Randomized Convex Optimization

Panagiotis REPOUSKOS

Sampling Methods for Convex Optimization

Panagiotis REPOUSKOS

Department of Informatics and Telecommunications, UoA

Joined work with:
A. Chalkis, I. Emiris, V. Fisikopoulos, G. Karagiorgos, E. Tsigaridas

Supported by Google Summer of Code 2019

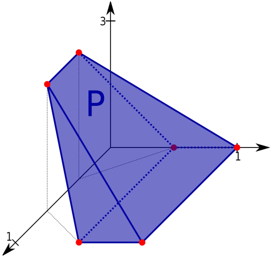
August 22, 2019

Section 1

Convex Optimization and Random Walks

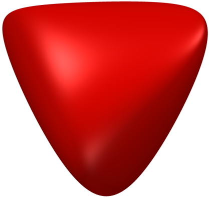
Linear Programming

$$\begin{aligned} \text{Minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$



Semidefinite Programming

$$\begin{aligned} \text{Minimize} \quad & C \cdot X \\ \text{subject to} \quad & A_i \cdot X \leq b_i \\ & X \succeq 0 \end{aligned}$$



[1] $C \cdot X := \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij}$

[2] The images were taken from Wikipedia

Randomized Convex Optimization

Panagiotis REPOUSKOS

Convex Optimization and Random Walks
Convex Optimization
Random Walks

Randomized Cutting Plane
Algorithm
Experiments

Simulated Annealing
Algorithm

Future Work

References

- We will use sampling to solve convex programs

- We will use sampling to solve convex programs
- We need to sample a convex body under some distribution

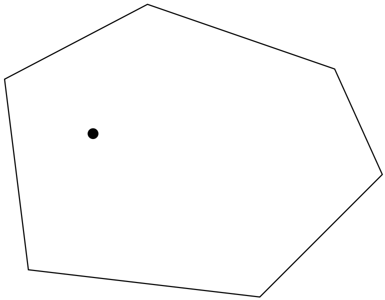
- We will use sampling to solve convex programs
- We need to sample a convex body under some distribution
- Use a geometric random walk

- We will use sampling to solve convex programs
- We need to sample a convex body under some distribution
- Use a geometric random walk
- A geometric random walk is a Markov Chain

- We will use sampling to solve convex programs
- We need to sample a convex body under some distribution
- Use a geometric random walk
- A geometric random walk is a Markov Chain
- We also need a boundary oracle

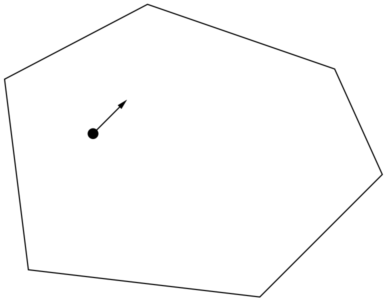
A Random Walk (Billiard Walk)

- Decide a distance H and repeat k times



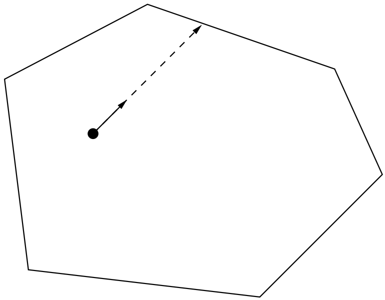
A Random Walk (Billiard Walk)

- Decide a distance H and repeat k times
  - Choose a direction and start moving



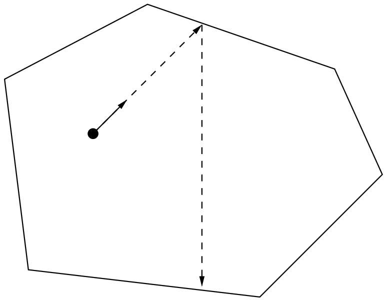
A Random Walk (Billiard Walk)

- Decide a distance H and repeat k times
  - Choose a direction and start moving
  - If reached the boundary before traveling H long, the trajectory is reflected



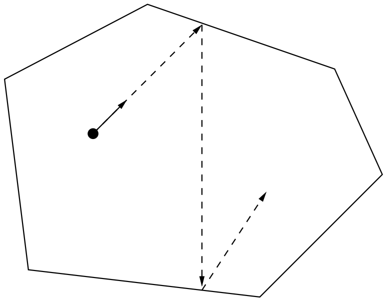
A Random Walk (Billiard Walk)

- Decide a distance H and repeat k times
  - Choose a direction and start moving
  - If reached the boundary before traveling H long, the trajectory is reflected



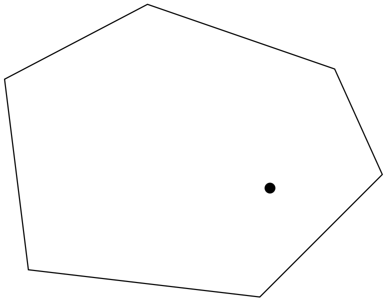
A Random Walk (Billiard Walk)

- Decide a distance H and repeat k times
  - Choose a direction and start moving
  - If reached the boundary before traveling H long, the trajectory is reflected



A Random Walk (Billiard Walk)

- Decide a distance H and repeat k times
  - Choose a direction and start moving
  - If reached the boundary before traveling H long, the trajectory is reflected

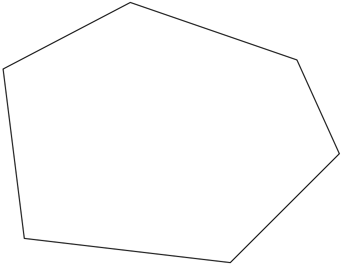


Section 2

Randomized Cutting Plane

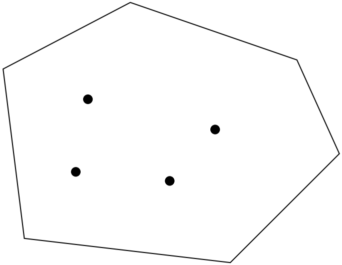
An Algorithm for Approximation

- Input: convex body K, objective function c



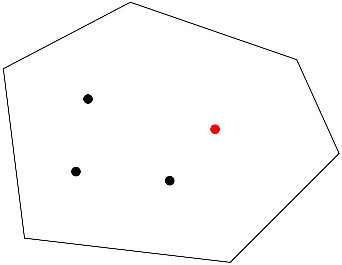
Algorithm

- Input: convex body K, objective function c
- Sample N points under the uniform distribution



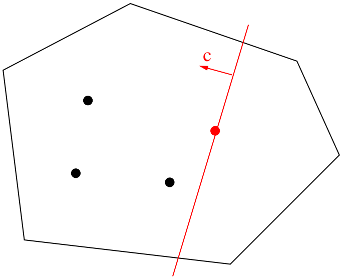
Algorithm

- Input: convex body K, objective function c
- Sample N points under the uniform distribution
- Find the point x minimizing the objective function

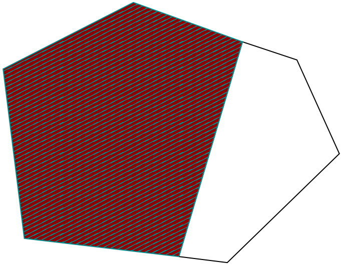


Algorithm

- Input: convex body K, objective function c
- Sample N points under the uniform distribution
- Find the point x minimizing the objective function
- Cut the convex body at x

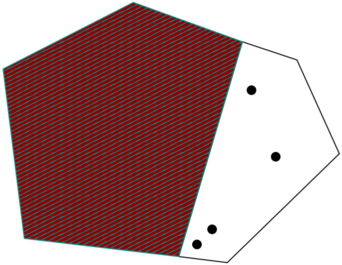


- Input: convex body K, objective function c
- Sample N points under the uniform distribution
- Find the point x minimizing the objective function
- Cut the convex body at x
- Repeat



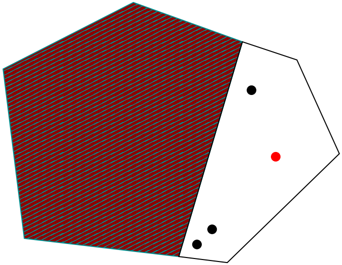
Algorithm

- Input: convex body K, objective function c
- Sample N points under the uniform distribution
- Find the point x minimizing the objective function
- Cut the convex body at x
- Repeat



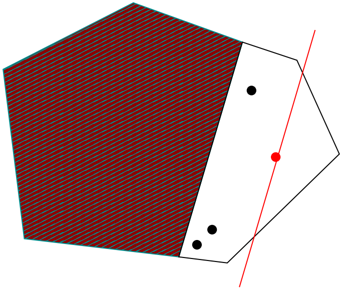
Algorithm

- Input: convex body K, objective function c
- Sample N points under the uniform distribution
- Find the point x minimizing the objective function
- Cut the convex body at x
- Repeat

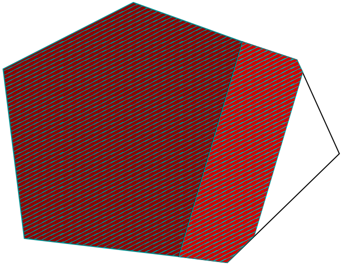


Algorithm

- Input: convex body K, objective function c
- Sample N points under the uniform distribution
- Find the point x minimizing the objective function
- Cut the convex body at x
- Repeat

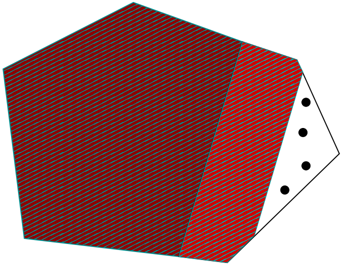


- Input: convex body K, objective function c
- Sample N points under the uniform distribution
- Find the point x minimizing the objective function
- Cut the convex body at x
- Repeat



Algorithm

- Input: convex body K, objective function c
- Sample N points under the uniform distribution
- Find the point x minimizing the objective function
- Cut the convex body at x
- Repeat



Expected Convergence [Dabbene, Shcherbakov, and Polyak]

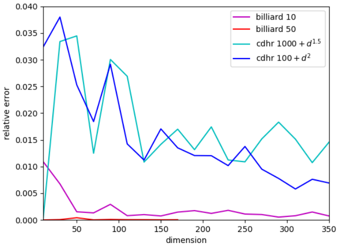
The expected number of steps k, to get a solution x_k s.t. $x_k - x^\star \leq \alpha$ is at most:

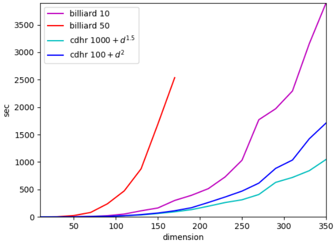
$$k = \lceil \frac{1}{\ln(N+1)} n \ln R/\alpha \rceil$$

where R is the diameter of the convex body in $\mathbb{R}^n$ and $N = \#$ samples.

Results for Polytopes

Our implementation using CDH&R[1] and Billiard Walk.



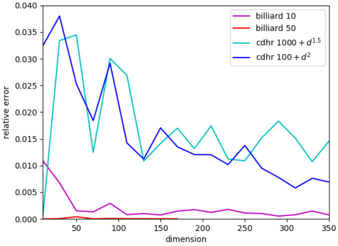


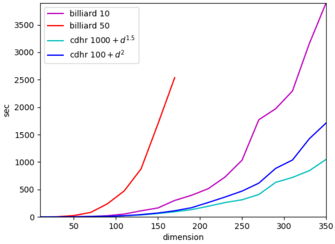
[1] Hit and Run with coordinate directions

[2] The polytopes tested are in n dimensions and have $4n$ facets.

Results for Polytopes

Our implementation using CDH&R[1] and Billiard Walk.





- Different random walks offer different benefits.

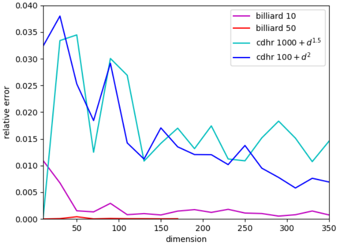
---

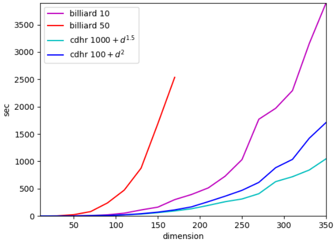
[1] Hit and Run with coordinate directions

[2] The polytopes tested are in n dimensions and have $4n$ facets.

Results for Polytopes

Our implementation using CDH&R[1] and Billiard Walk.



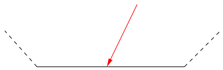


- Different random walks offer different benefits.
- Billiard offers precision, CDH&R offers speed.

[1] Hit and Run with coordinate directions

[2] The polytopes tested are in n dimensions and have $4n$ facets.

- The boundary oracle for polytopes entails linear equations.

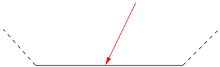


Linear Programming

$$\begin{aligned} \text{Minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

Moving to Spectrahedra

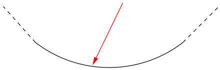
- The boundary oracle for polytopes entails linear equations.



Linear Programming

$$\begin{array}{ll} \text{Minimize} & c^\top x \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{array}$$

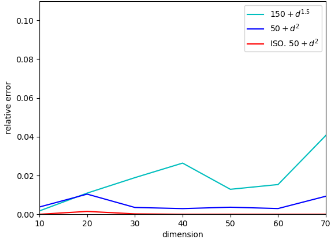
- The boundary oracle for spectrahedra entails eigenvalues (equivalently polynomial equations).

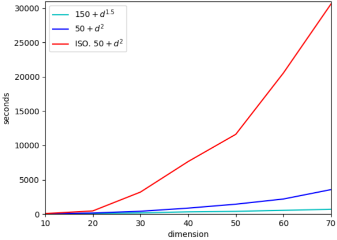


Semidefinite Programming

$$\begin{array}{ll} \text{Minimize} & C \cdot X \\ \text{subject to} & A_i \cdot X \leq b_i \\ & X \succeq 0 \end{array}$$

Using RDH&R[1] with / without implicit isotropization.



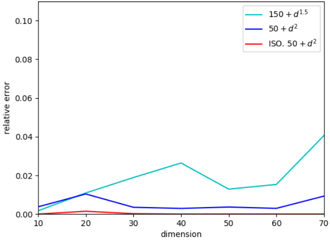


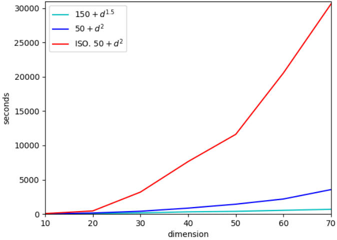
[1] Hit and Run with random directions

[2] The matrices in the experiments are of dimension 50

H&R on Spectrahedra

Using RDH&R[1] with / without implicit isotropization.





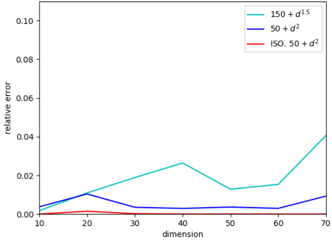
- This random walk doesn't scale well with dimensions

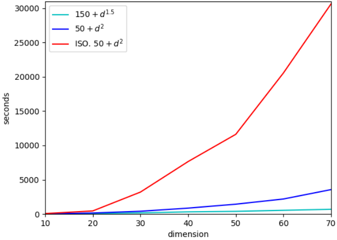
[1] Hit and Run with random directions

[2] The matrices in the experiments are of dimension 50

H&R on Spectrahedra

Using RDH&R[1] with / without implicit isotropization.





- This random walk doesn't scale well with dimensions
- We need better sampling for spectrahedra

[1] Hit and Run with random directions

[2] The matrices in the experiments are of dimension 50

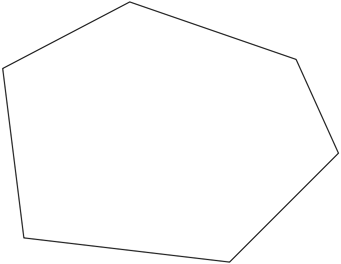
Section 3

Simulated Annealing

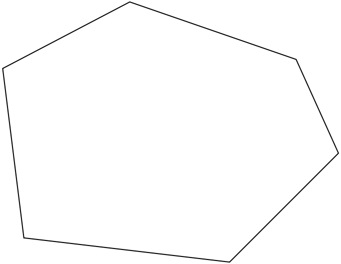
Beyond Uniform Distribution

Algorithm

- For $i = 1, 2, ..., I$

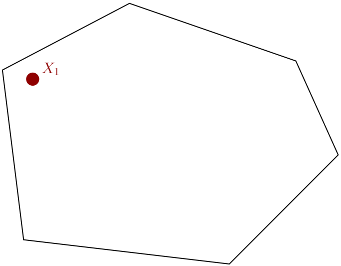


- For $i = 1, 2, \ldots, I$
  - set temperature $T_i = R\left(1 - \frac{1}{\sqrt{n}}\right)^i$

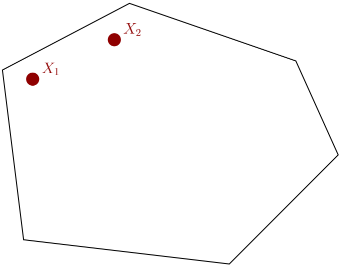


Algorithm

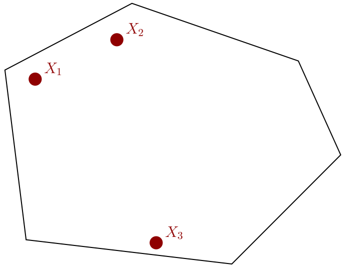
- For $i = 1, 2, ..., I$
  - set temperature $T_i = R\left(1 - \frac{1}{\sqrt{n}}\right)^i$
  - get a sample X_i using H&R with density $e^{-c \cdot x/T}$



- For $i = 1, 2, ..., I$
  - set temperature $T_i = R\left(1 - \frac{1}{\sqrt{n}}\right)^i$
  - get a sample X_i using H&R with density $e^{-c \cdot x/T}$

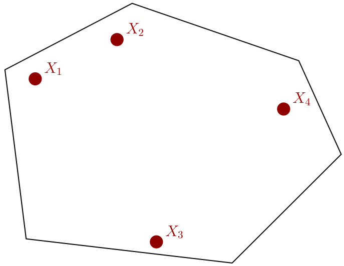


- For $i = 1, 2, ..., I$
  - set temperature $T_i = R\left(1 - \frac{1}{\sqrt{n}}\right)^i$
  - get a sample X_i using H&R with density $e^{-c \cdot x/T}$



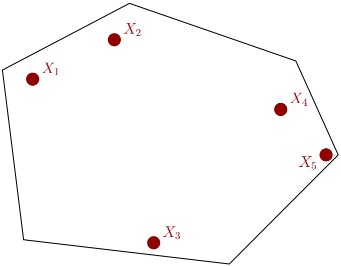
Algorithm

- For $i = 1, 2, ..., I$
  - set temperature $T_i = R\left(1 - \frac{1}{\sqrt{n}}\right)^i$
  - get a sample X_i using H&R with density $e^{-c \cdot x / T}$



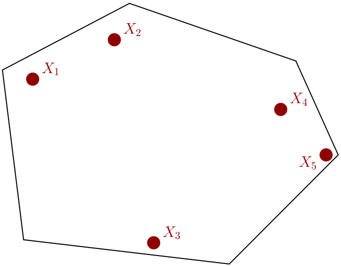
Algorithm

- For $i = 1, 2, ..., I$
  - set temperature $T_i = R\left(1 - \frac{1}{\sqrt{n}}\right)^i$
  - get a sample X_i using H&R with density $e^{-c \cdot x/T}$



Algorithm

- For $i = 1, 2, ..., I$
  - set temperature $T_i = R\left(1 - \frac{1}{\sqrt{n}}\right)^i$
  - get a sample X_i using H&R with density $e^{-c \cdot x/T}$



- Return X_I

Theorem [Kalai and Vempala]

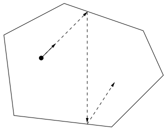
With probability $1-\delta$, $I = O(\sqrt{n}\log(Rn/\epsilon\delta))$ get a X_I s.t.

$$c \cdot X_I \leq \min_{x \in K} c \cdot x + \epsilon$$

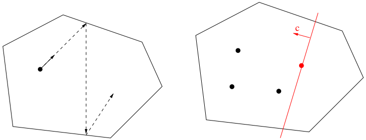
Section 4

Future Work

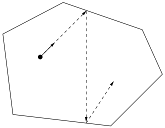
Future Work

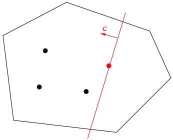


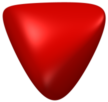
Future Work



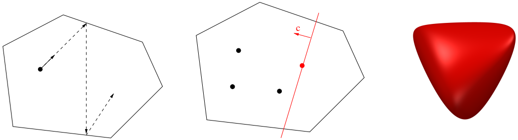
Future Work





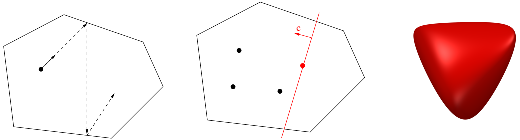


Future Work



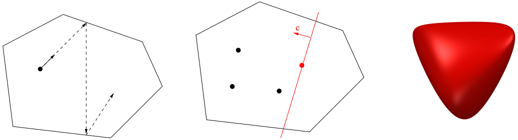
- Billiard Walk for spectrahedra - need to efficiently compute reflections

Future Work



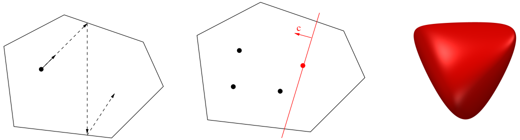
- Billiard Walk for spectrahedra - need to efficiently compute reflections
- Efficient boundary oracle for CDH&R in spectrahedra

Future Work



- Billiard Walk for spectrahedra - need to efficiently compute reflections
- Efficient boundary oracle for CDH&R in spectrahedra
- Study behavior of simulated annealing

Future Work



- Billiard Walk for spectrahedra - need to efficiently compute reflections
- Efficient boundary oracle for CDH&R in spectrahedra
- Study behavior of simulated annealing
- Try new random walks, including Hamiltonian Monte Carlo

R project for statistical computing

Google Summer of Code 2019

Project: Sampling Methods for Convex Optimization

Mentors: Vissarion Fisikopoulos
Elias Tsigaridas
Zafeirakis Zafeirakopoulos

[1] https://github.com/GeomScale/volume_approximation

References

F. Dabbene, P. S. Shcherbakov, and B. T. Polyak. A Randomized Cutting Plane Method with Probabilistic Geometric Convergence. *SIAM Journal on Optimization*, 20:3185–3207, Jan. 2010. ISSN 1052-6234, 1095-7189. doi: 10.1137/080742506. URL http://epubs.siam.org/doi/10.1137/080742506.

A. T. Kalai and S. Vempala. Simulated Annealing for Convex Optimization. *Mathematics of Operations Research*, 31(2):253–266, May 2006. ISSN 0364-765X, 1526-5471. doi: 10.1287/moor.1060.0194. URL http://pubsonline.informs.org/doi/abs/10.1287/moor.1060.0194.