Practical Polytope Volume Approximation

IOANNIS Z. EMIRIS, National and Kapodistrian University of Athens, Greece
VISSARION FISIKOPOULOS, Université libre de Bruxelles, Belgium

We experimentally study the fundamental problem of computing the volume of a convex polytope given as an intersection of linear halfspaces. We implement and evaluate randomized polynomial-time algorithms for accurately approximating the polytope's volume in high dimensions (e.g., few hundreds) based onhit-and-run random walks. To carry out this efficiently, we experimentally correlate the effect of parameters, such as random walk length and number of sample points, with accuracy and runtime. Our method is based on Monte Carlo algorithms with guaranteed speed and provably high probability of success for arbitrarily high precision. We exploit the problem's features in implementing a practical rounding procedure of polytopes, in computing only partial "generations" of random points, and in designing fast polytope boundary oracles. Our publicly available software is significantly faster than exact computation and more accurate than existing approximation methods. For illustration, volume approximations of Birkhoff polytopes $\mathcal{B}_{11}, \ldots, \mathcal{B}_{15}$ are computed, in dimensions up to 196, whereas exact methods have only computed volumes of up to $\mathcal{B}_{10}$.

Categories and Subject Descriptors: D.2.8 **[Software Engineering]:** Metrics—*Performance measures;* F.2.2 **[Analysis of Algorithms and Problem Complexity]:** Nonnumerical Algorithms and Problems—*Geometrical problems and computations;* G.3 **[Mathematics of Computing]:** Probability and Statistics—*Probabilistic algorithms (including Monte Carlo)*

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Volume approximation, general dimension, random walk, polytope oracle, algorithm engineering, open source software, birkhoff polytopes

ACM Reference format:
Ioannis Z. Emiris and Vissarion Fisikopoulos. 2018. Practical Polytope Volume Approximation. *ACM Trans. Math. Softw.* 44, 4, Article 38 (June 2018), 21 pages.
https://doi.org/10.1145/3194656

1 INTRODUCTION

A fundamental problem in optimization, discrete, and computational geometry and also Monte Carlo (MC) integration is to compute the volume of a convex body in general dimension or, more particularly, of a polytope. In the past 15 years, randomized algorithms for this problem have witnessed remarkable progress. Starting with the breakthrough polynomial-time algorithm of Dyer et al. (1991), subsequent results brought down the exponent on the dimension from 27 to 4 (Lovász

The first author is partially supported by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 734242 (LAMBDA).

Authors' addresses: I. Z. Emiris, Department of Informatics and Telecommunications, NKUA, Greece; email: emiris@di.uoa.gr; V. Fisikopoulos, Department of Computer Science, Université libre de Bruxelles CP 216, Boulevard du Triomphe, 1050 Brussels, Belgium; email: vissarion.fisikopoulos@gmail.com.

https://doi.org/10.1145/3194656

and Vempala 2006b), see Section 1.1. However, the question of an efficient implementation had remained open. We undertake this by showing that such an implementation is possible and accurate on a standard computer.

Preliminaries and Notation. A convex body S is abstractly represented by a *membership oracle*, a function that given S and a point returns whether the point lays in S. A more powerful oracle is the *boundary oracle*, that given S and a line returns the points where the line intersects the boundary of S, denoted ∂S.

As an important special case of a convex body, a polytope $P \subseteq \mathbb{R}^d$ can concretely be represented as the convex hull of a set V of points (V-polytope)

$$P := \text{conv}\{V\} = \{x \in \mathbb{R}^d \mid x = \sum_{i=1}^{|V|} \lambda_i v_i,\ \lambda_i \geq 0,\ \sum_{i=1}^{|V|} \lambda_i = 1\}.$$

Equivalently, P can be represented as the (bounded) intersection

$$P := \{x \in \mathbb{R}^d \mid Ax \leq b\},\ A \in \mathbb{R}^{m \times d}$$

of m halfspaces. The later is the case studied in this article.

The asymptotic notation $O^*(\cdot)$ hides poly-logarithmic factors in the argument. The input includes approximation factor $\epsilon > 0$; W denotes the most important runtime parameter, namely the random walk length. We denote R the ratio of the radius of the smallest enclosing ball of P over that of the largest enclosed ball in P.

1.1 Previous Work

Volume computation is #-P hard for V- and for H-polytopes (Dyer and Frieze 1988). Moreover, the volume of a rational polytope cannot be written in polynomial space (Lawrence 1991). The complexity of several computational problems related to volume computation of a polytope is surveyed in Khachiyan (1993) The computational complexity of the polytope volume computation is open if both representations are given (Büeler et al. 2000).

Several exact algorithms are surveyed in Büeler et al. (2000) and implemented in `VINCI` (Büeler and Enge 2000), and also in several other software packages like `Latte` (De Loera et al. 2013), `Normaliz` (Bruns et al. 2013), and Qhull (Barber et al. 1996), which, however, cannot handle general polytopes for dimension $d > 15$.

An interesting special case is the computation of the volume of the polytope of n by n doubly stochastic matrices known as n-Birkhoff polytope. In De Loera et al. (2008), they provide an explicit combinatorial formula for the volume. On the other hand, computing the volume of the n-Birkhoff polytope even for small values of n remains a computational challenge. This has been computed exactly for $n \leq 10$ using highly specialized software and high performance computing (Beck and Pixton 2003).

Regarding deterministic approximation, no polynomial-time algorithm can compute the volume of a convex body with less than exponential relative error (Elekes 1986). A stronger inapproximability result is proven in Bárány and Füredi (1987). The algorithm of Betke and Henk (1993) has error $\leq d!$. However, it is an interesting open question whether there exists a deterministic approximation algorithm for the special case of polytopes. Nevertheless, only randomized algorithms allow us to approximate the volume of a convex body to arbitrary accuracy in polynomial time. So we concentrate on them.

Let us examine the polynomial-time randomized algorithms. The landmark polynomial-time algorithm in Dyer et al. (1991) approximates the volume of a convex body with high probability and arbitrarily small relative error. We shall mostly follow the algorithm running in $O^*(d^5)$ membership calls (Kannan et al. 1997), which establishes Proposition 3.1 by a ball walk; the same complexity was later obtained by hit-and-run (Lovász 1999). All approaches up to this point transform the given object to near-isotropic position, then define a sequence of co-centric balls, and produce uniform point samples in their intersections with P. They use the ratios of the number of samples that fall in two bodies—defined by the intersection of P with two consecutive balls—to estimate the ratio of the volumes of the corresponding bodies. By multiplying all these ratios, we get an estimation of the volume of P. A slightly different approach is presented in Lovász and Vempala (2006b). They construct a sequence of log-concave functions and estimate ratios of integrals, instead of ratios of balls, using simulated annealing. The complexity reduces to $O^*(d^4)$ oracle calls by decreasing both number of phases and number of samples per phase to $O^*(\sqrt{d})$. Using hit-and-run, $O^*(d^3)$ calls still bound the time to sample each point. Moreover, they improve isoperimetric sandwiching to $O^*(d^4)$. While writing this article, a new algorithm with complexity $O^*(d^3)$ was announced with the additional assumption that the input polytope should be well-rounded (Cousins and Vempala 2015).

Concerning software for randomized algorithms, Cousins and Vempala (2016) presented recently `Matlab` code based on Lovász and Vempala (2006b) and Cousins and Vempala (2014). The latter offers a randomized algorithm for Gaussian volume (which has no direct reduction to or from volume) in $O^*(d^3)$, as a function of d. In Lovász and Deák (2012), they implement Lovász and Vempala (2006b), focusing on variance-decreasing techniques, and an empirical estimation of mixing time; they handle polytopes in dimensions up to 10 only. In Mohácsi and Deák (2015), they present a parallel version of this algorithm and computational results for convex bodies in dimensions ranging from 2 to 20. In Liu et al. (2007), they use a straightforward acceptance-rejection method, which is not expected to work in high dimensions; it was tested only for $d \leq 4$. An approach using thermodynamic integration (Jaekel 2011) offers only experimental guarantees on runtime and accuracy for $d \leq 80$.

The key ingredient of all approaches is *random sampling*, which is a fundamental problem of independent interest with important applications in, e.g., global optimization, statistics, machine learning, MC integration, and non-redundant constraint identification.

Several questions of sampling combinatorial structures such as contingency tables and, more generally, sampling lattice points in polytopes are related to sampling a polytope. These problems are hard, in general, in particular, sampling uniformly contingency tables when either the number of columns or rows are not constant is #P-hard (Dyer et al. 1997). There are many proposed methods in the bibliography for sampling contingency tables using Markov chain MC (Chen et al. 2005; Cryan and Dyer 2003; Diaconis and Sturmfels 1998). However, these methods work either for special cases or are only efficient in practice for small tables. Volume computation offer a potential tool for attacking these difficult problems (Kannan and Vempala 1997). The software developed here could be used as a tool toward a practical study in these topics.

Simple sampling methods are currently only known for bodies with standard shape, e.g., simplex, cube, or ellipsoid. Acceptance-rejection techniques are inefficient in high dimensions because the number of uniform points one needs to generate in a bounding box before finding one in P is exponential in d. A Markov chain is the only known method, and it may use geometric random walks such as the grid walk, the ball walk (or variants such as the Dikin walk (Kannan and Narayanan 2012), which uses Dikin ellipsoids instead of balls), and hit-and-run, see Simonovits (2003) for an overview. The Markov chain has to make a (large) number of steps, before the generated point becomes distributed approximately uniformly (which is the stationary limit distribution of the chain). We focus on hit-and-run, which yields the fastest algorithms today.

In contrast to other walks, hit-and-run is implemented by computing the intersection of a line with ∂P. In general, this reduces to binary search on the line, calling membership at every step. For

H-polytopes, the intersection is obtained by a *boundary oracle*; for this, we employ ray-shooting with respect to the m facet hyperplanes (Section 2). In exact form, it is possible to avoid linear-time queries by using space in $o(m^{\lfloor d/2 \rfloor})$, achieving queries in $O(\log m)$ (Ramos 1999). Duality may reduce boundary oracles to γ-nearest neighbor (NN) queries. The latter problem can be approximated in query time $O(dm^{(1+\gamma)^{-2}+o(1)})$ using $O(dm + m^{(1+\gamma)^{-2}+o(1)})$ space (Andoni and Indyk 2008). Alternatively, data depended methods achieve $O(1)$ time and $O(1/\gamma^{(d-O(1))})$ space (Arya et al. 2012). There is no asymptotic improvement by using the above methods but we examine their application experimentally.

Approximate membership or boundary oracles can be obtained by computing the exact oracle on an approximation of the polytope. Classic results, such as Dudley's, show that $O((1/\gamma)^{(d-1)/2})$ facets suffice to approximate a convex body of unit diameter within a Hausdorff distance of γ. This is optimized to $O(\sqrt{\text{vol}(\partial P)}/\gamma^{(d-1)/2})$ (Arya et al. 2012). In the case where the approximated convex body is itself a polytope, Reisner et al. (2001) study the approximation by a polytope whose vertices (or facets) are a subset of the vertices (or the facets) of the approximated polytope. The precision of the approximation is being measured using volume. The boundary oracle is dual to finding the extreme point in a given direction among a known pointset. This is γ-approximated through γ-coresets for measuring extent, in particular (directional) width, but requires a subset of $O((1/\gamma)^{(d-1)/2})$ points (Agarwal et al. 2005). The exponential dependence on d or the (super)linear dependence on m make all aforementioned methods of little practical use. Ray shooting has been studied in practice only in low dimensions, e.g., in six-dimensional V-polytopes (Zheng and Yamane 2013).

1.2 Contributions

We implement and experimentally study efficient algorithms for approximating the volume of polytopes. Our approach uses the iterative technique of co-centric balls intersecting a polytope P (cf. Section 3) introduced in Dyer et al. (1991). To define the sequence of concentric balls, we perform sandwiching. On the one hand, the algorithm computes the largest inscribed ball in P by a linear program. On the other hand, we need an enclosing ball; its computation shall also serve for rounding the given polytope. A crucial part of the volume algorithm—and of independent interest—is a random point sampler for polytopes. The sampler is used by the volume algorithm to generate random points in the intersection of a ball with P. Our C++ code is open-source on `github` and uses the `CGAL` and `Eigen` open-source libraries.

Our emphasis is to exploit the underlying characteristics of the problem to arrive at a practical and accurate method. Here we summarize the main pillars toward this goal as well as our results.

Scaling and Limits. A series of experiments establishes that our software handles dimensions substantially larger than existing exact approaches, e.g., cubes and products of simplices within an error of 2% for $d \leq 100$, in about 20 min. Our study indicates that it will run in dimensions 500 and 1000 in 6 and 60 days, respectively. These limits could be pushed even further by parallelization. In particular, we obtain a 5x speed-up in dimension 81 using 8 processor threads.

Random Walk Length. It is widely believed that the theoretical bound on random walk length W of hit-and-run is quite loose, and this is confirmed by our experiments, where we set $W = \Theta(d)$ and obtain a $<2\%$ error in up to 100 dimensions (Section 4).

Partial Generations of Random Points. The volume algorithm has to generate N random points in a sequence of convex bodies defines by a sequence of co-centric balls intersecting P. One main advantage of our method is that it creates partial "generations" of random points for every new body, as opposed to having always to generate N new points. Instead of increasing radii of the balls (as in theory, e.g., Dyer et al. (1991)), our method uses a sequence of concentric balls with *diminishing* radii, starting with one that (almost) encloses P and terminates with an inscribed ball. This allows us to "recycle" points generated in a larger ball but also lying in the smaller one, so as to sample only *partial generations* of points in the smaller intersection with P, instead of having to sample a full generation of N points per intersection. This has a significant effect on runtime since it reduces it by roughly a constant fraction of points per ball. Partial generations of points have been used in convex optimization (Bertsimas and Vempala 2004).

Rounding. Unlike most theoretical approaches that use an involved rounding procedure, we sample a set of points in P and compute an approximation of the minimum volume enclosing ellipsoid of this set, which is then linearly transformed to a ball. This procedure is repeated until the ratio of the minimum over the maximum ellipsoid axis reaches some user-defined threshold. This *iterative rounding* allows us to handle skinny polytopes efficiently, and takes few iterations in practice (see Section 4.3).

Boundary Oracle. We provide fast boundary oracles for sampling points (Section 2). For H-polytopes, the boundary oracle is in $O(md)$ by employing random direction hit-and-run (RDHR). We employ coordinate direction hit-and-run (CDHR), which further improves the oracle to $O(m)$ time complexity. Experiments show that, in practice, CDHR is much faster than RDHR without compromising a lot of accuracy. This difference in accuracy is diminished when the polytope has some symmetry (e.g., cubes or Birkhoff polytopes). See experiments in Section 4.

Polytope Database. We provide a database of polytopes to evaluate our implementation. The database contain polytope families for which it is very easy to evaluate the volume, like cubes, simplices, product of simplices, and families where this is difficult like the Birkhoff and order polytopes. Our software computes, in a few hours, volume estimations within an error of 2% for Birkhoff polytopes $\mathcal{B}_2, \ldots, \mathcal{B}_{10}$; $\text{vol}(\mathcal{B}_{10})$ has been exactly computed by specialized parallel software in a sequential time of years. More interestingly, it provides volume estimations for $\text{vol}(\mathcal{B}_{11}), \ldots, \text{vol}(\mathcal{B}_{15})$, of dimensions up to 196, whose exact values are unknown, within 9 hours. We exploit the symmetries of Birkhoff polytopes to compute efficiently more random points and thus imrpove the accuracy of our method.

In conclusion, our work shows that it is possible to approximate the volume of general H-polytopes in high dimensions (e.g., in the two hundreds) efficiently and accurately on standard computers today.

A preliminary version of most results from this article has appeared as Emiris and Fisikopoulos (2014).

1.3 Article Organization

The next section discusses walks and oracles. Section 3 presents the overall volume algorithm. Section 4 discusses our experiments, and we conclude with open questions in Section 5.

2 RANDOM WALKS AND ORACLES

In this section, we discuss theoretical and practical issues on polytope oracles and geometric random walks. Regarding polytope oracles we refer the reader to Grötschel et al. (1993). Geometric random walks are surveyed in Vempala (2005). Here we mainly focus on the hit-and-run paradigm (Smith 1984). The methods presented here are analysed experimentally in Section 4.

2.1 Hit-and-Run Random Walks

The main method to randomly sample a polytope is by (geometric) random walks. We shall focus on variants of hit-and-run that generate a uniform distribution of points (Smith 1984). Assume we possess procedure `Line`(p), which returns line ℓ through point $p \in P \subseteq \mathbb{R}^d$; ℓ will be specified below. The main procedure of hit-and-run is `Walk`(p, P, W) described below: it reads in point $p \in P$ and computes a new point $p' \in P$, which is meant to be uniformly distributed on P.

Procedure Walk(p, P, W)

```
Input: A point $p \in P$ and an integer $W$
Output: A point $p' \in P$
for $i$ from 1 to $W$ do
    $\ell \leftarrow$ Line($p$);
    move $p$ to a random point uniformly distributed on $P \cap \ell$;
end
return $p$;
```

We shall consider two variants of hit-and-run. In RDHR, `Line`(p) returns ℓ defined by a random vector uniformly distributed on the unit sphere centered at p. Equivalently, the vector coordinates are drawn from the standard normal distribution. In CDHR, `Line`(p) returns ℓ defined by a random vector uniformly distributed on the set $\{e_1, \dots, e_d\}$, of unit coordinate vectors $e_i = (0, \dots, 0, 1, 0, \dots, 0),\ i = 1, \dots, d$. This is a continuous variant of the grid walk; for the grid walk and other random walks see, e.g., Simonovits (2003).

Regarding mixing time, RDHR generates a uniformly distributed point in

$$O^*(d^3R^2) \text{ oracle calls with hidden constant } 10^{11}, \tag{1}$$

starting at an arbitrary point (Lovász and Vempala 2006a). Recall that R is the ratio of the radius of the smallest enclosing ball over that of the largest enclosed ball in P. As far as the authors know, the mixing time of CDHR has not been analysed. We offer experimental evidence that CDHR is faster than RDHR and sufficiently accurate so as to approximate volume accurately.

Procedure `Walk`(p, P, W) requires at every step a *boundary oracle*, which computes the intersection of line ℓ with ∂P. We discuss various implementations of this oracle.

2.2 Boundary Oracle by Facet Intersection

Given an H-polytope P the direct method to compute the intersection of line ℓ with ∂P is to examine all m hyperplanes. Let us consider `Walk` (p_0, P, W) and line $\ell = \{x \in \mathbb{R}^d : x = \lambda v + p_0, \lambda \in \mathbb{R}\}$, where point $p_0 \in \mathbb{R}^d$ lies on ℓ, and $v \in \mathbb{R}^d$ is the direction of ℓ. We compute the intersection of ℓ with the i-th hyperplane $a_i x = b_i$, $a_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$, namely

$$p_i := p_0 + \frac{b_i - a_i \cdot p_0}{a_i \cdot v} v,\ i \in \{1, 2, \dots, m\}.$$

We seek the intersection points p^+, p^- of ℓ with ∂P, namely

$$p^+ v = \min_{1 \le i \le m} \{p_i v \mid p_i v \ge 0\} \quad \text{and} \quad p^- v = \max_{1 \le i \le m} \{p_i v \mid p_i v \le 0\}.$$

This is computed in $O(md)$ arithmetic operations. In practice, only the $\lambda^\pm$ are computed, where $p^\pm = p_0 + \lambda^\pm v$.

In the context of the volume algorithm (Section 3), the intersection points of ℓ with ∂P are compared to the intersections of ℓ with the current sphere. Assuming the sphere is centered at the origin with radius R, its intersections with ℓ are $p = p_0 + \lambda v$ such that

$$\lambda^2 + 2\lambda p_0 v + |p_0|^2 - R^2 = 0. \tag{2}$$

If λ^+, λ^- give a negative sign when substituted to Equation (2), then p^+, p^- are the endpoints of the segment of ℓ lying in the intersection of P and the current ball. Otherwise, we have to compute one or two roots of Equation (2) since the segment has one or two endpoints on the sphere.

However, in CDHR, where ℓ and v are vertical, after the computation of the first pair p^+, p^-, all other pairs can each be computed in $O(m)$ arithmetic operations. This is because two sequential points produced by the walk differ only in one coordinate. Let j, k be the walk coordinates of the previous and the current steps, respectively. Then, assuming $P = \{x \in \mathbb{R}^d : Ax \leq b\}$, where $A \in \mathbb{R}^{m \times d}$, $\lambda^{\pm} = \max\{\lambda \mid A(p_0 \pm \lambda v) \leq b\}$. This becomes $\pm\lambda A v = \pm\lambda A_j \leq -Ap_0 + b$, where A_j is the j-th column of A. The two maximizations are solved in $O(md)$ operations: This is the bottleneck, namely computing vector $t = -Ap_0 + b \in \mathbb{R}^m$. At the next step, given point $p_0' = p_0 + ce_j$, where e_j is the j-th standard basis vector, and $c > 0$ is the (randomly chosen) walk length, we perform two maximizations of $\lambda : \pm\lambda A_k \leq t - cA_j$, in $O(m)$. In short, the oracle has $O(m)$ complexity, which has absorbed the cost of the first oracle that still takes $O(md)$. Hence the oracle incurs an amortized cost of $O(1)$ per hyperplane.

Remark 2.1. We compute random points as follows: starting from a point, we perform a hit-and-run walk and every W steps of the walk we keep the current point. Thus, we only compute one expensive boundary oracle computation of $O(md)$. After this, all oracle computations are in $O(m)$. Since, the number of sampled points is larger than the dimension of P, the amortized cost per boundary oracle computation is $O(m)$.

3 THE VOLUME ALGORITHM

This section details our polynomial-time methods for approximating the volume of P. Algorithms in this family can achieve any approximation ratio given by the user, i.e., they form a fully polynomial randomized approximation scheme (Simonovits 2003).

We consider that P is a full-dimensional H-polytope. If not, we can project it to a subspace in which it is full-dimensional. Suppose that P is lower-dimensional and given in the standard form $\{x \in \mathbb{R}^d \mid Ax = b,\ x \geq 0\}$, where $A \in \mathbb{R}^{d \times m}$, $b \in \mathbb{R}^m$. Using *Gauss-Jordan elimination,* the linear system $Ax = b$ can be transformed to its unique *reduced row echelon form* $[I|A']x = b' \in \mathbb{R}^m$, where I is the identity matrix and $A' \in \mathbb{R}^{d \times m-d}$. Then P can be written as $\{x' \in \mathbb{R}^{m-d} \mid A'x' \leq b'\}$, i.e., a full-dimensional H-polytope in $\mathbb{R}^{m-d}$.

3.1 Rounding and Sandwiching

This stage involves rounding P to reach a near isotropic position and, then, sandwiching, i.e., to compute ball B and scalar ρ such that $B \subseteq P \subseteq \rho B$. A convex body K is in θ-almost-isotropic position if for some $\theta \in (0,1)$, $||b(K)|| \leq \theta$ and for every $v \in \mathbb{R}^d$, we have

$$(1-\theta)||v||_2^2 \leq \frac{1}{\mathrm{vol}(K)} \int_{K-b(K)} (v^T x)^2 dx \leq (1+\theta)||v||_2^2,$$

where $b(K) := \int_K x dx$ is the center of gravity of K.

There is an abundance of methods in the literature for rounding and sandwiching (cf. Simonovits (2003) and references therein). However, here we develop a simple, efficient method that does not bring the polytopes into almost isotropic position but obtains approximate sufficiently accurate results experimentally in practice (cf. Section 4 and Table 4). The method does not compute a ball that covers P but a ball B' such that $B' \cap P$ contains almost all the volume of P. We sample a set S of $O(n)$ random points in P. Then we approximate the minimum volume ellipsoid $\mathcal{E}$ that covers S, and satisfies the inclusions $\frac{1}{(1+\varepsilon)d}\mathcal{E} \subseteq \mathrm{conv}(S) \subseteq \mathcal{E}$, in time $O(nd^2(\varepsilon^{-1} + \ln d + \ln\ln n))$ (Khachiyan

ALGORITHM 1: RoundingSandwiching(P, t_{round})

Input: H-polytope P, rounding threshold t_{round}
Output: rounded H-polytope P', center and radii of balls c, r, ρ, transformation matrix L_{prod}

```
compute the Chebychev ball $B(c, r)$;
generate a random point $p$ in $B(c, r)$;
$L_{prod} \leftarrow I$;  // $I$ is the identity matrix
repeat
    $S \leftarrow \emptyset$;
    for $i = 1$ to $N$ do
        $p \leftarrow$ Walk $(p, P, W)$;  // $W$ is the number of walk steps
        add $p$ in $S$;
    end
    compute min enclosing ellipsoid $\mathcal{E}$ of $S$, with p.s.d. $E$;
    set $\mathcal{E}min, \mathcal{E}max$ to be the min and max $\mathcal{E}$ axes;
    compute the Cholesky decomposition $L^T L$ of $E$;
    transform $P$ and $p$ w.r.t. $L$;
    $L_{prod} \leftarrow L_{prod} L$;
until $\mathcal{E}max/\mathcal{E}min < t_{round}$;
set $\rho$ the largest distance from $c$ to any point in $S$;
return $P, c, r, \rho, L_{prod}$ ;
```

1996). Let us write

$$\mathcal{E} = \{x \in \mathbb{R}^d \mid (x - c_{\mathcal{E}})^T\, E\, (x - c_{\mathcal{E}}) \leq 1\} \; = \{x \in \mathbb{R}^d \mid L^T (x - c_{\mathcal{E}}) \leq 1\}, \tag{3}$$

where $c_{\mathcal{E}}$ is the center of $\mathcal{E}$, $E \subseteq \mathbb{R}^{d \times d}$ is a positive semi-definite (p.s.d.) matrix and $E = L^T L$ its Cholesky decomposition. By substituting $x = (L^T)^{-1} y + c_{\mathcal{E}}$, we map the ellipsoid to the ball $\{y \in \mathbb{R}^d \mid y^T y \leq 1\}$. By applying this transformation to P, we obtain $P' = \{y \in \mathbb{R}^d \mid A(L^T)^{-1} y \leq b - Ac_{\mathcal{E}}\}$, which is the rounded polytope, and $\text{vol}(P) = \det(L)^{-1}\text{vol}(P')$. We iterate this procedure until the ratio of the minimum over the maximum ellipsoid axes reaches some user defined threshold denoted as t_{round}.

For sandwiching P, we first compute the *Chebychev ball* $B(c, r)$ of P, i.e., the largest inscribed ball in P. It suffices to solve the linear program: {maximize y, subject to $A_i x + R\|A_i\|_2 \leq b_i$, $i = 1, \ldots, m$, $y \geq 0$}, where A_i is the i-th row of A, and the optimal values of $y \in \mathbb{R}$ and $x \in \mathbb{R}^d$ yield, respectively, the radius r and the center c of the Chebychev ball.

Then we may compute a uniform random point in $B(c, r)$ and use it as a start to perform a random walk in P, eventually generating N random points. Now, compute the largest distance between any of the N points and c; this defines a (approximate) bounding ball $B(c, \rho)$. Finally, define the sequence of balls $B(c, 2^{i/d})$, $i = \alpha, \alpha + 1, \ldots, \beta$, where

$$\alpha = \lfloor d \log r \rfloor \text{ and } \beta = \lceil d \log \rho \rceil.$$

3.2 Multiphase Monte Carlo (MMC)

In this stage we construct a sequence of bodies $P_i := P \cap B(c, 2^{i/d})$, $i = \alpha, \alpha + 1, \ldots, \beta$, where $P_\alpha = B(c, 2^{\alpha/d}) \subseteq B(c, r)$ and P_β (almost) contains P. Then MMC approximates vol(P) by the telescopic product

$$\mathrm{vol}(P_\alpha)\prod_{i=\alpha+1}^{\beta}\frac{\mathrm{vol}(P_i)}{\mathrm{vol}(P_{i-1})},\ \text{ where }\ \mathrm{vol}(P_\alpha)=\frac{2\pi^{d/2}(2^{\lfloor\log r\rfloor})^d}{d\,\Gamma(d/2)},$$

the latter being the volume of the inscribed ball. This reduces to estimating the ratios $\mathrm{vol}(P_i)/\mathrm{vol}(P_{i-1})$, which is achieved by generating N uniformly distributed points in P_i and by counting how many of them fall in P_{i-1}. For point generation we use random walks. We set the walk length $W=\lfloor 10+d/10\rfloor=O(d)$, which is of the same order as in Lovász and Deák (2012) but significantly lower than theoretical bounds. This choice is corroborated experimentally (Section 4).

Unlike typical approaches, which generate points in P_i for $i=\alpha,\alpha+1,\dots,\beta$, here we proceed inversely, namely by generating points in P_i and counting how many fall within P_{i-1}. We start by generating an (almost) uniformly distributed random point $p\in P_\alpha$, which is easy since $P_\alpha = B(c,2^{\alpha/d})\subseteq B(c,r)$. Then we use p to start a random walk in P_β. Alternatively, we may start a random walk in $P_\alpha, P_{\alpha+1}, P_{\alpha+2}$ and so on, until we obtain a uniformly distributed point in P_β. In the second approach, the point computed in P_β is less biased with respect to $B(c,r)$.

Once we have a randomly distributed point in P_β, we initiate a random walk from this point using WN walk steps: every W steps, we output the current point. Such a random walk is called a *thread*, also employed in Lovász and Deák (2012). Algoritm 2 describes our method using a single thread. This way we generate N (almost) uniformly distributed points in P_β, then count how many of them fall into $P_{\beta-1}$. This yields an estimate of $\mathrm{vol}(P_\beta)/\mathrm{vol}(P_{\beta-1})$. We keep the points that lie in $P_{\beta-1}$, and use them to start walks so as to gather a total of N (almost) uniformly distributed points in $P_{\beta-1}$. We repeat until we compute the last ratio $\mathrm{vol}(P_{\alpha+1})/\mathrm{vol}(P_\alpha)$.

The implementation is based on a data structure S that stores the random points. In step $i>\alpha$, we wish to compute $\mathrm{vol}(P_{\beta-i})/\mathrm{vol}(P_{\beta-i-1})$ and S contains N random points in $P_{\beta-i+1}$ from the previous step. The computation in this step consists in removing from S the points not in $P_{\beta-i}$, then sampling $N-size(S)$ new points in $P_{\beta-i}$ and, finally, counting how many lie in $P_{\beta-i-1}$. Testing whether a point lies in some P_i reduces to testing whether $p\in B(2^{i/d})$ because $p\in P$.

One main advantage of our method is that it creates partial "generations" of random points for every new body P_i, as opposed to having always to generate N points. This has a significant effect on runtime since it reduces it by roughly a constant fraction of points per ball.

3.3 Complexity

The first $O^*(d^5)$ algorithm (Kannan et al. 1997) used a sequence of subsets defined as the intersection of the given body with a ball. It uses isotropic sandwiching to bound the number of balls by $O^*(d)$, it samples $N=400\epsilon^{-2}d\log d=O^*(d)$ points per ball, and follows a ball walk to generate each point with $O^*(d^3)$ oracle calls. Interestingly, both sandwiching and MMC each require $O^*(d^5)$ oracle calls. Later the same complexity was obtained by hit-and-run under the assumption the convex body is well sandwiched (Lovász 1999).

Proposition 3.1. (Kannan et al. 1997) *Assuming $B(0,1)\subseteq P\subseteq B(0,\rho)$, the algorithm of Kannan et al. (1997) returns an estimation of vol(P), which lies between $(1-\epsilon)$vol(P) and $(1+\epsilon)$vol(P), with probability $\geq 3/4$, by*

$$O\left(\frac{d^4\rho^2}{\epsilon^2}\ln d\ln\rho\ \ln^2\frac{d}{\epsilon}\right)=O^*(d^4\rho^2)$$

oracle calls with probability $\geq 9/10$, where we have assumed ϵ is fixed for the second bound. Sandwiching yields $\rho=\sqrt{d/\log(1/\epsilon)}$, implying a total of $O^*(d^5)$ calls.

The following lemma states the runtime of Algorithm 2, which is a variant of the algorithm analysed in Proposition 3.1. Although there is no theoretical bound on the approximation of the

ALGORITHM 2: VolEsti (P, ϵ, t_{round})

```
Input: H-polytope $P$ in dimension $d$, objective approximation $\epsilon$, rounding threshold $t_{round}$
Output: approximation of vol($P$)
$P, c, r, \rho, L \leftarrow RoundingSandwiching(P, t_{round})$;
$W \leftarrow \lfloor 10 + d/10 \rfloor$;
Generate the first random point $p$ in $P \cap B(c, \rho)$;
$S \leftarrow \{p\}$;
Set $\alpha \leftarrow \lfloor d \log r \rfloor$;   $\beta \leftarrow \lceil d \log \rho \rceil$;
$vol \leftarrow 2\pi^{d/2}(2^{\lfloor \log r \rfloor})^d / d\,\Gamma(d/2)$;
$i \leftarrow \beta$;
while $i > \alpha$ do
    Set $p$ to be an arbitrary point in $S$;
    $P_i \leftarrow P \cap B(c, 2^{i/d})$;
    $P_{\text{large}} \leftarrow P_i$;   $i \leftarrow i - 1$;   $P_{\text{small}} \leftarrow P_i$;
    $count\_prev \leftarrow size(S)$;   remove from $S$ the points not in $P_{\text{small}}$;   $count \leftarrow size(S)$;
    for $j = 1$ to $N - count\_prev$ do
        $p \leftarrow Walk(p, P_{\text{large}}, W)$;
        if $p \in B(c, 2^{i/d})$ then
            $count \leftarrow count + 1$;
            add $p$ in $S$;
        end
    end
    $vol \leftarrow vol \cdot (N/count)$;
end
return vol/ det($L$) ;
```

volume computed by of Algorithm 2 in terms of ϵ, our experimental analysis in Section 4 shows that in practice the achieved error is always better than the one proved in Proposition 3.1.

Lemma 3.2. *Given H-polytope P, Algorithm 2 performs k phases of rounding, for some $k \geq 1$, in total time $O^*(d^2mk)$. The Algorithm approximates vol(P), in $O(md^3 \log d \log(\rho/r))$ arithmetic operations, where r and ρ denote, respectively, the radii of the largest inscribed ball in P and of a cocentric ball covering P.*

Proof. Each rounding iteration decreases the diameter of P and runs in $O(nd^2(\varepsilon^{-1} + \ln d + \ln\ln(n)))$, where n stands for the number of sampled points, and ε is the approximation of the minimum volume ellipsoid of Equation (3). We generate $n = O(d)$ points, each after $O(d)$ steps, each step taking $O(m)$ arithmetic operations. The k roundings run in $O^*(d^2nk)$ total, where ε is fixed, hence the rounding stage runs in $O^*((d^3 + d^2m)k) = O^*(d^2mk)$ overall.

Algorithm 2 generates $d\log(\rho/r)$ balls and uses hit-and-run. In each ball intersected with P, we generate $\leq N = 400\epsilon^{-2} d \log d$ random points, where ϵ is the theoretical bound for volume approximation (see Proposition 3.1)—not to be confused with ε above. Each point is computed after $W = O(d)$ steps of CDHR. The boundary oracle of CDHR is implemented in Section 2. In particular, W CDHR steps require $O(dm + (W-1)m + Wd)$ arithmetic operations. It holds $d = O(m)$ and $W = \Omega(d)$. Thus, the amortized complexity of a CDHR step is $O(m)$. Overall, the algorithm needs $O(\epsilon^{-2} md^3 \log d \log(\rho/r))$ operations. □

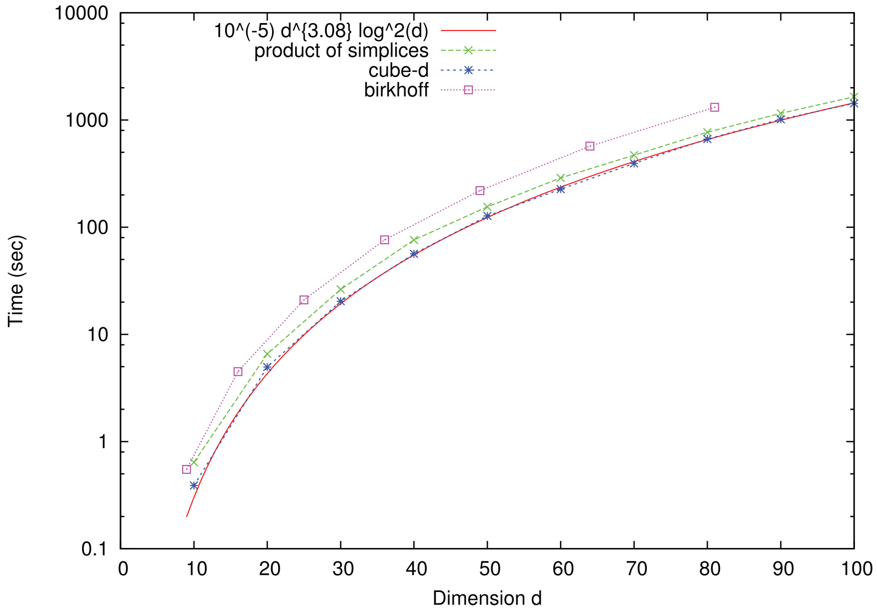

Fig. 1. Runtime of `VolEsti` w.r.t. dimension; $\epsilon = 1$, y-axis in logscale; fitting on cube-d results.

Interestingly, the number of roundings k is typically constant, for example $k = 1$ is enough to handle polytopes with $\rho/r = 100$ in dimensions up to 20.

Let us check the above bound with the experimental data for cubes, products of simplices, and Birkhoff polytopes, with $d \leq 100$ and $\epsilon = 1$, where $m = 2d$, $d + 2$ and $d + 1 + 2\sqrt{d}$, respectively, for the three classes, and for cubes $\log(\rho/r) \leq \log(\sqrt{d}) = O(\log d)$. Figure 1 shows that the three classes behave similarly. Performing a fit of $ad^b \log^2 d$, runtime follows $10^{-5} d^{3.08} \log^2 d$, which shows a smaller dependence on d than our bounds.

4 EXPERIMENTS

We implement and experimentally test the above algorithms and methods in the software package `VolEsti`. The code currently consists of around 2,500 lines in C++; it is open-source and publicly available at the following web page:

`https://github.com/vissarion/volume_approximation.`

It relies on the `CGAL` library (CGAL 2015) for its d-dimensional kernel parametrized with `double` types to represent objects such as points and vectors, for its linear programming solver (Fischer et al. 2013b), for the approximate minimum ellipsoid (Fischer et al. 2013a), and for generating random points in balls. We use `Eigen` (Guennebaud et al. 2010) for linear algebra, such as Cholesky decomposition, determinant computation, and matrix multiplication.

The memory consumption is dominated by the list of random points that need $O(dN)$ space during the entire execution of the algorithm (Section 3). Arithmetic uses the `double` data type of C++, except from the linear programming solver, which uses the *GNU Multiple Precision arithmetic library* to avoid double exponent overflow. We experimented with several pseudo-random number generators in Boost (Maurer 2000) and chose the fastest, namely Mersenne twister generator `mt19937`. All timings are on an Intel Core i5-2400 3.1GHz, 6MB L2 cache, 8GB RAM, 64-bit Debian GNU/Linux, unless otherwise stated.

4.1 Experimental Data

We construct the following database of polytopes to test our algorithm:

— cube-d: $\{x = (x_1, \ldots, x_d) \mid x_i \leq 1, x_i \geq -1, x_i \in \mathbb{R}$ for all $i = 1, \ldots, d\}$,
— cross-d: cross polytope, the dual of cube, i.e., $\text{conv}(\{-e_i, e_i,\ i = 1, \ldots, d\})$,
— Δ-d: the d-dimensional simplex $\text{conv}(\{e_i,$ for $i = 0, 1, \ldots, d\})$,
— Δ-d-d: product of two simplices, i.e $\{(p, p') \in \mathbb{R}^{2d} \mid p \in \Delta_d, p' \in \Delta_d\}$,
— skinny-cube-d: $\{x = (x_1, \ldots, x_d) \mid x_1 \leq 100, x_1 \geq -100, x_i \leq 1, x_i \geq -1, x_i \in \mathbb{R}\ i = 2, \ldots, d\}$, rotated by 30^o in the plane defined by the first two coordinate axes,
— $O(P)$: the order polytope of the poset P (defined below),
— $\mathcal{B}_n$: the n-Birkhoff polytope (defined below).

We have also tested polytopes from the VINCI web page. However, apart from the ones added to our database above, all other polytopes present no challenge for our approach.

Each experiment is repeated 100 times with $\epsilon = 1$ unless otherwise stated. The reported timing for each experiment is the mean of 100 timings. We report the min and max computed values, the mean μ, and the standard deviation. We measure the accuracy of our method by $(\text{vol}(P) - \mu)/\text{vol}(P)$ and $(\max - \min)/\mu$; unless otherwise stated, the mean error of approximation refers to the former quantity. Recall that $\text{vol}(P)$ denotes the exact volume. One should not confuse these quantities, which refer to the approximation error computed *in practice*, with ϵ, which refers to the *objective*, user-defined, approximation error. Comparing the practical and objective approximation error, our method turns out to be more accurate than indicated by the theoretical bounds. In particular, in all experiments all computed values are contained in the interval $((1 - \epsilon)\text{vol}(P), (1 + \epsilon)\text{vol}(P))$, while theoretical results in Kannan et al. (1997) guarantee only 75% of them. Actually, the above interval is larger than $[\min, \max]$.

In general, our experimental results show that our software can approximate the volume of general polytopes up to dimension 100 in less than 2 hours with mean approximation error at most 2% (cf. Table 1). Moreover, we run experiments for polytopes of dimension ≤ 200 that we do not know the volume. A natural question that arises is how the algorithm will scale to higher dimensions. By an interpolation in our experimental data (order polytopes, see Table 1), we can expect our software to compute an estimation in dimensions 500, 1000 in 6, 60 days, respectively.

4.2 Random Walks and Polytope Oracles

First, we compare the implementations of boundary oracles using membership oracles versus using facet intersection. By performing experiments with RDHR our algorithm approximates the volume of a 10-cube in 42.58 sec using the former, whereas it runs in 2.03 sec using the latter.

We compare RDHR to CDHR. The latter takes advantage of more efficient boundary oracle implementations as described in Section 2. Table 2 shows that our algorithm becomes faster with CDHR than with RDHR. An accuracy measure is $(\text{vol}(P) - \mu)/\text{vol}(P)$: CDHR may be less accurate (e.g., cube-10 in Table 2) but, since it is faster, we can obtain the same accuracy as RDHR in the same runtime by decreasing ϵ. We observe that CDHR exploits the symmetry of polytopes yielding smaller $[\min, \max]$ intervals, i.e., smaller variance of results, for cubes and Birkhoff polytopes. Random rotated cubes is a more fair comparison case, where CDHR is at most two times less accurate than RDHR (with respect to $(\text{vol}(P) - \mu)/\text{vol}(P)$, Table 2) reaching a 10x speed-up.

4.3 Choice of Parameters and Rounding

We consider two crucial parameters, the length of a random walk, denoted by W, and approximation ϵ, which determines the number N of random points. We set $W = \lfloor 10 + d/10 \rfloor$. Our

Table 1. Volume Approximation Experimental Results

P	d	m	vol(P)	N	μ	[min, max]	std-dev	$\frac{\text{vol}(P)-\mu}{\text{vol}(P)}$	VolEsti (sec)	Exact (sec)
cube-10	10	20	1.02E+003	9210	1.02E+003	[0.95E+003,1.10E+003]	3.16E+001	0.0030	0.42	0.01
cube-15	15	30	3.27E+004	16248	3.24E+004	[3.03E+004,3.43E+004]	9.41E+002	0.0088	1.44	0.40
cube-20	20	40	1.04E+006	23965	1.04E+006	[0.97E+006,1.11E+006]	3.15E+004	0.0028	4.62	swap
cube-50	50	100	1.12E+015	78240	1.12E+015	[1.00E+015,1.25E+015]	4.39E+013	0.0007	117.51	swap
cube-100	100	200	1.26E+030	184206	1.27E+030	[1.16E+030,1.40E+030]	4.82E+028	0.0081	1285.08	swap
Δ-10	10	11	2.75E-007	9210	2.76E-007	[2.50E-007,3.08E-007]	1.08E-008	0.0021	0.56	0.01
Δ-50	60	61	1.20E-082	98264	1.21E-082	[1.07E-082,1.38E-082]	6.44E-084	0.0068	183.12	0.01
Δ-100	100	101	1.07E-158	184206	1.07E-158	[9.95E-159,1.21E-158]	4.24E-160	0.0032	907.52	0.02
Δ-20-20	40	42	1.68E-037	59022	1.70E-037	[1.54E-037,1.87E-037]	7.33E-039	0.0088	53.13	0.01
Δ-40-40	80	82	1.50E-096	140224	1.50E-096	[1.32E-096,1.70E-096]	7.70E-098	0.0015	452.05	0.01
Δ-50-50	100	102	1.08E-129	184206	1.10E-129	[1.01E-129,1.19E-129]	4.65E-131	0.0154	919.01	0.02
cross-10	10	1024	2.82E-004	9210	2.82E-004	[2.69E+004,2.94E+004]	5.15E-006	0.0003	1.58	388.50
cross-11	11	2048	5.13E-005	10550	5.12E-005	[4.88E-005,5.43E-005]	1.15E-006	0.0010	5.19	6141.40
cross-12	12	4096	8.55E-006	11927	8.55E-006	[8.13E-006,9.02E-006]	1.69E-007	0.0007	12.21	—
cross-15	15	32768	2.50E-008	16248	2.50E-008	[2.33E-008,2.62E-008]	5.15E-010	0.0004	541.22	—
cross-18	18	262144	4.09E-011	20810	4.02E-011	[3.97E-011,4.08E-011]	5.58E-013	0.0165	5791.06	—
$O(P)$	10	40	8.27E-007	9210	8.30E-007	[7.06E-007,9.31E-007]	4.44E-008	0.0041	0.57	0.1
$O(P)$	20	60	8.27E-007	23965	8.23E-007	[6.99E-007,9.34E-007]	4.48E-008	0.0039	19.62	swap
$O(P)$	50	120	8.27E-007	78240	8.30E-007	[7.26E-007,9.37E-007]	4.46E-008	0.0041	385.86	swap
$O(P)$	100	220	8.27E-007	184206	8.35E-007	[6.85E-007,9.72E-007]	5.18E-008	0.0102	2866.11	swap
$\mathcal{B}_8$	49	64	4.42E-023	76279	4.46E-023	[4.05E-023, 7.32E-024]	1.93E+004	0.0092	192.97	1920.00
$\mathcal{B}_9$	64	81	2.60E-033	106467	2.58E-033	[2.23E-033, 3.07E-033]	2.13E-034	0.0069	499.56	8 days
$\mathcal{B}_{10}$	81	100	8.78E-046	142380	8.92E-046	[7.97E-046, 9.96E-046]	4.99E-047	0.0152	1034.74	6160 days
$\mathcal{B}_{11}$	100	121	?	184206	1.40E-060	[1.06E-060, 1.67E-060]	1.10E-061	?	2398.17	—
$\mathcal{B}_{12}$	121	144	?	232116	7.85E-078	[6.50E-078, 9.31E-078]	5.69E-079	?	4946.42	—
$\mathcal{B}_{13}$	144	169	?	286261	1.33E-097	[1.13E-097, 1.62E-097]	1.09E-098	?	9802.73	—
$\mathcal{B}_{14}$	169	196	?	346781	5.96E-120	[5.30E-120, 6.96E-120]	3.82E-121	?	17257.61	—
$\mathcal{B}_{15}$	196	225	?	413804	5.70E-145	[5.07E-145, 6.52E-145]	1.55E-145	?	31812.67	—

Notes: $\epsilon = 1$, "swap" indicates it ran out of memory and started swapping. "?" indicates that the exact volume vol(P), is unknown; "—" indicates it did not terminate after at least 10h. `VINCI` is used for exact volume computation except Birkhoff polytopes where `birkhoff` is used instead.

Table 2. Experiments with CDHR vs RDHR

			RDHR				CDHR			
P	d	ϵ	μ	[min, max]	(vol(P) − μ) /vol(P)	VolEsti (sec)	μ	[min, max]	(vol(P) − μ) /vol(P)	VolEsti (sec)
$\mathcal{B}_5$	16	1	2.27E-07	[1.66E-07,2.85E-07]	0.0072	22.90	2.25E-07	[1.87E-07,2.80E-07]	0.0003	4.06
$\mathcal{B}_6$	25	1	8.53E-13	[3.72E-13,1.22E-12]	0.0982	105.96	9.53E-13	[7.30E-13,1.15E-12]	0.0083	17.26
$\mathcal{B}_7$	36	1	2.75E-20	[1.78E-21,6.71E-20]	0.4259	479.40	4.82E-20	[3.86E-20,6.18E-20]	0.0056	56.64
cube-10	10	1	1.02E+03	[0.94E+03,1.10E+03]	0.0012	2.03	1026.83	[970.3117,1096.469]	0.0027	0.34
cube-20	20	1	1.04E+06	[9.38E+05,1.14E+06]	0.0033	25.44	1.04E+06	[9.74E+05,1.12E+06]	0.0028	4.62
cube-rot-10	10	1	1.02E+03	[9.81E+02,1.05E+03]	0.0084	1.91	1.03E+03	[9.82E+02 1.07E+03]	0.0096	0.39
cube-rot-50	50	1	1.12E+15	[1.06E+15,1.19E+15]	0.0023	1329.62	1.12E+15	[1.05E+15 1.19E+15]	0.0044	155.27
cube-rot-100	100	1	1.27E+30	[1.25E+30,1.29E+30]	0.0021	23202.08	1.26E+30	[1.14E+30 1.37E+30]	0.0028	2390.25

Notes: $W = 10$ is used; cube-rot-d is a randomly rotated cube of dimension d.

Table 3. Experiments with Varying W

							$(\text{vol}(P)-\mu)$	(max-min)
P	d	m	W	μ	[min, max]	std-dev	$/\text{vol}(P)$	$/\mu$
(*) cube-10	10	20	**10**	1026.953	[925.296,1147.101]	33.91331	0.0029	0.2160
cube-10	10	20	15	1024.157	[928.667,1131.928]	31.34121	0.0002	0.1985
cube-10	10	20	20	1026.910	[932.118,1144.601]	30.97023	0.0028	0.2069
cube-50	50	100	10	1.123E+15	[1.019E+15,1.257E+15]	4.135E+13	0.0022	0.2125
(*) cube-50	50	100	**15**	1.131E+15	[1.039E+15,1.237E+15]	3.882E+13	0.0044	0.1744
cube-50	50	100	20	1.127E+15	[1.033E+15,1.216E+15]	3.893E+13	0.0007	0.1629
cube-100	100	200	10	1.278E+30	[1.165E+30,1.402E+30]	4.819E+28	0.0081	0.1856
cube-100	100	200	15	1.250E+30	[1.243E+30,1.253E+30]	4.075E+27	0.0140	0.0083
(*) cube-100	100	200	**20**	1.263E+30	[1.190E+30,1.321E+30]	3.987E+28	0.0038	0.1038
Δ-20-20	40	42	10	1.699E-37	[1.527E-37,1.881E-37]	7.670E-39	0.0056	0.2083
(*) Δ-20-20	40	42	**14**	1.694E-37	[1.526E-37,1.892E-37]	7.096E-39	0.0025	0.2166
Δ-20-20	40	42	20	1.694E-37	[1.433E-37,1.836E-37]	7.006E-39	0.0024	0.2382
Δ-50-50	100	102	10	1.098E-129	[1.012E-129,1.189E-129]	4.652E-131	0.0154	0.1612
Δ-50-50	100	102	15	1.111E-129	[1.090E-129,1.139E-129]	1.610E-131	0.0281	0.0437
(*) Δ-50-50	100	102	**20**	1.079E-129	[1.011E-129,1.148E-129]	3.685E-131	0.0015	0.1266
$\mathcal{B}_{10}$	81	100	10	7.951E-55	[6.291E-55,9.077E-55]	8.533E-56	0.0946	0.3504
$\mathcal{B}_{10}$	81	100	15	8.124E-55	[7.451E-55,8.774E-55]	5.015E-56	0.0750	0.1629
(*) $\mathcal{B}_{10}$	81	100	**20**	7.489E-55	[7.398E-55,7.552E-55]	6.615E-57	0.1472	0.0106

Note: $\epsilon = 1$; (*) indicates minimum W where either $(\text{vol}(P) - \mu)/\text{vol}(P)$ or (max-min)μ is $< 1\%$.

experiments indicate that, with this choice, either $(\text{vol}(P) - \mu)/\text{vol}(P)$ or $(max - min)/\mu$ is $< 1\%$ up to $d = 100$ (Table 3). Moreover, for higher W the improvement in accuracy is not significant, which supports the claim that asymptotic bounds are unrealistically high. Figure 2 correlates runtime, expressed by the product NW, and accuracy, expressed by $(\max - \min)/\mu$ (which actually measures some "deviation"), with respect to W and ϵ (expressed by N). A positive observation is that accuracy tightly correlates with runtime: e.g., accuracy values close to or beyond 1 lie under the curve $NW = 10^5$, and those rounded to ≤ 0.3 lie roughly above $NW = 3 \cdot 10^5$. It also shows that increasing W converges faster than increasing N to a value beyond which the improvement in accuracy is not significant.

To experimentally test the effect of *rounding*, we construct skinny hypercubes skinny-cube-d. We rotate them to avoid CDHR, taking unfair advantage of the degenerate situation where the long edge is parallel to an axis. Table 4 on these and other polytopes shows that rounding reduces approximation error by two orders of magnitude. Without rounding, for polytope rv-8-11 one needs to multiply N (thus runtime) by 100 to achieve approximation error same as with rounding.

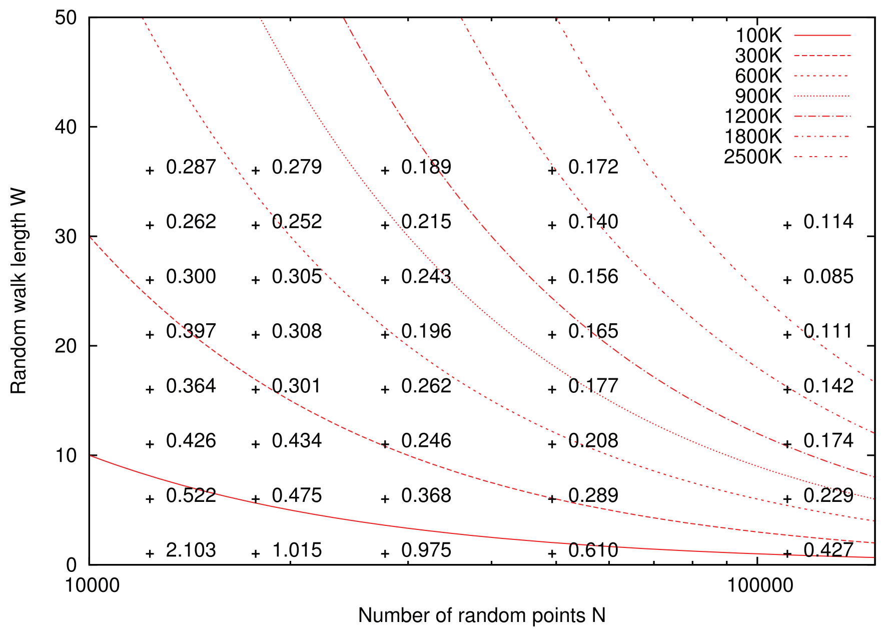

Fig. 2. Experiments with $\mathcal{B}_5$ on the effect of W and ϵ (or N) on accuracy, measured by (max-min)/μ (crosses), and runtime, measured by levels of $N \cdot W = c$, for $c = 10^5, \ldots, 2.5 \cdot 10^6$.

Table 4. Experiments with Rounding

P	vol(P)	N	μ	[min, max]	$\frac{\text{vol}(P)-\mu}{\text{vol}(P)}$	VolEsti(sec)
rv-8-11	3.047E+18	6654	1.595E+18	[6.038E+17,3.467E+18]	0.4766	1.48
rv-8-11	3.047E+18	665421	3.134E+18	[3.134E+18,3.134E+18]	0.0283	157.46
(*) rv-8-11	3.047E+18	6654	3.052E+18	[2.755E+18,3.383E+18]	0.0013	1.34
skinny-cube-10	1.024E+05	9210	5.175E+04	[2.147E+04,1.228E+05]	0.4946	0.69
(*) skinny-cube-10	1.024E+05	9210	1.029E+05	[8.445E+04,1.149E+05]	0.0050	0.71
skinny-cube-20	1.049E+08	23965	4.193E+07	[2.497E+07,7.259E+07]	0.6001	5.59
(*) skinny-cube-20	1.049E+08	23965	1.040E+08	[8.458E+07,1.163E+08]	0.0084	6.70

Note: (*): means that we use rounding.

4.4 Other Software

Exact volume computation concerns software computing the exact value of the volume, up to round-off errors in case it uses floating point arithmetic. We mainly test against VINCI 1.0.5 (Büeler and Enge 2000), which implements state-of-the art algorithms, cf. Table 1. For H-polytopes, the method based on Lawrence's general formula is numerically unstable resulting in wrong results in many examples (Büeler et al. 2000), and thus was excluded. Therefore, we focused on Lasserre's method. For all polytopes there is a threshold dimension for which VINCI cannot compute the volume: it takes a lot of time (e.g., >4 hrs for cube-20) and consumes all system memory, thus starts swapping.

Table 5. Experiments with qhull

	rv-15-				rv-10-				cube-			
P:	30	40	50	60	100	150	200	250	7	8	9	10
time (sec)	7.7	82.8	473.3	swap	37.3	107.8	282.5	449.0	0.1	2.2	119.5	> 5h

Note: "swap" indicates it ran out of memory and started swapping; ">5h" indicates it did not terminate after 5 hours.

Table 6. Comparison of the Software (Cousins and Vempala 2014) vs. VolEsti

	software of Cousins and Vempala [2014]					VolEsti				
P	[min, max]	std-dev	$\frac{\mathrm{vol}(P)-\mu}{\mathrm{vol}(P)}$	# total steps	time(sec)	[min, max]	std-dev	$\frac{\mathrm{vol}(P)-\mu}{\mathrm{vol}(P)}$	# total steps	time(sec)
cube-20	[5.11E+05, 1.55E+06]	1.67E+05	0.0198	7.96E+04	21.48	[9.74E+05, 1.12E+06]	3.15E+04	0.0028	3.61E+06	4.62
cube-30	[6.75E+08, 1.45E+09]	1.72E+08	0.0440	2.22E+05	49.24	[9.91E+08, 1.16E+09]	3.89E+07	0.0039	1.21E+07	17.96
cube-40	[7.90E+11, 1.38E+12]	1.67E+11	0.0731	4.30E+05	88.09	[1.01E+12, 1.23E+12]	4.46E+10	0.0039	2.84E+07	50.72
cube-50	[8.75E+14, 1.45E+15]	1.43E+14	0.0327	7.16E+05	148.06	[1.00E+15, 1.25E+15]	4.39E+13	0.0007	5.49E+07	117.51
cube-60	[8.89E+17, 1.43E+18]	1.64E+17	0.0473	1.15E+06	229.33	[1.06E+18, 1.27E+18]	4.00E+16	0.0051	9.42E+07	222.10
cube-70	[9.01E+20, 1.36E+21]	1.49E+20	0.0707	1.66E+06	427.82	[1.02E+21, 1.32E+21]	5.42E+19	0.0013	1.49E+08	358.93
cube-80	[9.30E+23, 1.36E+24]	1.46E+23	0.1145	2.30E+06	531.46	[1.13E+24, 1.30E+24]	4.42E+22	0.0009	2.21E+08	582.19
cube-90	[1.07E+27, 1.88E+27]	2.20E+26	0.0394	3.30E+06	701.54	[1.09E+27, 1.44E+27]	5.18E+25	0.0019	3.15E+08	875.69
cube-100	[9.53E+29, 1.64E+30]	1.93E+29	0.0357	4.19E+06	884.43	[1.17E+30, 1.40E+30]	4.82E+28	0.0081	4.33E+08	1285.08
$\mathcal{B}_8$	[2.12E-23, 2.45E-22]	6.25E-23	0.3970	9.31E+05	221.30	[4.05E-23, 7.32E-24]	1.93E+04	0.0092	1.01E+08	192.97
$\mathcal{B}_9$	[1.54E-33, 2.77E-33]	3.71E-34	0.1830	2.05E+06	420.07	[2.23E-33, 3.07E-33]	2.13E-34	0.0069	2.27E+08	499.56
$\mathcal{B}_{10}$	[3.39E-46, 1.92E-45]	4.75E-46	0.1207	3.69E+06	691.97	[7.97E-46, 9.96E-46]	4.99E-47	0.0152	4.62E+08	1034.74

Note: each experiment is run 10 times, total steps refer to the mean of the total number of hit-and-run steps in each execution.

Latte implements the same decomposition methods as VINCI; it is less prone to round-off error but slower (De Loera et al. 2013). Normaliz (Bruns et al. 2013) applies triangulation: it handles cubes for $d \leq 10$, in < 1 min, but for $d = 15$, it did not terminate after 5 hours. Qhull handles V-polytopes but does not terminate for cube-10 nor random polytope rv-15-60 (Table 5). This should be juxtaposed to the duals, namely our software approximates the volume of cross-10 in 2 sec with $<1\%$ error and rh-15-60 in 3.44 sec. LRS (Avis 2000) is not useful for H-polytopes as stated on its webpage: "If the volume option is applied to an H-representation, the results are not predictable." A general conclusion for exact software is that it cannot handle $d > 15$.

We compare with the most relevant approximation method, namely the Matlab implementation of Cousins and Vempala (2016), for bodies represented as the intersection of an H-polytope and an ellipsoid. They report that the code is optimized to achieve about 75% success rate for bodies of dimension ≤ 100 and objective $\epsilon \in [0.1, 0.2]$ (not to be confused with the ϵ of our method). Testing Cousins and Vempala (2016) with default options and $\epsilon = 0.1$, our implementation with $\epsilon = 1$ runs faster for $d < 80$, performs roughly 100 times more total hit-and-run steps and returns significantly more accurate results, e.g., from 4 to 100 times smaller error on cube-d when $d > 70$, and from 5 to 80 times on Birkhoff polytopes (Table 6).

4.5 Birkhoff Polytopes

The n-th Birkhoff polytope is

$$\mathcal{B}_n = \{x \in \mathbb{R}^{n \times n} \mid x_{ij} \geq 0, \ \sum_i x_{ij} = 1, \ \sum_j x_{ij} = 1, \ 1 \leq i, j \leq n\}.$$

It is the polytope of the perfect matchings of the complete bipartite graph $K_{n,n}$, the polytope of the $n \times n$ doubly stochastic matrices, or the Newton polytope of the determinant. Every point in the polytope can equivalently be seen as an $n \times n$ matrix or a point in $\mathbb{R}^{n \times n}$. These polytopes are well

Table 7. Asymptotic and Experimental Approximation of the Volume of $\mathcal{B}_n$

n	d	estimation	asymptotic (Canfield and McKay 2009)	$\frac{\text{estimation}}{\text{asymptotic}}$	exact	$\frac{\text{exact}}{\text{asymptotic}}$
3	4	1.12E+000	1.41E+000	0.793284	1.13E+000	0.797392
4	9	6.79E-002	7.61E-002	0.891934	6.21E-002	0.815930
5	16	1.41E-004	1.69E-004	0.834435	1.41E-004	0.834190
6	25	7.41E-009	8.62E-009	0.859866	7.35E-009	0.852792
7	36	5.67E-015	6.51E-015	0.871389	5.64E-015	0.866505
8	49	4.39E-023	5.03E-023	0.872949	4.42E-023	0.877863
9	64	2.58E-033	2.93E-033	0.881310	2.60E-033	0.887412
10	81	8.92E-046	9.81E-046	0.909205	8.78E-046	0.895549
11	100	1.40E-060	1.49E-060	0.934258	?	?
12	121	7.85E-078	8.38E-078	0.937051	?	?
13	144	1.33E-097	1.43E-097	0.933151	?	?
14	169	5.96E-120	6.24E-120	0.955008	?	?
15	196	5.70E-145	5.94E-145	0.959378	?	?

Note: "?" indicates that the exact volume is unknown.

studied in combinatorial geometry and offer an important benchmark. A complex-analytic method (Beck and Pixton 2003), implemented in package `birkhoff`, has managed to compute vol($\mathcal{B}_{10}$) in parallel execution, which corresponds to a single 1GHz processor running for almost 17 years.

Since $\dim \mathcal{B}_n = n^2 - 2n + 1$, we project $\mathcal{B}_n$ to a subspace of this dimension. Our software, with $\epsilon = 1$, computes the volume of polytopes up to $\mathcal{B}_{10}$ in < 1 hour with mean error of $\leq 1.5\%$ (Table 1). By setting $\epsilon = 0.5$, we obtain an error of 0.7% for vol($\mathcal{B}_{10}$), in 6 hours. The computed approximation has two correct digits.

More interestingly, using $\epsilon = 1$ we compute, in < 9 hours, an approximation as well as an interval of values for vol($\mathcal{B}_{11}$), ..., vol($\mathcal{B}_{15}$). The absence of exact values makes the computation of the approximation error impossible. However, the calculation in Table 7 provides an experimental proof of the quality of our approximation since it shows that the ratio of our estimation over the asymptotic upper bound of Canfield and McKay (2009) converges to 1.

We exploit the symmetries of Birkhoff polytopes to compute efficiently more random points. If a point p is in $\mathcal{B}_n$ then all points obtained by applying the permutations of the symmetric group in the rows and columns of p (seen as a matrix) lie in $\mathcal{B}_n$. Therefore, for each p we readily compute $2n! - 1$ more random points. In general, whenever a polytope possesses some known symmetry, producing easily random points increases accuracy without practically increasing runtime. As an illustration, for $\mathcal{B}_6$, exploiting symmetries reduces relative error $(\mathrm{vol}(P) - \mu)/\mathrm{vol}(P)$ from 0.138 to 0.011 and shrinks interval [min, max] from $[7.807E - 13, 1.731E - 12]$ to $[8.813e - 13, 1.016e - 12]$ (experiments repeated 10 times).

4.6 Order Polytopes

The order polytope of the poset P is the subset of $\mathbb{R}^{|P|}$—where $|P|$ is the number of elements in P—defined by the conditions

$$0 \leq f(x) \leq 1, \text{ for all } x \in P, \tag{4}$$

$$f(x) \leq f(y), \text{ if } x \leq y \text{ in } P. \tag{5}$$

Order polytopes have been defined in Stanley (1986). They are important polytopes in combinatorics since their volume equals the number of linear extensions of the poset P. However, computing this number is proven to be hard (Brightwell and Winkler 1991).

In the experiments of this article, we construct a poset P with 20 fixed edges and we vary the number of elements to test how our algorithm will scale with dimension. The relative error $(\text{vol}(P) - \mu)/\text{vol}(P)$ of our computation is less than 0.01 up to dimension 100. For more details, see Table 1.

4.7 Parallel Computation

We exploit a crucial property of the method, its straightforward parallelization. In particular, we assign threads of the random walk to computer processors threads. We use `boost` threads (William and Escriba 2007). We test the acceleration of parallel computation in our software by computing the volumes of $\mathcal{B}_8, \mathcal{B}_9, \mathcal{B}_{10}$. To carry out these experiments, we used an `Intel(R) Core(TM) i7-4700HQ CPU @ 2.40GHz` with 8 processor threads. The parallel timings are 35.46, 91.64, 217.57 secs while the sequential are 153.31, 399.25, 825.65 secs, respectively. Therefore, we obtain an acceleration of approximately five times.

5 CONCLUSION

Various theoretical and practical research questions arise from our study. For example, the point samples can also be seen as another means of representing the polytope: Could we implement fast membership simply using the point samples?

Our original motivation and ultimate goal is to extend our methods to V-polytopes and, more generally, to polytopes represented by an optimization oracle.

The rest of this section is devoted to an interesting connection between approximate NN and approximate boundary oracles. Duality reduces the boundary oracle problem to NN search and its variants. Given a pointset $B \subseteq \mathbb{R}^d$ and query point q, NN search returns a point $p \in B$ s.t. $\text{dist}(q,p) \leq \text{dist}(q,p')$ for all $p' \in B$, where $\text{dist}(q,p)$ is the Euclidean distance between points q, p. Let us consider, w.l.o.g., boundary intersection for line ℓ parallel to the x_d-axis: $\ell = \{x : x = \lambda v + p,\ \lambda \in \mathbb{R}\}$, $v = (0, \ldots, 0, -1)$. It reduces to two ray-shooting questions; it suffices to describe one, namely with the upward vertical ray, defined by $\lambda \leq 0$. We seek the first facet hyperplane hit which, equivalently, has the maximum negative signed vertical distance from p to any hyperplane H of the upper hull. This distance is denoted by $\text{sv}(p, H)$. Let us consider the standard (aka functional) duality transform between points p and non-vertical hyperplanes H:

$$p = (p_1, \ldots, p_d) \mapsto p^* : x_d = p_1 x_1 + \cdots + p_{d-1} x_{d-1} - p_d,$$
$$H : x_d = c_1 x_1 + \cdots + c_{d-1} x_{d-1} + c_0 \mapsto H^* = (c_1, \ldots, c_{d-1}, -c_0).$$

This transformation is self-dual, preserves point-hyperplane incidences, and negates vertical distance, hence $\text{sv}^*(p^*, H^*) = -\text{sv}(p, H)$, where $\text{sv}^*(\cdot,\cdot)$ is the signed vertical distance from hyperplane p^* to point H^* in dual space. Hence, our problem is equivalent to minimizing $\text{sv}^*(p^*, H^*) \geq 0$. Equivalently, we seek point H^* minimizing absolute vertical distance to hyperplane p^* on its halfspace of positive distances. In dual space, consider

$$\text{point } t = (t_1, \ldots, t_d), \text{ and hyperplane } p^* = q : x_d = q_1 x_1 + \cdots + q_{d-1} x_{d-1} + q_0 : \tag{6}$$

Then, define the following function $\text{sv}^*(q, t)$, which is minimized over all t in the positive halfspace of vector q:

$$\text{sv}^*(q, t) = t_d - (q_1 t_1 + \cdots + q_{d-1} t_{d-1} + q_0) \quad = -(q_0, q_1, \ldots, q_{d-1}, -1) \cdot (1, t_1, \ldots, t_{d-1}, t_d),$$

where the latter operation is inner product in "lifted" Euclidean space $\mathbb{R}^{d+1}$ of "lifted" points $t' = (1, t_1, \ldots, t_{d-1}, t_d)$ with "lifted" query point $q' = (q_0, q_1, \ldots, q_{d-1}, -1)$. Notice that the t' lie in convex position. Let

$$q'' = (q', 0),\ \ t'' = (t', \sqrt{M - \|t'\|_2^2}),\ \ \text{for } M \geq \max_t\{1 + \|t\|_2^2\},$$

following an idea of Basri et al. (2011). By the cosine rule,

$$\mathrm{dist}_{d+2}^2(q'', t'') = \|q'\|_2^2 + M + 2\mathrm{sv}^*(q, t),$$

where $\mathrm{dist}_{d+2}(\cdot, \cdot)$ stands for Euclidean distance in $\mathbb{R}^{d+2}$. Since the t'' lie on hyperplane $x_1 = 1$, optimizing $\mathrm{dist}_{d+2}(q'', t'')$ over a set of points t'' is equivalent to optimizing $\mathrm{dist}_{d+1}(\hat{q}, \hat{t})$ in $(d+1)$-dimensional space, where $\hat{q} = (q_1, \ldots, q_{d-1}, -1, 0)$, over points $\hat{t} = (t, \sqrt{M - 1 - \|t\|_2^2})$. Hence, point t minimizing $\mathrm{sv}^*(q, t) \geq 0$ corresponds to $\hat{t}$ minimizing $\mathrm{dist}_{d+1}^2(\hat{q}, \hat{t})$. Thus the problem is reduced to NN in $\mathbb{R}^{d+1}$. Ray shooting to the lower hull with same v reduces to farthest neighbor.

Preliminary tests of the above approach, using FLANN (Muja 2011) to perform NN queries, have shown a 40x speed-up for certain instances such as cross-polytopes in dimension 16. However, an approximate solution to these problems incurs an additive error to the corresponding original problem. Thus, we view this technique mainly with motivation for the design of algorithms that can use approximate boundary queries and hence take advantage of NN software to handle polytopes with a large number of facets.

ACKNOWLEDGMENTS

Most of the work was done while the second author was at the NKU Athens. The authors acknowledge discussions with Matthias Beck on Birkhoff polytopes, and Andreas Enge on VINCI, and help with experiments with NN software by Ioannis Psarros and Georgios Samaras, students at NKUA. Finally, we acknowledge useful comments by anonymous reviewers that helped to improve the presentation of the article.

REFERENCES

P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. 2005. Geometric approximation via coresets. In *Combinatorial and Computational Geometry*. MSRI, Berkeley, 1–30.

A. Andoni and P. Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* 51 (2008), 117–122.

S. Arya, G. D. da Fonseca, and D. M. Mount. 2012. Optimal area-sensitive bounds for polytope approximation. In *Proc. Symp. on Computational Geometry*. ACM, New York, 363–372.

D. Avis. 2000. lrs: A revised implementation of the reverse search vertex enumeration algorithm. In *Polytopes: Combinatorics & Computation*. Oberwolfach Seminars, Vol. 29. Birkhäuser, Basel, 177–198.

I. Bárány and Z. Füredi. 1987. Computing the volume is difficult. *Discrete Comput. Geom.* 2, 4 (1987), 319–326.

C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. 1996. The quickhull algorithm for convex hulls. *ACM Trans. of Math. Softw.* 22, 4 (1996), 469–483.

R. Basri, T. Hassner, and L. Zelnik-Manor. 2011. Approximate nearest subspace search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 2 (2011), 266–278.

M. Beck and D. Pixton. 2003. The Ehrhart polynomial of the Birkhoff polytope. *Discrete Comput. Geom.* 30, 4 (2003), 623–637.

D. Bertsimas and S. Vempala. 2004. Solving convex programs by random walks. *J. ACM* 51, 4 (2004), 540–556.

U. Betke and M. Henk. 1993. Approximating the volume of convex bodies. *Discrete Comput. Geom.* 10, 1 (1993), 15–21. http://dx.doi.org/10.1007/BF02573960.

G. Brightwell and P. Winkler. 1991. Counting linear extensions. *Order* 8, 3 (1991), 225–242.

W. Bruns, B. Ichim, and C. Söger. 2013. Normaliz. Algorithms for rational cones and affine monoids. Retrieved from http://www.math.uos.de/normaliz.

B. Büeler and A. Enge. 2000. VINCI. Retrieved from http://www.math.u-bordeaux1.fr/ aenge/index.php?category=software&page=vinci.

B. Büeler, A. Enge, and K. Fukuda. 2000. *Exact Volume Computation for Polytopes: A Practical Study*. Mathematics and Statistics, Vol. 29. Birkhäuser, Basel, 131–154.

E. Canfield and B. McKay. 2009. The asymptotic volume of the Birkhoff polytope. *Online J. Anal. Comb.* 4 (2009).

CGAL 2015. CGAL: Computational Geometry Algorithms Library. Retrieved from http://www.cgal.org.

Y. Chen, P. Diaconis, S. P. Holmes, and J. S. Liu. 2005. Sequential Monte Carlo methods for statistical analysis of tables. *J. Amer. Statist. Assoc.* 100, 469 (March 2005), 109–120.

B. Cousins and S. Vempala. 2014. A cubic algorithm for computing gaussian volume. In *Proc. Symp. on Discrete Algorithms*. SIAM/ACM, 1215–1228.

B. Cousins and S. Vempala. 2015. Bypassing KLS: Gaussian cooling and an $O^*(n^3)$ volume algorithm. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC'15)*. ACM, New York, NY, 539–548.

B. Cousins and S. Vempala. 2016. A practical volume algorithm. *Math. Program. Comput.* 8, 2 (2016), 133–160.

M. Cryan and M. Dyer. 2003. A polynomial-time algorithm to approximately count contingency tables when the number of rows is constant. *J. Comput. System Sci.* 67, 2 (2003), 291–310.

J. A. De Loera, B. Dutra, M. Köppe, S. Moreinis, G. Pinto, and J. Wu. 2013. Software for exact integration of polynomials over polyhedra. *Comput. Geom.: Theory Appl.* 46, 3 (April 2013), 232–252.

J. A. De Loera, F. Liu, and R. Yoshida. 2008. A generating function for all semi-magic squares and the volume of the Birkhoff polytope. *J. Algebraic Comb.* 30, 1 (2008), 113–139.

P. Diaconis and B. Sturmfels. 1998. Algebraic algorithms for sampling from conditional distributions. *Ann. Stat.* 26, 1 (02 1998), 363–397.

M. E. Dyer and A. M. Frieze. 1988. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.* 17, 5 (1988), 967–974.

M. Dyer, A. Frieze, and R. Kannan. 1991. A random polynomial-time algorithm for approximating the volume of convex bodies. *J. ACM* 38, 1 (1991), 1–17.

M. Dyer, R. Kannan, and J. Mount. 1997. Sampling contingency tables. *Random Struct. Algorithms* 10 (1997), 487–506.

G. Elekes. 1986. A geometric inequality and the complexity of computing volume. *Discrete Comput. Geom.* 1 (1986), 289–292.

I. Z. Emiris and V. Fisikopoulos. 2014. Efficient random-walk methods for approximating polytope volume. In *Proc. Symp. on Computational Geometry*. ACM, Kyoto, Japan, 318–325.

K. Fischer, B. Gärtner, T. Herrmann, M. Hoffmann, and S. Schönherr. 2013a. Bounding volumes. In *CGAL User and Reference Manual* (4.3 ed.). CGAL Editorial Board. Retrieved from http://doc.cgal.org/4.3/Manual/packages.html#PkgBoundingVolumesSummary.

K. Fischer, B. Gärtner, S. Schönherr, and F. Wessendorp. 2013b. Linear and quadratic programming solver. In *CGAL User and Reference Manual* (4.3 ed.). CGAL Editorial Board. http://doc.cgal.org/4.3/Manual/packages.html#PkgQPSolverSummary.

M. Grötschel, L. Lovász, and A. Schrijver. 1993. *Geometric Algorithms and Combinatorial Optimization* (2nd corrected ed.). Algorithms and Combinatorics, Vol. 2. Springer, Berlin.

G. Guennebaud, B. Jacob, et al. 2010. Eigen v3. Retrieved from http://eigen.tuxfamily.org.

U. Jaekel. 2011. A Monte Carlo method for high-dimensional volume estimation and application to polytopes. *Proced. Comput. Sci.* 4 (2011), 1403–1411.

R. Kannan, L. Lovász, and M. Simonovits. 1997. Random walks and an $O(n^5)$ volume algorithm for convex bodies. *Rand. Struct. Algor.* 11 (1997), 1–50.

R. Kannan and H. Narayanan. 2012. Random walks on polytopes and an affine interior point method for linear programming. *Math. Op. Res.* 37, 1 (2012), 1–20.

R. Kannan and S. Vempala. 1997. Sampling lattice points. In *Proc. Symp. Theory of Computing*. ACM, New York, 696–700.

L. Khachiyan. 1993. Complexity of polytope volume computation. In *New Trends in Discrete and Computational Geometry*. Springer, Berlin, 91–101.

L. G. Khachiyan. 1996. Rounding of polytopes in the real number model of computation. *Math. Oper. Res.* 21, 2 (1996), 307–320.

J. Lawrence. 1991. Polytope volume computation. *AMS Math. Comput.* 57, 195 (1991), 259–271.

S. Liu, J. Zhang, and B. Zhu. 2007. Volume computation using a direct Monte Carlo method. In *Comp. and Combinatorics.*, G. Lin (Ed.). LNCS, Vol. 4598. Springer, Berlin, 198–209.

L. Lovász. 1999. Hit-and-run mixes fast. *Math. Program.* 86 (1999), 443–461. Issue 3.

L. Lovász and I. Deák. 2012. Computational results of an $O(n^4)$ volume algorithm. *Eur. J. Op. Res.* 216, 1 (2012), 152–161.

L. Lovász and S. Vempala. 2006a. Hit-and-run from a corner. *SIAM J. Comput.* 35, 4 (2006), 985–1005.

L. Lovász and S. Vempala. 2006b. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *J. Comp. Syst. Sci.* 72, 2 (2006), 392–417.

J. Maurer. 2000. Boost: C++ Libraries. Chapter 23. Boost Random. Retrieved from www.boost.org/doc/libs/1_54_0/doc/html/ boost_random.html.

L. Mohácsi and I. Deák. 2015. A parallel implementation of an $O(n^4)$ volume algorithm. *Cent. Eur. J. Op. Res.* 3, 24 (2015), 1–28.

M. Muja. 2011. FLANN: Fast Library for Approximate Nearest Neighbors. Retrieved October 2013 from http://mloss.org/software/view/143/.

E. A. Ramos. 1999. On range reporting, ray shooting and K-level construction. In *Proc. Symp. on Computational Geometry.* ACM, New York, 390–399.

S. Reisner, C. Schütt, and E. Werner. 2001. Dropping a vertex or a facet from a convex polytope. *Forum Math.* 13, 3 (2001), 359–378.

M. Simonovits. 2003. How to compute the volume in high dimension? *Math. Program.* (2003), 337–374.

R. L. Smith. 1984. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Op. Res.* 32, 6 (1984), 1296–1308.

R. Stanley. 1986. Two poset polytopes. *DiscreteComput. Geom.* 1, 1 (1986), 9–23.

S. Vempala. 2005. Geometric random walks: A survey. *Comb. Comput. Geom.* 52 (2005), 573–612.

A. William and V. J. B. Escriba. 2007. Boost: C++ Libraries. Chapter 32. Thread. Retrieved from http://www.boost.org/doc/libs/1_56_0/doc/html/thread.html.

Y. Zheng and K. Yamane. 2013. Ray-shooting algorithms for robotics. *IEEE Trans. Autom. Sci. Eng.* 10, 4 (2013), 862–874.

Received March 2015; revised January 2017; accepted February 2018