distribution, but also that if the dictionary satisfies certain conditions, the main of which is (ε, α)-accuracy (see Appendix E and [5]), then the algorithm will generate S quickly.

Theorem 2. *Given any* $\mathbf{L} \succeq \mathbf{0}$*, dictionary* $\mathcal{D}$*,* $\mathbf{W} \succ \mathbf{0}$*,* $r \geq 1$ *and* $\alpha > 0$*,* α-DPP *returns* $S \sim \mathrm{DPP}(\alpha\mathbf{L})$*. Moreover, if* $r \geq d_{\text{eff}}(\alpha\mathbf{L}) \geq 1/2$*,* $\mathcal{D}$ *and* $\mathbf{W}$ *are* $(1/d_{\text{eff}}(\alpha\mathbf{L}), \alpha)$*-accurate,* $\mathcal{D}$ *satisfies* $|\mathcal{D}| \leq 10 d_{\text{eff}}(\alpha\mathbf{L})$*, and* $d_{\text{eff}}(\alpha\widehat{\mathbf{L}}) \leq 10 d_{\text{eff}}(\alpha\mathbf{L})$*, w.p.* $1-\delta$ α-DPP *runs in time*

$$\mathcal{O}\Big(\big[\min\{\alpha\kappa^2 d_{\text{eff}}(\alpha\mathbf{L}), 1\} \cdot n \cdot d_{\text{eff}}(\alpha\mathbf{L})^6 \log^2(n/\delta) + d_{\text{eff}}(\alpha\mathbf{L})^9 \log^3(n/\delta)\big] \cdot \log(1/\delta)\Big).$$

The main implication of our result is that the intermediate distribution based on marginals can be replaced more and more accurately with a uniform distribution as α becomes smaller. This results in having to compute marginals only for a $\min\{\alpha\kappa^2 d_{\text{eff}}(\alpha\mathbf{L}), 1\}$ fraction of the n items. This speedup can be significant when the rescaling α is very small, as is the case when we want to sample a small number of items out of a large collection. Compared to other exact DPP samplers, such as DPP-VFX, our α-DPP is strictly faster by roughly a $1/(\alpha\kappa^2 d_{\text{eff}}(\alpha\mathbf{L}))$ factor when implemented with an appropriate caching strategy for the estimates l_i (see Appendix E). Further, unlike MCMC samplers, α-DPP is an *exact* sampler. Moreover, there is no known MCMC approach that can achieve a runtime sub-linear in n when α is small as α-DPP.

An (ε, α)-accurate dictionary that also satisfies the other conditions can be generated using a slight modification of the BLESS algorithm [37], that we call BLESS-I algorithm, presented in Appendix C. However, note that since the marginals ℓ_i are equivalent to the ridge leverage scores [2] of item i, we can replace BLESS-I with any present or future algorithm for leverage score sampling that can be modified to be rescaling-aware [5, 37]. Moreover, note that BLESS-I also returns an estimate of $d_{\text{eff}}(\alpha\mathbf{L})$ that is sufficiently accurate to tune r and ε. At the same time, our analysis could be excessively conservative, and instead of trying to set r and ε using $d_{\text{eff}}(\alpha\mathbf{L})$ as suggested by Theorem 2, a more practical strategy is to start with a constant r and increase it slowly if the sampler is rejecting with a too low probability, using a doubling schedule to preserve overall time complexity.

Proof sketch. The proof is divided in two parts, proving that α-DPP is an exact sampler (Lemma 6) and that under the right conditions it is efficient (Lemma 7).

For the first part we once again rely on the approach of Proposition 1, but with the added difficulty of not being allowed to compute all the marginals. To avoid this bottleneck, we show that:

A) sampling $t \sim \text{Poisson}(r)$ and $\sigma \sim \text{Multinomial}(t, \{\ell_i/d_{\text{eff}}(\alpha\mathbf{L})\}_{i=1}^n)$; and

B) sampling n independent $s_i \sim \text{Poisson}(r'\ell_i(\alpha\mathbf{L}))$, and adding s_i copies of item i to σ,

are equivalent for an appropriate choice of r and r', i.e., we prove that the σ generated by both approach A and B follow the same distribution. However, unlike approach A, approach B does not require computing a normalization constant, i.e., it samples from *unnormalized* probabilities. Moreover, if we know an upper bound on the marginals we can further reduce the number of marginals that need to be computed. In our case we use the bound $\ell_i \leq \alpha\kappa^2$, and show that

C) sampling n Poisson independently $u_i \sim \text{Poisson}(r'\alpha\kappa^2)$, only if $u_i > 0$ computing ℓ_i and sampling $s_i \sim \text{Binomial}(u_i, \ell_i/(\alpha\kappa^2))$, and adding s_i copies of item i to σ

once again generates σ strictly equivalent to the ones of approach B and A. The added advantage of approach C over the others is that only the marginals of items with $u_i > 0$ are actually computed, and there is no need to compute a normalization constant. Starting from this new approach, to obtain our α-DPP sampler (Algorithm 1) we simply replace the n Poisson $u_i \sim \text{Poisson}(r'\alpha\kappa^2)$ with a single $u \sim \text{Poisson}(r'\alpha\kappa^2 n)$ followed by uniform sampling, and replace the exact ℓ_i with approximate l_i.

For the second part we derive a lower bound on the acceptance probability similar to the one from Dereziński et al. [13]. However, while they use an $n \times d_{\text{eff}}(\alpha\mathbf{L})$ Nyström approximation of the matrix $\mathbf{L}$, to avoid direct dependencies on n we are forced to use a less stable approximation $\widehat{\mathbf{L}}$. As a result, controlling $d_{\text{eff}}(\alpha\widehat{\mathbf{L}})$ requires a more careful analysis.

Beyond uniform subsampling. One of the implications of our analysis is that more adaptive upper bounds on the marginals ℓ_i could further speedup our α-DPP sampling approach. In particular, we chose uniform sampling, i.e., a uniform upper bound, for its conceptual simplicity and because knowing an upper bound κ^2 on the entries of the similarity matrix usually does not require looking at the items, e.g., κ^2 is always equal to 1 for Gaussian similarity, Cosine similarity or other self normalized similarities. However for other similarities, such as linear similarity, this bound could be very loose. A simple replacement is using the actual diagonal of $\mathbf{L}$, which requires to look at all items and $\mathcal{O}(n)$ time to compute but is usually very scalable. Ideally, one could imagine designing a

sequence of upper bounds starting from cheaper to more computationally expensive, where more advanced techniques such as random projection are used near the end to further filter candidate items.

3 Efficient reduction from k-DPP to rescaled DPP via binary search

Given our fast DPP sampler, we can see a k-DPP as a sampling process where we first sample $S \sim \mathrm{DPP}(\alpha\mathbf{L})$, check if the sample size $|S|$ is equal to k, and then accept or reject the sample accordingly. Rescaling $\mathbf{L}$ by a constant factor α only changes the expected size $d_{\text{eff}}(\alpha\mathbf{L})$ (and not the k-DPP), with $\alpha > 1$ increasing the expected size and $\alpha < 1$ decreasing it. Thus, it is natural to imagine that there exists some $\alpha_\star$ for which the acceptance probability is high. Indeed this was recently proven to be possible. Dereziński et al. [13] show that if the *mode* $m_{\alpha_\star}$ of $S_{\alpha_\star} \sim \mathrm{DPP}(\alpha_\star\mathbf{L})$ is equal to k, then we will accept with probability at least $\Omega(1/\sqrt{k})$. They also provide an algorithm to find such an $\alpha_\star$. However, this algorithm has a prohibitively high computational cost, $\widetilde{\mathcal{O}}(nk^{10} + k^{15})$, because ensuring that the mode of $\mathrm{DPP}(\alpha_\star\mathbf{L})$ is *exactly* k requires an extremely accurate approximation of $\mathbf{L}$. Instead, our approach is to run a binary search to find a good rescaling α, which will terminate once the acceptance probability is high enough, regardless of whether k is exactly the mode. Crucially, this binary search only requires a black box $\mathrm{DPP}(\alpha\mathbf{L})$ sampler (such as our α-DPP), and it only queries the sampler $\widetilde{\mathcal{O}}(\sqrt{k})$ many times. To prove that the binary search finds a good α in a small number of steps, we establish a new property (Lemma 3) of the Poisson Binomial distribution (the distribution of the subset sizes of $\mathrm{DPP}(\alpha\mathbf{L})$), which should be of independent interest.

3.1 Binary search

Our main result in this subsection is Algorithm 2, which requires only oracle access to the samples from a random-size DPP, and finds a rescaling $\hat{\alpha}$ which enables efficient rejection sampling from a k-DPP. Note that the provided oracle sampler does not have to be our α-DPP sampler, so the algorithm could be paired with other samplers.

Lemma 1. *Suppose that we are given an integer k, a range $I = [\alpha_{\min}, \alpha_{\max}]$ where $\alpha_{\max} = \gamma\alpha_{\min}$, and access to an oracle which, for any $\alpha \in I$, returns $S \sim \mathrm{DPP}(\alpha\mathbf{L})$. If there exists $\alpha_\star \in I$ such that k is the mode of $|S|$ for $S \sim \mathrm{DPP}(\alpha_\star\mathbf{L})$, then using $O\big(\sqrt{k}\log^2(k\log(\gamma)/\delta)\big)$ calls to the oracle we can find $\hat{\alpha} \in I$ such that with probability $1-\delta$ we have*

$$\Pr(|S| = k) = \Omega\big(\tfrac{1}{\sqrt{k}}\big), \qquad \textit{for} \quad S \sim \mathrm{DPP}(\hat{\alpha}\mathbf{L}).$$

The distribution of subset size $|S|$ for $S \sim \mathrm{DPP}(\mathbf{L})$ can be defined via the eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots$ of $\mathbf{L}$ (see [25]): if we let $b_i \sim \mathrm{Bernoulli}(\frac{\lambda_i}{\lambda_i+1})$ for $i \geq 1$, then $\sum_i b_i$ is distributed identically to $|S|$. This distribution is known as the *Poisson Binomial*, and it has been extensively studied in the probability literature [38]. The recent result of [13] on the probability of the mode of a Poisson Binomial shows that it is possible to find $\hat{\alpha}$ satisfying the condition of Lemma 1.

Lemma 2. *There is an absolute constant $0 < c < 1$ such that for any Poisson Binomial distribution $p : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$, with mode k^* we have $p(k^*) \geq \frac{c}{\sqrt{k^*+1}}$.*

This result, however, does not provide an efficient way of finding an $\hat{\alpha}$ such that the mode of the subset size distribution of $\mathrm{DPP}(\hat{\alpha}\mathbf{L})$ is k. We circumvent this problem by performing a binary search (Algorithm 2) that looks for such an $\hat{\alpha}$, but stops early when it reaches a sufficiently good candidate, avoiding excess computations. To make this rigorous, we establish the following new property of the Poisson Binomial distribution, which should be of independent interest.

Lemma 3. *Let $p : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a Poisson Binomial distribution, and let $k \geq 1$ satisfy $p(k) < \frac{c}{12\sqrt{3(k+1)}}$, where c comes from Lemma 2. Then, $P_{<k} = \sum_{i<k} p(k)$ and $P_{>k} = \sum_{i>k} p(k)$ satisfy:*

1. *if the mode of p is less than k, then $P_{>k} \leq \frac{1}{2} - \frac{c}{12}$;*
2. *if the mode of p is greater than k, then $P_{<k} \leq \frac{1}{2} - \frac{c}{12}$.*

Informally, the above result states the following: For any k, either its probability under the given Poisson Binomial is at least $\Omega(\frac{1}{\sqrt{k}})$, or this k splits the probability mass into two uneven parts, with the larger one containing the mode. Thus, as long as our candidate α does not yield high acceptance probability for k, it is easy to make the branching decision in the binary search by estimating the quantities $P_{>k}$ and $P_{<k}$ simply by repeated sampling from $\mathrm{DPP}(\alpha\mathbf{L})$. Note that if the condition

Algorithm 2 Binary search for initializing the k-DPP($\mathbf{L}$) sampler

Input: $0 < \alpha_{\min} < \alpha_{\max}$, sampling oracle for DPP($\alpha\mathbf{L}$), integer k and constants $C > 0, \delta \in (0,1)$

Output: $\hat{\alpha}$ such that DPP($\hat{\alpha}\mathbf{L}$) can be used to efficiently sample k-DPP($\mathbf{L}$)

1: **for** $s = \{1, \dots, \lceil \log(\gamma) \rceil\}$ **do**
2: **if** $\alpha_{\max}/\alpha_{\min} < (1 + \frac{1}{(k+3)^2})$ **then return** $\hat{\alpha} = \alpha_{\min}$
3: $\bar{\alpha} \leftarrow \sqrt{\alpha_{\min}\alpha_{\max}}$
4: Sample $S_1, ..., S_t \overset{\text{i.i.d.}}{\sim} \text{DPP}(\bar{\alpha}\mathbf{L})$ where $t = C\sqrt{k}\log(s/\delta)$
5: $\hat{P}_k \leftarrow \frac{1}{t}\sum_{i=1}^{t} \mathbf{1}_{[|S_i|=k]}$
6: **if** $\hat{P}_k \geq \frac{1}{2} \cdot \frac{e}{12\sqrt{3(k+1)}}$ **then return** $\hat{\alpha} = \bar{\alpha}$
7: $(\hat{P}_{<k}, \hat{P}_{>k}) \leftarrow \left(\frac{1}{t}\sum_{i=1}^{t} \mathbf{1}_{[|S_i|<k]}, \frac{1}{t}\sum_{i=1}^{t} \mathbf{1}_{[|S_i|>k]}\right)$
8: **if** $\hat{P}_{<k} > \hat{P}_{>k}$ **then** $(\alpha_{\min}, \alpha_{\max}) = (\bar{\alpha}, \alpha_{\max})$ **else** $(\alpha_{\min}, \alpha_{\max}) = (\alpha_{\min}, \bar{\alpha})$
9: **end for**

on $p(k)$ is not satisfied, then performing the branching decision could be very expensive, but our algorithm avoids this possibility. The proof of Lemma 1 (Appendix B) follows from Lemmas 2 and 3.

3.2 Constructing the initial interval

To initiate our binary search, we must first find a range of values $[\alpha_{\min}, \alpha_{\max}]$, which contains the desired $\alpha_\star$, and also construct a sampling oracle for DPP($\alpha\mathbf{L}$). The binary search procedure is deliberately presented in a way that is agnostic to how these two steps are accomplished, because a number of existing DPP samplers could be adapted to take advantage of Algorithm 2, including [30, 36, 10, 13]. Our implementation of these two steps is different than these previous approaches in that it takes advantage of the structure of the kernel so that it only has to look at a potentially small fraction of the data points. We achieve this with a modified version of the BLESS algorithm [37].

Lemma 4. *W.p.* $1-\delta$ BLESS-I *runs in time* $\widetilde{\mathcal{O}}\left(\min\{\alpha_{\max}\kappa^2, 1\}nk^6 + k^9\right)$ *and satisfies:*

1. *The interval* $[\alpha_{\min}, \alpha_{\max}]$ *is bounded by* $\frac{1}{4}(k-1)/\text{tr}(\mathbf{L}) \leq \alpha_{\min} \leq \alpha_{\max} \leq 8(k+2)/d_{\text{eff}}(\mathbf{L})$
2. *There is* $\alpha_\star \in [\alpha_{\min}, \alpha_{\max}]$ *for which* k *is the mode of* $|S|$ *where* $S \sim \text{DPP}(\alpha_\star\mathbf{L})$*;*
3. *The dictionary* $\mathcal{D}^{\alpha_{\max}}$ *satisfies the conditions from Theorem 2 for any* $\alpha \in [\alpha_{\min}, \alpha_{\max}]$.

The first two parts of the lemma ensure that the interval $I = [\alpha_{\min}, \alpha_{\max}]$ is a valid input for the binary search in Algorithm 2 and that its size $\gamma = \alpha_{\max}/\alpha_{\min} \leq 4\text{tr}(\mathbf{L})/d_{\text{eff}}(\mathbf{L})$ is bounded in the log-scale. The last part implies that α-DPP can be used by that algorithm as the oracle sampler.

At a high level, Algorithm 6 proceeds by starting with a small α^0 that is guaranteed to be a valid lower bound for the interval, and for which a dictionary $\mathcal{D}^0$ can be constructed simply via uniform sampling. Then we repeatedly double the α and refine the dictionary, until we reach α^i such that we can ensure that with high probability $d_{\text{eff}}(\alpha^i\mathbf{L}) \geq k+1$ which makes it a valid upper bound for the interval (then, this α^i becomes $\alpha_{\max}$).

3.3 Overall time complexity of k-DPP sampling

Putting together all the results from the previous sections, we can finally bound the computational complexity of our k-DPP sampler, which first uses BLESS-I (Algorithm 6) to construct a dictionary and search interval, and then applies the binary search of (Algorithm 2) using our α-DPP sampler (Algorithm 1) as the sampling oracle. Once again note that in the following computational analysis we will use conservative values for many parameters, notably r from α-DPP and q from BLESS-I, as they are suggested from the theory. However in practice it is always better to start from a more optimistic value, and keep doubling them only if the sampler repeatedly fails to accept. Importantly, samples generated this way will still be exactly distributed according to the DPP, as all the approximations used in our approach only influence the runtime of our algorithm, and not the correctness of its acceptance, which always holds.

By Lemma 4, the preprocessing step of running BLESS-I takes $\widetilde{\mathcal{O}}\left(\min\{\alpha_{\max}\kappa^2, 1\}nk^6 + k^9\right)$ and generates a dictionary $\mathcal{D}$ with size $\widetilde{\mathcal{O}}(k^3)$. Since $d_{\text{eff}}(\alpha\mathbf{L}) \leq d_{\text{eff}}(\alpha_{\max}\mathbf{L}) \leq \mathcal{O}(k)$ for all α in the search interval, each call to the α-DPP sampler also requires at most $\widetilde{\mathcal{O}}\left(\min\{\alpha_{\max}\kappa^2 k, 1\}nk^6 + k^9\right)$. Finally, the binary search invokes α-DPP at most $\widetilde{\mathcal{O}}(\sqrt{k})$ times so the overall runtime is $\widetilde{\mathcal{O}}\big((\min\{\alpha_{\max}\kappa^2 k, 1\}nk^6 + k^9) \cdot \sqrt{k}\big)$. We now provide a bound on $\alpha_{\max}$.

Lemma 5. *For any matrix* $\mathbf{L}$ *and* $0 < \alpha \leq 1$*, we have* $d_{\text{eff}}(\alpha\mathbf{L})/d_{\text{eff}}(\mathbf{L}) \geq \alpha \geq d_{\text{eff}}(\alpha\mathbf{L})/\text{tr}(\mathbf{L})$.

Applied to $\alpha_{\max}$, we obtain $\alpha_{\max} \leq d_{\text{eff}}(\alpha_{\max}\mathbf{L})/d_{\text{eff}}(\mathbf{L}) \leq \mathcal{O}(k/d_{\text{eff}}(\mathbf{L}))$, giving us the final runtime of $\widetilde{\mathcal{O}}\big((\min\{k^2\kappa^2/d_{\text{eff}}(\mathbf{L}), 1\}nk^6 + k^9) \cdot \sqrt{k}\big)$ reported in Theorem 1.

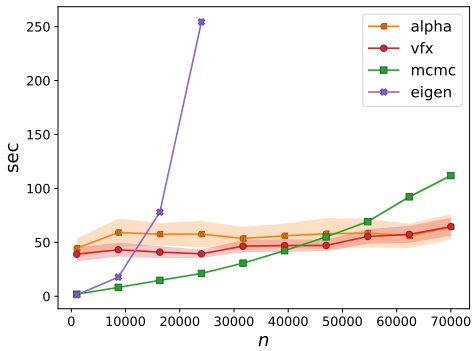

Figure 1: Small scale experiment

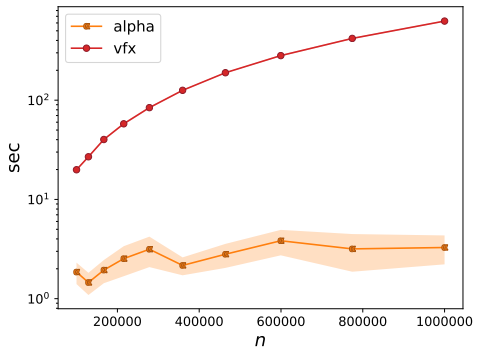

Figure 2: Large scale experiment.

4 Experiments

In this section, we evaluate our α-DPP sampler on a benchmark[2] introduced by [13] (see Appendix D). The benchmark uses subsets of the infinite MNIST dataset [33] with $d = 784$ and n varying up to 10^6. All experiments are executed on a 28 core Xeon E5-2680 v4. Each experiment is repeated multiple times, and we report mean values and a 95% confidence interval.

Baselines: we compare α-DPP with DPP-VFX [13], an MCMC sampler [3] and a sampler based on eigendecompositions [25, 22]. All algorithms are implemented in `python` as part[3] of `DPPy` [20]. Due to their similar input, we use the same oversampling parameters (see Appendix D) for α-DPP and DPP-VFX. We run the MCMC sampler for $\mathcal{O}(nk)$ iterations to guarantee mixing [3]. For more details on hyperparameter tuning we refer to Appendix D.

Results We begin by reporting results on a smaller subset of data (Figure 1) where even the non-efficient samplers can be run. We use an `rbf` similarity with $\sigma = \sqrt{3d}$, and set $k = 10$ to match the number of digit classes in MNIST. Note that for $n = 70000$ BLESS-I estimates $d_{\text{eff}}(\mathbf{L}) \approx 300$, validating our assumption of $d_{\text{eff}}(\mathbf{L}) \gg k$. Thanks to this mismatch, we can see how α-DPP maintains a *constant* runtime as n grows, and increasingly matches or outpaces competing baselines as n grows. In particular, it becomes faster than the eigendecomposition based sampler (which cannot scale beyond $n = 24000$) or the MCMC sampler. However, the gap is still sufficiently small that DPP-VFX, the previously fastest k-DPP sampler available, remains competitive. Note that, unlike the MCMC and eigendecomposition based samplers, α-DPP and DPP-VFX sample from a k-DPP by repeatedly sampling from a random-size DPP until they generate a sample with size exactly k. While our theory ensures that this will happen after only a small number of rejections, this creates some overhead cost relative to the other two methods, which is noticeable for small values of n.

For larger datasets we consider only the scalable samplers, α-DPP and DPP-VFX. We consider again an `rbf` similarity, but this time we choose n up to 10^6 and $\sigma = \sqrt{10}$. This further increases the gap between k and $d_{\text{eff}}(\mathbf{L})$, with BLESS-I estimating $d_{\text{eff}}(\mathbf{L}) \approx 1000$. We report results in Figure 2, with runtime shown in log-scale. In this regime, the gap between DPP-VFX and α-DPP widens, as DPP-VFX cannot use rescaling to reduce the final dictionary size from $d_{\text{eff}}(\mathbf{L})$ to $d_{\text{eff}}(\hat{\alpha}\mathbf{L}) \approx k$, and has to compute n marginal probabilities since it does not leverage uniform intermediate subsampling. In particular, thanks to the uniform sampling step, we see that α-DPP's runtime does not grow as n

[2] https://github.com/LCSL/dpp-vfx

[3] Our implementation of α-DPP is included in the supplementary material, and it is also available in `DPPy`.

grows, since all the expensive computations are performed in the small intermediate subset which is hardly sensitive to n. We note that, due to using a smaller dictionary, α-DPP requires about 2-5x more trials in the rejection sampling step, which leads to larger variance in the runtime.

In Figure 3, we report the fraction of data that is observed by α-DPP in the large scale experiment. This quantity, denoted as β in Theorem 1, is responsible for much of the computational gains of the algorithm over DPP-VFX, reported in Figure 2. Note that the remaining $1 - \beta$ portion of the data does not ever need to be loaded into the program's memory, which leads to a significant reduction in memory accesses. We observe that as the data size increases, the fraction of items observed by α-DPP goes down to as little as 1% for $n = 10^6$, which is why the runtime of α-DPP stays roughly flat, whereas the runtime of DPP-VFX grows.

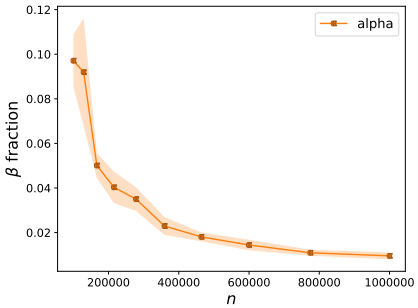

Figure 3: Fraction of items observed by α-DPP.

Broader impact

DPPs were discovered in the 70s by Odile Macchi to model repulsion of particle distributions in fermions, so improvements in samplers may help in modelling physical simulations. In bringing faster DPP samplers to machine learning we aim to enable a better handling of diversity through this rigorous theoretical framework.

Acknowledgments and Disclosure of Funding

MD thanks the NSF for funding via the NSF TRIPODS program.

References

[1] Raja Hafiz Affandi, Alex Kulesza, Emily Fox, and Ben Taskar. Nystrom approximation for large-scale determinantal processes. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 85–98, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

[2] Ahmed El Alaoui and Michael W. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 775–783, Montreal, Canada, December 2015.

[3] Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 103–115, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[4] Rémi Bardenet, Frédéric Lavancier, Xavier Mary, and Aurélien Vasseur. On a few statistical applications of determinantal point processes. *ESAIM: Procs*, 60:180–202, 2017. doi: 10.1051/proc/201760180.

[5] Daniele Calandriello, Alessandro Lazaric, and Michal Valko. Distributed adaptive sampling for kernel matrix approximation. In *AISTATS*, 2017.

[6] Andrea Caponnetto and Ernesto De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.

[7] Elisa Celis, Vijay Keswani, Damian Straszak, Amit Deshpande, Tarun Kathuria, and Nisheeth Vishnoi. Fair and diverse DPP-based data summarization. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 716–725, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[8] Laming Chen, Guoxin Zhang, and Eric Zhou. Fast greedy map inference for determinantal point process to improve recommendation diversity. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5622–5633. Curran Associates, Inc., 2018.

[9] John N Darroch et al. On the distribution of the number of successes in independent trials. *The Annals of Mathematical Statistics*, 35(3):1317–1321, 1964.

[10] Michał Dereziński. Fast determinantal point processes via distortion-free intermediate sampling. In *Proceedings of the 32nd Conference on Learning Theory*, 2019.

[11] Michał Dereziński and Michael W Mahoney. Determinantal point processes in randomized numerical linear algebra. *arXiv preprint arXiv:2005.03185*, 2020. Accepted for publication, Notices of the AMS.

[12] Michał Dereziński, Manfred K. Warmuth, and Daniel Hsu. Leveraged volume sampling for linear regression. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 2510–2519. Curran Associates, Inc., 2018.

[13] Michał Dereziński, Daniele Calandriello, and Michal Valko. Exact sampling of determinantal point processes with sublinear time preprocessing. In *Advances in Neural Information Processing Systems*, pages 11542–11554, 2019.

[14] Michał Dereziński, Manfred K. Warmuth, and Daniel Hsu. Correcting the bias in least squares regression with volume-rescaled sampling. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

[15] Michał Dereziński, Burak Bartan, Mert Pilanci, and Michael W Mahoney. Debiasing distributed second order optimization with surrogate sketching and scaled regularization. *arXiv preprint arXiv:2007.01327*, 2020. Accepted for publication, Proc. NeurIPS 2020.

[16] Michał Dereziński, Rajiv Khanna, and Michael W Mahoney. Improved guarantees and a multiple-descent curve for the column subset selection problem and the nyström method. *arXiv preprint arXiv:2002.09073*, 2020. Accepted for publication, Proc. NeurIPS 2020.

[17] Michal Derezinski, Feynman Liang, and Michael Mahoney. Bayesian experimental design using regularized determinantal point processes. In *International Conference on Artificial Intelligence and Statistics*, pages 3197–3207, 2020.

[18] Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 1117–1126, Miami, FL, USA, January 2006.

[19] Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. Bayesian low-rank determinantal point processes. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 349–356, Boston, MA, USA, September 2016.

[20] Guillaume Gautier, Rémi Bardenet, and Michal Valko. DPPy: Sampling determinantal point processes with Python. *Journal of Machine Learning Research - Machine Learning Open Source Software (JMLR-MLOSS)*, 2019.

[21] Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Discovering diverse and salient threads in document collections. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 710–720, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[22] Jennifer Ann Gillenwater. Approximate inference for determinantal point processes. 2014.

[23] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2069–2077. Curran Associates, Inc., 2014.

[24] Venkatesan Guruswami and Ali K. Sinop. Optimal column-based low-rank matrix reconstruction. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1207–1214, Kyoto, Japan, January 2012.

[25] J Ben Hough, Manjunath Krishnapur, Yuval Peres, Bálint Virág, et al. Determinantal processes and independence. *Probability surveys*, 3:206–229, 2006.

[26] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4206–4214. Curran Associates, Inc., 2016.

[27] SK Katti and A Vijaya Rao. Handbook of the poisson distribution, 1968.

[28] Alex Kulesza and Ben Taskar. k-DPPs: Fixed-Size Determinantal Point Processes. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1193–1200, Bellevue, WA, USA, June 2011.

[29] Alex Kulesza and Ben Taskar. *Determinantal Point Processes for Machine Learning*. Now Publishers Inc., Hanover, MA, USA, 2012.

[30] Claire Launay, Bruno Galerne, and Agnès Desolneux. Exact Sampling of Determinantal Point Processes without Eigendecomposition. *arXiv e-prints*, art. arXiv:1802.08429, Feb 2018.

[31] Chengtao Li, Stefanie Jegelka, and Suvrit Sra. Efficient sampling for k-determinantal point processes. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1328–1337, Cadiz, Spain, 09–11 May 2016. PMLR.

[32] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 510–520, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9.

[33] Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. In Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

[34] Zelda E. Mariet and Suvrit Sra. Elementary symmetric polynomials for optimal experimental design. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2136–2145. 2017.

[35] Mojmir Mutny, Michal Derezinski, and Andreas Krause. Convergence analysis of block coordinate algorithms with determinantal sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 3110–3120, 2020.

[36] Jack Poulson. High-performance sampling of generic determinantal point processes. ArXive:1905.00165v1, 2019.

[37] Alessandro Rudi, Daniele Calandriello, Luigi Carratino, and Lorenzo Rosasco. On fast leverage score sampling and optimal learning. In *Advances in Neural Information Processing Systems 31*, pages 5672–5682. 2018.

[38] Wenpin Tang and Fengmin Tang. The poisson binomial distribution–old & new. *arXiv preprint arXiv:1908.10024*, 2019.

[39] Cheng Zhang, Hedvig Kjellström, and Stephan Mandt. Determinantal point processes for mini-batch diversification. In *33rd Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, 11 August 2017 through 15 August 2017*. AUAI Press Corvallis, 2017.

A Correctness and efficiency of α-DPP (Algorithm 1)

In this section we prove the theorems stated in Sections 2 and 3 claiming the correctness and efficiency of DPP-VFX. In particular, we split Theorem 2 into two parts.

Lemma 6. *Given any psd matrix* $\mathbf{L}$*, dictionary* $\mathcal{D}$*, positive weights* $\mathbf{W}$*,* $r \geq 1$*, and positive* $\alpha > 0$*,* α-DPP *returns* $S \sim \mathrm{DPP}(\alpha\mathbf{L})$*.*

Lemma 7. *If* $r \geq d_{\text{eff}}(\alpha\mathbf{L}) \geq 1/2$*,* $\mathcal{D}$ *and* $\mathbf{W}$ *are* $(1/d_{\text{eff}}(\alpha\mathbf{L}), \alpha)$*-accurate,* $\mathcal{D}$ *satisfies* $|\mathcal{D}| \leq 10 d_{\text{eff}}(\alpha\mathbf{L})$*, and* $d_{\text{eff}}(\alpha\widehat{\mathbf{L}}) \leq 10 d_{\text{eff}}(\alpha\mathbf{L})$*, then with probability* $1-\delta$*,* α-DPP *(Algorithm 1) runs in time*

$$\mathcal{O}\Big(\big[\min\{\alpha\kappa^2 d_{\text{eff}}(\alpha\mathbf{L}), 1\} \cdot n \cdot d_{\text{eff}}(\alpha\mathbf{L})^6 \log^2(n/\delta) + d_{\text{eff}}(\alpha\mathbf{L})^9 \log^3(n/\delta)\big] \cdot \log(1/\delta)\Big).$$

A.1 Notation

We start by introducing some additional notation. First, let us describe the so-called kernel-based view of DPPs. We associate with our similarity matrix $\mathbf{L}$ a similarity function or a kernel[4] function $K(\cdot,\cdot) : [n] \times [n] \to \mathbb{R}$ such that $K(i,j)$ is equal to the (i,j)-th entry of $\mathbf{L}$.

We also generalize the notation just defined in a way that given multi-sets A and B, $K(A,B) \stackrel{\text{def}}{=} \mathbf{L}_{A,B}$ returns the matrix containing the corresponding rows and columns of $\mathbf{L}$. Note that if A or B contains duplicates (e.g., the i-th index appears twice in A) the matrix $K(A,B)$ will consequently contain duplicate rows and columns. Finally, note that in this notation the original matrix can be written as $\mathbf{L} = K([n],[n])$.

We also denote with $\varphi(\cdot) : [n] \to \mathbb{R}^D$ the so-called feature map associated with $\mathbf{L}$ and $K(\cdot,\cdot)$ such that $K(i,j) = \varphi(i)^\top \varphi(j)$, where D can be arbitrarily large or infinite.[5] Just as with K, we also extend $\varphi(\cdot)$ to operate on multi-set, such that given $A = \{i_1, \dots, i_m\}$ (potentially with duplicates $i_j = i_l$), we have $\varphi(A) = [\varphi(i_1), \dots, \varphi(i_m)]^\top \in \mathbb{R}^{m \times D}$.

Using the above notation, we have $\mathbf{L} = K([n],[n]) = \varphi([n])\varphi([n])^\top$. Note also that the corresponding operator $\varphi([n])^\top\varphi([n])$ can be decomposed as a sum of outer products $\varphi([n])^\top\varphi([n]) = \sum_{i=1}^n \varphi(i)\varphi(i)^\top$.

We also use the following notation to indicate common sampling distributions:

- $u \sim \mathrm{Poisson}(\lambda)$ as a non-negative integer sampled from a Poisson distribution with intensity $0 < \lambda$;
- $\rho \sim \mathrm{Uniform}(u, [n])$ as a set of size u sampled uniformly i.i.d. with replacement from $[n]$; i.e., $\rho = (\rho_1, \dots, \rho_u) \stackrel{\text{i.i.d.}}{\sim} (1/n, \dots, 1/n)$.
- $z \sim \mathrm{Bernoulli}(p)$ as the $\{0,1\}$ r.v. sampled from a Bernoulli distribution w.p. $0 \leq p \leq 1$;
- $s \sim \mathrm{Binomial}(k,p)$ as the non-negative integer in the range $[0,k]$ sampled from a Binomial distribution with $0 \leq k$ Bernoulli repetitions each with probability $0 \leq p \leq 1$
- $(s_1, \dots, s_n) \sim \mathrm{GenBinomial}(k, \{p_i\}_{i=1}^n)$ as the vector of positive integers $[0,k]^n$ with $0 \leq k$ sampled according to $\mathbf{P}\left((s_1, \dots, s_n)\right) = \frac{n!}{\prod_{i=1}^n s_i!} \prod_{i=1}^n p_i^{s_i}$ such that $\sum_{i=1}^n s_i = k$.
- $\sigma \sim \mathrm{Multinomial}(k, \{p_i\}_{i=1}^n)$ as a set of size k sampled i.i.d. with replacement from $[n]$ according to probabilities $0 \leq p_i \leq 1$ with $\sum_{i=1}^n p_i = 1$, i.e., $\sigma = (\sigma_1, \dots, \sigma_k) \stackrel{\text{i.i.d.}}{\sim} (p_1, \dots, p_n)$.

[4] Note that we are defining the kernel K as a function on indices, but since we focus on DPPs defined on PSD matrices, everything can be immediately extended to any standard PSD kernel $K(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined on an arbitrary input space $\mathcal{X}$.

[5] Again we focus on a feature map from indices to a finite dimensional space. All the results can be immediately extended to a feature map $\varphi(\cdot) : \mathcal{X} \to \mathcal{H}$ that maps from an arbitrary input space into a reproducing kernel Hilbert space, e.g., Gaussian kernel and Gaussian feature maps. Notice that in our setting given a PSD matrix $\mathbf{L}$ the eigenspace of $\mathbf{L}$ suffices to construct an appropriate feature map $\varphi(\cdot) : [n] \to \mathbb{R}^n$ with $D = n$. In particular, we have an explicit expression for $\varphi(\cdot)$ based on the eigendecomposition $\mathbf{L} = \mathbf{U}\Sigma\mathbf{U}^\top$ of $\mathbf{L}$. Since $\mathbf{L}$ is psd, Σ is a diagonal matrix with non-negative entries, and we can define $\Sigma^{+/2}$ as the square root of its pseudo-inverse. Then the feature map becomes $\varphi(\cdot) \stackrel{\text{def}}{=} \Sigma^{+/2}\mathbf{U}^\top K([n],\cdot)$. A similar argument can be made using the Cholesky decomposition of $\mathbf{L}$.

A.2 Proof of Lemma 6 (exact sampling)

To prove that α-DPP is an exact sampler we show that σ is sampled according to an appropriate R-DPP, and that therefore we can invoke Proposition 1.

Proof of Lemma 6. Given the approximate marginals l_i from Equation 1, let us denote with $\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) \overset{def}{=} \sum_{i=1}^n l_i$ their sum, or approximate effective dimension. Note that Algorithm 1 *never* computes $\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})$ explicitly, nor does it compute all approximate marginals l_i. Nonetheless, our first claim is that the inner loop of α-DPP is proposing a candidate σ sampled according to the approximate marginals l_i even without computing them all.

Lemma 8. *The set σ generated by Algorithm 1 before Line 11 is distributed as*

$$\sigma = \{\sigma_1, \dots, \sigma_t\} \overset{\text{i.i.d.}}{\sim} (l_1/\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}), \dots, l_n/\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})), \qquad t \sim \textit{Poisson}\left(re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})\right).$$

Then, we show that the rejection sampling step of Line 11 is valid.

Lemma 9. *Given any psd matrix $\mathbf{L}$, dictionary $\mathcal{D}$, positive weights $\mathbf{W}$, $r \geq 1$, and positive $\alpha > 0$, the acceptance probability $\frac{\mathrm{e}^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})}\det(\mathbf{I}+\alpha\widetilde{\mathbf{L}}_\sigma)}{\mathrm{e}^{t/r}\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})} \leq 1$ is valid.*

Let $\widetilde{\sigma}$ denote the random variable distributed as σ is after exiting the repeat loop. Combining Lemma 9 with the fact that $t \sim \text{Poisson}(re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}))$ is a Poisson r.v. it follows that

$$\begin{aligned}
\Pr(\widetilde{\sigma} \in A) &\propto \mathbb{E}_\sigma\left[\mathbf{1}_{[\sigma\in A]}\frac{\mathrm{e}^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})}\det(\mathbf{I}+\widetilde{\mathbf{L}}_\sigma)}{\mathrm{e}^{t/r}\det(\mathbf{I}+\widehat{\mathbf{L}})}\right] \\
&\propto \sum_{t=0}^{\infty}\frac{(r\,\mathrm{e}^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}))^t}{\mathrm{e}^{r\,\mathrm{e}^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}\,t!}\cdot\mathrm{e}^{-t/r}\,\mathbb{E}_\sigma\left[\mathbf{1}_{[\sigma\in A]}\det(\mathbf{I}+\widetilde{\mathbf{L}}_\sigma)\mid t\right] \\
&\propto \mathbb{E}_{t'}\left[\mathbb{E}_\sigma\left[\mathbf{1}_{[\sigma\in A]}\det(\mathbf{I}+\widetilde{\mathbf{L}}_\sigma)\mid t=t'\right]\right] \quad \text{for } t' \sim \text{Poisson}(r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})),
\end{aligned}$$

which matches the numerator of Definition 1 for a $\text{R-DPP}^{r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}_{\{l_i/\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})\}_{i=1}^n}$. All that remains is to show that the distribution integrates properly, i.e., the denominator also matches. We do this by generalizing a determinantal equality to our modified reweighting.

Proposition 2 ([10]). *If $t \sim \textit{Poisson}(r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}))$ and $(\sigma_1, \dots, \sigma_t) \overset{\text{i.i.d.}}{\sim} \left(\frac{l_1}{\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}, \dots, \frac{l_n}{\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}\right)$ then*

$$\mathbb{E}_{t,\sigma}\left[\det\left(\mathbf{I}+\widetilde{\mathbf{L}}_\sigma\right)\right] = \det\left(\mathbf{I}+\mathbf{L}\right).$$

This shows that $\widetilde{\sigma} \sim \text{R-DPP}^{r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}_{\{l_i/\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})\}_{i=1}^n}$. The claim follows from Proposition 1. ■

Proof of Lemma 8. Before starting, we will use two well known connections of the Poisson distribution with GenBinomial and Binomial r.v. [27]. The first useful Poisson property is that for any set of positive weights $\{\lambda_i\}_{i=1}^n$ the random variables $X_1 \sim \text{Poisson}(\lambda_1), \dots, X_n \sim \text{Poisson}(\lambda_n)$ and the random variables

$$k \sim \text{Poisson}\big(\textstyle\sum_{i=1}^n \lambda_i\big), \qquad \{X_1, \dots, X_n\}|k \sim \text{GenBinomial}\left(k, \frac{\lambda_i}{\sum_{i=1}^n \lambda_i}\right)$$

are equally distributed. Note that for this identity to hold we do not need to explicitly compute $\sum_{i=1}^n \lambda_i$, as we can simply sample n Poisson r.v.-s and obtain the normalization effect in the GenBinomial sample for free using the conditioning on k. The second useful Poisson property we will use is that if $k \sim \text{Poisson}(\lambda)$ and $Y|k \sim \text{Binomial}(k, p)$ then $Y \sim \text{Poisson}(\lambda \cdot p)$.

Algorithm 3 α-DPP reformulation in terms of u_i and s_i

Input: α, $\mathbf{L}$, an m-element dictionary $\mathcal{D}$, weight matrix $\mathbf{W} \in \mathbb{R}^{m\times m}$, $r \geq 1$

Set $\widehat{\mathbf{L}} = \mathbf{W}^{1/2}\mathbf{L}_{\mathcal{D},\mathcal{D}}\mathbf{W}^{1/2} \in \mathbb{R}^{m\times m}$
repeat
  for $i = \{1, \dots, n\}$ **do**
    Sample $u_i \sim \text{Poisson}(re^{1/r}\alpha\kappa^2)$
    if $u_i > 0$ **then**
      Compute l_i using Equation 1
      Sample $s_i \sim \text{Binomial}(u_i, l_i/(\alpha\kappa^2))$
    else
      Set $s_i = 0$
    end if
    Add s_i copies of i to σ
  end for
  Set $t = |\sigma|$, $[\widetilde{\mathbf{L}}_\sigma]_{ij} = \frac{1}{r\sqrt{l_{\sigma_i} l_{\sigma_j}}}[\mathbf{L}]_{\sigma_i\sigma_j}$
  Sample $Acc \sim \text{Bernoulli}\Big(\mathrm{e}^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})-t/r}\det(\mathbf{I}+\alpha\widetilde{\mathbf{L}}_\sigma)/\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})\Big)$
until $Acc =$ true
Sample $\widetilde{S} \sim \text{DPP}\big(\alpha\widetilde{\mathbf{L}}_\sigma\big)$
return $S = \{\sigma_i : i \in \widetilde{S}\}$

Let us denote with $u_i = \sum_{j=1}^{u} \mathbb{I}\{\rho_j = i\}$ the multiplicity of index i in ρ, such that we have a set of n random variables $\{u_i\}_{i=1}^n$ and that $u = \sum_{i=1}^n u_i$. Then, from the previous relationship, we can instantly see that sampling $u_i \sim \text{Poisson}(re^{1/r}b)$ is equivalent to sampling

$$u \sim \text{Poisson}\big(\sum_{i=1}^n re^{1/r}\alpha\kappa^2\big) = \text{Poisson}(re^{1/r}n\alpha\kappa^2),$$
$$\{u_1, \dots, u_n\}|u \sim \text{GenBinomial}\left(u, \tfrac{re^{1/r}\alpha\kappa^2}{re^{1/r}n\alpha\kappa^2}\right) = \text{GenBinomial}\left(u, \left\{\tfrac{1}{n}\right\}\right).$$

We can now connect u_i and ρ. In particular sampling $\rho|u \sim \text{Uniform}(u, [n])$ is equivalent to sampling $\{u_1, \dots, u_n\}|u \overset{\text{i.i.d.}}{\sim} \text{GenBinomial}\left(u, \left\{\frac{1}{n}\right\}\right)$ and then adding u_i copies of i to $\rho|u$ for each $i \in [n]$.

Starting from this characterization, let us now denote with $s_i = \sum_{j=1}^t \mathbb{I}\{\sigma_j = i\}$ the multiplicity of index i in σ, such that we have a set of n random variables $\{s_i\}_{i=1}^n$ and that $t = \sum_{i=1}^n s_i$. We can now formally describe Line 7 of Algorithm 1 as a binomial sampling step: first we sample $u_i \sim \text{Poisson}(re^{1/r}\alpha\kappa^2)$, and then we sample $s_i|u_i \sim \text{Binomial}(u_i, l_i/(\alpha\kappa^2))$. To see this, we can just sum over all z_j that correspond to the i-th element, of which we have exactly u_i, and remember that a sum of i.i.d. Bernoulli is a Binomial. We also have to take care of the fact that the Binomial probability is well defined, i.e., smaller than 1, but it is easy to see that $l_i \leq \alpha[\mathbf{L}]_{i,i} \leq \alpha\kappa^2$ and $l_i/(\alpha\kappa^2) \leq 1$. We can now use the second fact about Poissons, namely that sampling $u_i \sim \text{Poisson}(re^{1/r}b)$ and then $s_i|u_i \sim \text{Binomial}(u_i, \frac{l_i}{b})$ is equivalent to sampling $s_i \sim \text{Poisson}(re^{1/r}b \cdot \frac{l_i}{b}) = \text{Poisson}(re^{1/r}l_i)$.

Finally we can once again use the equivalence between Poisson and GenBinomial sampling to see that sampling $s_i \sim \text{Poisson}(re^{1/r}l_i)$ for each $i \in [n]$ is equivalent to sampling

$$t \sim \text{Poisson}\big(\textstyle\sum_{i=1}^n ql_i\big) = \text{Poisson}(re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})), \quad \{s_1, \dots, s_n\}|t \sim \text{GenBinomial}\left(t, \frac{l_i}{\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}\right),$$

which in turn implies that by adding s_i copies of the index i to σ, which is what Algorithm 1 is doing, we are sampling according to

$$\sigma = \{\sigma_1, \dots, \sigma_t\} \overset{\text{i.i.d.}}{\sim} (l_1/\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}), \dots, l_n/\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})), \qquad t \sim \text{Poisson}(re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})),$$

without ever explicitly computing $\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})$.

For completeness, we also include the two implicit reformulations of Algorithm 1 that we just described as Algorithm 3 and Algorithm 4. Note that all three algorithms are strictly equivalent, but depending on the actual implementation they have different complexities. For example, Algorithm 4

Algorithm 4 α-DPP reformulation in terms of s_i

Input: α, $\mathbf{L}$, an m-element dictionary $\mathcal{D}$, weight matrix $\mathbf{W} \in \mathbb{R}^{m\times m}$, $r \geq 1$

repeat
  for $i = \{1, \dots, n\}$ **do**
    Compute l_i using Equation 1
    Sample $s_i \sim \text{Poisson}(re^{1/r} l_i)$
    Add s_i copies of i to σ
  end for
  Set $t = |\sigma|$, $[\widetilde{\mathbf{L}}_\sigma]_{ij} = \frac{1}{r\sqrt{l_{\sigma_i} l_{\sigma_j}}}[\mathbf{L}]_{\sigma_i\sigma_j}$
  Sample $Acc \sim \text{Bernoulli}\Big(\mathrm{e}^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})-t/r}\det(\mathbf{I}+\alpha\widetilde{\mathbf{L}}_\sigma)/\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})\Big)$
until $Acc =$ true
Sample $\widetilde{S} \sim \text{DPP}\big(\alpha\widetilde{\mathbf{L}}_\sigma\big)$
return $S = \{\sigma_i : i \in \widetilde{S}\}$

needs to compute all marginals in advance. We chose to include Algorithm 1 in the main paper as the version that more clearly highlights the uniform sampling step. ■

Proof of Lemma 9. The first reason we introduced the kernel-based DPP notation is to be able to succinctly use Sylvester's identity to equate determinants in the matrix and feature view of the DPP, i.e.,

$$\det(\mathbf{I}+\mathbf{L}) = \det(\mathbf{I}+\varphi([n])\varphi([n])^\top) = \det(\mathbf{I}+\varphi([n])^\top\varphi([n])) = \det\left(\mathbf{I}+\textstyle\sum_{i=1}^n \varphi(i)^\top\varphi(i)\right),$$

where the size of the identity matrix[6] $\mathbf{I}$ is either n or D and it is clear from the context. Similarly, the denominator $\det(\mathbf{I}+\alpha\widehat{\mathbf{L}}) = \det(\mathbf{I}+\alpha\mathbf{W}^{1/2}\mathbf{L}_{\mathcal{D},\mathcal{D}}\mathbf{W}^{1/2})$ in the rejection loop becomes

$$\det(\mathbf{I}+\alpha\mathbf{W}^{1/2}\mathbf{L}_{\mathcal{D},\mathcal{D}}\mathbf{W}^{1/2}) = \det(\mathbf{I}+\alpha\mathbf{W}^{1/2}\varphi(\mathcal{D})\varphi(\mathcal{D})^\top\mathbf{W}^{1/2}) = \det(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D})).$$

Finally, given σ let us denote with $\widetilde{\varphi}(i) = \frac{1}{\sqrt{rl_i}}\varphi(i)$ a rescaled feature map, where once again we extend $\widetilde{\varphi}(\sigma) = \text{Diag}(\frac{1}{\sqrt{rl_{\sigma_i}}})_{i=1}^m\varphi(\sigma)$ to multi-sets. Then the numerator in the rejection loop becomes

$$\begin{aligned}\det(\mathbf{I}+\alpha\widetilde{\mathbf{L}}_\sigma) &= \det(\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)\widetilde{\varphi}(\sigma)^\top)\\ &= \det(\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma)) = \det\left(\mathbf{I}+\alpha\textstyle\sum_{j=1}^t \frac{1}{rl_{\sigma_j}}\varphi(\sigma_j)\varphi(\sigma_j)^\top\right).\end{aligned}$$

The second reason we introduce this notation is that the formulation of the approximate marginals l_i is much simplified and becomes (see [5, 37] for details)

$$l_i = \alpha[\mathbf{L}-\alpha\mathbf{L}_{[n],\mathcal{D}}^\top(\alpha\mathbf{L}_{\mathcal{D},\mathcal{D}}+\mathbf{W}^{-1})^{-1}\mathbf{L}_{[n],\mathcal{D}}]_{i,i} = \alpha\varphi(i)^\top(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\varphi(i). \quad (2)$$

Using the kernel-based view of DPPs and the reformulation of most quantities, we can now move from characterizing the distribution of σ, to computing the final acceptance probability $\mathbf{P}(Acc|\sigma)$. In particular, to guarantee correctness we must guarantee that the rejection step is valid, i.e., that the acceptance probability is bounded by 1. For this we rewrite the acceptance condition as

$$\frac{\det(\mathbf{I}+\alpha\widetilde{\mathbf{L}}_\sigma)}{\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})} = \frac{\det(\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma))}{\det(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))}.$$

Similarly to [13], we can use the inequality $\det(\mathbf{I}+\mathbf{A}) \leq \exp\{\text{tr}(\mathbf{A})\}$, which follows immediately by applying the bound $1+x \leq e^x$ to each singular value of $\mathbf{A}$. We obtain

$$\begin{aligned}&\frac{\det(\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma))}{\det(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))}\\ &= \det\left((\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma))(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\right)\\ &= \det\left(\mathbf{I}+(\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma)-\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\right)\\ &\leq \exp\left\{\text{tr}\left((\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma)-\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\right)\right\}\\ &= \exp\{\underbrace{\text{tr}\left(\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma)(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\right)}_{(a)} - \underbrace{\text{tr}\left(\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D})(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\right)}_{(b)}\}.\end{aligned}$$

[6] Or an identity operator on an RKHS in general

For (b), we can see that by definition $\operatorname{tr}\left(\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D})(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\right) = d_{\text{eff}}(\alpha\widehat{\mathbf{L}})$.
For (a), we have

$$
\begin{aligned}
(a) &= \operatorname{tr}\left(\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma)(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\right) = \operatorname{tr}\left(\alpha\widetilde{\varphi}(\sigma)(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\widetilde{\varphi}(\sigma)^\top\right)\\
&= \sum_{j=1}^{t}\alpha\widetilde{\varphi}(\sigma_j)^\top(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\widetilde{\varphi}(\sigma_j) = \sum_{j=1}^{t}\frac{1}{rl_{\sigma_j}}\alpha\varphi(\sigma_j)^\top(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\varphi(\sigma_j)\\
&= \sum_{j=1}^{t}\frac{1}{rl_{\sigma_j}}l_{\sigma_j} = \frac{t}{r}.
\end{aligned}
$$

Putting (a) and (b) together we have

$$
\begin{aligned}
\mathbf{P}(Acc=\text{true}|\sigma) &= \frac{\det(\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma))}{\det(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))}\cdot\exp\left\{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})-\frac{t}{r}\right\}\\
&\leq \exp\left\{\frac{t}{r}-d_{\text{eff}}(\alpha\widehat{\mathbf{L}})\right\}\cdot\exp\left\{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})-\frac{t}{r}\right\} = e^0 = 1.
\end{aligned}
$$

■

Proof of Proposition 2. We first rewrite the equality as

$$
\mathbb{E}_{t,\sigma}\left[\det\left(\mathbf{I}+\sum_{j=1}^{t}\frac{1}{rl_{\sigma_j}}\varphi(\sigma_j)\varphi(\sigma_j)^\top\right)\right] = \det\left(\mathbf{I}+\sum_{i=1}^{n}\varphi(i)\varphi(i)^\top\right).
$$

Dereziński [10] showed the following identity when sampling $t\sim\text{Poisson}(r)$ and then sampling a multi-set σ with t elements i.i.d. from any arbitrary distribution,

$$
\mathbb{E}_{t,\sigma}\left[\det\left(\mathbf{I}+\varphi(\sigma)^\top\varphi(\sigma)\right)\right] = \det\left(\mathbf{I}+r\mathbb{E}_{\sigma_1}\left[\varphi(\sigma_1)\varphi(\sigma_1)^\top\right]\right).
$$

Applying this to our $t\sim\text{Poisson}(r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}))$ and the distribution of σ we have

$$
\begin{aligned}
\mathbb{E}_{t,\sigma}\left[\det\left(\mathbf{I}+\sum_{j=1}^{t}\frac{1}{rl_{\sigma_j}}\varphi(\sigma_j)\varphi(\sigma_j)^\top\right)\right] &= \det\left(\mathbf{I}+r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})\mathbb{E}_{\sigma_1}\left[\varphi(\sigma_1)\varphi(\sigma_1)^\top\right]\right)\\
&= \det\left(\mathbf{I}+r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})\sum_{i=1}^{n}\frac{l_i}{\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}\frac{1}{rl_i}\varphi(\sigma_i)\varphi(\sigma_i)^\top\right)\\
&= \det\left(\mathbf{I}+\sum_{i=1}^{n}\varphi(i)\varphi(i)^\top\right).
\end{aligned}
$$

■

A.3 Proof of Lemma 7 (efficiency)

Proof of Lemma 7. We need to lower bound the acceptance probability $\mathbf{P}(Acc|\sigma)$. Note that this is equivalent to lower bounding $\mathbb{E}[Acc=\text{true}]$ since it is a $\{0,1\}$ random variable.

$$
\begin{aligned}
\mathbf{P}(Acc=\text{true}) &= \mathbb{E}_{\sigma,t}\left[\frac{e^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})}\det(\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma))}{e^{t/r}\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})}\right] = \frac{e^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})}}{\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})}\mathbb{E}_{\sigma,t}\left[\frac{\det(\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma))}{e^{t/r}}\right]\\
&= \frac{e^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})}}{\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})}\cdot\sum_{t=0}^{\infty}\mathbb{E}_{\sigma}\left[\det(\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma))|\ t\right]\frac{1}{e^{t/r}}\frac{\left(re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})\right)^t}{t!\cdot e^{re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}},
\end{aligned}
$$

where we expanded the expectation with respect to $t\sim\text{Poisson}(re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}))$. Focusing on the last term we have

$$
\frac{1}{e^{t/r}}\frac{\left(re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})\right)^t}{t!\cdot e^{re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}}. = \frac{1}{e^{t/r}}\frac{r^te^{t/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})^t}{t!\cdot e^{re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}} = \frac{r^t\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})^t}{t!\cdot e^{re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}} = \frac{(r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}))^t}{t!\cdot e^{r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}}\frac{e^{r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}}{e^{re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}}.
$$

Recognizing that $\frac{(r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}))^t}{t!\cdot e^{r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}}$ is the density of a $t \sim \text{Poisson}(r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}))$, we can apply Proposition 2,

$$\begin{aligned}
\mathbf{P}(Acc = \text{true}) &= \frac{e^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})}}{\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})} \cdot \sum_{t=0}^{\infty} \mathbb{E}_\sigma\left[\det(\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma))|\ t\right] \tfrac{1}{e^{t/r}} \frac{\left(re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})\right)^t}{t!\cdot e^{re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}} \\
&= \frac{e^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})}}{\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})} \frac{e^{r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}}{e^{re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}} \cdot \sum_{t=0}^{\infty} \mathbb{E}_\sigma\left[\det(\mathbf{I}+\alpha\widetilde{\varphi}(\sigma)^\top\widetilde{\varphi}(\sigma))|\ t\right] \frac{(r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}))^t}{t!\cdot e^{r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}} \\
&= \frac{e^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})}}{\det(\mathbf{I}+\alpha\widehat{\mathbf{L}})} \frac{e^{r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}}{e^{re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}} \cdot \det(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n])) \\
&= \frac{e^{d_{\text{eff}}(\alpha\widehat{\mathbf{L}})}e^{r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}}{e^{re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})}} \frac{\det(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))}{\det(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))}.
\end{aligned}$$

To lower bound this quantity we will again upper bound the inverse $\frac{\det(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))}{\det(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))}$ as follows

$$\begin{aligned}
\tfrac{\det(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))}{\det(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))} &\leq \exp\left\{\operatorname{tr}\left((\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}) - \alpha\varphi([n])^\top\varphi([n]))(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1}\right)\right\} \\
&= \exp\left\{\operatorname{tr}\left(\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D})(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1}\right) - d_{\text{eff}}(\alpha\mathbf{L})\right\}.
\end{aligned}$$

Inverting the relationship and putting it all together we have

$$\begin{aligned}
&\mathbf{P}(Acc_\sigma = \text{true}) \\
&\geq \exp\left\{d_{\text{eff}}(\alpha\widehat{\mathbf{L}}) + r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) - re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) + d_{\text{eff}}(\alpha\mathbf{L}) - \operatorname{tr}\left(\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D})(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1}\right)\right\}.
\end{aligned}$$

Using the bound $e^{1/r} \leq 1 + 1/r + 1/r^2$ for $r \geq 1$ we simplify

$$r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) - re^{1/r}\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) \geq r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) - r\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) - \widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) - \widetilde{d}_{\text{eff}}(\alpha\mathbf{L})/r \geq -\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) - \widetilde{d}_{\text{eff}}(\alpha\mathbf{L})/r$$

and obtain the final

$$\begin{aligned}
&\mathbf{P}(Acc_\sigma = \text{true}) \\
&\geq \exp\left\{d_{\text{eff}}(\alpha\widehat{\mathbf{L}}) - \widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) + d_{\text{eff}}(\alpha\mathbf{L}) - \operatorname{tr}\left(\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D})(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1}\right) - \widetilde{d}_{\text{eff}}(\alpha\mathbf{L})/r\right\}.
\end{aligned}$$

Using the definition of (ε, α)-accuracy, we have

$$\begin{aligned}
\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) &= \sum_{i=1}^{n} \alpha\varphi(i)^\top(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\varphi(i) \\
&= \operatorname{tr}\left(\alpha\varphi([n])(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\varphi([n])^\top\right) \\
&\leq \tfrac{1}{1-\varepsilon}\operatorname{tr}\left(\alpha\varphi([n])(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1}\varphi([n])^\top\right) \\
&= \tfrac{1}{1-\varepsilon}d_{\text{eff}}(\alpha\mathbf{L}) = (1+\tfrac{\varepsilon}{1-\varepsilon})d_{\text{eff}}(\alpha\mathbf{L}),
\end{aligned}$$

and therefore $-\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) + d_{\text{eff}}(\alpha\mathbf{L}) \geq \frac{\varepsilon}{1-\varepsilon}d_{\text{eff}}(\alpha\mathbf{L})$. On the other side

$$\begin{aligned}
\operatorname{tr}\left(\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D})(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1}\right) &\leq \tfrac{1}{1-\varepsilon}\operatorname{tr}\left(\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D})(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\right) \\
&= \tfrac{1}{1-\varepsilon}d_{\text{eff}}(\alpha\widehat{\mathbf{L}}) = (1+\tfrac{\varepsilon}{1-\varepsilon})d_{\text{eff}}(\alpha\widehat{\mathbf{L}}),
\end{aligned}$$

and therefore $\widetilde{d}_{\text{eff}}(\alpha\widehat{\mathbf{L}}) - \operatorname{tr}\left(\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D})(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1}\right) \geq \frac{\varepsilon}{1-\varepsilon}d_{\text{eff}}(\alpha\widehat{\mathbf{L}})$. Putting it all together, we obtain our result $\mathbf{P}(Acc_\sigma = \text{true}) \geq \exp\{-(\varepsilon(d_{\text{eff}}(\alpha\mathbf{L}) + d_{\text{eff}}(\alpha\widehat{\mathbf{L}})) + \widetilde{d}_{\text{eff}}(\alpha\mathbf{L})/r))\}$.

To bound $\varepsilon d_{\text{eff}}(\alpha\mathbf{L})$ we simply use the fact that the dictionary is $1/d_{\text{eff}}(\alpha\mathbf{L})$ accurate. Secondly to bound $\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})/r$ we use the fact that by Equation 2 and Proposition 5

$$\begin{aligned}
\widetilde{d}_{\text{eff}}(\alpha\mathbf{L}) = \sum_{i=1}^{n} l_i &= \sum_{i=1}^{n} \alpha\varphi(i)^\top(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\varphi(i) \\
&\leq \frac{1}{1-\varepsilon}\sum_{i=1}^{n} \alpha\varphi(i)^\top(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1}\varphi(i) = \frac{1}{1-\varepsilon}\sum_{i=1}^{n} \ell_i(\alpha\mathbf{L}) = \frac{d_{\text{eff}}(\alpha\mathbf{L})}{1-\varepsilon}.
\end{aligned}$$

Combining this with the fact that $\varepsilon \leq 1/2$ and that $r \geq d_{\text{eff}}(\alpha\mathbf{L})$ we have that $\widetilde{d}_{\text{eff}}(\alpha\mathbf{L})/r \leq 2$.

Finally, to bound $\varepsilon d_{\text{eff}}(\alpha\widehat{\mathbf{L}})$, first we bound

$$\begin{aligned}
d_{\text{eff}}(\alpha\widehat{\mathbf{L}}) &= \operatorname{tr}(\alpha\mathbf{W}^{1/2}\mathbf{L}_{\mathcal{D}}\mathbf{W}^{1/2}(\alpha\mathbf{W}^{1/2}\mathbf{L}_{\mathcal{D}}\mathbf{W}^{1/2}+\mathbf{I})^{-1}) \\
&= \operatorname{tr}(\alpha\mathbf{W}^{1/2}\varphi(\mathcal{D})\varphi(\mathcal{D})^\top\mathbf{W}^{1/2}(\alpha\mathbf{W}^{1/2}\varphi(\mathcal{D})\varphi(\mathcal{D})^\top\mathbf{W}^{1/2}+\mathbf{I})^{-1}) \\
&= \operatorname{tr}(\alpha\varphi(\mathcal{D})^\top\mathbf{W}^{1/2}\mathbf{W}^{1/2}\varphi(\mathcal{D})^\top(\alpha\varphi(\mathcal{D})^\top\mathbf{W}^{1/2}\mathbf{W}^{1/2}\varphi(\mathcal{D})^\top+\mathbf{I})^{-1}) \\
&= \operatorname{tr}(\alpha\mathbf{W}\varphi(\mathcal{D})^\top(\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D})^\top+\mathbf{I})^{-1}\varphi(\mathcal{D})^\top) \\
&= \sum_{j=1}^m [\mathbf{W}]_{j,j}\alpha\varphi(\mathcal{D}_j)^\top(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\varphi(\mathcal{D}_j) \\
&\leq \sum_{j=1}^m [\mathbf{W}]_{j,j}\tfrac{1}{1-\varepsilon}\alpha\varphi(\mathcal{D}_j)^\top(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1}\varphi(\mathcal{D}_j) = \sum_{j=1}^m [\mathbf{W}]_{j,j}\tfrac{1}{1-\varepsilon}\ell_{\mathcal{D}_j}(\alpha\mathbf{L})
\end{aligned}$$

where the last inequality used again Equation 2 and Proposition 5. To continue we have to use the following result for BLESS, the specific dictionary construction algorithm used by α-DPP, which follows immediately from Proposition 4 in Appendix C.

Proposition 3. *For some $\alpha' \geq \alpha$, let $\mathcal{D}$ be a dictionary generated using* BLESS-I *ran with parameter $q \geq 54\kappa^2\frac{(2\varepsilon+1)^2}{\varepsilon^2}\log(12n^2/\delta)$ and $\varepsilon \leq \min\{1/2, 1/d_{\text{eff}}(\alpha'\mathbf{L})\}$. Then w.p. $1-\delta$*

- *the dictionary and weights are (ε,α')-accurate,*
- *the weights $\mathbf{W}$ obtained satisfy $[\mathbf{W}]_{j,j} \leq \max\{\frac{1}{1-\varepsilon}\frac{1}{q\ell_{\mathcal{D}_j}(\alpha'\mathbf{L})}, 1\}$,*
- *the size of the dictionary $m = |\mathcal{D}|$ is bounded as $m/q \leq 2d_{\text{eff}}(\alpha'\mathbf{L})$.*

Applying this to the previous bound, and using the $(1/d_{\text{eff}}(\alpha\mathbf{L}),\alpha)$-accuracy, $\varepsilon \leq 1/2$ and the fact that $\ell_{\mathcal{D}_j}(\alpha\mathbf{L}) \leq \ell_{\mathcal{D}_j}(\alpha'\mathbf{L})$ for $\alpha' \geq \alpha$ we obtain

$$\begin{aligned}
\sum_{j=1}^m [\mathbf{W}]_{j,j}\tfrac{1}{1-\varepsilon}\ell_j(\alpha\mathbf{L}) &\leq \sum_{j=1}^m \max\{\tfrac{d_{\text{eff}}(\alpha'\mathbf{L})}{d_{\text{eff}}(\alpha'\mathbf{L})-1}\tfrac{1}{d_{\text{eff}}(\alpha'\mathbf{L})^2\ell_{\mathcal{D}_j}(\alpha'\mathbf{L})}, 1\}2\ell_{\mathcal{D}_j}(\alpha\mathbf{L}) \\
&= 2\sum_{j=1}^m \max\{\tfrac{d_{\text{eff}}(\alpha'\mathbf{L})}{d_{\text{eff}}(\alpha'\mathbf{L})-1}\tfrac{1}{d_{\text{eff}}(\alpha'\mathbf{L})^2}\tfrac{\ell_{\mathcal{D}_j}(\alpha\mathbf{L})}{\ell_{\mathcal{D}_j}(\alpha'\mathbf{L})}, \ell_{\mathcal{D}_j}(\alpha\mathbf{L})\} \\
&\leq 2\sum_{j=1}^m \max\{\tfrac{d_{\text{eff}}(\alpha'\mathbf{L})}{d_{\text{eff}}(\alpha'\mathbf{L})-1}\tfrac{1}{d_{\text{eff}}(\alpha'\mathbf{L})^2}, \ell_{\mathcal{D}_j}(\alpha\mathbf{L})\} \\
&\leq 2\sum_{j=1}^m \left(\tfrac{d_{\text{eff}}(\alpha'\mathbf{L})}{d_{\text{eff}}(\alpha'\mathbf{L})-1}\tfrac{1}{d_{\text{eff}}(\alpha'\mathbf{L})^2} + \ell_{\mathcal{D}_j}(\alpha\mathbf{L})\right).
\end{aligned}$$

To conclude, we have that since BLESS does not include duplicates in $\mathcal{D}$,

$$\sum_{j=1}^m \ell_{\mathcal{D}_j}(\alpha\mathbf{L}) \leq \sum_{i=1}^n \ell_i(\alpha\mathbf{L}) = d_{\text{eff}}(\alpha\mathbf{L}).$$

Now using the second result from Proposition 3 on m we have

$$\sum_{j=1}^m \tfrac{d_{\text{eff}}(\alpha'\mathbf{L})}{d_{\text{eff}}(\alpha'\mathbf{L})-1}\tfrac{1}{d_{\text{eff}}(\alpha'\mathbf{L})^2} = m\tfrac{d_{\text{eff}}(\alpha'\mathbf{L})}{d_{\text{eff}}(\alpha'\mathbf{L})-1}\tfrac{1}{d_{\text{eff}}(\alpha'\mathbf{L})^2} \leq 2d_{\text{eff}}(\alpha'\mathbf{L})\tfrac{d_{\text{eff}}(\alpha'\mathbf{L})}{d_{\text{eff}}(\alpha'\mathbf{L})-1}\tfrac{1}{d_{\text{eff}}(\alpha'\mathbf{L})^2} \leq 2.$$

■

B Proofs for the binary search algorithm

In this section we present omitted proofs for the binary search algorithm. The key properties of a Poisson Binomial which we will use are summarized in the following two lemmas.

Lemma 10. *Let $p : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a Poisson Binomial distribution. Then:*

1. *p is unimodal, i.e., if k^* is the mode of p, then $p(1) \leq ... \leq p(k^*) \geq p(k^*+1) \geq ...$;*
2. *p is log-concave, i.e., $log(p)$ is a concave function over the support of p;*
3. *the median of p is one of k^*-1, k^* and k^*+1.*

Lemma 11 ([9])**.** *Given a Poisson Binomial with mean $\bar{k}$ and mode k^*, let $k \stackrel{def}{=} \lfloor \bar{k} \rfloor$. Then:*

$$k^* = \begin{cases} k & \text{if} \quad k \leq \bar{k} < k + \frac{1}{k+2}, \\ k \ \text{ or } \ k+1 & \text{if} \quad k + \frac{1}{k+2} \leq \bar{k} \leq k + 1 - \frac{1}{n-k+1}, \\ k+1 & \text{if} \quad k + 1 - \frac{1}{n-k+1} < \bar{k} \leq k+1. \end{cases}$$

Note that these statements are independent of how we break ties in the definitions of the mode and the median, but for the sake of concreteness, suppose that we round down when choosing between a pair of (consecutive) mode/median candidates.

Lemma 3 (restated)**.** *Let $p : \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a Poisson Binomial distribution, and let $k \geq 1$ satisfy $p(k) < \frac{c}{12\sqrt{3(k+1)}}$, where c comes from Lemma 2. Then, $P_{<k} = \sum_{i<k} p(k)$ and $P_{>k} = \sum_{i>k} p(k)$ satisfy:*

1. *if the mode of p is less than k, then $P_{>k} \leq \frac{1}{2} - \frac{c}{12}$;*
2. *if the mode of p is greater than k, then $P_{<k} \leq \frac{1}{2} - \frac{c}{12}$.*

Proof of Lemma 3. Let k^* be the mode of p and let p^* denote $p(k^*)$. From Lemma 10 it follows that $k \neq k^*$. Suppose that $k^* > k$ (which implies that $k^* \geq 2$) and define:

$$t \stackrel{def}{=} \min\left\{ i \in \{1, ..., k^*\} \quad \text{subject to} \quad p(k^*-i) < \frac{p^*}{(1+\beta p^*)^i} \right\},$$

where $\beta = 2 + c/2.5$ is chosen so that the following inequalities (used later) hold: (a) $\frac{1}{\beta} \leq \frac{1}{2} - \frac{c}{12}$, (b) $\frac{1}{(1+\beta)^2} \geq \frac{1}{12}$ and (c) $\mathrm{e}^{\beta} \leq 12$. If no i exists satisfying the above constraint, then we let $t = k^*+1$ and use $p(-1) = 0$ for convenience. We consider two cases.

Case 1: $t \leq \lceil c/p^* \rceil + 1$. Since $p^* \geq \frac{c}{\sqrt{k^*+1}}$, it follows that $t \leq \left\lceil \sqrt{k^*+1} \right\rceil + 1$. Note that if $k > k^* - t$ then $k+1 \geq k^*+1 - \left\lceil \sqrt{k^*+1} \right\rceil \geq (k^*+1)/3$ and $p(k) \geq p^*(1+\beta p^*)^{-1/p^*} > \frac{c}{\mathrm{e}^{\beta}\sqrt{k^*+1}} \geq \frac{c}{12\sqrt{3(k+1)}}$ which is a contradiction, so we must have $k \leq k^* - t$. Furthermore, using the definition of t as well as unimodality and log-concavity of p, for any $i \geq t$ we have:

$$\frac{p(k^*-i+1)}{p(k^*-i)} \geq \frac{p(k^*-t+1)}{p(k^*-t)} \geq 1 + \beta p^*.$$

Thus, $p(k^*-i) < \frac{p^*}{(1+\beta p^*)^i}$ for all $i \geq t$ and it follows that:

$$P_{<k} \leq \sum_{i>t} p(k^*-i) \leq \frac{p^*}{(1+\beta p^*)^t} \sum_{i \geq 1} \frac{1}{(1+\beta p^*)^i} \leq \frac{p^*}{(1+\beta p^*)^t} \frac{1}{\beta p^*} \leq \frac{1}{\beta} \leq \frac{1}{2} - \frac{c}{12}.$$

Case 2: $t > \lceil c/p^* \rceil + 1$. This implies that for any $i \leq \lceil c/p^* \rceil$ we have $p(k^*-i) \geq p^*(1+\beta p^*)^{-c/p^*} > \frac{c}{12\sqrt{3(k+1)}} > p(k)$ so $k \leq k^* - \lceil c/p^* \rceil - 1$. Note that the median of p is no less than k^*-1 so:

$$P_{<k} \leq \underbrace{\sum_{i<k^*-1} p(i)}_{\leq 1/2} - \underbrace{\sum_{i=2}^{\lceil c/p^* \rceil + 1} p(k^*-i)}_{B}.$$

If $p^* \geq \frac{c}{2\beta}$, then it suffices to note that

$$B \geq p(k^*-2) \geq \frac{p^*}{(1+\beta p^*)^2} \geq \min\Big\{\frac{c/(2\beta)}{(1+c/2)^2}, \frac{1}{(1+\beta)^2}\Big\} \geq \frac{c}{12},$$

whereas if $p^* < \frac{c}{2\beta}$, then, we have:

$$B = p^*\bigg(\frac{1-(1+\beta p^*)^{-\lceil c/p^*\rceil-1}}{1-(1+\beta p^*)^{-1}} - 1 - (1+\beta p^*)^{-1}\bigg) = \frac{1}{\beta}\Big(1-(1+\beta p^*)^{-\lceil c/p^*\rceil}\Big) - \frac{p^*}{1+\beta p^*}$$
$$\geq \frac{1}{\beta}\big(1-2^{-\beta c}-c/2\big) \geq \frac{1}{\beta}\big(1-(1-\beta c/3)-c/2\big) = \frac{c}{3} - \frac{c}{2\beta} > \frac{c}{12},$$

which completes the proof when $k^* > k$, and the case of $k^* < k$ follows analogously. ■

We are now ready to establish the correctness of the binary search procedure that is used to prove Lemma 1, with pseudo-code given in Algorithm 2. In the proof we will use the following standard form of the Chernoff bound.

Lemma 12 (Chernoff bound)**.** *Let $X_1, ..., X_t$ be independent Bernoulli variables and let $\bar{X} = \frac{1}{t}\sum_i X_i$. Then, for any $0 < \epsilon \leq 1$, we have:*

$$\Pr\big(|\bar{X} - \mathbb{E}[\bar{X}]| \geq \epsilon \cdot \mathbb{E}[\bar{X}]\big) \leq 2e^{-\epsilon^2 t\,\mathbb{E}[\bar{X}]/3}.$$

Lemma 1 (restated)**.** *Suppose that we are given an integer k, a range $I = [\alpha_{\min}, \alpha_{\max}]$ where $\alpha_{\max} = \gamma\alpha_{\min}$, and access to an oracle which, for any $\alpha \in I$, returns $S \sim \mathrm{DPP}(\alpha\mathbf{L})$. If there exists $\alpha_\star \in I$ such that k is the mode of $|S|$ for $S \sim \mathrm{DPP}(\alpha_\star\mathbf{L})$, then using $O\big(\sqrt{k}\log^2(k\log(\gamma)/\delta)\big)$ calls to the oracle we can find $\hat{\alpha} \in I$ such that with probability $1-\delta$ we have*

$$\Pr(|S| = k) = \Omega\big(\tfrac{1}{\sqrt{k}}\big), \qquad \textit{for} \quad S \sim \mathrm{DPP}(\hat{\alpha}\mathbf{L}).$$

Proof of Lemma 1. Let $\mathrm{PB}(\alpha\mathbf{L})$ denote the size distribution of $\mathrm{DPP}(\alpha\mathbf{L})$. Since the binary search is performed in the log-scale, it takes at most $O(\log(k\log(\gamma)))$ steps to reduce the interval ratio $\frac{\alpha_{\max}}{\alpha_{\min}}$ from γ to $1 + \frac{1}{(k+3)^2}$. We first establish concentration of $\hat{P}_k$ around its mean $\mathbb{E}[\hat{P}_k] = \Pr(|S_1| = k) = p(k)$, where p is the probability function of $\mathrm{PB}(\bar{\alpha}\mathbf{L})$. Define $f(x) = \Pr(\bar{X} \geq q/2)$ where $\bar{X} = \frac{1}{t}\sum_{i=1}^t X_i$ and X_i are drawn i.i.d. from $\mathrm{Bernoulli}(x)$, with $q = \frac{c}{12\sqrt{3(k+1)}}$. Lemma 12 implies that, choosing a sufficiently large constant C in Algorithm 2, we have:

$$\max\big\{f(q/4),\, 1-f(q)\big\} \leq 2\mathrm{e}^{-tq/12} \leq \frac{\delta}{4s^2},$$

where s is the number of the current branching step. Note that if $p(k) > q$ then $\Pr(\hat{P}_k < q/2) \leq 1 - f(q) \leq \delta/(4s^2)$ whereas if $p(k) < q/4$, then $\Pr(\hat{P}_k \geq q/2) \leq f(q/4) \leq \delta/(4s^2)$, so putting this together we conclude that:

$$\Pr\Big(\big(\hat{P}_k \geq \tfrac{q}{2} \Rightarrow p(k) \geq \tfrac{q}{4}\big) \wedge \big(\hat{P}_k < \tfrac{q}{2} \Rightarrow p(k) < q\big)\Big)$$
$$= 1 - \Pr\Big(\big(\hat{P}_k \geq \tfrac{q}{2} \wedge p(k) < \tfrac{q}{4}\big) \vee \big(\hat{P}_k < \tfrac{q}{2} \wedge p(k) \geq q\big)\Big) \geq 1 - \frac{\delta}{4s^2}.$$

Thus, conditioning on the above high probability event ensures that when the **if** statement in Line 6 of Algorithm 2 succeeds then $\hat{\alpha}$ satisfies the condition from Lemma 1 because $p(k) \geq q/4 = \Omega(\frac{1}{\sqrt{k}})$, and when the if statement fails, then the assumption of Lemma 3 is satisfied because $p(k) < q$.

We now move on to the branching step of the binary search (Line 8). Our assumptions ensure that the initial interval $(\alpha_{\min}, \alpha_{\max})$ contains an $\alpha_\star$ such that k is the mode of $\mathrm{PB}(\alpha_\star\mathbf{L})$. Our goal is to show that the branching step preserves this invariant throughout the procedure. As discussed above, when entering the branching step, with high probability we have $p(k) < q$, so that we can use Lemma 3. Note that $\mathbb{E}[\hat{P}_{<k}] = P_{<k}$ and $\mathbb{E}[\hat{P}_{>k}] = P_{>k}$, as defined in the lemma, and the goal of the branching statement is to determine whether $P_{<k} > P_{<k}$, since that tells us on which side of k is the

mode of $\mathrm{PB}(\bar{\alpha}K)$. Conditioned on a high probability event, we know that either $P_{<k} \leq \frac{1}{2} - \frac{c}{12}$ or $P_{>k} \leq \frac{1}{2} - \frac{c}{12}$. Suppose the former holds. Then, we have:

$$P_{>k} = 1 - (P_{<k} + p(k)) \geq 1 - (\tfrac{1}{2} - \tfrac{c}{12} + \tfrac{c}{12\sqrt{3}}) \geq \tfrac{1}{2} + \tfrac{c}{30},$$

and an analogous bound follows for $P_{<k}$ in the latter case. If $P_{>k} \geq \frac{1}{2} + \frac{c}{30}$ (call it event E), then we can once again apply Lemma 12 to show that (for a sufficiently large constant C),

$$\begin{aligned}\Pr\big(\hat{P}_{<k} > \hat{P}_{>k} \mid E\big) &\geq \Pr\big(\hat{P}_{<k} \geq \tfrac{1}{2} \mid E\big) \geq \Pr\big(|\hat{P}_{<k} - \mathbb{E}[P_{<k}]| < \tfrac{c}{30} \mid E\big) \\ &\geq 1 - 2\exp\big\{-(\tfrac{c}{30})^2 \tfrac{1}{2} t/3\big\} \geq 1 - \delta/(4s^2),\end{aligned}$$

and an analogous claim follows when $P_{>k} \leq \frac{1}{2} - \frac{c}{12}$. Conditioning on this high probability event implies (via Lemma 3) that the interval constructed after branching still satisfies the invariant. A union bound implies that the probability that any of the events we have conditioned on fails (throughout the algorithm) is bounded by $\sum_{s\geq 1} \frac{2\delta}{4s^2} \leq \delta$. Thus, with probability $1-\delta$ the last interval used in the search will still satisfy the invariant. It remains to show that when the **if** statement in Line 2 succeeds then either $\alpha_{\min}$ or $\alpha_{\max}$ satisfies the claim of Lemma 1. To that end, since $k \geq \lfloor d_{\text{eff}}(\alpha_{\min}\mathbf{L})\rfloor$, we have:

$$\begin{aligned}d_{\text{eff}}(\alpha_{\max}\mathbf{L}) &< d_{\text{eff}}\big((1 + \tfrac{1}{(k+3)^2})\alpha_{\min}\mathbf{L}\big) \leq \big(1 + \tfrac{1}{(k+3)^2}\big) d_{\text{eff}}(\alpha_{\min}\mathbf{L}) \\ &\leq d_{\text{eff}}(\alpha_{\min}\mathbf{L}) + \frac{1}{\lfloor d_{\text{eff}}(\alpha_{\min}\mathbf{L})\rfloor + 3}.\end{aligned}$$

Now, there are two cases. Either $\lfloor d_{\text{eff}}(\alpha_{\max}\mathbf{L})\rfloor = \lfloor d_{\text{eff}}(\alpha_{\min}\mathbf{L})\rfloor$, in which case Lemma 11 immediately implies that there are at most two possible modes of the Poisson Binomial $\mathrm{PB}(\alpha\mathbf{L})$ among all values of $\alpha \in [\alpha_{\min}, \alpha_{\max}]$, and they must be achieved by $\alpha_{\min}$ and by $\alpha_{\max}$. If $\lfloor d_{\text{eff}}(\alpha_{\max}\mathbf{L})\rfloor = \lfloor d_{\text{eff}}(\alpha_{\min}\mathbf{L})\rfloor + 1$, then the same conclusion is reached by observing that:

$$d_{\text{eff}}(\alpha_{\max}\mathbf{L}) \leq \lfloor d_{\text{eff}}(\alpha_{\max}\mathbf{L})\rfloor + \frac{1}{\lfloor d_{\text{eff}}(\alpha_{\min}\mathbf{L})\rfloor + 3} \leq \lfloor d_{\text{eff}}(\alpha_{\max}\mathbf{L})\rfloor + \frac{1}{\lfloor d_{\text{eff}}(\alpha_{\max}\mathbf{L})\rfloor + 2},$$

so, by Lemma 11, the mode of $\mathrm{PB}(\alpha_{\max}\mathbf{L})$ must be $\lfloor d_{\text{eff}}(\alpha_{\max}\mathbf{L})\rfloor$, and once again there are only two possible modes in the interval $\alpha \in [\alpha_{\min}, \alpha_{\max}]$. With high probability, one of these modes must be k, which completes the proof. ■

C BLESS-I algorithm

In this section we present the omitted BLESS-I algorithm with proofs of its accuracy and efficiency. For simplicity, in the entirety of this section we will assume that $k \geq 2$. Note that this can be relaxed, at the only cost of slightly more complex constants (e.g., $\alpha_{\text{init}} = \max\{k-1, 1\}/\mathrm{tr}(\mathbf{L})$) instead of $\alpha_{\text{init}} = (k-1)/\mathrm{tr}(\mathbf{L})$). Moreover, the case $k = 1$ is qualitatively different, as in a 1-DPP the marginal and joint distribution coincide, making it much simpler to sample from.

C.1 BLESS

We begin by reporting the BLESS algorithm [37] and several of its properties. Note that BLESS was originally introduced as a ridge leverage score (RLS) sampling algorithm. However in the context of DPPs the RLS of an item coincides exactly with its marginal inclusion probability, i.e., $\ell_i(\mathbf{L})$ is the RLS of the i-th item. Therefore we can leverage any RLS sampler both to generate dictionaries as well as RLS estimate for α-DPP. We choose to use BLESS as a starting point because, to our knowledge, it is the only rescaling-aware RLS sampler existing in the literature. We report BLESS, in its rejection sampling version, in full in Algorithm 5 with the only notational difference of using a rescaling $\alpha \leq 1$ rather than a regularization λ, with a conversion $\alpha \approx 1/(\lambda n)$ between the two.

Proposition 4 (Thm. 1 by Rudi et al. [37]). *For some $\alpha' \geq \alpha$, let $\mathcal{D}$ be a dictionary generated using* BLESS *ran with parameter $q \geq 54\kappa^2 \frac{(2\varepsilon+1)^2}{\varepsilon^2} \log(12n^2/\delta)$. Then w.p. $1-\delta$ for all i,*

- *the dictionary $\mathcal{D}^i$ and weights are (ε, α^i)-accurate,*
- *the approximate marginals l_j computed using $\mathcal{D}^i$ satisfy $\frac{1}{1+\varepsilon}\ell_j(\alpha^i) \leq l_j \leq \frac{1}{1-\varepsilon}\ell_j(\alpha^i)$.*

Algorithm 5 BLESS (rejection-based version)

Input: $\mathbf{L} \in \mathbb{R}^{n\times n}$, $q > 0$, k, $\alpha_{\max}$

Initialize $i = 0$, $\alpha^0 = 1/\mathrm{tr}(\mathbf{L})$, $\widehat{d}_{\text{eff}}(\alpha^0\mathbf{L}) = \frac{1}{2}(k-1)$

Initialize $\mathcal{D}^0$ by sampling $q\alpha^0 n\kappa^2$ elements $\mathcal{D}^0 \overset{\text{i.i.d.}}{\sim} (1/n, \dots, 1/n)$ and weight $w_j^0 = 1/(q\alpha^0\kappa^2)$.

for $i = \{1, \dots, \lceil \log_2(\alpha_{\max}/\alpha^0)\rceil\}$ **do**

  Set $\alpha^i = 2\alpha^{i-1}$, $b^i = \min\{q\alpha^i\kappa^2, 1\}$

  for $j = \{1, \dots, n\}$ **do**

    Sample $u_j^i \sim \text{Bernoulli}(b^i)$

    if $u_j^i = 1$ **then**

      Compute l_j^i using Equation 1 and $\mathcal{D}^{i-1}$

      Sample $z_j^i \sim \text{Bernoulli}(\min\{ql_j, b^i\}/b^i)$

    end if

  end for

  Set $\sigma^i = \{j \in [n] : z_j^i = 1\}$, $\mathcal{D}^i = \sigma^i$, $w_j^i = 1/\min\{ql_{\sigma_j^i}, b^i\}$

end for

return $\mathcal{D}^{\alpha^{\lceil \log_2(\alpha_{\max}/\alpha^0)\rceil}}$

- *the size of the dictionary* $m^i = |\mathcal{D}^i|$ *is bounded as* $d_{\text{eff}}(\alpha^i\mathbf{L})/2 \le m/q \le 2d_{\text{eff}}(\alpha^i\mathbf{L})$,

and the algorithm runs in $\mathcal{O}\left((\min\{\alpha_{\max}n\kappa^2, 1\}d_{\text{eff}}(\alpha_{\max}\mathbf{L})^2\log(n/\delta)^3)\log(\alpha_{\max}\mathrm{tr}(\mathbf{L}))\right)$ *time.*

Note that all results presented in Proposition 4 are only reformulations of Theorem 1 from [37]. The only exception is the lower bound $m/q \ge d_{\text{eff}}(\alpha^i\mathbf{L})/2$, since the original BLESS analysis was only interested in showing that $m/q \le d_{\text{eff}}(\alpha^i\mathbf{L})/2$. However, the same concentration argument of Lemma 6 in [37] also holds for the lower bound we report here.

C.2 Modification to BLESS

In order to use BLESS in our approach for DPP sampling, we need to make a few modifications. Compared to BLESS, our BLESS-I (Algorithm 6):

- automatically computes an appropriate $\alpha_{\max}$ rather than taking it as input;
- introduces a novel α_{init} to initialize α^0 that both takes into account the desired DPP size k and is a valid lower bound for the interval search;
- automatically computes an appropriate $\alpha_{\min}$ rather than setting $\alpha_{\min} = \alpha^0$;
- uses the last $d_{\text{eff}}(\alpha_{\max}\mathbf{L})$ estimate to generate a dictionary $\mathcal{D}^{\alpha_{\max}}$ that is guaranteed to be $(1/d_{\text{eff}}(\alpha_{\max}\mathbf{L}), \alpha_{\max})$-accurate.

Lemma 5 (restated)**.** *For any matrix* $\mathbf{L}$ *and* $0 < \alpha \le 1$*, we have* $d_{\text{eff}}(\alpha\mathbf{L})/d_{\text{eff}}(\mathbf{L}) \ge \alpha \ge d_{\text{eff}}(\alpha\mathbf{L})/\mathrm{tr}(\mathbf{L})$.

Proof of Lemma 5. From the definition $d_{\text{eff}}(\alpha\mathbf{L}) = \mathrm{tr}(\alpha\mathbf{L}(\alpha\mathbf{L}+\mathbf{I})^{-1})$. Then the first half comes from

$$\mathrm{tr}(\alpha\mathbf{L}(\alpha\mathbf{L}+\mathbf{I})^{-1}) \le \mathrm{tr}(\alpha\mathbf{L}(\mathbf{I})^{-1}) = \alpha\mathrm{tr}(\mathbf{L}),$$

while for the second half we have

$$\mathrm{tr}(\alpha\mathbf{L}(\alpha\mathbf{L}+\mathbf{I})^{-1}) \ge \alpha\mathrm{tr}(\mathbf{L}(\mathbf{L}+\mathbf{I})^{-1}) = \alpha d_{\text{eff}}(\mathbf{L}).$$

■

Lemma 4 (restated)**.** *W.p.* $1-\delta$ BLESS-I *runs in time* $\widetilde{\mathcal{O}}\left(\min\{\alpha_{\max}\kappa^2, 1\}nk^6 + k^9\right)$ *and satisfies:*

1. *The interval* $[\alpha_{\min}, \alpha_{\max}]$ *is bounded by* $\frac{1}{4}(k-1)/\mathrm{tr}(\mathbf{L}) \le \alpha_{\min} \le \alpha_{\max} \le 8(k+2)/d_{\text{eff}}(\mathbf{L})$
2. *There is* $\alpha_\star \in [\alpha_{\min}, \alpha_{\max}]$ *for which* k *is the mode of* $|S|$ *where* $S \sim \mathrm{DPP}(\alpha_\star\mathbf{L})$*;*
3. *The dictionary* $\mathcal{D}^{\alpha_{\max}}$ *satisfies the conditions from Theorem 2 for any* $\alpha \in [\alpha_{\min}, \alpha_{\max}]$.

Algorithm 6 BLESS modified to compute the search interval (BLESS-I)

Input: $\mathbf{L} \in \mathbb{R}^{n\times n}$, $q > 0$, k

1: Initialize $i = 0$, $\alpha^0 = \alpha_{\text{init}} = (k-1)/(n\kappa^2)$, $\widehat{d}_{\text{eff}}(\alpha^0\mathbf{L}) = \frac{1}{2}(k-1)$

2: Initialize $\mathcal{D}^0$ by sampling $q\alpha^0 n\kappa^2$ elements $\mathcal{D}^0 \overset{\text{i.i.d.}}{\sim} (1/n, \dots, 1/n)$ and weight $w_j^0 = 1/(q\alpha^0\kappa^2)$.

3: **while** $\widehat{d}_{\text{eff}}(\alpha^i\mathbf{L}) \leq 2(k+2)$ **do**
4: Set $i = i+1$, $\alpha^i = 2\alpha^{i-1}$, $\alpha_{\max} = \alpha^i$, $b^i = \min\{q\alpha^i\kappa^2, 1\}$
5: **for** $j = \{1, \dots, n\}$ **do**
6: Sample $u_j^i \sim \text{Bernoulli}(b^i)$
7: **if** $u_j^i = 1$ **then**
8: Compute l_j^i using Equation 1 and $\mathcal{D}^{i-1}$
9: Sample $z_j^i \sim \text{Bernoulli}(\min\{ql_j, b^i\}/b^i)$
10: **end if**
11: **end for**
12: Set $\sigma^i = \{j \in [n] : z_j^i = 1\}$, $\mathcal{D}^i = \sigma^i$, $w_j^i = 1/\min\{ql_{\sigma_j^i}, 1\}$
13: set $\widehat{d}_{\text{eff}}(\alpha^i\mathbf{L}) = |\mathcal{D}^i|/q$
14: **if** $\widehat{d}_{\text{eff}}(\alpha^{i-1}\mathbf{L}) \leq \frac{1}{2}(k-1)$ and $\widehat{d}_{\text{eff}}(\alpha^i\mathbf{L}) > \frac{1}{2}(k-1)$ **then**
15: Set $\alpha_{\min} = \alpha^{i-1}$
16: **end if**

Computing approximate RLS.

17: **end while**
18: Set $\mathcal{D}^{\alpha_{\max}} = \emptyset$, $q' = q\widehat{d}_{\text{eff}}(\alpha^i\mathbf{L})^2$, $b^{\max} = \min\{q'\alpha^i\kappa^2, 1\}$
19: **for** $j = \{1, \dots, n\}$ **do**
20: Sample $u_j^{\max} \sim \text{Bernoulli}(b^{\max})$
21: **if** $u_j^{\max} = 1$ **then**
22: Compute $l_j^{\max}$ using Equation 1 and $\mathcal{D}^i$
23: Sample $z_j^{\max} \sim \text{Bernoulli}(\min\{q'l_j^{\max}, b^{\max}\}/b^{\max})$
24: If $z_j^{\max} = 1$, add j to $\mathcal{D}^{\alpha_{\max}}$ with weight $w_j^{\alpha_{\max}} = \frac{1}{\min\{q'l_j^{\max}, b^{\max}\}}$
25: **end if**
26: **end for**

Final dictionary construction.

27: **return** $\alpha_{\min}$, $\alpha_{\max}$, $\mathcal{D}^{\alpha_{\max}}$

Proof of Lemma 4. Throughout the proof we will make use of Proposition 4, in particular that $\frac{1}{2}d_{\text{eff}}(\alpha^i\mathbf{L}) \leq \widehat{d}_{\text{eff}}(\alpha^i\mathbf{L}) \leq 2d_{\text{eff}}(\alpha^i\mathbf{L})$. Note that by inverting the relationship we also have the reciprocal guarantee $\frac{1}{2}\widehat{d}_{\text{eff}}(\alpha^i\mathbf{L}) \leq d_{\text{eff}}(\alpha^i\mathbf{L}) \leq 2\widehat{d}_{\text{eff}}(\alpha^i\mathbf{L})$.

Claim (1): size of the interval. Applying Lemma 5 we have that $\alpha_{\max} \leq d_{\text{eff}}(\alpha_{\max}\mathbf{L})/d_{\text{eff}}(\mathbf{L})$. We need now to further upper bound $d_{\text{eff}}(\alpha_{\max}\mathbf{L})$ BLESS-I's terminating condition (Line 3) only guarantees the lower bound $\widehat{d}_{\text{eff}}(\alpha_{\max}\mathbf{L}) \geq 2(k+2)$. To this end we will use a property of RLS (see Lemma 3 from [37]) that says that if $\alpha^i > \alpha^{i-1}$ then $d_{\text{eff}}(\alpha^i\mathbf{L}) \leq \frac{\alpha^i}{\alpha^{i-1}} d_{\text{eff}}(\alpha^{i-1}\mathbf{L})$. In our case, $\alpha^i/\alpha^{i-1} = 2$ and $d_{\text{eff}}(\alpha^i\mathbf{L}) \leq 2d_{\text{eff}}(\alpha^{i-1}\mathbf{L})$. Now, let i be the index before the loop exit condition in Algorithm 6 is satisfied (i.e., $\alpha_{\max} = \alpha^{i+1}$). Then we have $\widehat{d}_{\text{eff}}(\alpha^i\mathbf{L}) \leq 2(k+2)$, using Proposition 4 we further bound $d_{\text{eff}}(\alpha^i\mathbf{L}) \leq 4(k+2)$, which implies that $d_{\text{eff}}(\alpha^{i+1}\mathbf{L}) = d_{\text{eff}}(\alpha_{\max}\mathbf{L}) \leq 8(k+2)$. Going back to our bound we obtain $\alpha_{\max} \leq d_{\text{eff}}(\alpha_{\max}\mathbf{L})/d_{\text{eff}}(\mathbf{L}) \leq 8(k+2)/d_{\text{eff}}(\mathbf{L})$.

The side of $\alpha_{\min}$ is much simpler. From Lemma 5 we have that $\alpha_{\min} \geq d_{\text{eff}}(\alpha_{\min}\mathbf{L})/\text{tr}(\mathbf{L})$, and from the algorithm we know that $\widehat{d}_{\text{eff}}(\alpha_{\min}\mathbf{L}) \geq \frac{1}{2}(k-1)$. Combining this with Proposition 4 we get

$$\alpha_{\min} \geq d_{\text{eff}}(\alpha_{\min}\mathbf{L})/\text{tr}(\mathbf{L}) \geq \tfrac{1}{2}\widehat{d}_{\text{eff}}(\alpha_{\min}\mathbf{L})/\text{tr}(\mathbf{L}) \geq \tfrac{1}{4}(k-1)/\text{tr}(\mathbf{L}).$$

Claim (2): validity of the interval. To begin, remember from Lemma 11 that the mode m_α of the sample size of $\text{DPP}(\alpha\mathbf{L})$ is bounded by $\lfloor d_{\text{eff}}(\alpha\mathbf{L})\rfloor \leq m_\alpha \leq \lfloor d_{\text{eff}}(\alpha\mathbf{L})\rfloor + 1$. To guarantee the validity of our interval, we show that $m_{\alpha_{\min}} \leq k$, and $m_{\alpha_{\max}} \geq k+1$. Due to the monotonicity of the mode of a Poisson Binomial distribution (see Lemma 10) we have therefore that starting from $m_{\alpha_{\min}}$ the mode increases with α, until it reaches k for some $\alpha_\star \in [\alpha_{\min}, \alpha_{\max}]$, and then continue increasing until it reaches $k+1 \leq m_{\alpha_{\max}}$.

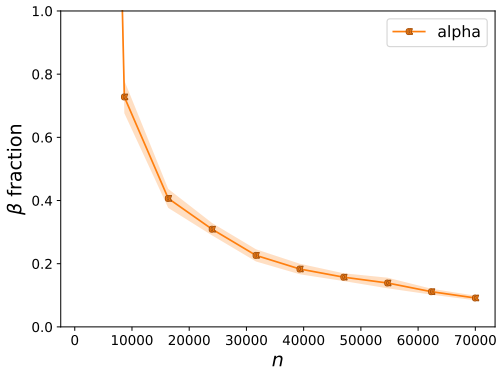

Figure 4: Fraction of items observed by α-DPP on the small scale experiment.

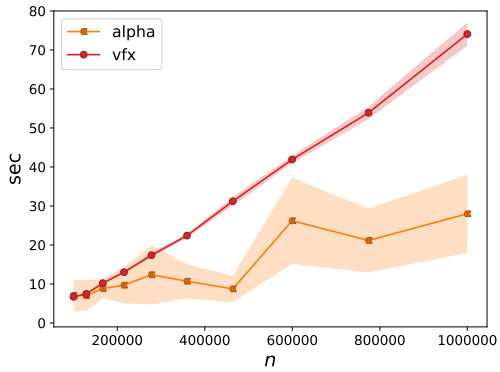

Figure 5: Large scale experiment using linear similarity.

Concretely, we have that

$$m_{\alpha_{\min}} \overset{\text{Lemma 11}}{\leq} d_{\text{eff}}(\alpha_{\min}\mathbf{L}) + 1 \overset{\text{Proposition 4}}{\leq} 2 \cdot \widehat{d}_{\text{eff}}(\alpha_{\min}\mathbf{L}) + 1 < 2 \cdot \tfrac{1}{2}(k-1) + 1 = k,$$

where the last inequality is due to the condition from Line 14 in BLESS-I. Similarly

$$m_{\alpha_{\max}} \overset{\text{Lemma 11}}{\geq} d_{\text{eff}}(\alpha_{\max}\mathbf{L}) - 1 \overset{\text{Proposition 4}}{\geq} \tfrac{1}{2} \cdot \widehat{d}_{\text{eff}}(\alpha_{\max}\mathbf{L}) - 1 > \tfrac{1}{2} \cdot 2(k+2) - 1 = k+1,$$

where this time the last inequality is due to the condition from Line 3 in BLESS-I. Finally, we have to guarantee that α_{init} is also a valid lower bound, or we will never able to correctly set $\alpha_{\min}$. This is easy to show using Lemma 5

$$m_{\alpha_{\text{init}}} \leq d_{\text{eff}}(\alpha_{\text{init}}\mathbf{L}) + 1 \overset{\text{Lemma 5}}{\leq} \alpha_{\text{init}}\text{tr}(\mathbf{L}) + 1 = \tfrac{\text{tr}(\mathbf{L})}{n\kappa^2}(k-1) + 1 \leq k - 1 + 1 = k,$$

making it a valid initialization for the lower bound.

Claim (3): quality of $\mathcal{D}^{\max}$. At the end of the main loop, due to Proposition 4 we have that $\widehat{d}_{\text{eff}}(\alpha^i\mathbf{L}) \geq \frac{1}{2}d_{\text{eff}}(\alpha^i\mathbf{L})$, and that since $\alpha^i = 2\alpha^{i-1} = 2\alpha_{\max}$, $d_{\text{eff}}(\alpha^i\mathbf{L}) \geq d_{\text{eff}}(\alpha_{\max}\mathbf{L})$. Therefore, setting $q' = 4\widehat{d}_{\text{eff}}(\alpha^i\mathbf{L})^2 q$ is sufficient to invoke Proposition 4 with $\varepsilon = 1/d_{\text{eff}}(\alpha_{\max}\mathbf{L})$ and obtain an $(1/d_{\text{eff}}(\alpha_{\max}), \alpha_{\max})$-accurate dictionary. Moreover, it is easy to see that for any $\alpha' \geq \alpha$ and $\varepsilon' \leq \varepsilon$, an (ε', α')-accurate dictionary is also an (ε, α)-accurate dictionary (see Proposition 5). Since $\alpha_{\max} \geq \alpha$ for the whole duration of the binary search, and therefore $d_{\text{eff}}(\alpha_{\max}\mathbf{L}) \geq d_{\text{eff}}(\alpha\mathbf{L})$, our $\mathcal{D}^{\max}$ dictionary is sufficiently accurate for the whole duration of the binary search. ■

D Additional experimental details

Both DPP-VFX and α-DPP rely on BLESS or BLESS-I to generate their input dictionaries. For this preprocessing phase, the major hyperparameters to tune are q_{BLESS} and q_{dpp}, i.e., the q and q' parameters indicated in Algorithm 6.[7]

Note that theory suggests to set $q_{\text{BLESS}} \approx \mathcal{O}(\log(n))$ and $q_{\text{dpp}} \approx \mathcal{O}(d_{\text{eff}}(\alpha\mathbf{L})^2)$, but they can be freely tuned since both α-DPP and DPP-VFX remain exact samplers for any hyperparameter choice. However, q_{BLESS} and q_{dpp} do impact acceptance rate and runtime, and even more importantly too low values can result in empty dictionaries which force the algorithm to be stopped.

In our case, we start with $q_{\text{BLESS}} = 2$ and $q_{\text{dpp}} = 2$, and increase them until the `DPPy` implementation does not return an empty dictionary. We also keep the same value for α-DPP and DPP-VFX so that for similar α they operate with similarly accurate and large dictionaries. The final values are $q_{\text{BLESS}} = 5$ and $q_{\text{dpp}} = 10$ for the small scale experiment (Figure 1), and $q_{\text{BLESS}} = 4$ and $q_{\text{dpp}} = 5$ for the large scale experiment (Figure 2).

[7] Following `DPPy`'s API, these hyperparameters are denoted as `rls_oversample_bless` and `rls_oversample_dppvfx` in our code.

For completeness, in addition to the fraction of observed items in the large scale experiment (Figure 3), we also report the fraction of observed items in the small scale experiment (Figure 4). We note that, for the small scale experiment, until n exceeds 10000, α-DPP is still observing all items, and only when the item collection becomes sufficiently large uniform sampling starts to play a role.

Finally, we report another experiment taken directly from the benchmark of Dereziński et al. [13] where a linear similarity is used instead of `rbf` similarity. We see that in this setting $d_{\text{eff}}(\mathbf{L})$ grows slower with n, since the similarity/kernel is less expressive. As a consequence the gap between α-DPP and DPP-VFX (i.e., the advantage of using uniform intermediate sampling) is reduced, but remains impactful.

E Miscellaneous proofs

In this section we present omitted miscellaneous facts and proofs for completeness.

Definition 2. *Given a psd matrix $\mathbf{L}$, its ith ridge leverage score (RLS) $\ell_i(\mathbf{L})$ is the ith diagonal entry of $\mathbf{L}(\mathbf{I}+\mathbf{L})^{-1}$. The sum $\sum_{i=1}^n \ell_i(\mathbf{L}) = d_{\text{eff}}(\mathbf{L})$ of the RLSs is equal to the effective dimension of $\mathbf{L}$.*

Definition 3 ([2, 5])**.** *A dictionary $\mathcal{D}$ and its associated weighting matrix $\mathbf{W}$ are (ε,α)-accurate if $\|\alpha\mathbf{L}(\mathbf{I}+\alpha\mathbf{L})^{-1}(\mathbf{I}-\overline{\mathbf{W}})\| \leq \varepsilon$, where $\overline{\mathbf{W}} \in \mathbb{R}^{n\times n}$ is diagonal with $\overline{\mathbf{W}}_{i,i} = \sum_{j=1}^m w_j \mathbb{I}\{\mathcal{D}_j = i\}$.*

Proposition 5 ([2, 5])**.** *A dictionary $\mathcal{D}$ and its associated weighting matrix $\mathbf{W}$ are (ε,α)-accurate if*

$$\|(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1/2}(\alpha\varphi([n])^\top\varphi([n]) - \alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1/2}\| \leq \varepsilon,$$

or equivalently

$$\|(\mathbf{I}/\alpha+\varphi([n])^\top\varphi([n]))^{-1/2}(\varphi([n])^\top\varphi([n]) - \varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))(\mathbf{I}/\alpha+\varphi([n])^\top\varphi([n]))^{-1/2}\| \leq \varepsilon,$$

or yet equivalently

$$(1-\varepsilon)(\mathbf{I}/\alpha+\varphi([n])^\top\varphi([n])) \preceq \mathbf{I}/\alpha+\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}) \preceq (1+\varepsilon)(\mathbf{I}/\alpha+\varphi([n])^\top\varphi([n])).$$

Note that using Proposition 5 it is easy to see that for any $\alpha' \geq \alpha$ and $\varepsilon' \leq \varepsilon$, an (ε',α')-accurate dictionary is also an (ε,α)-accurate dictionary since $\mathbf{I}/\alpha' \preceq \mathbf{I}/\alpha$ and therefore

$$\begin{aligned}
&\|(\mathbf{I}/\alpha+\varphi([n])^\top\varphi([n]))^{-1/2}(\varphi([n])^\top\varphi([n]) - \varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))(\mathbf{I}/\alpha+\varphi([n])^\top\varphi([n]))^{-1/2}\| \\
&\leq \|(\mathbf{I}/\alpha'+\varphi([n])^\top\varphi([n]))^{-1/2}(\varphi([n])^\top\varphi([n]) - \varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))(\mathbf{I}/\alpha'+\varphi([n])^\top\varphi([n]))^{-1/2}\| \\
&\leq \varepsilon' \leq \varepsilon.
\end{aligned}$$

Moreover, using basic algebraic manipulation we can see that for any matrix/operator $\mathbf{A}$ we have

$$(\mathbf{I}+\mathbf{A}\mathbf{A}^\top)^{-1} = \mathbf{I} - \mathbf{A}(\mathbf{I}+\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top,$$

which applied to $\mathbf{A} = \sqrt{\alpha}\mathbf{W}^{1/2}\varphi(\mathcal{D})$ gives us the following reformulation from [5, 37]:

$$\begin{aligned}
l_i &= \alpha[\mathbf{L} - \mathbf{L}_{[n],\mathcal{D}}^\top(\alpha\mathbf{L}_{\mathcal{D},\mathcal{D}} + \mathbf{W}^{-1})^{-1}\mathbf{L}_{[n],\mathcal{D}}]_{i,i} \\
&= \alpha[\varphi([n])\varphi([n])^\top - \alpha\varphi([n])\varphi(\mathcal{D})^\top(\alpha\varphi(\mathcal{D})\varphi(\mathcal{D})^\top + \mathbf{W}^{-1})^{-1}\varphi(\mathcal{D})\varphi([n])^\top]_{i,i} \\
&= \alpha[\varphi([n])\left(\mathbf{I} - \alpha\varphi(\mathcal{D})^\top(\alpha\varphi(\mathcal{D})\varphi(\mathcal{D})^\top + \mathbf{W}^{-1})^{-1}\varphi(\mathcal{D})\right)\varphi([n])^\top]_{i,i} \\
&= \alpha[\varphi([n])\left(\mathbf{I} - \alpha\varphi(\mathcal{D})^\top\mathbf{W}^{1/2}(\alpha\varphi(\mathcal{D})\mathbf{W}\varphi(\mathcal{D})^\top + \mathbf{I})^{-1}\mathbf{W}^{1/2}\varphi(\mathcal{D})\right)\varphi([n])^\top]_{i,i} \\
&= \alpha[\varphi([n])(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\varphi([n])^\top]_{i,i} \\
&= \alpha\varphi(i)^\top(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\varphi(i).
\end{aligned}$$

Applying Proposition 5 to the reformulation it is easy to see that

$$\begin{aligned}
&\alpha\varphi(i)^\top(\mathbf{I}+\alpha\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\varphi(i) = \varphi(i)^\top(\mathbf{I}/\alpha+\varphi(\mathcal{D})^\top\mathbf{W}\varphi(\mathcal{D}))^{-1}\varphi(i) \\
&\leq \tfrac{1}{1-\varepsilon}\varphi(i)^\top(\mathbf{I}/\alpha+\varphi([n])^\top\varphi([n]))^{-1}\varphi(i) = \tfrac{1}{1-\varepsilon}\alpha\varphi(i)^\top(\mathbf{I}+\alpha\varphi([n])^\top\varphi([n]))^{-1}\varphi(i) = \ell_i(\mathbf{L}).
\end{aligned}$$

Caching strategy. Note that if we invoke α-DPP multiple times for a fixed α, we do not need to recompute all approximations l_i from scratch each time. Rather, we first store an eigendecomposition

of $\widehat{\mathbf{L}}$ to be able to quickly compute $(\alpha \mathbf{L}_{\mathcal{D},\mathcal{D}} + \mathbf{W}^{-1})^{-1}$ in quadratic rather than cubic time. Then, for each item i we store a *cache* of the current upper bound, which is initialized to $\alpha\kappa^2$ and then lowered to l_i when l_i is actually computed. This way we never need to recompute the same l_i twice, and the runtime improves. In particular, computing a single marginal l_i requires $\mathcal{O}(k^6)$ time. So, if all l_i were computed from scratch, then the inner loop of Algorithm 1 would require $\alpha_{\max}\kappa^2 kn \cdot k^6$ to compute $\alpha_{\max}\kappa^2 kn$ marginals l_i, one for each item in ρ. On the other hand, computing all l_i for all items once and for all would require $n \cdot k^6$ time, and then sampling would be near-constant time using an appropriate multinomial sampler (see [13]). In our case, using the caching strategy we can get the best of both worlds $\widetilde{\mathcal{O}}(\min\{\alpha_{\max}\kappa^2 k, 1\} \cdot n \cdot k^6)$ since we never compute any l_i more than once.