The Journal of Open Source Software

PolytopeWalk: Sparse MCMC Sampling over Polytopes

Benny Sun[1] **and Yuansi Chen**[2]

1 Department of Statistics, Duke University **2** Department of Mathematics, ETH Zurich

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

High dimensional sampling is an important computational tool in statistics and other disciplines, with applications ranging from Bayesian statistical uncertainty quantification, metabolic modeling in systems biology, to volume computation. We present `PolytopeWalk`, a new scalable Python library designed for uniform sampling over polytopes. The library provides an end-to-end solution, that includes preprocessing algorithms such as facial reduction and initialization methods. Six state-of-the-art MCMC algorithms on polytopes are implemented, including the Dikin, Vaidya, and John Walk. Additionally, we introduce novel sparse constrained formulations of these algorithms, enabling efficient sampling from sparse polytopes of the form $\mathcal{K}_2 = \{x \in \mathbb{R}^d \mid Ax = b, x \succeq_k 0\}$. This implementation maintains sparsity in A, ensuring scalability to high dimensional settings $(d > 10^4)$. We also include novel implementation of several pre-processing steps, including the Facial Reduction algorithm. `PolytopeWalk` is available at github.com/ethz-randomwalk/polytopewalk with documentation at polytopewalk.readthedocs.io/.

Statement of Need

High dimensional sampling is a fundamental problem in many computational disciplines such as statistics, probability, and operation research. For example, sampling is applied in portfolio optimization (Calès et al. (2018)), metabolic networks in systems biology (Heirendt et al. (2018)) and volume approximation over convex shapes (Simonovits (2003)). Markov chain Monte Carlo (MCMC) sampling algorithms offer a natural and scalable solution to this problem. These algorithms construct a Markov chain whose stationary distribution matches the target distribution. By running the chain for a large number of steps to ensure mixing, MCMC algorithms can efficiently generate approximately independent samples close to the target distribution, while not suffering from the curse of dimension issues.

This package focuses on sampling from a uniform distribution over a user-specified polytope. We define the polytope as the following. Let $A \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$ and define $x \succeq_k y$ to mean that the last k-coordinates of x are greater than or equal to the corresponding coordinates of y, i.e., $\{x_{d-k+1} - y_{d-k+1} \geq 0, ..., x_d - y_d \geq 0\}$. Depending on whether we allow equality constraints, the sampling problem can be formalized in two forms:

1. The full-dimensional form:

$$\mathcal{K}_1 = \{x \in \mathbb{R}^d \mid Ax \leq b\}, \tag{1}$$

   where $\mathcal{K}_1$ is specified via n inequality constraints.

2. The constrained form:

$$\mathcal{K}_2 = \{x \in \mathbb{R}^d \mid Ax = b, x \succeq_k 0\}, \tag{2}$$

The Journal of Open Source Software

where $\mathcal{K}_2$ is specified via n equality constraints and k coordinate inequality constraints.

Large polytopes with sparse constraints are common in many applications. For instance, the largest human metabolic network RECON3D is modeled as a 13543-dimensional sparse polytope (King et al. (2015)). Additionally, linear programming datasets from `NetLib` are naturally in the constrained form, where A matrix is sparse. These applications motivate the need for MCMC algorithms that leverage the large and sparse $\mathcal{K}_2$ formulation. We develop novel interior-point-method-based MCMC algorithms optimized for large and sparse constrained polytopes. By exploiting sparsity, our algorithms scale well in both per-iteration cost and sampling efficiency as a function of increasing dimension, enabling effective sampling from polytopes with dimensions exceeding 10^4.

Interior-point-method-based MCMC sampling algorithms on a polytope are modifications of the Ball Walk (Vempala (2005)), incorporating key concepts from interior-point methods in optimization. These algorithms operate in two primary steps. First, the algorithm generates a proposal distribution whose covariance matrix is state-dependent and equal to the inverse of the Hessian matrix of a specified barrier function, capturing the local geometry of the polytope. Second, the algorithm employs the Metropolis-Hastings accept-reject step to ensure that its stationary distribution is uniform on the polytope. Using a state-dependent proposal distribution that adapts to the polytope's local geometry, these MCMC algorithms achieve an improved mixing rate. Specific algorithms in this class include the Dikin Walk (Sachdeva & Vishnoi (2015)), Vaidya Walk (Chen et al. (2018)), John Walk (Chen et al. (2018)), and Lee Sidford Walk (Laddha et al. (2019)), with theoretical guarantees therein. Each of these methods leverages different barrier functions and are specialized for sampling distributioned truncated on a polytope.

In `PolytopeWalk`, we implement 4 interior-point-method-based MCMC sampling algorithms in both the sparse, constrained formulation and the full-dimensional form. `PolytopeWalk` makes meaningful strides in the open-source development of MCMC, speeding up calculations for sparse high-dimensional sampling. On our Github, we demonstrate the scalability of `PolytopeWalk` against packages like `Volesti`. Finally, we provide an an open-source implementation of the Facial Reduction algorithm, used to handle degeneracy in polytopes.

Package Overview

`PolytopeWalk` is an open-source library written in C++ with Python wrapper code. There are 3 main components of the package: preprocessing, sampling, and post processing. `PolytopeWalk` provides accelerated MCMC sampling algorithms in both the $\mathcal{K}_1$ formulation~(1) and $\mathcal{K}_2$ formulation~(2). The source code is written in low-level C++ with `Eigen` for optimized linear algebra operations, `glpk` for a fast sparse linear programming solver, and `pybind` to enable Python binding.

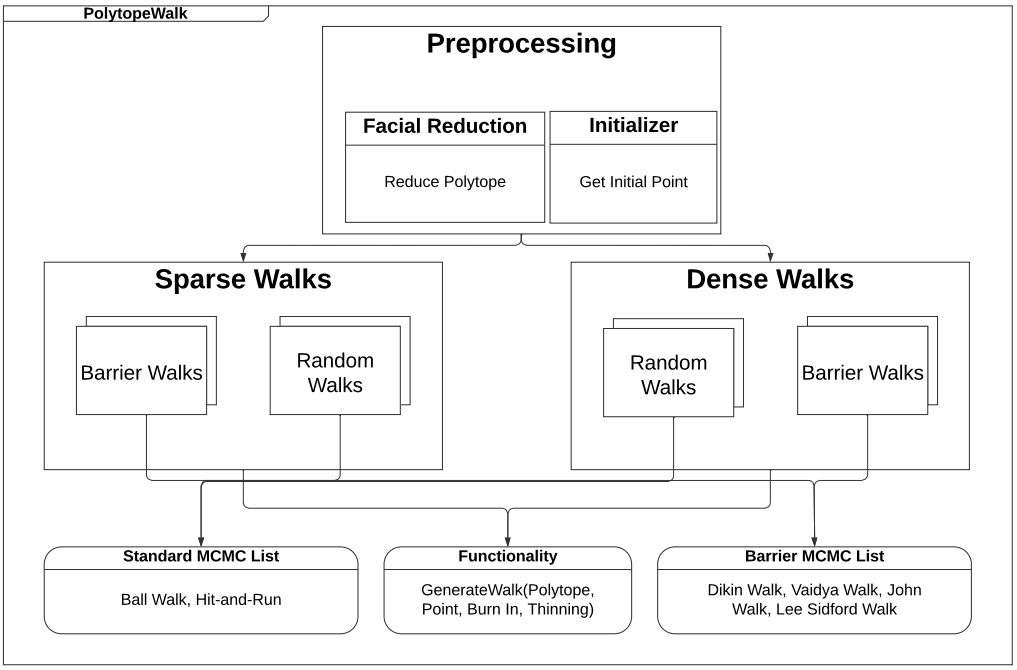

Figure 1: Code Structure of Package

Random Walk Algorithms

There are 2 main submodules: `dense` and `sparse`. The `dense` module includes implementations of the 6 random walks algorithms in the full-dimensional formulation. The `sparse` module includes implementations in the sparse, constrained formulation. Table I provides a broad overview of their respective mixing rates and cited paper.

Name	Mixing Time	Author
`Ball Walk`	$\tau(d^2R^2/r^2)$	Vempala (2005)
`Hit and Run`	$\tau(d^2R^2/r^2)$	Lovasz (1999)
`Dikin Walk`	$\tau(nd)$	Sachdeva et al. (2015)
`Vaidya Walk`	$\tau(n^{1/2}d^{3/2})$	Chen et al. (2018)
`John Walk`	$\tau(d^{2.5})$	Chen et al. (2018)
`Lee Sidford Walk`	$\tau(d^2)$	Laddha et al. (2019)

The mixing times refer to the required number of steps to converge to stationary distribution. In each, d refers to the dimension of the polytope and n refers to the number of boundaries ($\mathcal{K}_1$ dimensions). In the first 2 walks, R^2/r^2 means where the convex body contains a ball of radius r and is mostly contained in a ball of radius R.

Preprocessing Algorithms

`PolytopeWalk` comes with 2 preprocessing algorithms: initialization and facial reduction.

Initialization: If the user cannot specify a point inside of the polytope to start, `PolytopeWalk` provides a class to compute an initial point well within the polytope for both the full-dimensional formulation and constrained formulation. In $\mathcal{K}_2$, this is akin to solving the linear program

maximizing $\delta \in \mathbb{R}$ s.t. $Ax = b$ and $x \succeq_k \vec{\delta}$. In $\mathcal{K}_1$, this is equivalent to maximizing $\delta \in \mathbb{R}$ such that $Ax + \delta \cdot \mathbb{1} \leq b$ where $\mathbb{1}$ is the all-ones vector $\in \mathbb{R}^n$.

Facial Reduction: We adopt the facial reduction algorithm implementation from Drusvyatskiy's research (Drusvyatskiy et al. (2017)) (Im & Wolkowicz (2023)). The facial reduction algorithm deals with cases of degeneracy in the polytope. In the constrained formulation $\mathcal{K}_2 = \{x \in \mathbb{R}^d \mid Ax = b, x \succeq_k 0\}$, degeneracy occurs when there is a lack of strict feasibility in the polytope: there does not exist an $x \in \mathbb{R}^d$ such that $Ax = b$ and $x \succ_k 0$. The facial reduction algorithm eliminates variables in the last k dimensions fixed at 0. It is far more commonly used in linear programming than MCMC sampling. Degeneracy exists in polytopes when the lower-dimensional polytope is embedded in a higher dimension. For example, if one attempts to use MCMC algorithms to sample from a 2-dimensional hypercube in 5 dimensions, the proposal distributions which generate an ellipsoid in 5 dimensions will certainly reject each proposal. It is crucial that one projects the polytope back into the 2 dimensional form to run the MCMC sampler efficiently. Facial Reduction accomplishes this task, ensuring numerical stability for sampling.

Package Comparison

Feature	`PolytopeWalk`	`Volesti`	`WalkR`	`Polyrun`
Constrained Formulation	Y	N	Y	Y
Sparse Friendly	Y	N	N	N
C++ Implementation	Y	Y	N	N
Facial Reduction	Y	N	N	N
Dikin Walk	Y	Y	Y	N
Vaidya Walk	Y	Y	N	N
John Walk	Y	Y	N	N
Lee-Sidford Walk	Y	N	N	N

Table II contrasts the features of `PolytopeWalk` with `Volesti` (Chalkis & Fisikopoulos (2021)), `WalkR` (Yao & Kane (2017)), and `Polyrun` (Ciomek & Kadziński (2021)). `PolytopeWalk` is the first open-source package that enables users to leverage sparsity and use constrained formulations of the polytopes to generate points uniformly with fast MCMC algorithms. We are also one of the first to primarily focus on barrier walk MCMC samplers. `PolytopeWalk` includes a C++ implementation with corresponding Python wrapper code. Conversely, `Volesti` is implemented in C++ with some of its code represented in the Python library `Dingo`. `Polyrun` only works on Java and `WalkR` on R. Notably, `WalkR` was removed from the CRAN repository, motivating further open source MCMC sampling development.

Acknowledgements

Much of the work was done while Yuansi Chen was an assistant professor in the Department of Statistical Science at Duke University. Both authors are partially supported by NSF CAREER Award DMS-2237322, Sloan Research Fellowship and Ralph E. Powe Junior Faculty Enhancement Awards.

References

Calès, L., Chalkis, A., Emiris, I. Z., & Fisikopoulos, V. (2018). Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and financial crises. *CoRR*, *abs/1803.05861*. http://arxiv.org/abs/1803.05861

Chalkis, A., & Fisikopoulos, V. (2021). Volesti: Volume approximation and sampling for convex polytopes in r. *The R Journal*, *13*(2), 561. https://doi.org/10.32614/rj-2021-077

Chen, Y., Dwivedi, R., Wainwright, M. J., & Yu, B. (2018). Fast MCMC sampling algorithms on polytopes. *Journal of Machine Learning Research*, *19*(55), 1–86. http://jmlr.org/papers/v19/18-158.html

Ciomek, K., & Kadziński, M. (2021). Polyrun: A java library for sampling from the bounded convex polytopes. *SoftwareX*, *13*, 100659.

Drusvyatskiy, D., Wolkowicz, H., & others. (2017). The many faces of degeneracy in conic optimization. *Foundations and Trends® in Optimization*, *3*(2), 77–170.

Heirendt, L., Arreckx, S., Pfau, T., Mendoza, S. N., Richelle, A., Heinken, A., Haraldsdóttir, H. S., Wachowiak, J., Keating, S. M., Vlasov, V., Magnusdóttir, S., Ng, C. Y., Preciat, G., Žagare, A., Chan, S. H. J., Aurich, M. K., Clancy, C. M., Modamio, J., Sauls, J. T., … Fleming, R. M. T. (2018). *Creation and analysis of biochemical constraint-based models: The COBRA toolbox v3.0*. https://arxiv.org/abs/1710.04038

Im, H., & Wolkowicz, H. (2023). Revisiting degeneracy, strict feasibility, stability, in linear programming. *European Journal of Operational Research*, *310*(2), 495–510.

King, Z. A., Lu, J., Dräger, A., Miller, P., Federowicz, S., Lerman, J. A., Ebrahim, A., Palsson, B. O., & Lewis, N. E. (2015). BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research*, *44*(D1), D515–D522. https://doi.org/10.1093/nar/gkv1049

Laddha, A., Lee, Y. T., & Vempala, S. S. (2019). Strong self-concordance and sampling. *CoRR*, *abs/1911.05656*. http://arxiv.org/abs/1911.05656

Sachdeva, S., & Vishnoi, N. K. (2015). A simple analysis of the dikin walk. *CoRR*, *abs/1508.01977*. http://arxiv.org/abs/1508.01977

Simonovits, M. (2003). How to compute the volume in high dimension? *Mathematical Programming*, *97*. https://link.springer.com/article/10.1007/s10107-003-0447-x

Vempala, S. (2005). Geometric random walks: A survey. *Combinatorial and Computational Geometry*, *52*. https://faculty.cc.gatech.edu/~vempala/papers/survey.pdf

Yao, A. Y. Z., & Kane, D. (2017). Walkr: MCMC sampling from non-negative convex polytopes. *Journal of Open Source Software*, *2*(11), 61. https://doi.org/10.21105/joss.00061