Efficient Random-Walk Methods for Approximating Polytope Volume

Vissarion Fisikopoulos

Joint work with I.Z. Emiris

Dept. of Informatics & Telecommunications, University of Athens

Visiting Scholar at NIMS, South Korea

SoCG 10 Jun. 2014

The volume computation problem

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

Output: Volume of P

- #-P hard for vertex and for halfspace repres. [DyerFrieze'88]

The volume computation problem

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

Output: Volume of P

- #-P hard for vertex and for halfspace repres. [DyerFrieze'88]
- open if both vertex & halfspace representation is available

The volume computation problem

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

Output: Volume of P

- #-P hard for vertex and for halfspace repres. [DyerFrieze'88]
- open if both vertex & halfspace representation is available
- no deterministic poly-time algorithm can compute the volume with less than exponential relative error [Elekes'86]

The volume computation problem

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

Output: Volume of P

- #-P hard for vertex and for halfspace repres. [DyerFrieze'88]
- open if both vertex & halfspace representation is available
- no deterministic poly-time algorithm can compute the volume with less than exponential relative error [Elekes'86]
- randomized poly-time algorithm approximates the volume of a convex body with high probability and arbitrarily small relative error [DyerFriezeKannan'91] $O^*(d^{23}) \to O^*(d^4)$ [LovVemp'04]

Implementations

Exact: `VINCI` [Bueler et al'00], `Latte` [deLoera et al], `Qhull` [Barber et al], `LRS` [Avis], `Normaliz` [Bruns et al]

- triangulation, sign decomposition methods
- cannot compute in high dimensions (e.g. > 20)

Randomized:

- [LovàszDeàk'12] cannot compute in > 10 dimensions
- Matlab code by Cousins & Vempala based on [LovVemp'04]
- Ours: based on [DyerFriezeKannan'91],…, [KannanLovàszSimon.'97]

How do we compute a random point in a polytope P?

- easy for simple shapes like simplex or cube

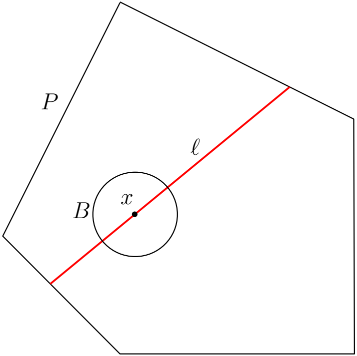
How do we compute a random point in a polytope $\mathbb{P}$?

- easy for simple shapes like simplex or cube

- BUT for arbitrary polytopes we need *random walks*

  e.g. grid walk, ball walk, hit-and-run

Random Directions Hit-and-Run (RDHR)



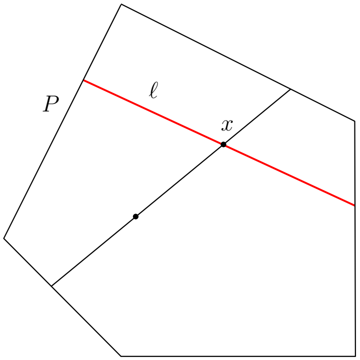
Input: point $x \in P$ and polytope $P \subset \mathbb{R}^d$

Output: a new point in P

1. line ℓ through x, uniform on $B(x, 1)$
2. set x to be a uniform disrtibuted point on $P \cap \ell$

Iterate this for W steps and return x.

Random Directions Hit-and-Run (RDHR)



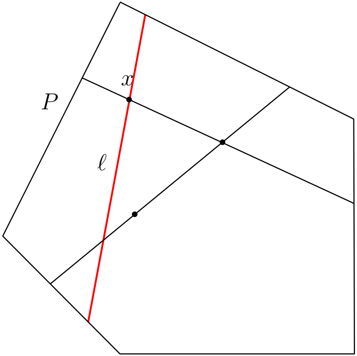
Input: point $x \in P$ and polytope $P \subset \mathbb{R}^d$

Output: a new point in P

1. line ℓ through x, uniform on $B(x, 1)$
2. set x to be a uniform disrtibuted point on $P \cap \ell$

Iterate this for W steps and return x.

Random Directions Hit-and-Run (RDHR)



Input: point $x \in P$ and polytope $P \subset \mathbb{R}^d$

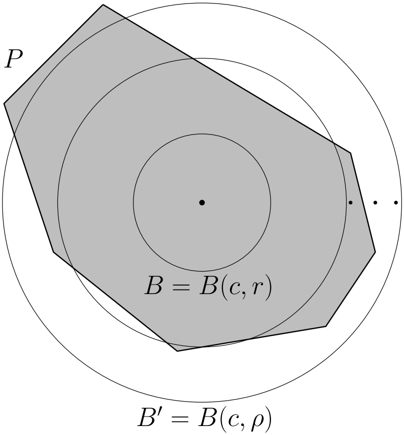
Output: a new point in P

1. line ℓ through x, uniform on $B(x, 1)$
2. set x to be a uniform disrtibuted point on $P \cap \ell$

Iterate this for W steps and return x.

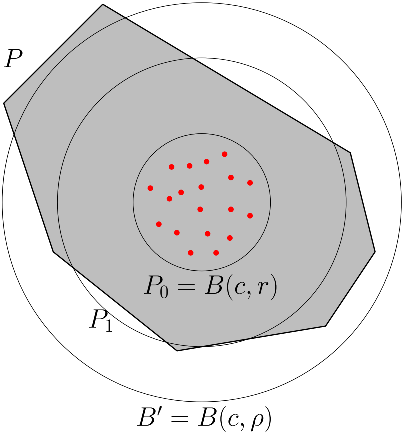
- x is unif. random distrib. in P after $W = O^*(d^3)$ steps, where $O^*(\cdot)$ hides log factors [LovaszVempala'06]
- to generate many random points iterate this procedure

Multiphase Monte Carlo (Sequence of balls)



- $B(c, 2^{i/d})$, $i = \alpha, \alpha+1, \ldots, \beta$,
  $\alpha = \lfloor d \log r \rfloor$, $\beta = \lceil d \log \rho \rceil$
- $P_i := P \cap B(c, 2^{i/d})$, $i = \alpha, \alpha+1, \ldots, \beta$,
  $P_\alpha = B(c, 2^{\alpha/d}) \subseteq B(c, r)$

Multiphase Monte Carlo (Generating random points)

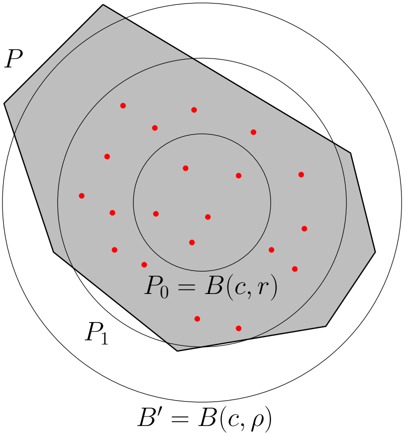


- $B(c, 2^{i/d})$, $i = \alpha, \alpha+1, \dots, \beta$,
  $\alpha = \lfloor d \log r \rfloor$, $\beta = \lceil d \log \rho \rceil$
- $P_i := P \cap B(c, 2^{i/d})$, $i = \alpha, \alpha+1, \dots, \beta$,
  $P_\alpha = B(c, 2^{\alpha/d}) \subseteq B(c, r)$

1. Generate rand. points in P_i
2. Count how many rand. points in P_i fall in P_{i-1}

$$\mathrm{vol}(P) = \mathrm{vol}(P_\alpha) \prod_{i=\alpha+1}^{\beta} \frac{\mathrm{vol}(P_i)}{\mathrm{vol}(P_{i-1})}$$

Multiphase Monte Carlo (Generating random points)



- $B(c, 2^{i/d})$, $i = \alpha, \alpha+1, \dots, \beta$,
  $\alpha = \lfloor d \log r \rfloor$, $\beta = \lceil d \log \rho \rceil$
- $P_i := P \cap B(c, 2^{i/d})$, $i = \alpha, \alpha+1, \dots, \beta$,
  $P_\alpha = B(c, 2^{\alpha/d}) \subseteq B(c, r)$

1. Generate rand. points in P_i
2. Count how many rand. points in P_i fall in P_{i-1}

$$\mathrm{vol}(P) = \mathrm{vol}(P_\alpha) \prod_{i=\alpha+1}^{\beta} \frac{\mathrm{vol}(P_i)}{\mathrm{vol}(P_{i-1})}$$

Complexity [KannanLS'97]

Assuming $B(c, 1) \subseteq P \subseteq B(c, \rho)$, the volume algorithm returns an estimation of vol(P), which lies between $(1-\epsilon)$vol(P) and $(1+\epsilon)$vol(P) with probability $\geq 3/4$, making

$$O^*(d^5)$$

oracle calls, where ρ is the radius of a bounding ball for P.
Isotropic sandwitching: $\rho = O^*(\sqrt{d})$ and ball walk.

Runtime

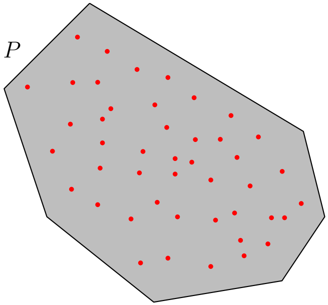
- generates $d \log(\rho)$ balls
- generate $N = 400\epsilon^{-2} d \log d$ random points in each ball $\cap P$
- each point is computed after $O^*(d^3)$ random walk steps

Modifications towards practicality

- $W = \lfloor 10 + d/10 \rfloor$ random walk steps (vs. $O^*(d^3)$ which hides constant 10^{11}) achieve $< 1\%$ error in up to 100 dim.
- sample partial generations of $\leq N$ points in each ball $\cap P$ (starting from the largest ball)
- coordinate (vs. random) directions hit-and-run (CDHR)
- implement boundary oracles with $O(m)$ runtime in CDHR

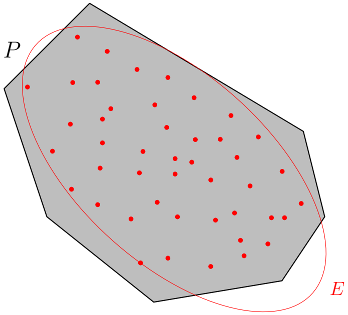
Iterative rounding

1. compute set S of random points in P



Iterative rounding

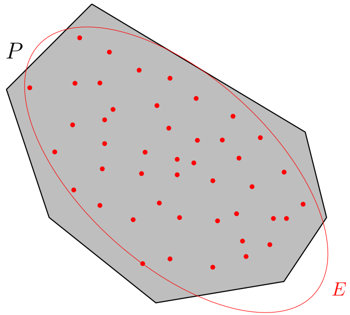
1. compute set S of random points in P
2. compute (approximate) minimum volume ellipsoid E covers S



Iterative rounding

1. compute set S of random points in P
2. compute (approximate) minimum volume ellipsoid E covers S
3. compute L that maps E to the unit ball B and apply L to P

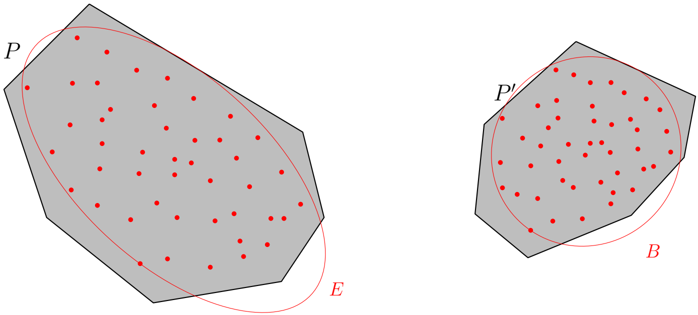
Iterate until the ratio of the maximum over the minumum ellipsoid axis reaches some user-defined threshold.



Iterative rounding

1. compute set S of random points in P
2. compute (approximate) minimum volume ellipsoid E covers S
3. compute L that maps E to the unit ball B and apply L to P

Iterate until the ratio of the maximum over the minumum ellipsoid axis reaches some user-defined threshold.



Efficiently handle skinny polytopes in practice.

Experimental results

- approximate the volume of a series of polytopes (cubes, random, cross, birkhoff) up to dimension 100 in less than 2 hours with mean approximation error at most 1%

Experimental results

- approximate the volume of a series of polytopes (cubes, random, cross, birkhoff) up to dimension 100 in less than 2 hours with mean approximation error at most 1%
- randomized vs. exact software (`VINCI`, etc.): there is a threshold dimension $(\mathrm{d} < 15)$ for which exact software halts

Experimental results

- approximate the volume of a series of polytopes (cubes, random, cross, birkhoff) up to dimension 100 in less than 2 hours with mean approximation error at most 1%
- randomized vs. exact software (`VINCI`, etc.): there is a threshold dimension ($d < 15$) for which exact software halts
- computed value always in $[(1-\epsilon)\text{vol}(P), (1+\epsilon)\text{vol}(P)]$ (vs. prob. 3/4 [KLS'97]) up to $d = 100$

Experimental results

- approximate the volume of a series of polytopes (cubes, random, cross, birkhoff) up to dimension 100 in less than 2 hours with mean approximation error at most 1%
- randomized vs. exact software (`VINCI`, etc.): there is a threshold dimension ($d < 15$) for which exact software halts
- computed value always in $[(1-\epsilon)\mathrm{vol}(P), (1+\epsilon)\mathrm{vol}(P)]$ (vs. prob. 3/4 [KLS'97]) up to $d = 100$
- CDHR faster and more accurate than RDHR

Experimental results

- approximate the volume of a series of polytopes (cubes, random, cross, birkhoff) up to dimension 100 in less than 2 hours with mean approximation error at most 1%
- randomized vs. exact software (`VINCI`, etc.): there is a threshold dimension ($d < 15$) for which exact software halts
- computed value always in $[(1-\epsilon)\text{vol}(P), (1+\epsilon)\text{vol}(P)]$ (vs. prob. 3/4 [KLS'97]) up to $d = 100$
- CDHR faster and more accurate than RDHR
- Compute the volume of Birkhoff polytopes $B_{11}, \ldots, B_{15}$ in few hrs whereas exact methods have only computed that of B_{10} by specialized software in ~ 1 year of parallel computation

Volumes of Birkhoff polytopes

n	d	estimation	asymptotic [CanfieldMcKay09]	$\frac{\text{estimation}}{\text{asymptotic}}$	exact	$\frac{\text{exact}}{\text{asymptotic}}$
3	4	1.12E+000	1.41E+000	0.7932847	1.13E+000	0.7973923
4	9	6.79E-002	7.61E-002	0.8919349	6.21E-002	0.8159296
5	16	1.41E-004	1.69E-004	0.8344350	1.41E-004	0.8341903
6	25	7.41E-009	8.62E-009	0.8598669	7.35E-009	0.8527922
7	36	5.67E-015	6.51E-015	0.8713891	5.64E-015	0.8665047
8	49	4.39E-023	5.03E-023	0.8729497	4.42E-023	0.8778632
9	64	2.62E-033	2.93E-033	0.8960767	2.60E-033	0.8874117
10	81	8.14E-046	9.81E-046	0.8305162	8.78E-046	0.8955491
11	100	1.40E-060	1.49E-060	0.9342584	???	???
12	121	7.85E-078	8.38E-078	0.9370513	???	???
13	144	1.33E-097	1.43E-097	0.9331517	???	???
14	169	5.96E-120	6.24E-120	0.9550089	???	???
15	196	5.70E-145	5.94E-145	0.9593786	???	???

Ongoing work

1. random walks for polytopes described by optimization oracles
   e.g. resultant polytopes [Emiris,F,Konaxis,Penaranda SoCG'12]

2. use approximate oracles (utilizing approximate NN)

3. volume of more general convex bodies
   e.g. spectahedra

Conclusion

- Practical volume estimation in high dimensions (e.g. 100)
- Software framework for testing theoretical ideas (e.g. new geometric random walks)

Code

- `http://sourceforge.net/projects/randgeom`

Conclusion

- Practical volume estimation in high dimensions (e.g. 100)
- Software framework for testing theoretical ideas (e.g. new geometric random walks)

Code

- `http://sourceforge.net/projects/randgeom`

THANK YOU