Zonotope Hit-and-run for Efficient Sampling from Projection DPPs

Guillaume Gautier [1 2] **Rémi Bardenet** [1] **Michal Valko** [2]

Abstract

Determinantal point processes (DPPs) are distributions over sets of items that model diversity using kernels. Their applications in machine learning include summary extraction and recommendation systems. Yet, the cost of sampling from a DPP is prohibitive in large-scale applications, which has triggered an effort towards efficient approximate samplers. We build a novel MCMC sampler that combines ideas from combinatorial geometry, linear programming, and Monte Carlo methods to sample from DPPs with a fixed sample cardinality, also called projection DPPs. Our sampler leverages the ability of the hit-and-run MCMC kernel to efficiently move across convex bodies. Previous theoretical results yield a fast mixing time of our chain when targeting a distribution that is close to a projection DPP, but not a DPP in general. Our empirical results demonstrate that this extends to sampling projection DPPs, i.e., our sampler is more sample-efficient than previous approaches which in turn translates to faster convergence when dealing with costly-to-evaluate functions, such as summary extraction in our experiments.

1. Introduction

Determinantal point processes (DPPs) are distributions over configurations of points that encode diversity through a kernel function. DPPs were introduced by Macchi (1975) and have then found applications in fields as diverse as probability (Hough et al., 2006), number theory (Rudnick & Sarnak, 1996), statistical physics (Pathria & Beale, 2011), Monte Carlo methods (Bardenet & Hardy, 2016), and spatial statistics (Lavancier et al., 2015). In machine learning, DPPs over finite sets have been used as a model of diverse sets of items, where the kernel function takes the form of a finite matrix, see Kulesza & Taskar (2012) for a comprehensive survey. Applications of DPPs in machine learning (ML) since this survey also include recommendation tasks (Kathuria et al., 2016; Gartrell et al., 2017), text summarization (Dupuy & Bach, 2016), or models for neural signals (Snoek et al., 2013).

Sampling generic DPPs over finite sets is expensive. Roughly speaking, it is cubic in the number r of items in a DPP sample. Moreover, generic DPPs are sometimes specified through an $n \times n$ kernel matrix that needs diagonalizing before sampling, where n is the number of items to pick from. In text summarization, r would be the desired number of sentences for a summary, and n the number of sentences of the corpus to summarize. Thus, sampling quickly becomes intractable for large-scale applications (Kulesza & Taskar, 2012). This has motivated research on fast sampling algorithms. While fast exact algorithms exist for specific DPPs such as uniform spanning trees (Aldous, 1990; Broder, 1989; Propp & Wilson, 1998), generic DPPs have so far been addressed with approximate sampling algorithms, using random projections (Kulesza & Taskar, 2012), low-rank approximations (Kulesza & Taskar, 2011; Gillenwater et al., 2012; Affandi et al., 2013), or using Markov chain Monte Carlo techniques (Kang, 2013; Li et al., 2016a; Rebeschini & Karbasi, 2015; Anari et al., 2016; Li et al., 2016b). In particular, there are polynomial bounds on the mixing rates of natural MCMC chains with arbitrary DPPs as their limiting measure; see Anari et al. (2016) for cardinality-constrained DPPs, and Li et al. (2016b) for the general case.

In this paper, we contribute a non-obvious MCMC chain to approximately sample from *projection DPPs*, which are DPPs with a fixed sample cardinality. Leveraging a combinatorial geometry result by Dyer & Frieze (1994), we show that sampling from a projection DPP over a finite set can be relaxed into an easier continuous sampling problem with a lot of structure. In particular, the target of this continuous sampling problem is supported on the volume spanned by the columns of the feature matrix associated to the projection DPP, a convex body also called a *zonotope*. This zonotope can be partitioned into tiles that uniquely correspond to DPP realizations, and the relaxed target distribution is flat on each tile. Previous MCMC approaches to sampling projections DPPs can be viewed as attempting

[1]Univ. Lille, CNRS, Centrale Lille, UMR 9189 — CRIStAL [2]INRIA Lille — Nord Europe, SequeL team. Correspondence to: Guillaume Gautier <g.gautier@inria.fr>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

moves between neighboring tiles. Using linear programming, we propose an MCMC chain that moves more freely across this tiling. Our chain is a natural transformation of a fast mixing hit-and-run Markov chain (Lovász & Vempala, 2003) on the underlying zonotope; this empirically results in more uncorrelated MCMC samples than previous work. While the results of Anari et al. (2016) and their generalization by Li et al. (2016b) apply to projection DPPs, our experiments support the fact that our chain mixes faster.

The rest of the paper is organized as follows. In Section 2, we introduce projection DPPs and review existing approaches to sampling. In Section 3, we introduce zonotopes and we tailor the hit-and-run algorithm to our needs. In Section 4, we empirically investigate the performance of our MCMC kernel on synthetic graphs and on a summary extraction task, before concluding in Section 5.

2. Sampling Projections DPPs

In this section, we introduce projection DPPs in two equivalent ways, respectively following Hough et al. (2006), Kulesza & Taskar (2012), and Lyons (2003). Both definitions shed a different light on the algorithms in Section 3.

2.1. Projection DPPs as Particular DPPs

Let $E = [n] \triangleq \{1, \dots n\}$. Let also $\mathbf{K}$ be a real symmetric positive semidefinite $n \times n$ matrix, and for $I \subset E$, write $\mathbf{K}_I$ for the square submatrix of $\mathbf{K}$ obtained by keeping only rows and columns indexed by $I \subset E$. The random subset $X \subset E$ is said to follow a DPP on $E = \{1, \dots, n\}$ with kernel $\mathbf{K}$ if

$$\mathbb{P}\left[I \subset X\right] = \det \mathbf{K}_I, \quad \forall I \subset E. \tag{1}$$

Existence of the DPP described by (1) is guaranteed provided $\mathbf{K}$ has all its eigenvalues in $[0, 1]$, see e.g., Kulesza & Taskar (2012, Theorem 2.3). Note that (1) encodes the repulsiveness of DPPs. In particular, for any distinct $i, j \in [n]$,

$$\begin{aligned}\mathbb{P}\left[\{i, j\} \subset X\right] &= \begin{vmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ij} \\ \mathbf{K}_{ji} & \mathbf{K}_{jj} \end{vmatrix} \\ &= \mathbb{P}\left[\{i\} \in X\right] \mathbb{P}\left[\{j\} \in X\right] - \mathbf{K}_{ij}^2 \\ &\leq \mathbb{P}\left[\{i\} \in X\right] \mathbb{P}\left[\{j\} \in X\right].\end{aligned}$$

In other words, $\mathbf{K}_{ij}$ encodes departure from independence. Similarly, for constant $\mathbf{K}_{ii}, \mathbf{K}_{jj}$, the larger $\mathbf{K}_{ij}^2$, the less likely it is to have items i and j co-occur in a sample.

Projection DPPs are the DPPs such that the eigenvalues of $\mathbf{K}$ are either 0 or 1, that is, $\mathbf{K}$ is the matrix of an *orthogonal* projection. Projection DPPs are also sometimes called elementary DPPs (Kulesza & Taskar, 2012). One can show that samples from a projection DPP with kernel matrix $\mathbf{K}$ almost surely contain $r = \mathrm{Tr}(\mathbf{K})$ points and that general DPPs are mixtures of projection DPPs, see e.g., Kulesza & Taskar (2012, Theorem 2.3).

2.2. Building Projection DPPs from Linear Matroids

Let $r < n$, and let $\mathbf{A}$ be a full-rank $r \times n$ real matrix with columns $(a_j)_{j \in [n]}$. The linear matroid $M[\mathbf{A}]$ is defined as the pair $(E, \mathcal{B})$, with $E = [n]$ and

$$\mathcal{B} = \left\{ B \subset [n] : |B| = r, \{a_j\}_{j \in B} \text{ are independent} \right\}. \tag{2}$$

A set of indices $B \subset [n]$ is in $\mathcal{B}$ if and only if it indexes a basis of the columnspace of $\mathbf{A}$. Because of this analogy, elements of $\mathcal{B}$ are called *bases* of the matroid $M[\mathbf{A}]$. Note that elementary algebra yields that for all $B_1, B_2 \in \mathcal{B}$ and $x \in B_1 \setminus B_2$, there exists an element $y \in B_2 \setminus B_1$ such that

$$(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}. \tag{3}$$

Property (3) is known as the *basis-exchange* property. It is used in the definition of general matroids (Oxley, 2003).

Lyons (2003) defines a projection DPP as the probability measure on $\mathcal{B}$ that assigns to $B \in \mathcal{B}$ a mass proportional to $|\det \mathbf{B}|^2$, where $\mathbf{B} \triangleq \mathbf{A}_{:B}$ is the square matrix formed by the r columns of $\mathbf{A}$ indexed by B. Note that this squared determinant is also the squared volume of the parallelotope spanned by the columns indexed by B. In this light, sampling a projection DPP is akin to volume sampling (Deshpande & Rademacher, 2010). Finally, observe that the Cauchy-Binet formula gives the normalization

$$\sum_{B \in \mathcal{B}} |\det \mathbf{A}_{:B}|^2 = \det \mathbf{A}\mathbf{A}^\intercal,$$

so that the probability mass assigned to B is

$$\frac{\det \mathbf{A}^\intercal_{B:} \det \mathbf{A}_{:B}}{\det \mathbf{A}\mathbf{A}^\intercal} = \det \left[\mathbf{A}^\intercal \left[\mathbf{A}\mathbf{A}^\intercal\right]^{-1} \mathbf{A} \right]_B.$$

Letting

$$\mathbf{K} = \mathbf{A}^\intercal \left[\mathbf{A}\mathbf{A}^\intercal\right]^{-1} \mathbf{A}, \tag{4}$$

gives the equivalence between Sections 2.1 and 2.2.

A fundamental example of DPP defined by a matroid is the random set of edges obtained from a uniform spanning tree (Lyons, 2003). Let G be a connected graph with $r + 1$ vertices and n edges $\{e_i\}_{i \in [n]}$. Let now $\mathbf{A}$ be the first r rows of the vertex-edge incidence matrix of G. Then $B \subset [n]$ is a basis of $M[\mathbf{A}]$ if and only if $\{e_i\}_{i \in B}$ form a spanning tree of G (Oxley, 2003). The transfer current theorem of Burton & Pemantle (1993) implies that the uniform distribution on $\mathcal{B}$ is a projection DPP, with kernel matrix (4).

2.3. On Projection DPPs and k-DPPs in ML

Projection DPPs are DPPs with realizations of constant cardinality $k = r$, where r is the rank of $\mathbf{K}$. This constant cardinality is desirable when DPPs are used in summary extraction (Kulesza & Taskar, 2012; Dupuy & Bach, 2016) and the size of the required output is predefined. Another way of constraining the cardinality of a DPP is to condition on the event $|X| = k$, which leads to the so-called k-DPPs (Kulesza & Taskar, 2012). Projection DPPs and k-DPPs are in general different objects. In particular, a k-DPP is not a DPP in the sense of (1) unless its kernel matrix $\mathbf{K}$ is a projection. In that sense, k-DPPs are non-DPP objects that generalize projection DPPs. In this paper, we show that projection DPPs can benefit from fast sampling methods. It is not obvious how to generalize our algorithm to k-DPPs.

In ML practice, using projection DPPs is slightly different from using a k-DPP. In some applications, typically with graphs, the DPP is naturally a projection, such as uniform spanning trees described in Section 2.2. But quite often, kernels are built feature-by-feature. That is, for each data item $i \in [n]$, a normalized vector of features $\phi_i \in \mathbb{R}^r$ is chosen, a marginal relevance q_i is assigned to item i, and a matrix $\mathbf{L}$ is defined as

$$\mathbf{L}_{ij} = \sqrt{q_i}\phi_i\phi_j\sqrt{q_j}. \tag{5}$$

In text summarization, for instance, items i, j could be sentences, q_i the marginal relevance of sentence i to the user's query, and ϕ_i features such as tf-idf frequencies of a choice of words, and one could draw from a k-DPP associated to $\mathbf{L}$ through $\mathbb{P}[X = I] \propto \det \mathbf{L}_I$, see e.g., Kulesza & Taskar (2012, Section 4.2.1).

Alternately, let $\mathbf{A}$ be the matrix with columns $(\sqrt{q_i}\phi_i)_{i\in[r]}$, and assume $r < n$ and $\mathbf{A}$ is full-rank. The latter can be ensured in practice by adding a small i.i.d. Gaussian noise to each entry of $\mathbf{A}$. The projection DPP with kernel $\mathbf{K}$ in (4) will yield samples of cardinality r, almost surely, and such that the corresponding columns of $\mathbf{A}$ span a large volume, hence feature-based diversity. Thus, if the application requires an output of length p, one can pick $r = p$, as we do in Section A. Alternatively, if we want an output of size approximately p, we can pick $r \geq p$ and independently thin the resulting sample, which preserves the DPP structure (Lavancier et al., 2015).

2.4. Exact Sampling of Projection DPPs

Hough et al. (2006) give an algorithm to sample general DPPs, which is based on a subroutine to sample projection DPPs. Consider a projection DPP with kernel $\mathbf{K}$ such that $\mathrm{Tr}(\mathbf{K}) = r$, Hough et al.'s (2006) algorithm follows the chain rule to sample a vector $(x_1, \dots, x_r) \in [n]^r$ with successive conditional densities

$$p(x_{\ell+1} = i | x_1 = i_1, \dots, x_\ell = i_\ell) \propto \mathbf{K}_{ii} - \mathbf{K}_{i,I_\ell}\mathbf{K}_{I_\ell}^{-1}\mathbf{K}_{I_\ell,i},$$

where $I_\ell = \{i_1, \dots, i_\ell\}$. Forgetting order, $\{x_1, \dots, x_r\}$ are a draw from the DPP (Hough et al., 2006, Proposition 19), see also Kulesza & Taskar (2012, Theorem 2.3) for a detailed treatment of DPPs on $[n]$.

While exact, this algorithm runs in $\mathcal{O}(nr^3)$ operations and requires computing and storing the $n \times n$ matrix $\mathbf{K}$. Storage can be diminished if one has access to $\mathbf{A}$ in (4), through QR decomposition of $\mathbf{A}^\intercal$. Still, depending on n and r, sampling can become intractable. This has sparked interest in fast approximate sampling methods for DPPs, which we survey in Section 2.5.

Interestingly, there exist *fast* and *exact* methods for sampling some specific DPPs, which are not based on the approach of Hough et al. (2006). We introduced the DPP behind uniform spanning trees on a connected graph G in Section 2.2. Random walk algorithms such as the ones by Aldous (1990), Broder (1989), and Propp & Wilson (1998) sample uniform spanning trees in time bounded by the cover time of the graph, for instance, which is $\mathcal{O}(r^3)$ and can be $o(r^3)$ (Levin et al., 2009), where G has $r + 1$ vertices. This compares favorably with the algorithm of Hough et al. (2006) above, since each sample contains r edges. The Aldous-Broder algorithm, for instance, starts from an empty set $\mathcal{T} = \emptyset$ and an arbitrary node x_0, and samples a simple random walk $(X_t)_{t\in\mathbb{N}}$ on the edges of G, starting from $X_0 = x_0$, and adding edge $[X_t, X_{t+1}]$ to $\mathcal{T}$ the first time it visits vertex X_{t+1}. The algorithm stops when each vertex has been seen at least once, that is, at the cover time of the graph.

2.5. Approximate Sampling of Projection DPPs

There are two main sets of methods for approximate sampling from general DPPs. The first set uses the general-purpose tools from numerical algebra and the other is based on MCMC sampling.

Consider $\mathbf{K} = \mathbf{C}^\intercal\mathbf{C}$ with $\mathbf{C}$ of size $d \times n$, for some $d \ll n$ (Kulesza & Taskar, 2011), but still too large for exact sampling using the method of Hough et al. (2006), then Gillenwater et al. (2012) show how projecting $\mathbf{C}$ can give an approximation with bounded error. When this decomposition of the kernel is not possible, Affandi et al. (2013) adapt Nyström sampling (Williams & Seeger, 2001) to DPPs and bound the approximation error for DPPs and k-DPPs, which thus applies to projection DPPs.

Apart from general purpose approximate solvers, there exist MCMC-based methods for approximate sampling from projection DPPs. In Section 2.2, we introduced the *basis-exchange* property, which implies that once we remove an element from a basis B_1 of a linear matroid, any other basis B_2 has an element we can take and add to B_1 to make it a basis again. This means we can construct a connected

Algorithm 1 `basisExchangeSampler`

Input: Either $\mathbf{A}$ or $\mathbf{K}$
Initialize $i \leftarrow 0$ and pick $B_0 \in \mathcal{B}$ as defined in (2)
while Not converged **do**
  Draw $u \sim \mathcal{U}_{[0,1]}$
  if $u < \frac{1}{2}$ **then**
    Draw $s \sim \mathcal{U}_{B_i}$ and $t \sim \mathcal{U}_{[n]\setminus B_i}$
    $P \leftarrow (B_i \setminus \{s\}) \cup \{t\}$
    Draw $u' \sim \mathcal{U}_{[0,1]}$
    if $u' < \frac{\mathrm{Vol}^2(\mathbf{A}_{:P})}{\mathrm{Vol}^2(\mathbf{B}_i)+\mathrm{Vol}^2(\mathbf{A}_{:P})} = \frac{\det \mathbf{K}_P}{\det \mathbf{K}_{B_i}+\det \mathbf{K}_P}$ **then**
      $B_{i+1} \leftarrow P$
    else
      $B_{i+1} \leftarrow B_i$
    end if
  else
    $B_{i+1} \leftarrow B_i$
  end if
  $i \leftarrow i+1$
end while

graph G_{be} with $\mathcal{B}$ as vertex set, and we add an edge between two bases if their symmetric difference has cardinality 2. G_{be} is called the *basis-exchange graph*. Feder & Mihail (1992) show that the simple random walk on G_{be} has limiting distribution the uniform distribution on $\mathcal{B}$ and mixes fast, under conditions that are satisfied by the matroids involved by DPPs.

If the uniform distribution on $\mathcal{B}$ is not the DPP we want to sample from,[1] we can add an accept-reject step after each move to make the desired DPP the limiting distribution of the walk. Adding such an acceptance step and a probability to stay at the current basis, Anari et al. (2016); Li et al. (2016b) give precise polynomial bounds on the mixing time of the resulting Markov chains. This Markov kernel on $\mathcal{B}$ is given in Algorithm 1. Note that we use the acceptance ratio of Li et al. (2016b). In the following, we make use of the notation Vol defined as follows. For any $P \subset [n]$,

$$\mathrm{Vol}^2(\mathbf{A}_{:P}) \triangleq \det \mathbf{A}^\intercal{}_{P:}\mathbf{A}_{:P} \propto \det \mathbf{K}_P, \tag{6}$$

which corresponds to the squared volume of the parallelotope spanned by the columns of $\mathbf{A}$ indexed by P. In particular, for subsets P such that $|P| > r$ or such that $|P| = r$, $P \notin \mathcal{B}$ we have $\mathrm{Vol}^2(\mathbf{A}_{:P}) = 0$. However, for $B \in \mathcal{B}$, $\mathrm{Vol}^2(\mathbf{B}) = |\det \mathbf{A}_{:B}|^2 > 0$.

We now turn to our contribution, which finds its place in this category of MCMC-based approximate DPP samplers.

[1] It may not even *be* a DPP (Lyons, 2003, Corollary 5.5).

3. Hit-and-run on Zonotopes

Our main contribution is the construction of a fast-mixing Markov chain with limiting distribution a given projection DPP. Importantly, we assume to know $\mathbf{A}$ in (4).

Assumption 1. *We know a full-rank $r \times n$ matrix $\mathbf{A}$ such that $\mathbf{K} = \mathbf{A}^\intercal(\mathbf{A}\mathbf{A}^\intercal)^{-1}\mathbf{A}$.*

As discussed in Section 2.3, this is not an overly restrictive assumption, as many ML applications start with building the feature matrix $\mathbf{A}$ rather than the similarity matrix $\mathbf{K}$.

3.1. Zonotopes

We define the *zonotope* $\mathcal{Z}(\mathbf{A})$ of $\mathbf{A}$ as the r-dimensional volume spanned by the column vectors of $\mathbf{A}$,

$$\mathcal{Z}(\mathbf{A}) = \mathbf{A}[0,1]^n. \tag{7}$$

As an affine transformation of the unit hypercube, $\mathcal{Z}(\mathbf{A})$ is a r-dimensional polytope. In particular, for a basis $B \in \mathcal{B}$ of the matroid $M[\mathbf{A}]$, the corresponding $\mathcal{Z}(\mathbf{B})$ is a r-dimensional parallelotope with volume $\mathrm{Vol}(\mathbf{B}) = |\det \mathbf{B}|$, see Figure 1(a). On the contrary, any $P \subset [n]$, such that $|P| = r, P \notin \mathcal{B}$ also yields a parallelotope $\mathcal{Z}(\mathbf{A}_{:P})$, but its volume is null. In the latter case, the exchange move in Algorithm 1 will never be accepted and the state space of the corresponding Markov chain is indeed $\mathcal{B}$.

Our algorithm relies on the proof of the following.

Proposition 1 (see Dyer & Frieze, 1994 for details)**.**

$$\mathrm{Vol}(\mathcal{Z}(\mathbf{A})) = \sum_{B\in\mathcal{B}} \mathrm{Vol}(\mathbf{B}) = \sum_{B\in\mathcal{B}} |\det \mathbf{B}| \tag{8}$$

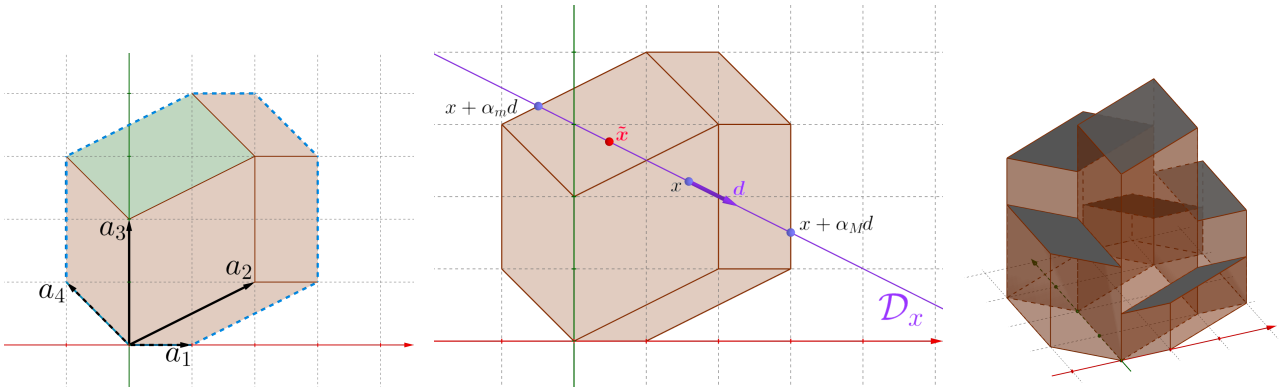
Proof. In short, for a good choice of $c \in \mathbb{R}^n$, Dyer & Frieze (1994) consider for any $x \in \mathcal{Z}(\mathbf{A})$, the following linear program (LP) noted $P_x(\mathbf{A}, c)$,

$$\begin{array}{ll} \min\limits_{y\in\mathbb{R}^n} & c^\intercal y \\ \text{s.t.} & \mathbf{A}y = x \\ & 0 \le y \le 1. \end{array} \tag{9}$$

Standard LP results (Luenberger & Ye, 2008) yield that the unique optimal solution y^* of $P_x(\mathbf{A}, c)$ takes the form

$$y^* = \mathbf{A}\xi(x) + \mathbf{B}_x u, \tag{10}$$

with $u \in [0,1]^r$ and $\xi(x) \in \{0,1\}^n$ such that $\xi(x)_i = 0$ for $i \in B_x$. In case the choice of B_x is ambiguous, Dyer & Frieze (1994) take the smallest in the lexicographic order. Decomposition (10) allows locating any point $x \in \mathcal{Z}(\mathbf{A})$ as falling inside a uniquely defined parallelotope $\mathcal{Z}(\mathbf{B}_x)$ shifted by $\xi(x)$. Manipulating the optimality conditions of (9), Dyer & Frieze (1994) prove that each basis B can be realized as a B_x for some x, and that $x' \in \mathcal{Z}(\mathbf{B}_x) \Rightarrow B_x =$



(a) (b) (c)

Figure 1. (a) The dashed blue lines define the contour of $\mathcal{Z}(\mathbf{A})$ where $\mathbf{A} = \left(\begin{smallmatrix} 1 & 2 & 0 & -1 \\ 0 & 1 & 2 & 1 \end{smallmatrix}\right)$. Each pair of column vectors corresponds to a parallelogram, the green one is associated to $\mathcal{Z}(\mathbf{B})$ with $B = \{2, 4\}$. (b) A step of hit-and-run on the same zonotope. (c) Representation of π_v for the same zonotope.

$B_{x'}$. This allows to write $\mathcal{Z}(\mathbf{A})$ as the tiling of all $\mathcal{Z}(\mathbf{B})$, $B \in \mathcal{B}$, with disjoint interiors. This leads to Proposition 1. □

Note that c is used to fix the tiling of the zonotope, but the map $x \mapsto B_x$ depends on this linear objective. Therefore, the tiling of $\mathcal{Z}(\mathbf{A})$ is may not be unique. An arbitrary c gives a valid tiling, as long as there are no ties when solving (9). Dyer & Frieze (1994) use a nonlinear mathematical trick to fix c. In practice (Section 4.1), we generate a random Gaussian c once and for all, which makes sure no ties appear during the execution, with probability 1.

Remark 1. *We propose to interpret the proof of Proposition 1 as a volume sampling algorithm: if one manages to sample an x uniformly on $\mathcal{Z}(\mathbf{A})$, and then extracts the corresponding basis $B = B_x$ by solving (9), then B is drawn with probability proportional to* $\mathrm{Vol}(\mathbf{B}) = |\det \mathbf{B}|$.

Remark 1 is close to what we want, as sampling from a projection DPP under Assumption 1 boils down to sampling a basis B of $M[\mathbf{A}]$ proportionally to the *squared* volume $|\det \mathbf{B}|^2$ (Section 2.2). In the rest of this section, we explain how to efficiently sample x uniformly on $\mathcal{Z}(\mathbf{A})$, and how to change the volume into its square.

3.2. Hit-and-run and the Simplex Algorithm

$\mathcal{Z}(\mathbf{A})$ is a convex set. Approximate uniform sampling on large-dimensional convex bodies is one of the core questions in MCMC, see e.g., Cousins & Vempala (2016) and references therein. The hit-and-run Markov chain (Turčin, 1971; Smith, 1984) is one of the preferred practical and theoretical solutions (Cousins & Vempala, 2016).

We describe the Markov kernel $P(x, z)$ of the hit-and-run Markov chain for a generic target distribution π supported on a convex set C. Sample a point y uniformly on the unit sphere centered at x. Letting $d = y - x$, this defines the line $\mathcal{D}_x \triangleq \{x + \alpha d \,;\, \alpha \in \mathbb{R}\}$. Then, sample z from any Markov kernel $Q(x, \cdot)$ supported on $\mathcal{D}_x$ that leaves the restriction of π to $\mathcal{D}_x$ invariant. In particular, Metropolis-Hastings kernel (MH, Robert & Casella 2004) is often used with uniform proposal on $\mathcal{D}_x$, which favors large moves across the support C of the target, see Figure 1(b). The resulting Markov kernel leaves π invariant, see e.g., Andersen & Diaconis (2007) for a general proof. Furthermore, the hit-and-run Markov chain has polynomial mixing time for log concave π (Lovász & Vempala, 2003, Theorem 2.1).

To implement Remark 1, we need to sample from $\pi_u \propto \mathbb{1}_{\mathcal{Z}(\mathbf{A})}$. In practice, we can choose the secondary Markov kernel $Q(x, \cdot)$ to be MH with uniform proposal on $\mathcal{D}_x$, as long as we can determine the endpoints $x + \alpha_m(y - x)$ and $x + \alpha_M(y - x)$ of $\mathcal{D}_x \cap \mathcal{Z}(\mathbf{A})$. In fact, zonotopes are tricky convex sets, as even an oracle saying whether a point belongs to the zonotope requires solving LPs (basically, it is Phase I of the simplex algorithm). As noted by Lovász & Vempala (2003, Section 4.4), hit-and-run with LP is the state-of-the-art for computing the volume of large-scale zonotopes. Thus, by definition of $\mathcal{Z}(\mathbf{A})$, this amounts to solving two more LPs: α_m is the optimal solution to the linear program

$$\begin{aligned} \min_{\lambda \in \mathbb{R}^n, \alpha \in \mathbb{R}} \quad & \alpha \\ \text{s.t.} \quad & x + \alpha d = \mathbf{A}\lambda \\ & 0 \le \lambda \le 1, \end{aligned} \tag{11}$$

while α_M is the optimal solution of the same linear program with objective $-\alpha$. Thus, a combination of hit-and-run and LP solvers such as Dantzig's simplex algorithm (Luenberger & Ye, 2008) yields a Markov kernel with invariant distribution $\mathbb{1}_{\mathcal{Z}(\mathbf{A})}$, summarized in Algorithm 2.

Algorithm 2 `unifZonoHitAndRun`

Input: $\mathbf{A}$
Initialization:
$i \leftarrow 0$
$x_0 \leftarrow \mathbf{A}u$ with $u \sim \mathcal{U}_{[0,1]^n}$
while Not converged **do**
  Draw $d \sim \mathcal{U}_{\mathbb{S}^{r-1}}$ and let $\mathcal{D}_{x_i} \triangleq x_i + \mathbb{R}d$
  Draw $\widetilde{x} \sim \mathcal{U}_{\mathcal{D}_{x_i} \cap \mathcal{Z}(A)}$ #Solve 2 LPs, see (11)
  $x_{i+1} \leftarrow \widetilde{x}$
  $i \leftarrow i + 1$
end while

Algorithm 3 `extractBasis`

Input: $\mathbf{A}, c, x \in \mathcal{Z}(\mathbf{A})$
Compute y^* the opt. solution of $P_x(\mathbf{A}, c)$ #1 LP, see (9)
$B \leftarrow \{i \,;\, y_i^* \in]0, 1[\}$
return B

The acceptance in MH is 1 due to our choice of the proposal and the target. By the proof of Proposition 1, running Algorithm 2, taking the output chain (x_i) and extracting the bases (B_{x_i}) with Algorithm 3, we obtain a chain on $\mathcal{B}$ with invariant distribution proportional to the volume of $\mathbf{B}$.

In terms of theoretical performance, this Markov chain inherits Lovász & Vempala's (2003) mixing time as it is a simple transformation of hit-and-run targeting the uniform distribution on a convex set. We underline that this is not a pathological case and it already covers a range of applications, as changing the feature matrix $\mathbf{A}$ yields another zonotope, but the target distribution on the zonotope stays uniform. Machine learning practitioners do not use volume sampling for diversity sampling yet, but nothing prevents it, as it already encodes the same feature-based diversity as squared volume sampling (i.e., DPPs). Nevertheless, our initial goal was to sample from a projection DPP with kernel $\mathbf{K}$ under Assumption 1. We now modify the Markov chain just constructed to achieve that.

3.3. From Volume to Squared Volume

Consider the probability density function on $\mathcal{Z}(\mathbf{A})$

$$\pi_v(x) = \frac{|\det \mathbf{B}_x|}{\det \mathbf{A}\mathbf{A}^\intercal} \mathbb{1}_{\mathcal{Z}(\mathbf{A})}(x),$$

represented on our example in Figure 1(c). Observe, in particular, that π_v is constant on each $\mathcal{Z}(\mathbf{B})$. Running the hit-and-run algorithm with this target instead of π_u in Section 3.2, and extracting bases using Algorithm 3 again, we obtain a Markov chain on $\mathcal{B}$ with limiting distribution $\nu(B)$ proportional to the squared volume spanned by column vectors of $\mathbf{B}$, as required. To see this, note that $\nu(B)$ is the volume of the "skyscraper" built on top of $\mathcal{Z}(\mathbf{B})$ in Figure 1(c), that is $\mathrm{Vol}(\mathbf{B}) \times \mathrm{Vol}(\mathbf{B})$.

Algorithm 4 `volZonoHitAndRun`

Input: $\mathbf{A}, c, x, B$
Draw $d \sim \mathcal{U}_{\mathbb{S}^{r-1}}$ and let $\mathcal{D}_x \triangleq x + \mathbb{R}d$
Draw $\widetilde{x} \sim \mathcal{U}_{\mathcal{D}_x \cap \mathcal{Z}(\mathbf{A})}$ #Solve 2 LPs, see (11)
$\widetilde{B} \leftarrow$ `extractBasis(`$\mathbf{A}, c, \widetilde{x}$`)` #Solve 1 LP, see (9)
Draw $u \sim \mathcal{U}_{[0,1]}$
if $u < \frac{\mathrm{Vol}(\widetilde{\mathbf{B}})}{\mathrm{Vol}(\mathbf{B})} = \left|\frac{\det \mathbf{A}_{:\widetilde{B}}}{\det \mathbf{A}_{:B}}\right|$ **then**
  return $\widetilde{x}, \widetilde{B}$
else
  return x, B
end if

Algorithm 5 `zonotopeSampler`

Input: $\mathbf{A}, c$
Initialization:
$i \leftarrow 0$
$x_i \leftarrow \mathbf{A}u$, with $u \sim \mathcal{U}_{[0,1]^n}$
$B_i \leftarrow$ `extractBasis(`$\mathbf{A}, c, x_i$`)`
while Not converged **do**
  $x_{i+1}, B_{i+1} \leftarrow$ `volZonoHitAndRun(`$\mathbf{A}, c, x_i, B_i$`)`
  $i \leftarrow i + 1$
end while

The resulting algorithm is shown in Algorithm 5. Note the acceptance ratio in the subroutine Algorithm 4 compared to Algorithm 2, since the target of the hit-and-run algorithm is not uniform anymore.

3.4. On Base Measures

As described in Section 2.3, it is common in ML to specify a marginal relevance q_i of each item $i \in [n]$, i.e., the *base measure* of the DPP. Compared to a uniform base measure, this means replacing $\mathbf{A}$ by $\widetilde{\mathbf{A}}$ with columns $\widetilde{a}_i = \sqrt{q_i} a_i$. Contrary to $\mathbf{A}$, in Algorithm 4, both the zonotope and the acceptance ratio are scaled by the corresponding products of $\sqrt{q_i}$s. We could equally well define $\widetilde{\mathbf{A}}$ by multiplying each column of $\mathbf{A}$ by q_i instead of its square root, and leave the acceptance ratio in Algorithm 4 use columns of the original $\mathbf{A}$. By the arguments in Section 3.3, the chain (B_i) would leave the same projection DPP invariant. In particular, we have some freedom in how to introduce the marginal relevance q_i, so we can choose the latter solution that simply scales the zonotope and its tiles to preserve outer angles, while using unscaled volumes to decide acceptance. This way, we do not create harder-to-escape or sharper corners for hit-and-run, which could lead the algorithm to be stuck for a while (Cousins & Vempala, 2016, Section 4.2.1). Finally, since hit-and-run is efficient at moving across convex bodies (Lovász & Vempala, 2003), the rationale is that if hit-and-run was empirically mixing fast before scaling, its performance should not decrease.

4. Experiments

We investigate the behavior of our Algorithm 5 on synthetic graphs in Section 4.1, in summary extraction in Section 4.2, and on MNIST in Appendix A.

4.1. Non-uniform Spanning Trees

We compare Algorithm 1 studied by Anari et al. (2016); Li et al. (2016b) and our Algorithm 5 on two types of graphs, in two different settings. The graphs we consider are the complete graph K_{10} with 10 vertices (and 45 edges) and a realization $\mathrm{BA}(20, 2)$ of a Barabási-Albert graph with 20 vertices and parameter 2. We chose BA as an example of structured graph, as it has the preferential attachment property present in social networks (Barabási & Albert, 1999). The input matrix $\mathbf{A}$ is a weighted version of the vertex-edge incidence matrix of each graph for which we keep only the 9 (resp. 19) first rows, so that it satisfies Assumption 1. For more generality, we introduce a base measure, as described in Section 2.3 and 3.4, by reweighting the columns of $\mathbf{A}$ with i.i.d. uniform variables in $[0, 1]$. Samples from the corresponding projection DPP are thus spanning trees drawn proportionally to the products of their edge weights.

For Algorithm 5, a value of the linear objective c is drawn once and for all, for each graph, from a standard Gaussian distribution. This is enough to make sure no ties appear during the execution, as mentioned in Section 3.1. This linear objective is kept fixed throughout the experiments so that the tiling of the zonotope remains the same. We run both algorithms for 70 seconds, which corresponds to roughly 50 000 iterations of Algorithm 5. Moreover, we run 100 chains in parallel for each of the two algorithms. For each of the 100 repetitions, we initialize the two algorithms with the same random initial basis, obtained by solving (9) once, with $x = \mathbf{A}u$ and $u \sim \mathcal{U}_{[0,1]^n}$. For both graphs, the total number $|\mathcal{B}|$ of bases is of order 10^8, so computing total variation distances is impractical. We instead compare Algorithms 1 and 5 based on the estimation of inclusion probabilities $\mathbb{P}[S \subset B]$ for various subsets $S \subset [n]$ of size 3. We observed similar behaviors across 3-subsets, so we display here the typical behavior on a 3-subset.

The inclusion probabilities are estimated via a running average of the number of bases containing the subsets S. Figures 2(a) and 3(a) show the behavior of both algorithms vs. MCMC iterations for the complete graph K_{10} and a realization of $\mathrm{BA}(20, 2)$, respectively. Figures 2(b) and 3(b) show the behavior of both algorithms vs. wall-clock time for the complete graph K_{10} and a realization of $\mathrm{BA}(20, 2)$, respectively. In these four figures, bold curves correspond to the median of the relative errors, whereas the frontiers of colored regions indicate the first and last deciles of the relative errors.

In Figures 2(c) and 3(c) we compute the Gelman-Rubin statistic (Gelman & Rubin, 1992), also called the potential scale reduction factor (PSRF). We use the PSRF implementation of CODA (Plummer et al., 2006) in R, on the 100 binary chains indicating the presence of the typical 3-subset in the current basis.

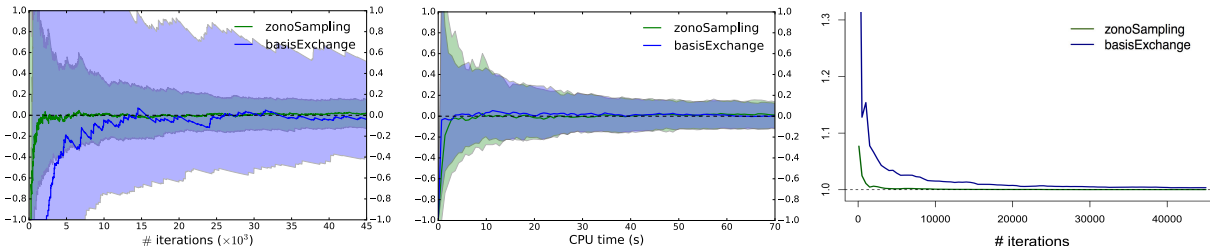
In terms of number of iterations, our Algorithm 5 clearly mixes faster. Relatedly, we observed typical acceptance rates for our algorithm an order of magnitude larger than Algorithm 1, while simultaneously attempting more global moves than the local basis-exchange moves of Algorithm 1. The high acceptance is partly due to the structure of the zonotope: the uniform proposal in the hit-and-run algorithm already favors bases with large determinants, as the length of the intersection of D_x in Algorithm 4 with any $\mathcal{Z}(\mathbf{B})$ is an indicator of its volume, see also Figure 1(b).

Under the time-horizon constraint, see Figures 2(b) and 3(b), Algorithm 1 has time to perform more than 10^6 iterations compared to roughly 50 000 steps for our chain. The acceptance rate of Algorithm 5 is still 10 times larger, but the time required to solve the linear programs at each MCMC iteration clearly hinders our algorithm in terms of CPU time. Both algorithms are comparable in performance, but given its large acceptance, we would expect our algorithm to perform better if it was allowed to do even only 10 times more iterations. Now this is implementation-dependent, and our current implementation of Algorithm 5 is relatively naive, calling the simplex algorithm in the GLPK (Oki, 2012) solver with CVXOPT (Andersen et al., 2008) from Python. We think there are big potential speed-ups to realize in the integration of linear programming solvers in our code. Moreover, we initialize our simplex algorithms randomly, while the different LPs we solve are related, so there may be additional smart mathematical speed-ups in using the path followed by one simplex instance to initialize the next.

Finally, we note that the performance of our Algorithm 5 seems stable and independent of the structure of the graph, while the performance of the basis-exchange Algorithm 1 seems more graph-dependent. Further investigation is needed to make stronger statements.

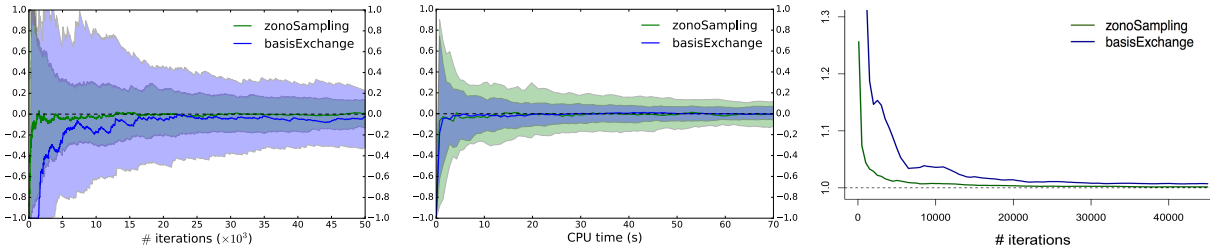
4.2. Text Summarization

Looking at Figures 2 and 3, our algorithm will be most useful when the bottleneck is mixing vs. number of iterations rather than CPU time. For instance, when integrating a costly-to-evaluate function against a projection DPP, the evaluation of the integrand may outweigh the cost of one iteration. To illustrate this, we adapt an experiment of Kulesza & Taskar (2012, Section 4.2.1) on minimum Bayes risk decoding for summary extraction. The idea is to find a



(a) Relative error vs. MCMC iterations. (b) Relative error vs. wall-clock time. (c) PSRF vs. MCMC iterations.

Figure 2. Comparison of Algorithms 1 and 5 on the complete graph K_{10}.



(a) Relative error vs. MCMC iterations. (b) Relative error vs. wall-clock time. (c) PSRF vs. MCMC iterations.

Figure 3. Comparison of Algorithms 1 and 5 on a realization of BA$(20, 2)$.

subset Y of sentences of a text that maximizes

$$\frac{1}{R}\sum_{r=1}^{R} \textsc{Rouge-1F}\left(Y, Y_r\right), \tag{12}$$

where $(Y_r)_r$ are sampled from a projection DPP. ROUGE-1F is a measure of similarity of two sets of sentences. We summarize this 64-sentence article as a subset of 11 sentences. In this setting, evaluating once ROUGE-1F in the sum (12) takes $0.1s$ on a modern laptop, while one iteration of our algorithm is $10^{-3}s$. Our Algorithm 5 can thus compute (12) for $R = 10\,000$ in about the same CPU time as Algorithm 1, an iteration of which costs $10^{-5}s$. We show in Figure 4 the value of (12) for 3 possible summaries $\left(Y^{(i)}\right)_{i=1}^{3}$ chosen uniformly at random in $\mathcal{B}$, over 50 independent runs. The variance of our estimates is smaller, and the number of different summaries explored is about 50%, against 10% for Algorithm 1. Evaluating (12) using our algorithm is thus expected to be closer to the maximum of the underlying integral. Details are given in Appendix B.

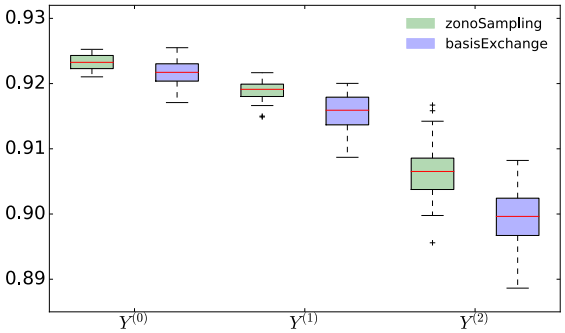


Figure 4. Summary extraction results

5. Discussion

We proposed a new MCMC kernel with limiting distribution being an arbitrary projection DPP. This MCMC kernel leverages optimization algorithms to help making global moves on a convex body that represents the DPP. We provided empirical results supporting its fast mixing when compared to the state-of-the-art basis-exchange chain of Anari et al. (2016); Li et al. (2016b). Future work will focus on an implementation: while our MCMC chain mixes faster, when compared based on CPU time our algorithm suffers from having to solve linear programs at each iteration. We note that even answering the question whether a given point belongs to a zonotope involves linear programming, so that chord-finding procedures used in slice sampling (Neal, 2003, Sections 4 and 5) would not provide significant computational savings.

We also plan to investigate theoretical bounds on the mixing time of our Algorithm 4. We can build upon the work of Anari et al. (2016), as our Algorithm 4 is also a weighted extension of our Algorithm 2, and the polynomial bounds for the vanilla hit-and-run algorithm (Lovász & Vempala, 2003) already apply to the latter. Note that while not targeting a DPP, our Algorithm 2 already samples items with feature-based repulsion, and could be used independently if the determinantal aspect is not crucial to the application.

Acknowledgments The research presented was supported by French Ministry of Higher Education and Research, CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020, and French National Research Agency projects EXTRA-LEARN (n.ANR-14-CE24-0010-01) and BOB (n.ANR-16-CE23-0003).

References

Affandi, R. H., Kulesza, A., Fox, E. B., and Taskar, B. Nyström approximation for large-scale determinantal processes. *International Conference on Artificial Intelligence and Statistics*, 31:85–98, 2013.

Aldous, D. J. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM Journal on Discrete Mathematics*, 3(4):450–465, 1990.

Anari, N., Gharan, S. O., and Rezaei, A. Monte-Carlo Markov chain algorithms for sampling strongly Rayleigh distributions and determinantal point processes. In *Conference on Learning Theory*, pp. 23–26, 2016.

Andersen, H. C. and Diaconis, P. W. Hit and run as a unifying device. *Journal de la Société Française de Statistique*, 148(4):5–28, 2007.

Andersen, M., Dahl, J., and Vandenberghe, L. CVXOPT: A python package for convex optimization, 2008.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *Science*, 286:11, 1999.

Bardenet, R. and Hardy, A. Monte-Carlo with determinantal point processes. *arXiv preprint arXiv:1605.00361*, 2016.

Broder, A. Generating random spanning trees. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pp. 442–447. IEEE, 1989.

Burton, R. and Pemantle, R. Local characteristics, entropy and limit theorems for spanning trees and domino tilings via transfer impedances. *The Annals of Probability*, 21: 1329–1371, 1993.

Cousins, B. and Vempala, S. A practical volume algorithm. *Mathematical Programming Computation*, 8(2): 133–160, 2016.

Deshpande, A. and Rademacher, L. Efficient volume sampling for row/column subset selection. In *Foundations of Computer Science*, 2010.

Dupuy, C. and Bach, F. Learning determinantal point processes in sublinear time. *arXiv preprint arXiv:1610.05925*, 2016.

Dyer, M. and Frieze, A. Random walks, totally unimodular matrices, and a randomised dual simplex algorithm. *Mathematical Programming*, 64(1-3):1–16, 1994.

Feder, T. and Mihail, M. Balanced matroids. *Proceedings of the twenty-fourth annual ACM*, pp. 26–38, 1992.

Gartrell, M., Paquet, U., and Koenigstein, N. Low-rank factorization of determinantal point processes for recommendation. In *AAAI Conference on Artificial Intelligence*, pp. 1912–1918, 2017.

Gelman, A. and Rubin, D. B. Inference from iterative simulation using multiple sequences. *Statist. Sci.*, 7(4):457–472, 11 1992.

Gillenwater, J., Kulesza, A., and Taskar, B. Discovering diverse and salient threads in document collections. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 710–720, 2012.

Hough, J. B., Krishnapur, M., Peres, Y., and Virág, B. Determinantal processes and independence. *Probability surveys*, 2006.

Kang, B. Fast determinantal point process sampling with application to clustering. In *Neural Information Processing Systems*, pp. 2319–2327, 2013.

Kathuria, T., Deshpande, A., and Kohli, P. Batched gaussian process bandit optimization via determinantal point processes. *Neural Information Processing Systems*, pp. pp. 4206–4214, 2016.

Kulesza, A. and Taskar, B. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2-3):123–286, 2012.

Kulesza, A. and Taskar, B. k-dpps: Fixed-size determinantal point processes. *International Conference on Machine Learning*, pp. 1193–1200, 2011.

Lavancier, F., Møller, J., and Rubak, E. Determinantal point process models and statistical inference. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 77(4):853–877, 2015.

Levin, D. A., Peres, Y., and Wilmer, E. L. *Markov chains and mixing times*. American Mathematical Soc., 2009.

Li, C., Jegelka, S., and Sra, S. Efficient sampling for k-determinantal point processes. In *Artificial Intelligence and Statistics*, pp. 1328–1337, 2016a.

Li, C., Jegelka, S., and Sra, S. Fast mixing markov chains for strongly rayleigh measures, dpps, and constrained sampling. In *Neural Information Processing Systems*, pp. 4188–4196, 2016b.

Liang, D. and Paisley, J. Landmarking manifolds with gaussian processes. In *International Conference on Machine Learning*, pp. 466–474, 2015.

Loper, E. and Bird, S. NLTK: The natural language toolkit. In *Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pp. 63–70, 2002.

Lovász, L. and Vempala, S. Hit and run is fast and fun. Technical Report MSR-TR-2003-05, 2003.

Luenberger, D. G. and Ye, Y. *Linear and nonlinear programming*. Springer, fourth edition, 2008.

Lyons, R. Determinantal probability measures. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 2003.

Macchi, O. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122, 1975.

Neal, R. M. Slice sampling. *Annals of statistics*, pp. 705–741, 2003.

Oki, E. Gnu linear programming kit, version 4.61. In *Linear Programming and Algorithms for Communication Networks - A Practical Guide to Network Design, Control, and Management*. 2012.

Oxley, J. What is a matroid? *Cubo Matemática Educacional*, 5.3:179–218, 2003.

Pathria, R. K. and Beale, P. D. *Statistical Mechanics*. 2011.

Plummer, M., Best, N., Cowles, K., and Vines, K. Coda: Convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7–11, 2006.

Propp, J. G. and Wilson, D. B. How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *Journal of Algorithms*, 27(2):170–217, 1998.

Rebeschini, P. and Karbasi, A. Fast mixing for discrete point processes. In *Conference on Learning Theory*, pp. 1480–1500, 2015.

Robert, C. P. and Casella, G. *Monte-Carlo Statistical Methods*. Springer-Verlag, New York, 2004.

Rudnick, Z. and Sarnak, P. Zeros of principal L-functions and random matrix theory. *Duke Mathematical Journal*, 81(2):269–322, 1996.

Smith, R. L. Efficient Monte-Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32:1296–1308, 1984.

Snoek, J., Zemel, R., and Adams, R. P. A determinantal point process latent variable model for inhibition in neural spiking data. In *Neural Information Processing Systems*, pp. 1932–1940, 2013.

Turčin, V. F. On the computation of multidimensional integrals by the monte-carlo method. *Theory of Probability & Its Applications*, 16(4):720–724, 1971.

Williams, C. and Seeger, M. Using the Nyström method to speed up kernel machines. In *Neural Information Processing Systems*, pp. 682–688, 2001.