Volume and edge-skeleton computation in high dimensions

Vissarion Fisikopoulos

Computer Science Department, Algorithms Group

INRIA 5 Nov. 2014

Outline of the talk

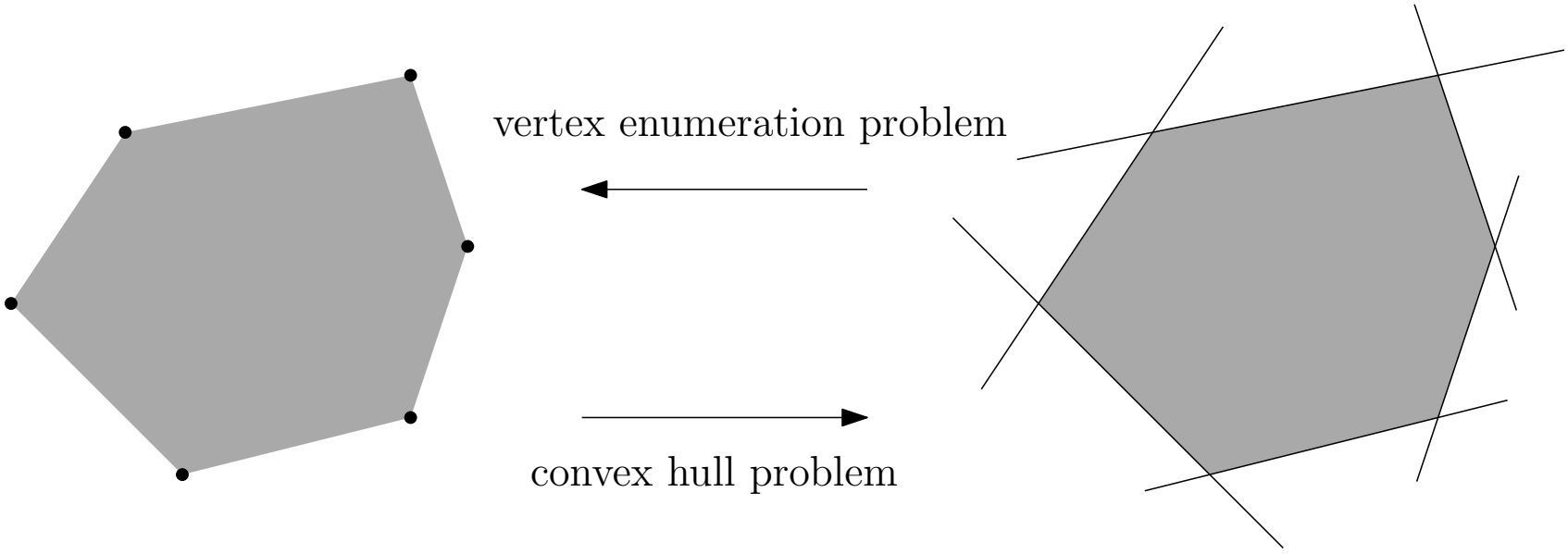
Edge-skeleton computation [Emiris-F-Gaertner]

Practical volume approximation [Emiris-F]

Classical Polytope Representations

A convex polytope $P \subseteq \mathbb{R}^d$ can be represented as the

1. convex hull of a pointset $\{p_1, \ldots, p_n\}$ (V-representation)
2. intersection of halfspaces $\{h_1, \ldots, h_m\}$ (H-representation)



- These problems are equivalent by polytope duality.

Algorithmic Issues

- For general dimension d there is no polynomial algorithm for the convex hull (or vertex enumeration) problem since m can be $O\left(n^{\lfloor d/2 \rfloor}\right)$ [McMullen'70].

- It is open whether there exist a total poly-time algorithm for the convex hull (or vertex enumeration) problem, *i.e. runs in poly-time in* n, m, d.

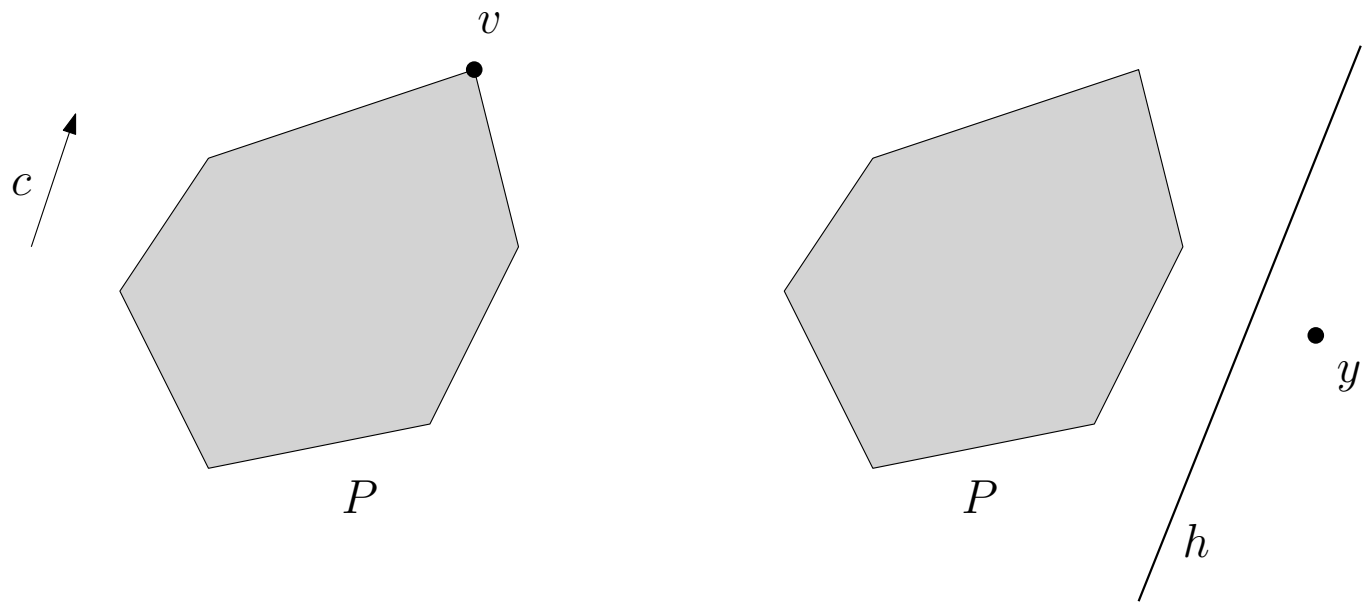
- Vertex enumeration special cases
  1. Simple polytopes: $O(n\,m\,d)$ [Avis,Fukuda'92]
  2. $0/1$ polytopes: $O(n\,d\,T_{LP})$ [Bussieck, Lubbecke'98]

Polytope Oracles

Implicit representation for a polytope $P \subseteq \mathbb{R}^d$.

OPT_P: Given direction $c \in \mathbb{R}^d$ return the vertex $v \in P$ that maximizes $c^T v$.

SEP_P: Given point $y \in \mathbb{R}^d$, return yes if $y \in P$ otherwise a hyperplane h that separates y from P.



Well-described polytopes and oracles

Definition

A rational polytope $P \subseteq \mathbb{R}^d$ is well-described (with a parameter φ) if there exists an H-representation for P in which every inequality has encoding length at most φ. The encoding length of P is $\langle P \rangle = d + \varphi$.

Proposition (Grötschel et al.'93)

For a well-described polytope, we can compute OPT_P from SEP_P (and vice versa) in oracle polynomial-time. The runtime (polynomially) depends on d and φ.

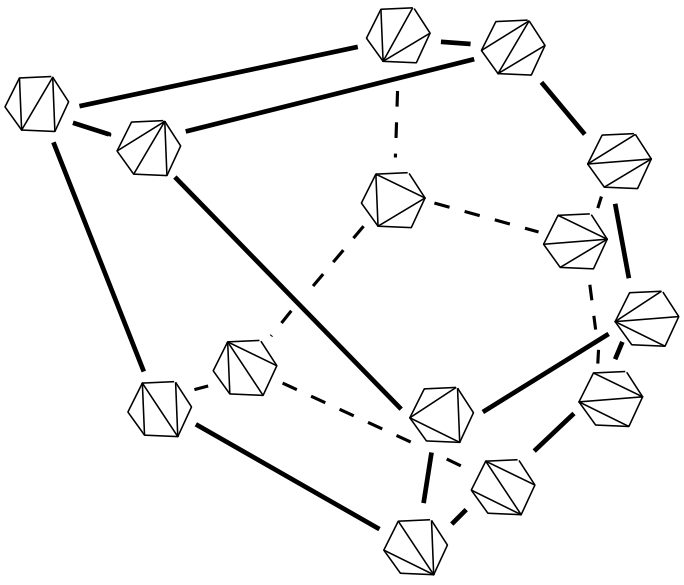
Why oracles?

- Polynomial time algorithms for combinatorial optimization problems using the ellipsoid method [Grötschel-Lovász-Schrijver'93]

- Randomized polynomial-time algorithm for approximating the volume of convex bodies [Dyer-Frieze-Kannan '90]

Original Motivation

(1) Secondary, Resultant polytopes



- Vertices $\longrightarrow$ reg. triangulations of a pointset's convex hull
- $\mathrm{OPT}_{\mathrm{P}}$ is available via a triangulation computation [Emiris-F-Konaxis-Peñaranda '12]

(2) Minkowski sums

- Applications in Computational Algebraic Geometry, Geometric Modelling, Combinatorics

Vertex enumeration with edge-directions

Given OPT_P and a superset D of the edge directions $D(P)$ of $P \subseteq \mathbb{R}^d$, compute the vertices P.

Proposition (Rothblum-Onn '07)

Let $P \subseteq \mathbb{R}^d$ *given by* OPT_P, *and* $D \supseteq D(P)$. *All vertices of* P *can be computed in*

$O(|D|^{d-1})$ *calls to* $\mathrm{OPT}_P + O(|D|^{d-1})$ *arithmetic operations.*

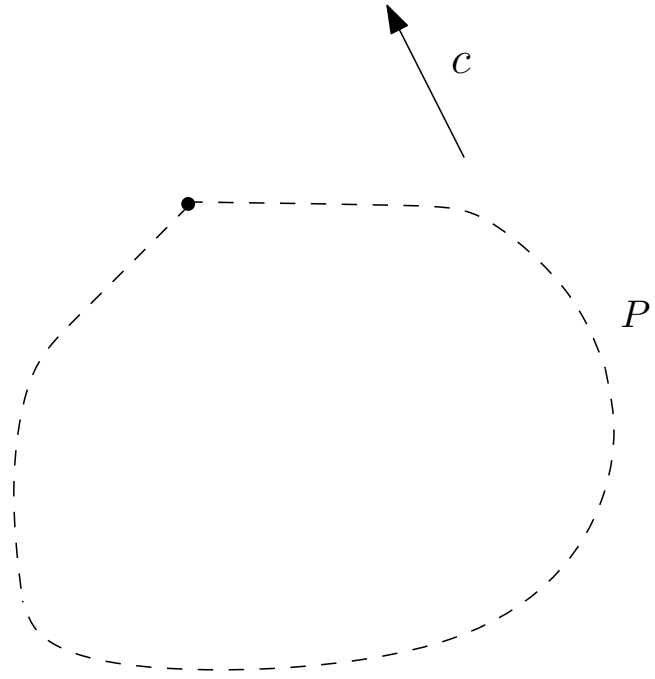
The edge-skeleton algorithm

Input:

- OPT_P
- Edge vec. P (dir. & len.): D

Output:

- Edge-skeleton of P



Sketch of **Algorithm**:

- Compute a vertex of P ($x = \mathrm{OPT}_P(c)$ for arbitrary $c^T \in \mathbb{R}^d$)

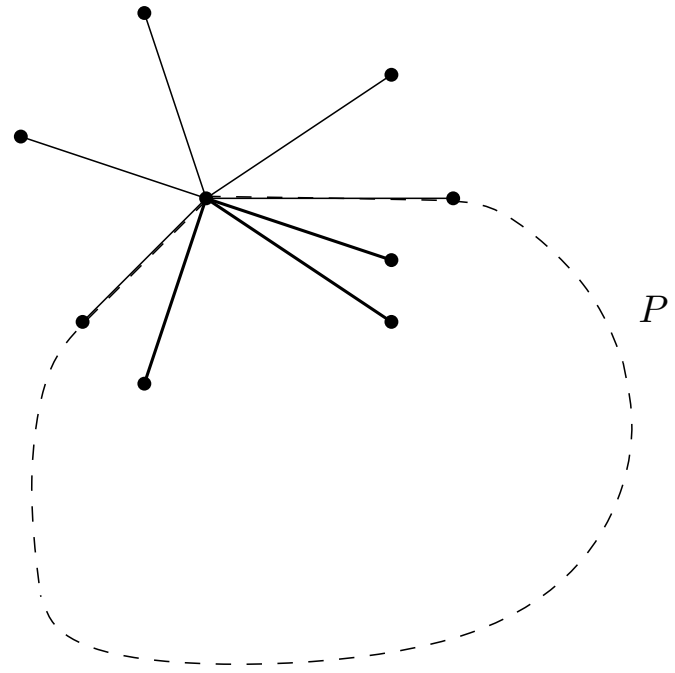
The edge-skeleton algorithm

Input:

- OPT_P
- Edge vec. P (dir. & len.): D

Output:

- Edge-skeleton of P



Sketch of **Algorithm**:

- Compute a vertex of P ($x = \mathrm{OPT}_P(c)$ for arbitrary $c^T \in \mathbb{R}^d$)
- Compute segments $S = \{(x, x + d), \text{ for all } d \in D\}$

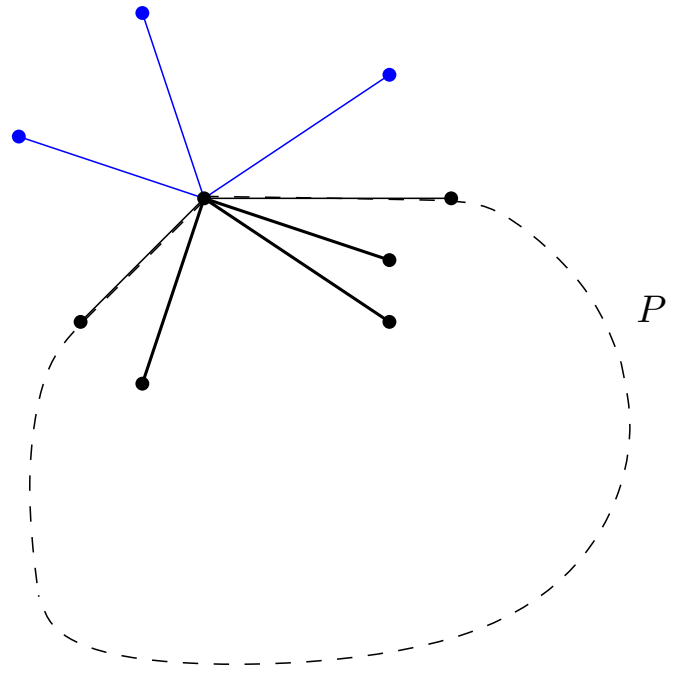
The edge-skeleton algorithm

Input:

- OPT_P
- Edge vec. P (dir. & len.): D

Output:

- Edge-skeleton of P



Sketch of **Algorithm**:

- Compute a vertex of P ($x = \mathrm{OPT}_P(c)$ for arbitrary $c^T \in \mathbb{R}^d$)
- Compute segments $S = \{(x, x + d), \text{ for all } d \in D\}$
- Remove from S all segments (x, y) s.t. $y \notin P$ ($\mathrm{OPT}_P \to \mathrm{SEP}_P$)

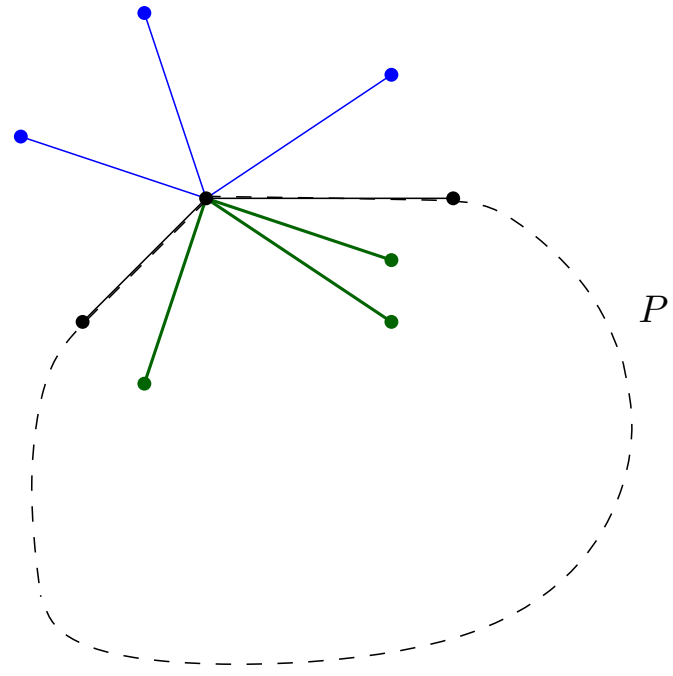
The edge-skeleton algorithm

Input:

- OPT_P
- Edge vec. P (dir. & len.): D

Output:

- Edge-skeleton of P



Sketch of **Algorithm**:

- Compute a vertex of P ($x = \mathrm{OPT}_P(c)$ for arbitrary $c^T \in \mathbb{R}^d$)
- Compute segments $S = \{(x, x + d), \text{ for all } d \in D\}$
- Remove from S all segments (x, y) s.t. $y \notin P$ ($\mathrm{OPT}_P \to \mathrm{SEP}_P$)
- Remove from S the segments that are not extreme

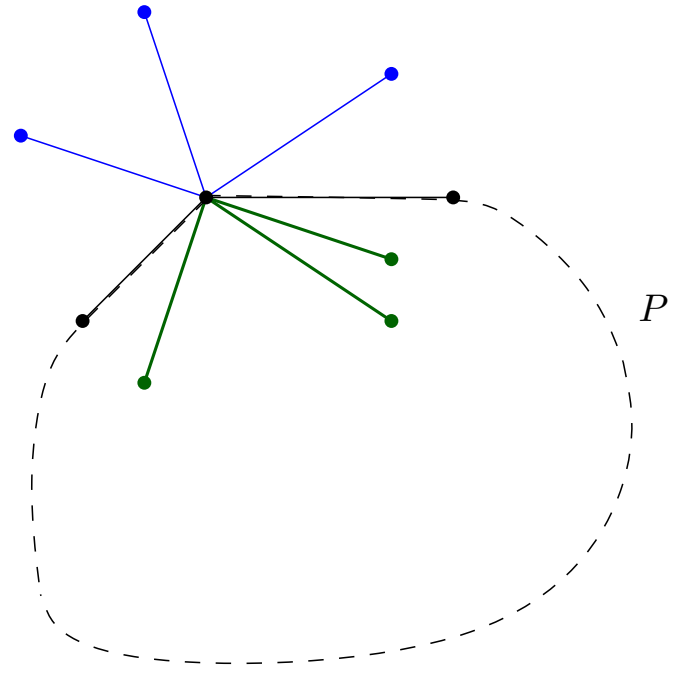
The edge-skeleton algorithm

Input:

- OPT_P
- Edge vec. P (dir. & len.): D

Output:

- Edge-skeleton of P



Sketch of **Algorithm**:

- Compute a vertex of P ($x = \mathrm{OPT}_P(c)$ for arbitrary $c^T \in \mathbb{R}^d$)
- Compute segments $S = \{(x, x + d), \text{ for all } d \in D\}$
- Remove from S all segments (x, y) s.t. $y \notin P$ ($\mathrm{OPT}_P \rightarrow \mathrm{SEP}_P$)
- Remove from S the segments that are not extreme

Can be altered to work with edge directions only

Complexity

Theorem

Given OPT_P *and a superset of edge directions* D *of a well-described polytope* $P \subseteq \mathbb{R}^d$*, the edge skeleton of* P *can be computed in oracle total polynomial-time*

$$O\left(n\,|D|\,\left(T + \mathbb{LP}(d^3|D|\langle B\rangle) + d\log n\right)\right),$$

- n *the number of vertices of* P*,*
- T : *runtime of oracle conversion algorithm for* P *and* D*,*
- $\langle B\rangle$ *is the binary encoding length of the vector set* D *and* P*,*
- $\mathbb{LP}(\langle A\rangle + \langle b\rangle + \langle c\rangle)$ *runtime of max* $c^T x$ *over* $\{x : Ax \leq b\}$*.*

Applications

Corollary

The edge skeleton of resultant, secondary polytopes and signed Minkowski sums (knowing the edge directions of the summands) can be computed in oracle total polynomial-time.

Corollary

The edge skeletons of polytopes appearing in convex combinatorial optimization [Rothblum-Onn '04] and convex integer programming [De Loera et al. '09] problems can be computed in oracle total polynomial-time.

Outline

Edge-skeleton computation [Emiris-F-Gaertner]

Practical volume approximation [Emiris-F]

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

Output: Volume of P

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

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] $O^*(d^{23}) \rightarrow O^*(d^4)$ [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. > 15)

Randomized:

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

Goal: compute accurate estimations in about 100 dimensions

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

- easy for simple shapes like simplex or cube

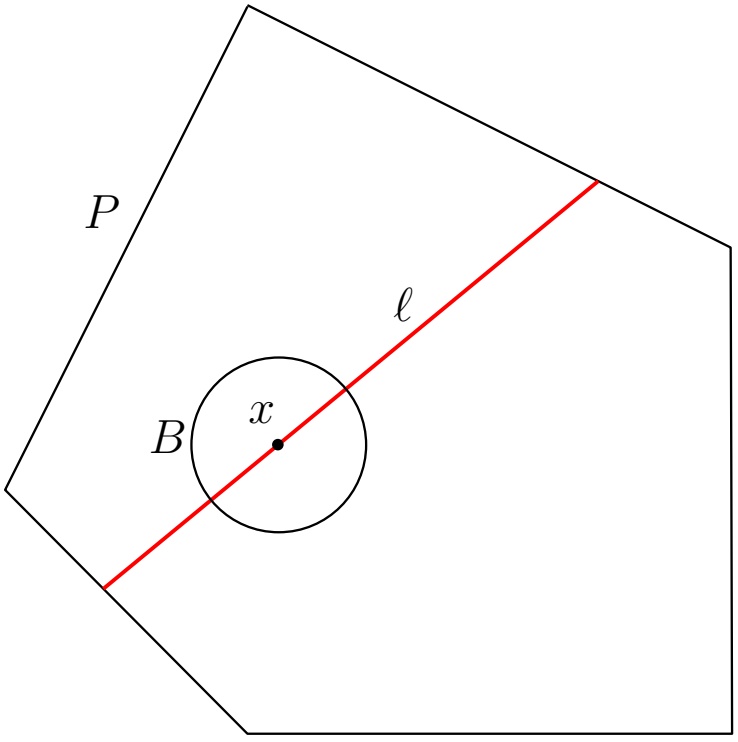
How do we compute a random point in a polytope P?

- 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)



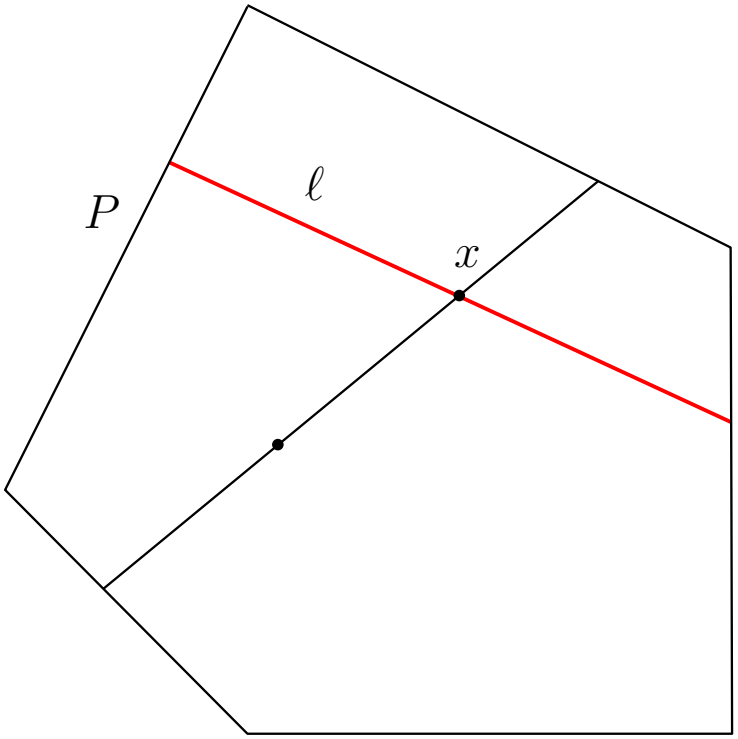
Input: point $\mathrm{x} \in \mathrm{P}$ and polytope $\mathrm{P} \subset \mathbb{R}^{\mathrm{d}}$

Output: a new point in P

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

Iterate this for W steps and return x.

Random Directions Hit-and-Run (RDHR)



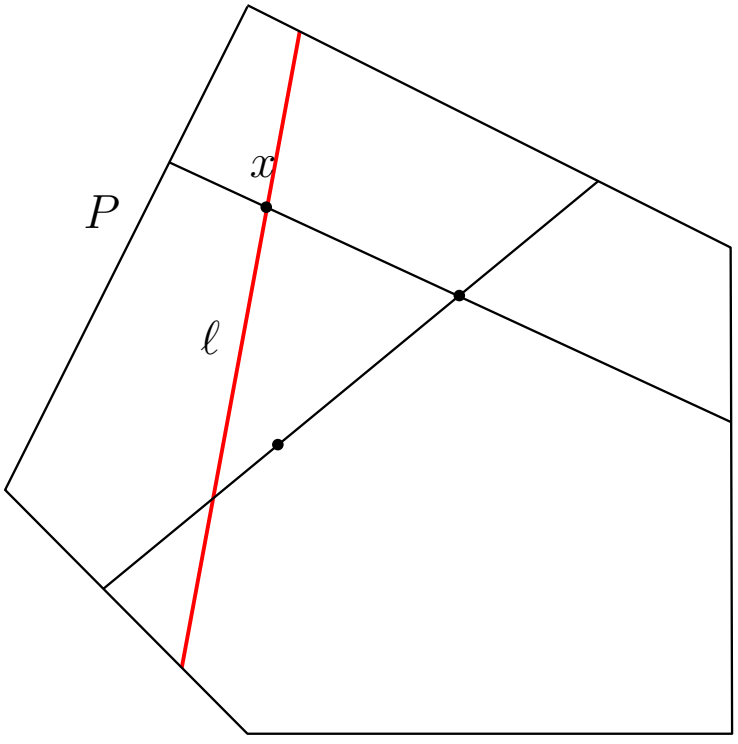
Input: point $x \in P$ and polytope $P \subset \mathbb{R}^d$

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 $\mathrm{x} \in \mathrm{P}$ and polytope $\mathrm{P} \subset \mathbb{R}^{\mathrm{d}}$

Output: a new point in P

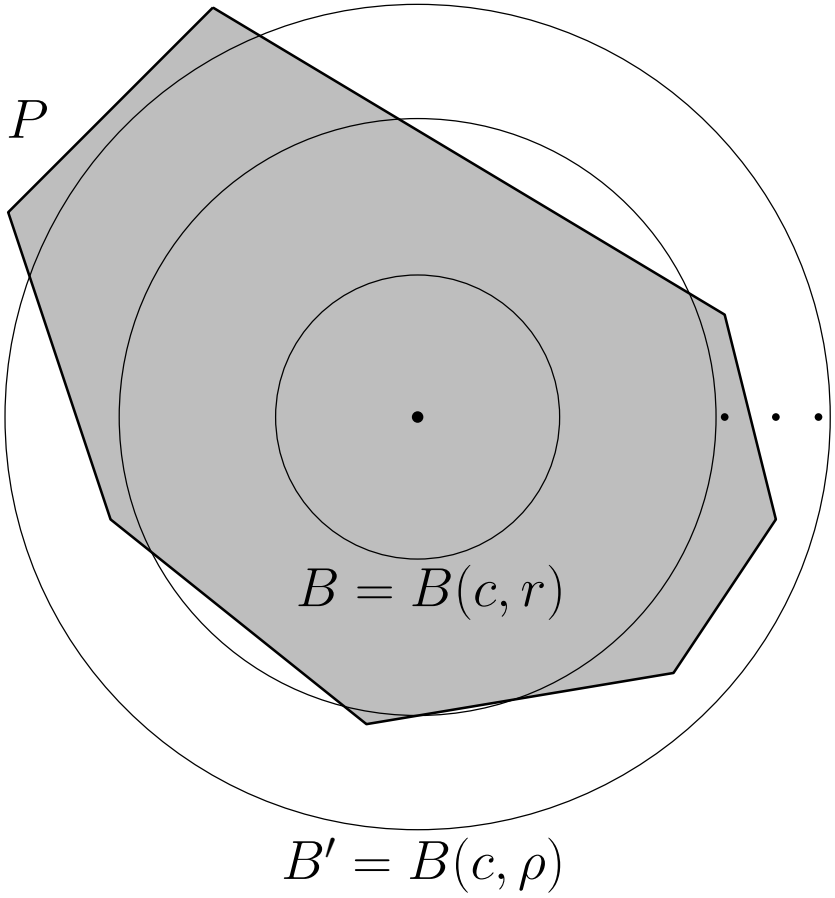
1. line ℓ through x, uniform on $\mathrm{B}(\mathrm{x}, 1)$
2. set x to be a uniform disrtibuted point on $\mathrm{P} \cap \ell$

Iterate this for W steps and return x.

- x is unif. random distrib. in P after $W = \mathrm{O}^*(\mathrm{d}^3)$ steps, where $\mathrm{O}^*(\cdot)$ hides log factors [LovaszVempala'06]

- to generate many random points iterate this procedure

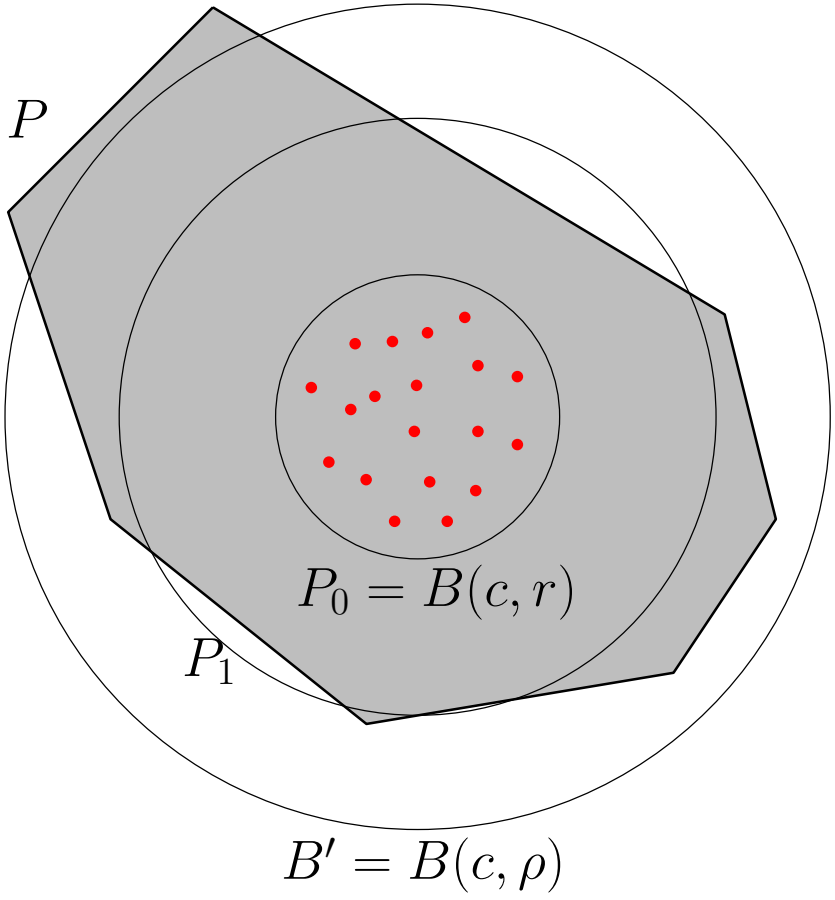
Multiphase Monte Carlo (Sequence of balls)



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

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

Multiphase Monte Carlo (Generating random points)



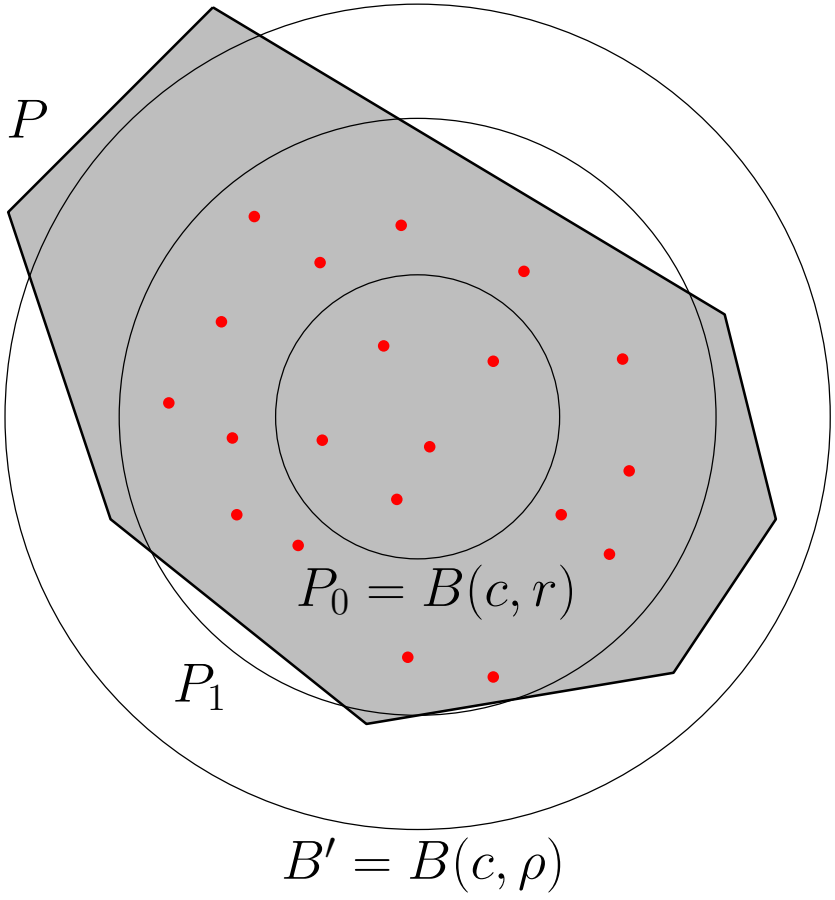
- $B(c, 2^{i/d})$, $i = \alpha, \alpha+1, \ldots, \beta$,
  $\alpha = \lfloor d \log r \rfloor$, $\beta = \lceil d \log \rho \rceil$

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

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

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

Multiphase Monte Carlo (Generating random points)



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

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

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

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

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

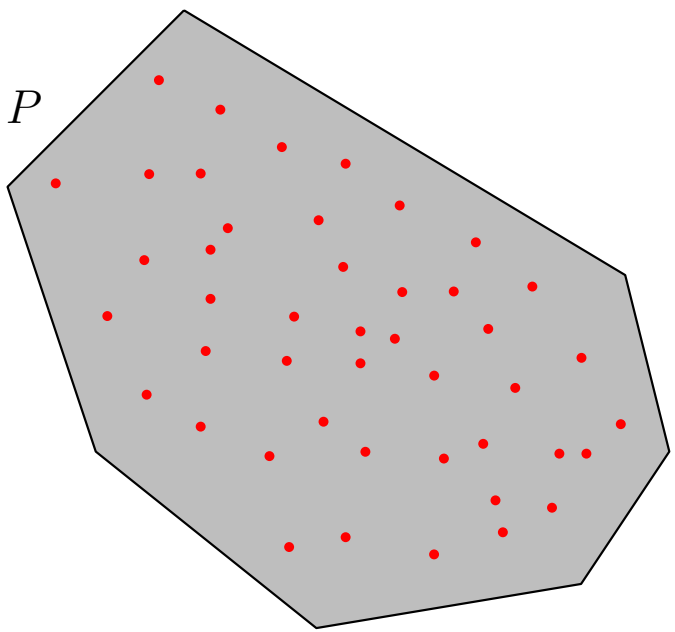
- generates $d \log(\rho)$ balls
- generate $N = 400\epsilon^{-2} d \log d$ random points in each ball $\cap P$
- each point is computed after $O^*(d^3)$ random walk steps

Modifications towards practicality

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

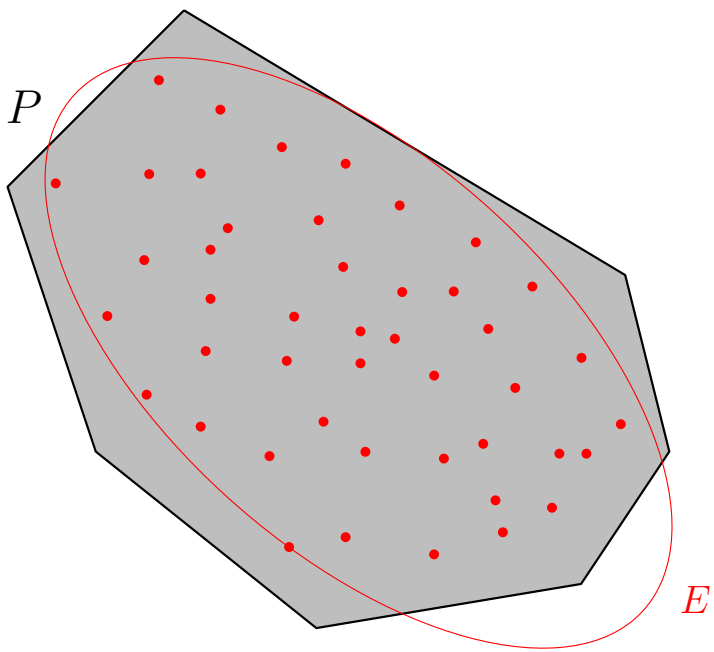
Iterative rounding

1. compute set S of random points in P



Iterative rounding

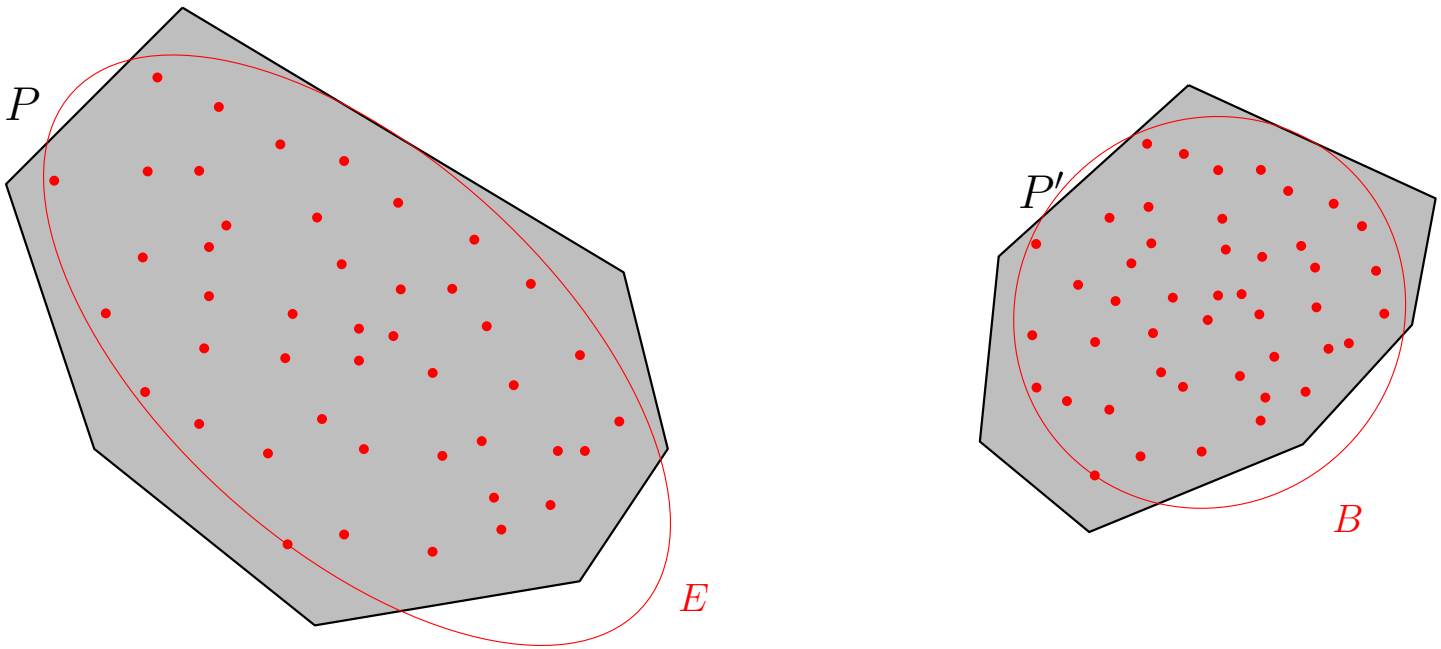
1. compute set S of random points in P
2. compute (approximate) minimum volume ellipsoid E covers S



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

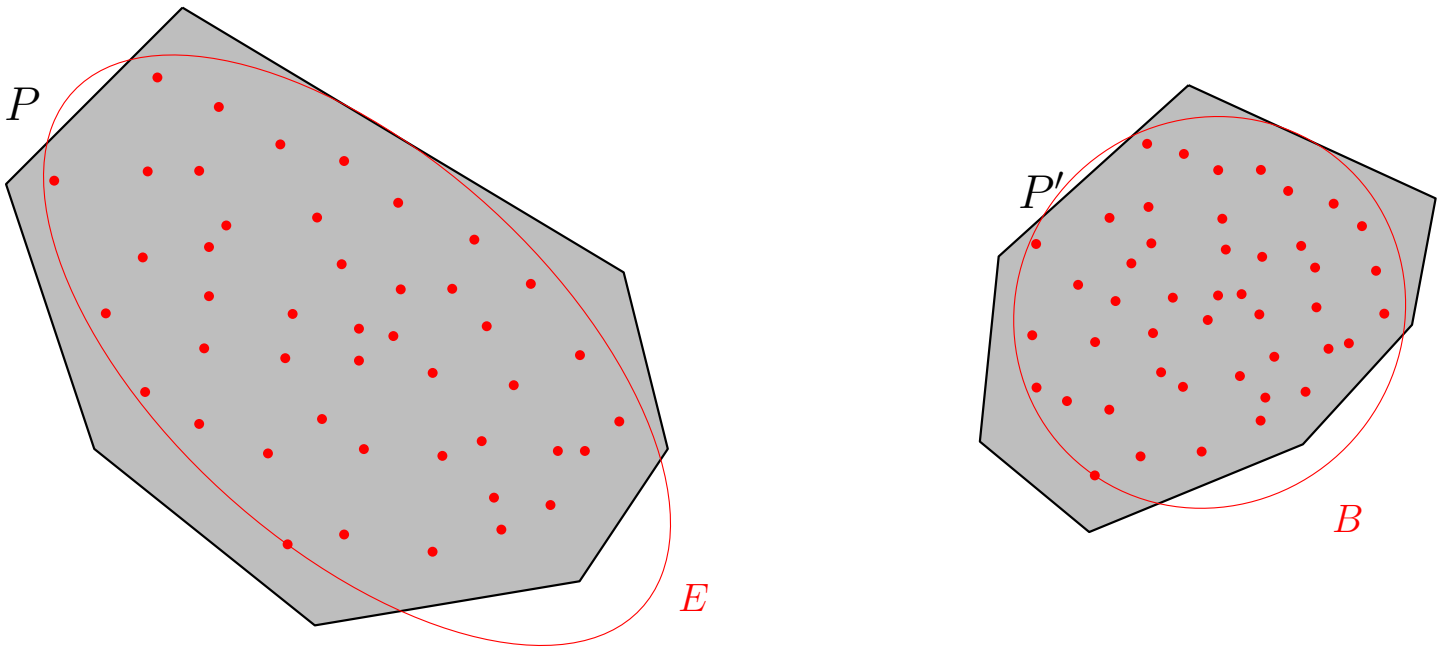
Iterate until the ratio of the maximum over the minumum ellipsoid axis reaches some user-defined threshold.



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 ($\mathrm{d} < 15$) for which exact software halts
- computed value always in $[(1-\epsilon)\mathrm{vol}(\mathrm{P}), (1+\epsilon)\mathrm{vol}(\mathrm{P})]$ (vs. prob. $3/4$ [KLS'97]) up to $\mathrm{d} = 100$

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 ($\mathrm{d} < 15$) for which exact software halts
- computed value always in $[(1-\epsilon)\mathrm{vol}(\mathrm{P}), (1+\epsilon)\mathrm{vol}(\mathrm{P})]$ (vs. prob. $3/4$ [KLS'97]) up to $\mathrm{d} = 100$
- CDHR faster and more accurate than RDHR

Experimental results

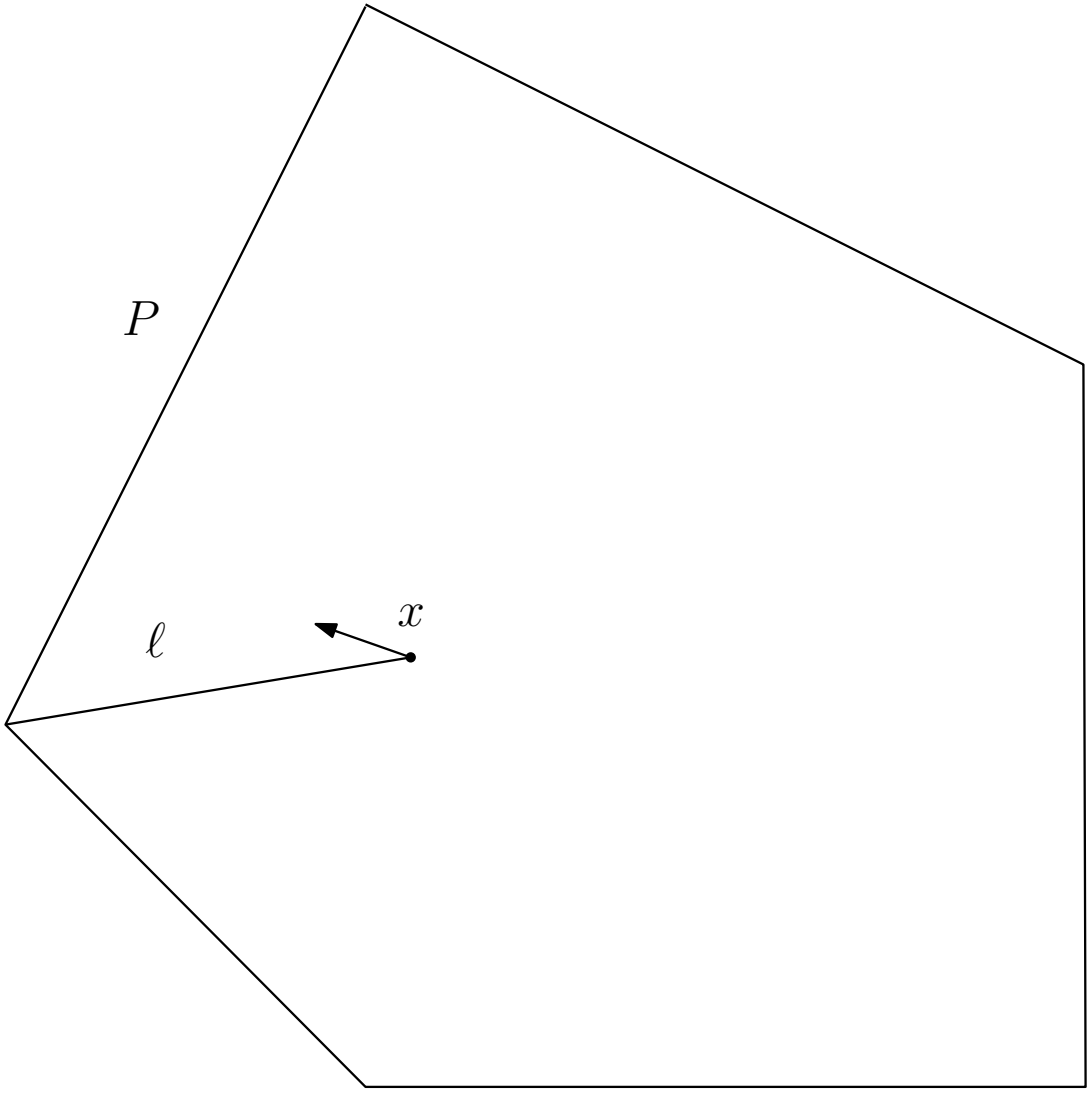
- approximate the volume of a series of polytopes (cubes, random, cross, birkhoff) up to dimension 100 in less than 2 hours with mean approximation error at most 1%
- randomized vs. exact software (`VINCI`, etc.): there is a threshold dimension ($\mathrm{d} < 15$) for which exact software halts
- computed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

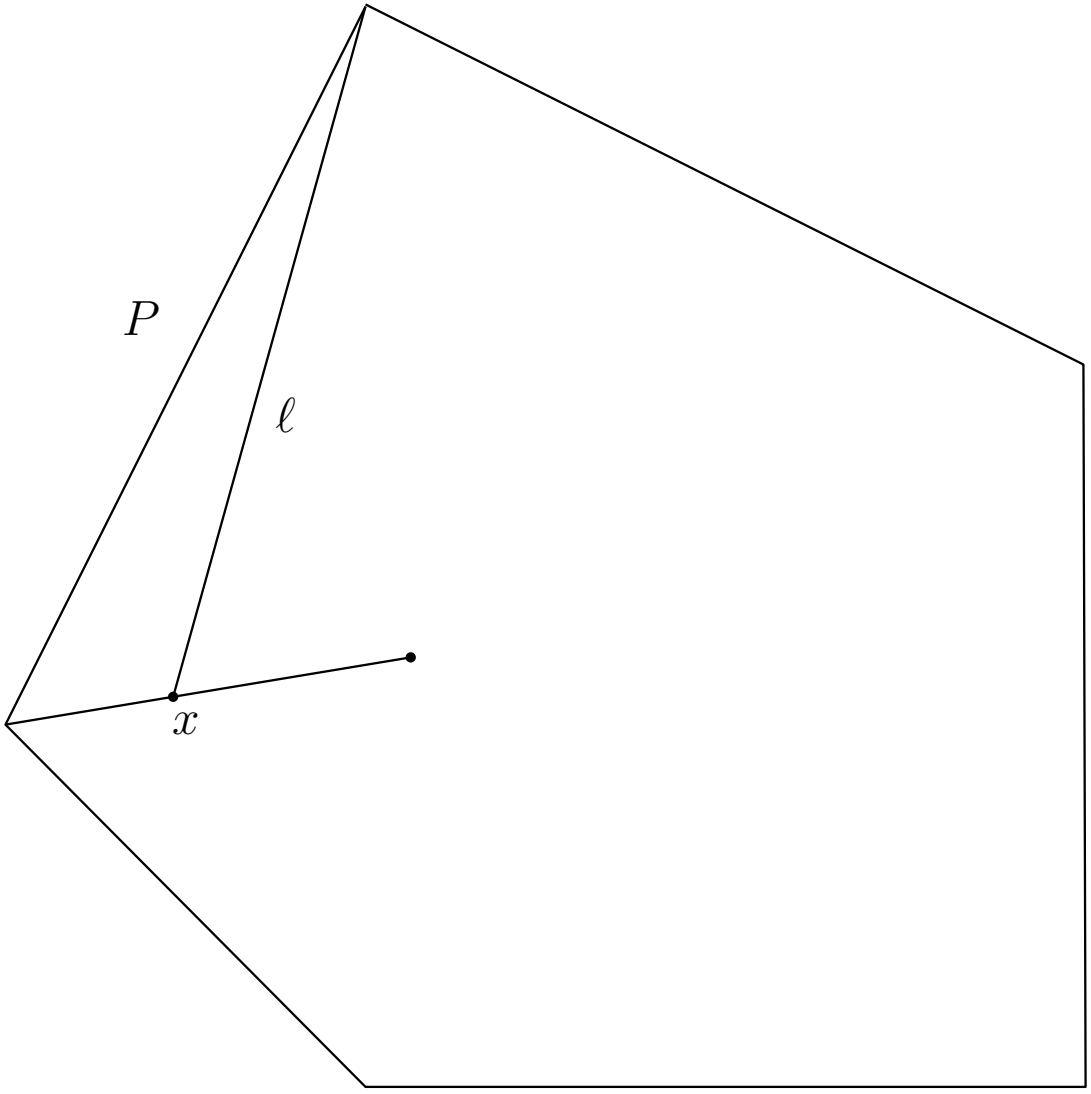
Open problem

Random walks for polytopes described by optimization oracles



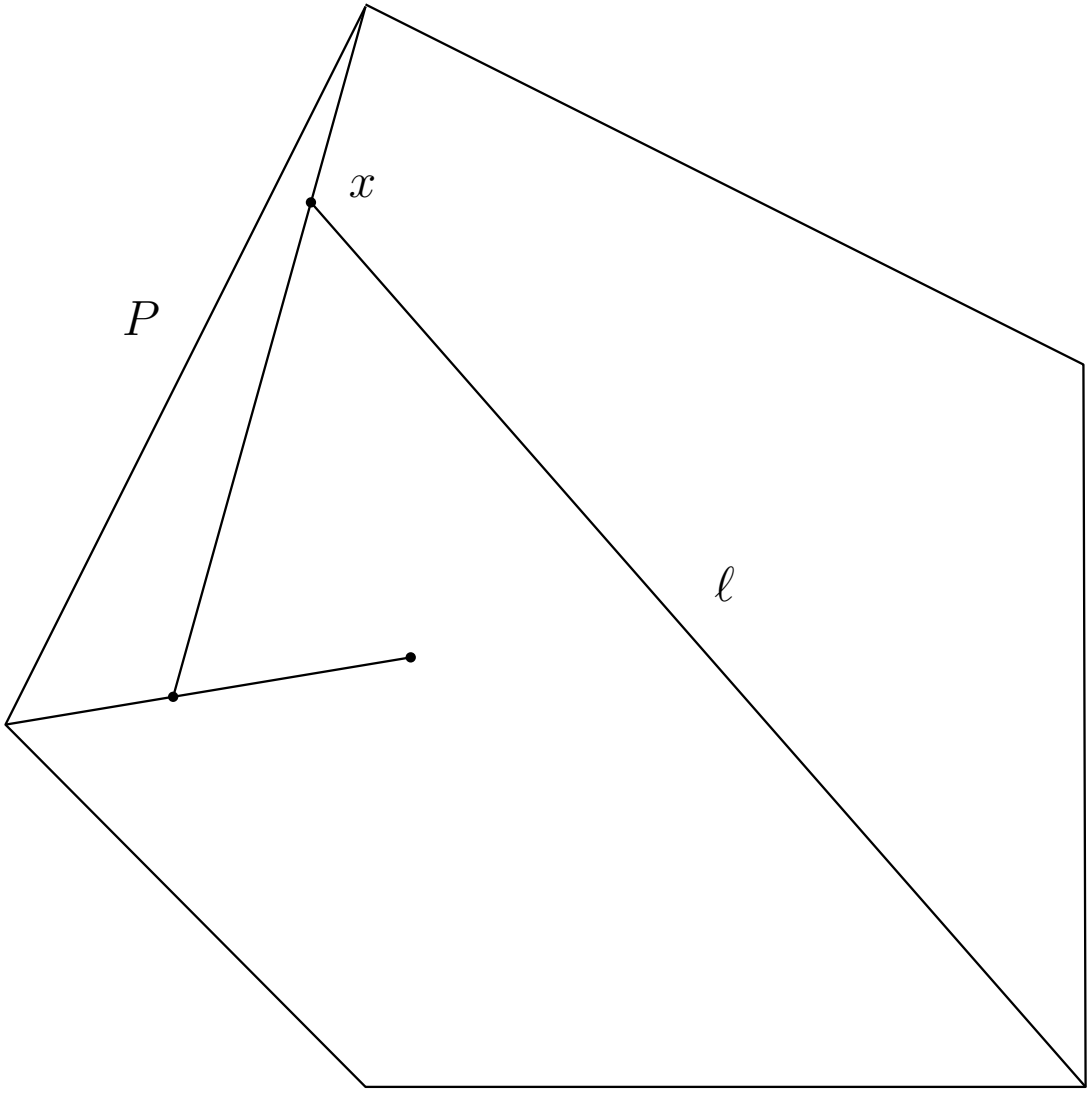
Open problem

Random walks for polytopes described by optimization oracles



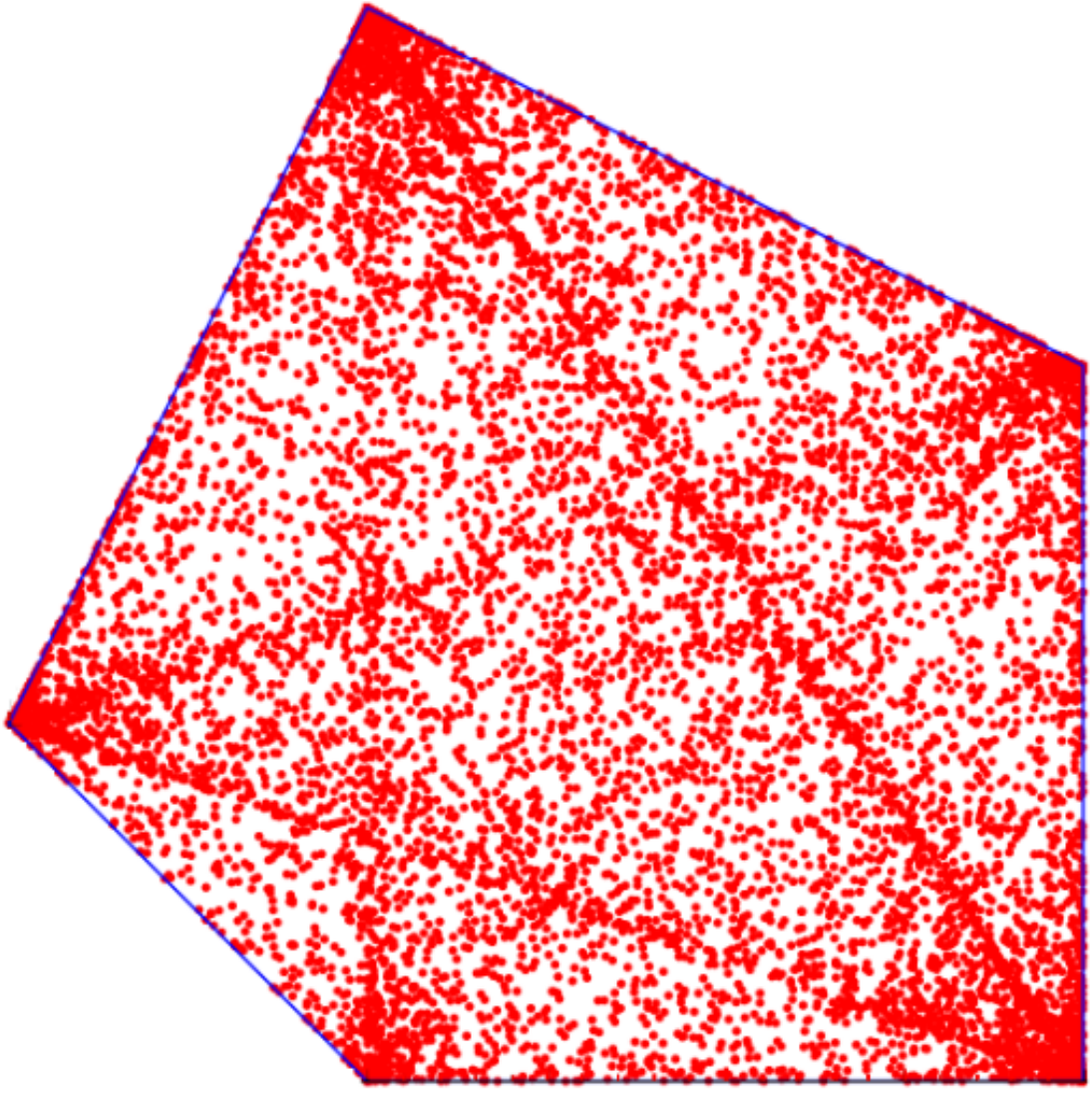
Open problem

Random walks for polytopes described by optimization oracles



Open problem

Random walks for polytopes described by optimization oracles



Ongoing and future work

1. Lattice point enumeration using random walks
   (joint with R.Yoshida, A.Campo)

2. Volume of more general convex bodies
   e.g. spectahedra

3. Other #P-hard problems
   e.g. counting linear extensions of posets

4. Use approximate oracles (utilizing approximate NN)

Conclusion

- Output-sensitive vertex enumeration for special cases of polytopes
- 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

- Output-sensitive vertex enumeration for special cases of polytopes
- 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!