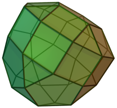
Smoothed Analysis of Algorithms
Part I: Simplex Method and Local Search

Heiko Röglin
Department of Computer Science
universität bonn

16 January 2013

Many problems and algorithms seem well understood.

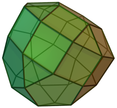


Linear Programming
efficient algorithms (ellipsoid, interior point)

Knapsack Problem (KP)
NP-hard, FPTAS exists

Traveling Salesperson Problem (TSP)
NP-hard, even hard to approximate

Many problems and algorithms seem well understood.



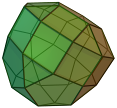
Linear Programming
efficient algorithms (ellipsoid, interior point)
Simplex method performs well in practice.

Knapsack Problem (KP)
NP-hard, FPTAS exists
very easy problem, solvable in almost linear time

Traveling Salesperson Problem (TSP)
NP-hard, even hard to approximate
local search methods yield very good solutions

Discrete Optimization

Many problems and algorithms seem well understood.



Linear Programming

efficient algorithms (ellipsoid, interior point)

Simplex method performs well in practice.

Knapsack Problem (KP)

NP-hard, FPTAS exists

very easy problem, solvable in almost linear time

Traveling Salesperson Problem (TSP)

NP-hard, even hard to approximate

local search methods yield very good solutions

$\Rightarrow$ big gap between theory and practice

Outline

Outline

1. **Linear Programming**
   Why is the simplex method usually efficient?
   Smoothed Analysis – analysis of algorithms beyond worst case
2. **Traveling Salesperson Problem**
   Why is local search successful?
3. **Smoothed Analysis**
   Overview of known results

Outline

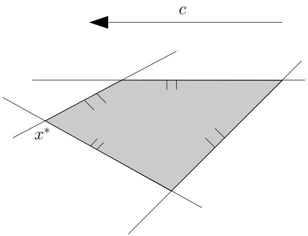
Outline

1. **Linear Programming**
   Why is the simplex method usually efficient?
   Smoothed Analysis – analysis of algorithms beyond worst case
2. **Traveling Salesperson Problem**
   Why is local search successful?
3. **Smoothed Analysis**
   Overview of known results

Linear Programs (LPs)

- variables: $x_1, \ldots, x_n \in \mathbb{R}$
- linear objective function:
  $\max c^T x = c_1 x_1 + \ldots + c_n x_n$
- m linear constraints:

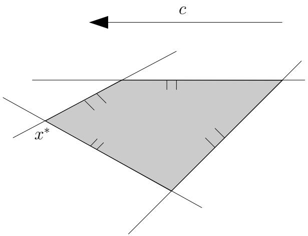
$$\begin{array}{ccc} a_{1,1}x_1 & + \ldots + & a_{1,n}x_n \leq b_1 \\ & \vdots & \\ a_{m,1}x_1 & + \ldots + & a_{m,n}x_n \leq b_m \end{array}$$



Linear Programs (LPs)

- variables: $x_1, \ldots, x_n \in \mathbb{R}$
- linear objective function:
  $\max c^T x = c_1 x_1 + \ldots + c_n x_n$
- m linear constraints:

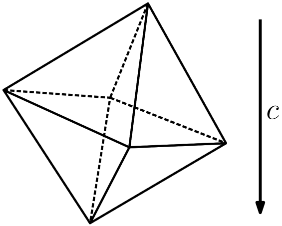
$$\begin{array}{ccc} a_{1,1}x_1 & + \ldots + & a_{1,n}x_n \leq b_1 \\ & \vdots & \\ a_{m,1}x_1 & + \ldots + & a_{m,n}x_n \leq b_m \end{array}$$



Complexity of LPs

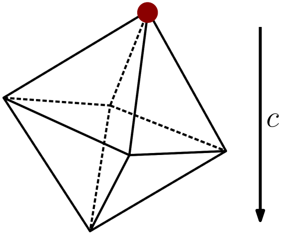
LPs can be solved in **polynomial time** by the ellipsoid method [Khachiyan 1979] and the interior point method [Karmarkar 1984].

Simplex Algorithm



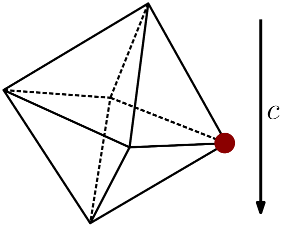
Simplex Algorithm

- Start at some vertex of the polytope.



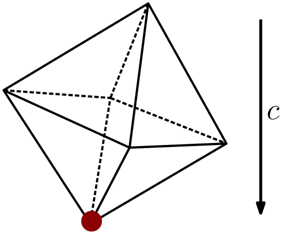
Simplex Algorithm

- Start at some vertex of the polytope.



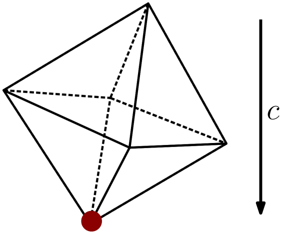
Simplex Algorithm

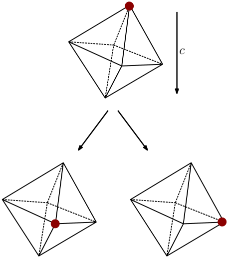
- Start at some vertex of the polytope.
- Walk along the edges of the polytope in the direction of the objective function $c^T x$.



Simplex Algorithm

- Start at some vertex of the polytope.
- Walk along the edges of the polytope in the direction of the objective function $c^T x$.
- local optimum $=$ global optimum





Simplex Algorithm

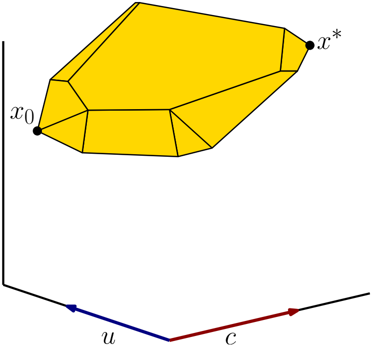
- Start at some vertex of the polytope.
- Walk along the edges of the polytope in the direction of the objective function $c^T x$.
- local optimum $=$ global optimum

Pivot Rules

- Which vertex is chosen if there are multiple options?
- Different **pivot rules** suggested: random, steepest descent, shadow vertex pivot rule, ...

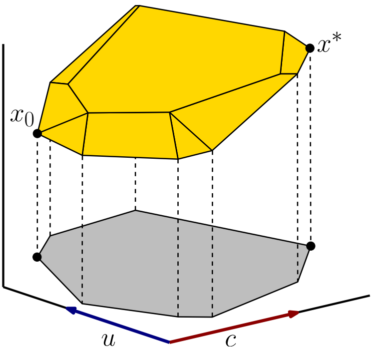
Shadow Vertex Pivot Rule

- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.



Shadow Vertex Pivot Rule

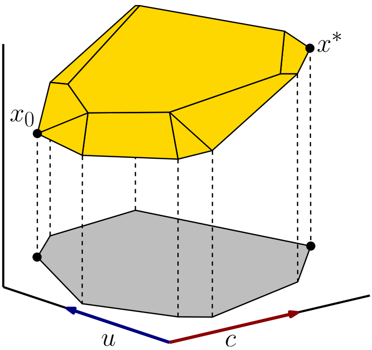
- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.



Simplex Algorithm – Shadow Vertex Pivot Rule

Shadow Vertex Pivot Rule

- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.

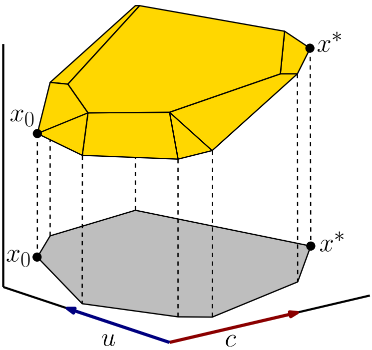


- The shadow is a polygon.

Simplex Algorithm – Shadow Vertex Pivot Rule

Shadow Vertex Pivot Rule

- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.

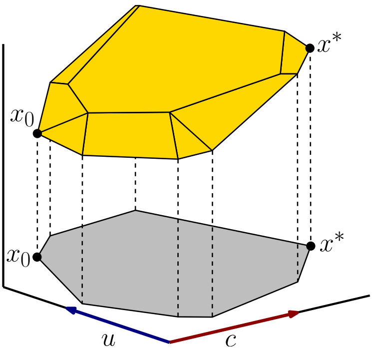


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.

Simplex Algorithm – Shadow Vertex Pivot Rule

Shadow Vertex Pivot Rule

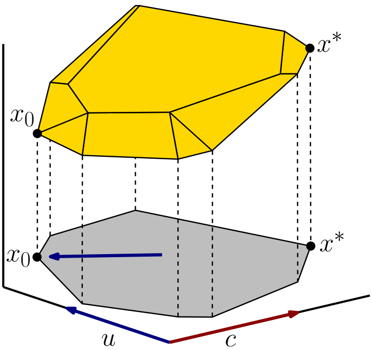
- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.



- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

Shadow Vertex Pivot Rule

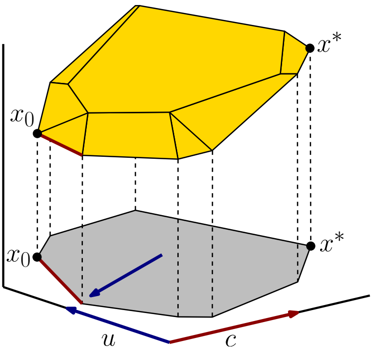
- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.
- **Start at x_0 and follow the edges of the shadow.**



- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

Shadow Vertex Pivot Rule

- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.
- **Start at x_0 and follow the edges of the shadow.**

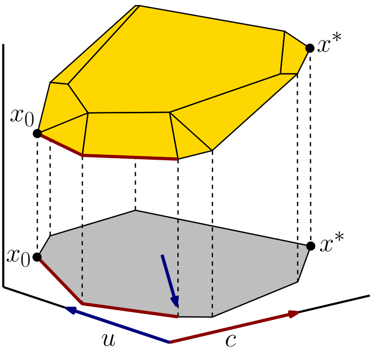


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

Simplex Algorithm – Shadow Vertex Pivot Rule

Shadow Vertex Pivot Rule

- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.
- **Start at x_0 and follow the edges of the shadow.**

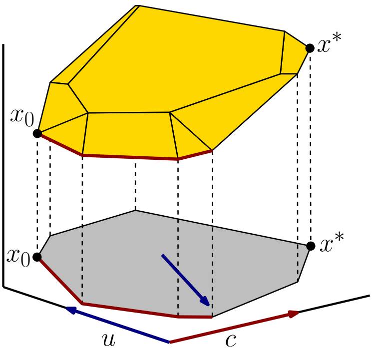


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

Simplex Algorithm – Shadow Vertex Pivot Rule

Shadow Vertex Pivot Rule

- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.
- **Start at x_0 and follow the edges of the shadow.**

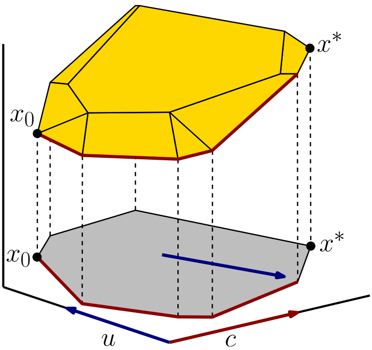


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

Simplex Algorithm – Shadow Vertex Pivot Rule

Shadow Vertex Pivot Rule

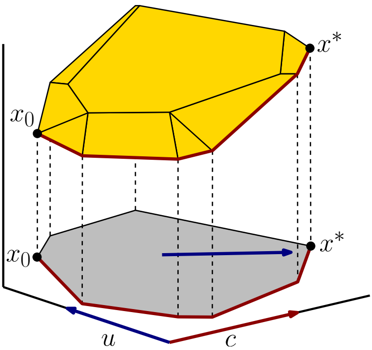
- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.
- **Start at x_0 and follow the edges of the shadow.**



- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

Shadow Vertex Pivot Rule

- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.
- **Start at x_0 and follow the edges of the shadow.**

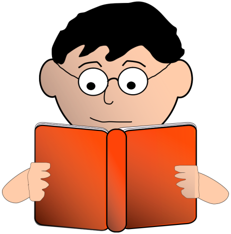


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

Simplex Algorithm – Running Time

Theoreticians say...

- shadow vertex pivot rule requires exponential number of steps
- no pivot rule with sub-exponential number of steps known
- ellipsoid and interior point methods are efficient



Theoreticians say...

- shadow vertex pivot rule requires exponential number of steps
- no pivot rule with sub-exponential number of steps known
- ellipsoid and interior point methods are efficient

Engineers say...

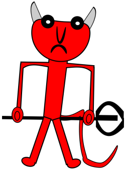
- simplex method usually fastest algorithm in practice
- requires usually only $\Theta(m)$ steps
- clearly outperforms ellipsoid method

Reason for Gap between Theory and Practice

- Worst-case complexity is too pessimistic!
- There are (artificial) worst-case LPs on which the simplex method is not efficient. These LPs, however, do not occur in practice.

  e.g., $a_{1,i} = 2^i$, $\sum_i a_{2,i} \equiv 3 \bmod 5$, $\ldots$

Adversary

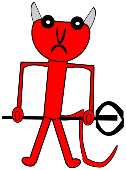


"I will trick your algorithm!"

Reason for Gap between Theory and Practice

- Worst-case complexity is too pessimistic!
- There are (artificial) worst-case LPs on which the simplex method is not efficient. These LPs, however, do not occur in practice.
  e.g., $a_{1,i} = 2^i$, $\sum_i a_{2,i} \equiv 3 \bmod 5$, $\ldots$
- This phenomenon occurs not only for the simplex method, but also for many other problems and algorithms.

Adversary

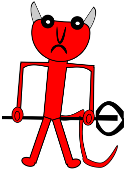


"I will trick your algorithm!"

Reason for Gap between Theory and Practice

- Worst-case complexity is too pessimistic!
- There are (artificial) worst-case LPs on which the simplex method is not efficient. These LPs, however, do not occur in practice.
  e.g., $a_{1,i} = 2^i$, $\sum_i a_{2,i} \equiv 3 \bmod 5$, $\ldots$
- This phenomenon occurs not only for the simplex method, but also for many other problems and algorithms.

Adversary



"I will trick your algorithm!"

Goal

Find a more realistic performance measure that is not just based on the worst case.

Observation: In worst-case analysis, the adversary is too powerful.

Idea: Let's weaken him!

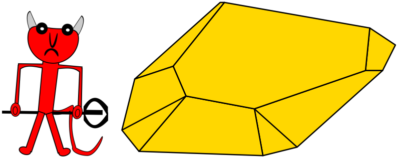
Smoothed Analysis

Observation: In worst-case analysis, the adversary is too powerful.

Idea: Let's weaken him!

Perturbed LPs

- Step 1: Adversary specifies arbitrary LP:
  $\max c^T x$ subject to $a_1^T x \leq b_1 \ldots a_n^T x \leq b_n$.
  W. l. o. g. $\|(a_i, b_i)\| = 1$.



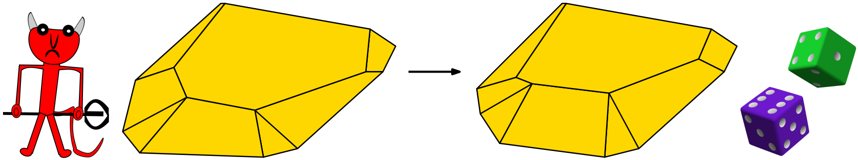
Smoothed Analysis

Observation: In worst-case analysis, the adversary is too powerful.

Idea: Let's weaken him!

Perturbed LPs

- Step 1: Adversary specifies arbitrary LP:
  max $c^T x$ subject to $a_1^T x \leq b_1 \ldots a_n^T x \leq b_n$.
  W. l. o. g. $\|(a_i, b_i)\| = 1$.
- Step 2: Add Gaussian random variable with standard deviation σ to each coefficient in the constraints.

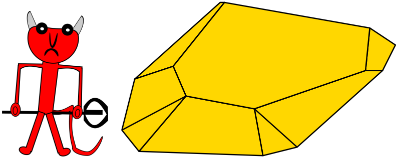


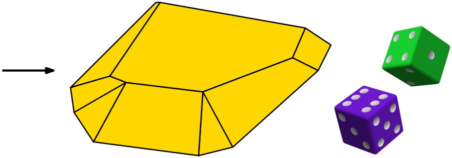
Observation: In worst-case analysis, the adversary is too powerful.

Idea: Let's weaken him!

Perturbed LPs

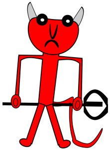
- Step 1: Adversary specifies arbitrary LP:
  $\max c^T x$ subject to $a_1^T x \leq b_1 \ldots a_n^T x \leq b_n$.
  W. l. o. g. $\|(a_i, b_i)\| = 1$.
- Step 2: Add Gaussian random variable with standard deviation σ to each coefficient in the constraints.



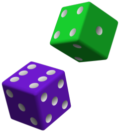


Smoothed Running Time

$=$ worst expected running time the adversary can achieve



Step 1: Adversary chooses input I

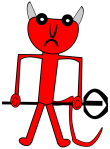


Step 2: Random perturbation $I \to \mathrm{per}_\sigma(I)$

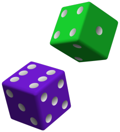
Formal Definition:

$\mathrm{LP}(n, m)$ = set of LPs with n variables and m constraints

$T(I)$ = number of steps of simplex method on LP I



Step 1: Adversary chooses input I



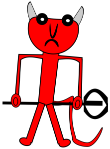
Step 2: Random perturbation $I \to \mathrm{per}_\sigma(I)$

Formal Definition:

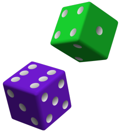
$\mathrm{LP}(n, m)$ = set of LPs with n variables and m constraints

$T(I)$ = number of steps of simplex method on LP I

smoothed run time $T^{\mathrm{smooth}}(n, m, \sigma) = \max_{I \in \mathrm{LP}(n,m)} \mathbf{E}[T(\mathrm{per}_\sigma(I))]$



Step 1:
Adversary
chooses input I



Step 2: Random
perturbation
$I \to \mathrm{per}_\sigma(I)$

Formal Definition:

$\mathrm{LP}(n, m)$ = set of LPs with n variables and m constraints

$T(I)$ = number of steps of simplex method on LP I

smoothed run time $T^{\mathrm{smooth}}(n, m, \sigma) = \max_{I \in \mathrm{LP}(n,m)} \mathbf{E}\left[T(\mathrm{per}_\sigma(I))\right]$

Why do we consider this model?

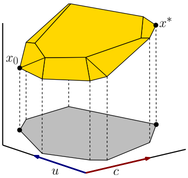
- First step models unknown structure of the input.
- Second step models random influences, e.g., measurement errors, numerical imprecision, rounding, ...
- smoothed running time low $\Rightarrow$ bad instances are unlikely to occur
- σ determines the amount of randomness

Smoothed Analysis of the Simplex Algorithm

Lemma [Spielman and Teng (STOC 2001)]

For every fixed plane and every LP the adversary can choose, after the perturbation, the expected number of edges on the shadow is

$$O\left(\mathrm{poly}\left(n, m, \sigma^{-1}\right)\right).$$

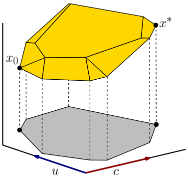


Smoothed Analysis of the Simplex Algorithm

Lemma [Spielman and Teng (STOC 2001)]

For every fixed plane and every LP the adversary can choose, after the perturbation, the expected number of edges on the shadow is

$$O\left(\mathrm{poly}\left(n, m, \sigma^{-1}\right)\right).$$



Theorem [Spielman and Teng (STOC 2001)]

The smoothed running time of the simplex algorithm with shadow vertex pivot rule is

$$O\left(\mathrm{poly}\left(n, m, \sigma^{-1}\right)\right).$$

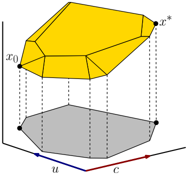
Already for small perturbations exponential running time is unlikely.

Smoothed Analysis of the Simplex Algorithm

Lemma [Spielman and Teng (STOC 2001)]

For every fixed plane and every LP the adversary can choose, after the perturbation, the expected number of edges on the shadow is

$$O\left(\mathrm{poly}\left(n, m, \sigma^{-1}\right)\right).$$



Theorem [Spielman and Teng (STOC 2001)]

The smoothed running time of the simplex algorithm with shadow vertex pivot rule is

$$O\left(\mathrm{poly}\left(n, m, \sigma^{-1}\right)\right).$$

Already for small perturbations exponential running time is unlikely.

Main Difficulties in Proof of Theorem:

- x_0 is found in phase I $\to$ no Gaussian distribution of coefficients
- In phase II, the plane onto which the polytope is projected is not independent of the perturbations.

Improved Analysis

Theorem [Vershynin (FOCS 2006)]

The smoothed number of steps of the simplex algorithm with shadow vertex pivot rule is

$$O\left(\mathrm{poly}\left(n, \log m, \sigma^{-1}\right)\right).$$

only polylogarithmic in the number of constraints m

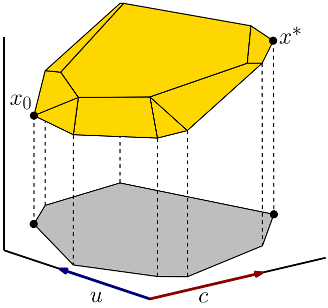
Improved Analysis

Theorem [Vershynin (FOCS 2006)]

The smoothed number of steps of the simplex algorithm with shadow vertex pivot rule is

$$O\left(\mathrm{poly}\left(n, \log m, \sigma^{-1}\right)\right).$$

only polylogarithmic in the number of constraints m



- Phase I: add vertex x_0 in random direction. With constant prob. this does not change optimal solution.
  $\Rightarrow$ The plane is not correlated with the perturbed polytope.

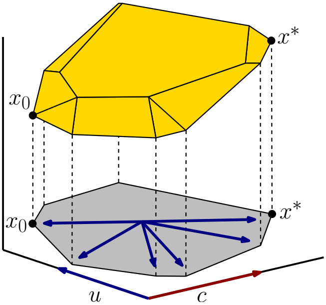
Improved Analysis

Theorem [Vershynin (FOCS 2006)]

The smoothed number of steps of the simplex algorithm with shadow vertex pivot rule is

$$O\left(\mathrm{poly}\left(n, \log m, \sigma^{-1}\right)\right).$$

only polylogarithmic in the number of constraints m



- Phase I: add vertex x_0 in random direction. With constant prob. this does not change optimal solution.
  $\Rightarrow$ The plane is not correlated with the perturbed polytope.
- With high prob. no angle between consecutive vertices is too small.

Outline

Outline

1. **Linear Programming**
   Why is the simplex method usually efficient?
   smoothed analysis – analysis of algorithms beyond worst case
2. **Traveling Salesperson Problem**
   Why is local search successful?
3. **Smoothed Analysis**
   Overview of known results

Traveling Salesperson Problem (TSP)

- **Input:** weighted (complete) graph $G = (V, E, d)$ with $d : E \to \mathbb{R}_+$

Traveling Salesperson Problem

Traveling Salesperson Problem (TSP)

- **Input:** weighted (complete) graph $G = (V, E, d)$ with $d : E \to \mathbb{R}_+$
- **Goal:** Find Hamiltonian cycle of minimum length.

Traveling Salesperson Problem (TSP)

- **Input:** weighted (complete) graph $G = (V, E, d)$ with $d : E \to \mathbb{R}_+$
- **Goal:** Find Hamiltonian cycle of minimum length.

One of the most intensively studied problems in optimization – **both in theory and practice**.

Traveling Salesperson Problem (TSP)

- **Input:** weighted (complete) graph $G = (V, E, d)$ with $d : E \to \mathbb{R}_+$
- **Goal:** Find Hamiltonian cycle of minimum length.

One of the most intensively studied problems in optimization – **both in theory and practice**.

Metric TSP: APX-hard **Euclidean TSP:** PTAS exists

2-Opt Algorithm

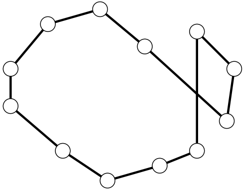
Numerous Experimental Studies:
(TSPLIB, DIMACS Implementation Challenge)

- The PTAS is too slow on large instances.
- The most successful algorithms (w. r. t. quality and running time) in practice rely on local search.

Numerous Experimental Studies:
(TSPLIB, DIMACS Implementation Challenge)

- The PTAS is too slow on large instances.
- The most successful algorithms (w. r. t. quality and running time) in practice rely on local search.

2-Opt:



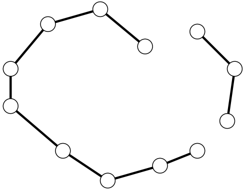
1. Start with an arbitrary tour.

2-Opt Algorithm

Numerous Experimental Studies:
(TSPLIB, DIMACS Implementation Challenge)

- The PTAS is too slow on large instances.
- The most successful algorithms (w. r. t. quality and running time) in practice rely on local search.

2-Opt:



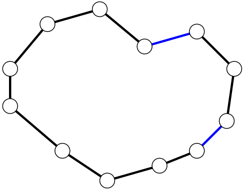
1. Start with an arbitrary tour.
2. Remove two edges from the tour.

2-Opt Algorithm

Numerous Experimental Studies:
(TSPLIB, DIMACS Implementation Challenge)

- The PTAS is too slow on large instances.
- The most successful algorithms (w. r. t. quality and running time) in practice rely on local search.

2-Opt:



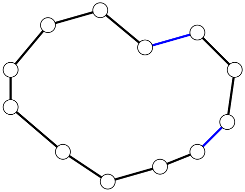
1. Start with an arbitrary tour.
2. Remove two edges from the tour.
3. Complete the tour by two other edges.

2-Opt Algorithm

Numerous Experimental Studies:
(TSPLIB, DIMACS Implementation Challenge)

- The PTAS is too slow on large instances.
- The most successful algorithms (w. r. t. quality and running time) in practice rely on local search.

2-Opt:

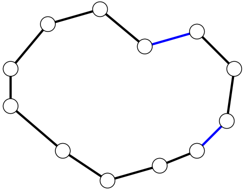


1. Start with an arbitrary tour.
2. Remove two edges from the tour.
3. Complete the tour by two other edges.
4. Repeat steps 2 and 3 until no local improvement is possible anymore.

Numerous Experimental Studies:
(TSPLIB, DIMACS Implementation Challenge)

- The PTAS is too slow on large instances.
- The most successful algorithms (w. r. t. quality and running time) in practice rely on local search.
- approximation ratio: ≈ 1.05
  number of steps: $\leq n \cdot \log n$

2-Opt:



1. Start with an arbitrary tour.
2. Remove two edges from the tour.
3. Complete the tour by two other edges.
4. Repeat steps 2 and 3 until no local improvement is possible anymore.

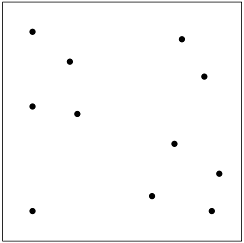
Smoothed Analysis

Worst Case [Englert, R., Vöcking (SODA 2007)]

Even for 2-dim. Euclidean instances, the worst-case run time is $2^{\Omega(n)}$.

Worst Case [Englert, R., Vöcking (SODA 2007)]

Even for 2-dim. Euclidean instances, the worst-case run time is $2^{\Omega(n)}$.

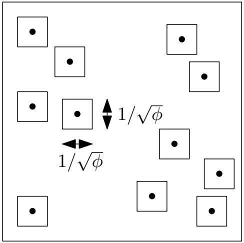


Smoothed Analysis:

Adversary chooses for each point i a probability density $f_i : [0,1]^d \to [0,\phi]$ according to which it is chosen.

Worst Case [Englert, R., Vöcking (SODA 2007)]

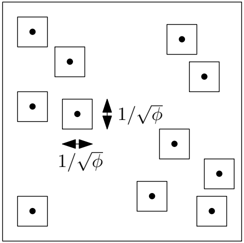
Even for 2-dim. Euclidean instances, the worst-case run time is $2^{\Omega(n)}$.



Smoothed Analysis:
Adversary chooses for each point i a probability density $f_i : [0, 1]^d \to [0, \phi]$ according to which it is chosen.

Worst Case [Englert, R., Vöcking (SODA 2007)]

Even for 2-dim. Euclidean instances, the worst-case run time is $2^{\Omega(n)}$.



Smoothed Analysis:

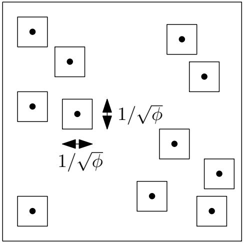
Adversary chooses for each point i a probability density $f_i : [0,1]^d \to [0,\phi]$ according to which it is chosen.

Adversary more powerful than before. He determines also the type of noise. $\phi \sim 1/\sigma$

Smoothed Analysis

Worst Case [Englert, R., Vöcking (SODA 2007)]

Even for 2-dim. Euclidean instances, the worst-case run time is $2^{\Omega(n)}$.



Smoothed Analysis:

Adversary chooses for each point i a probability density $f_i : [0,1]^d \to [0,\phi]$ according to which it is chosen.

Adversary more powerful than before. He determines also the type of noise. $\phi \sim 1/\sigma$

Smoothed Analysis [Englert, R., Vöcking (SODA 2007)]

The smoothed number of 2-Opt steps is $\tilde{O}(n^{4.33} \cdot \phi^{2.67})$.

Simple Polynomial Bound

Theorem

The smoothed number of 2-Opt steps is $O(n^7 \phi^3 \log^2 n)$.

Simple Polynomial Bound

Theorem

The smoothed number of 2-Opt steps is $O(n^7 \phi^3 \log^2 n)$.

Proof.

- Consider a 2-Opt step $(e_1, e_2) \to (e_3, e_4)$.
- $\Delta(e_1, e_2, e_3, e_4) = d(e_1) + d(e_2) - d(e_3) - d(e_4)$

Simple Polynomial Bound

Theorem

The smoothed number of 2-Opt steps is $O(n^7 \phi^3 \log^2 n)$.

Proof.

- Consider a 2-Opt step $(e_1, e_2) \to (e_3, e_4)$.
- $\Delta(e_1, e_2, e_3, e_4) = d(e_1) + d(e_2) - d(e_3) - d(e_4)$
- Every step decreases tour length by at least

$$\Delta = \min_{\substack{e_1, e_2, e_3, e_4 \in E \\ \Delta(e_1, e_2, e_3, e_4) > 0}} \Delta(e_1, e_2, e_3, e_4).$$

Simple Polynomial Bound

Theorem

The smoothed number of 2-Opt steps is $O(n^7\phi^3\log^2 n)$.

Proof.

- Consider a 2-Opt step $(e_1, e_2) \to (e_3, e_4)$.
- $\Delta(e_1, e_2, e_3, e_4) = d(e_1) + d(e_2) - d(e_3) - d(e_4)$
- Every step decreases tour length by at least

$$\Delta = \min_{\substack{e_1,e_2,e_3,e_4 \in E \\ \Delta(e_1,e_2,e_3,e_4)>0}} \Delta(e_1, e_2, e_3, e_4).$$

- Initial tour has length at most $\sqrt{d}n$. Hence,

$$\#\text{ 2-Opt Steps} \le \frac{\sqrt{d}n}{\Delta}.$$

Simple Polynomial Bound

Theorem

The smoothed number of 2-Opt steps is $O(n^7 \phi^3 \log^2 n)$.

Proof.

- Consider a 2-Opt step $(e_1, e_2) \to (e_3, e_4)$.
- $\Delta(e_1, e_2, e_3, e_4) = d(e_1) + d(e_2) - d(e_3) - d(e_4)$
- Every step decreases tour length by at least

$$\Delta = \min_{\substack{e_1, e_2, e_3, e_4 \in E \\ \Delta(e_1, e_2, e_3, e_4) > 0}} \Delta(e_1, e_2, e_3, e_4).$$

- Initial tour has length at most $\sqrt{d}n$. Hence,

$$\text{\# 2-Opt Steps} \leq \frac{\sqrt{d}n}{\Delta}.$$

- Union bound over $O(n^4)$ steps + calculations:
  $\Pr[\Delta \leq \varepsilon] = O(n^4 \cdot \phi^3 \cdot \varepsilon \cdot \log(1/\varepsilon))$ □

Idea for Improvement

- The bound is too pessimistic: Not every step yields the smallest possible improvement $\Delta \approx 1/(n^4 \log n)$.

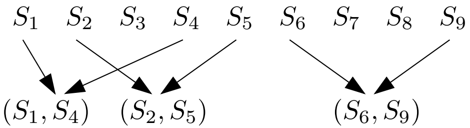
Idea for Improvement

- The bound is too pessimistic: Not every step yields the smallest possible improvement $\Delta \approx 1/(n^4 \log n)$.
- Consider two consecutive steps: They yield $\Delta + \Delta_2 > 2\Delta$.

Idea for Improvement

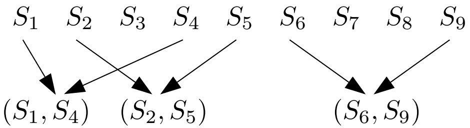
- The bound is too pessimistic: Not every step yields the smallest possible improvement $\Delta \approx 1/(n^4 \log n)$.
- Consider two consecutive steps: They yield $\Delta + \Delta_2 > 2\Delta$.
- Consider linked pair: $(e_1, e_2) \to (e_3, e_4)$ and $(e_3, e_5) \to (e_6, e_7)$.

- The bound is too pessimistic: Not every step yields the smallest possible improvement $\Delta \approx 1/(n^4 \log n)$.
- Consider two consecutive steps: They yield $\Delta + \Delta_2 > 2\Delta$.
- Consider linked pair: $(e_1, e_2) \to (e_3, e_4)$ and $(e_3, e_5) \to (e_6, e_7)$.
- Sequence of t consecutive steps, contains $\Omega(t)$ linked pairs:



Idea for Improvement

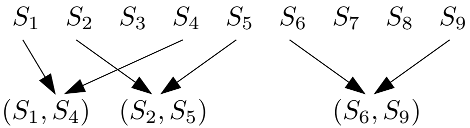
- The bound is too pessimistic: Not every step yields the smallest possible improvement $\Delta \approx 1/(n^4 \log n)$.
- Consider two consecutive steps: They yield $\Delta + \Delta_2 > 2\Delta$.
- Consider linked pair: $(e_1, e_2) \to (e_3, e_4)$ and $(e_3, e_5) \to (e_6, e_7)$.
- Sequence of t consecutive steps, contains $\Omega(t)$ linked pairs:



- $\Delta_{\text{Linked}} \approx 1/(n^{3+1/3} \log^{2/3} n)$.
worst and second worst step are unlikely to form a linked pair

Idea for Improvement

- The bound is too pessimistic: Not every step yields the smallest possible improvement $\Delta \approx 1/(n^4 \log n)$.
- Consider two consecutive steps: They yield $\Delta + \Delta_2 > 2\Delta$.
- Consider linked pair: $(e_1, e_2) \to (e_3, e_4)$ and $(e_3, e_5) \to (e_6, e_7)$.
- Sequence of t consecutive steps, contains $\Omega(t)$ linked pairs:



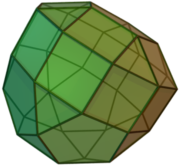
- $\Delta_{\mathrm{Linked}} \approx 1/(n^{3+1/3} \log^{2/3} n)$.
  worst and second worst step are unlikely to form a linked pair
- This idea yields $\tilde{O}(n^{4.33} \cdot \phi^{2.67})$.

Outline

Outline

1. **Linear Programming**
   Why is the simplex method usually efficient?
   smoothed analysis – analysis of algorithms beyond worst case
2. **Traveling Salesperson Problem**
   Why is local search successful?
3. **Smoothed Analysis**
   Overview of known results

Overview of Results on Smoothed Analsyis

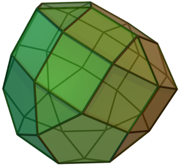


Linear Programming

Simplex Method [Spielman, Teng (STOC 2001)]

$\rightarrow$ Gödel Prize 2008, Fulkerson Prize 2009

Overview of Results on Smoothed Analsyis



Linear Programming

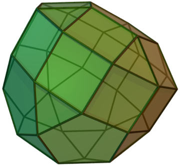
Simplex Method [Spielman, Teng (STOC 2001)]

$\rightarrow$ Gödel Prize 2008, Fulkerson Prize 2009

Perceptron Algo [Blum, Dunagan (SODA 2002)]

Interior Point Algo [Dunagan, Spielman, Teng (MathProg 2011)]

Overview of Results on Smoothed Analsyis



Linear Programming

Simplex Method [Spielman, Teng (STOC 2001)]
$\rightarrow$ Gödel Prize 2008, Fulkerson Prize 2009

Perceptron Algo [Blum, Dunagan (SODA 2002)]

Interior Point Algo [Dunagan, Spielman, Teng (MathProg 2011)]

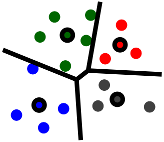
Combinatorial Optimization

Complexity of Binary Optimization Problems [Beier, Vöcking (STOC 2004)]

2-Opt Algo for TSP [Englert, R., Vöcking (SODA 2007)]

SSP Algo for Min-Cost Flow Problem [Brunsch, Cornelissen, Manthey, R. (SODA 2013)]

Overview of Results on Smoothed Analsyis



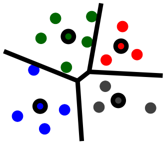
Machine Learning

k-Means [Arthur, Manthey, R. (FOCS 2009)]

PAC-Learning [Kalai, Samorodnitsky, Teng (FOCS 2009)]

Belief Propagation [Brunsch, Cornelissen, Manthey, R. (WALCOM 2013)]

$\rightarrow$ (more in Kamiel's talk at 14.00)



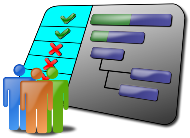
Machine Learning

k-Means [Arthur, Manthey, R. (FOCS 2009)]

PAC-Learning [Kalai, Samorodnitsky, Teng (FOCS 2009)]

Belief Propagation [Brunsch, Cornelissen, Manthey, R. (WALCOM 2013)]

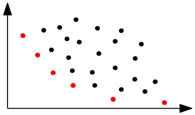
$\rightarrow$ (more in Kamiel's talk at 14.00)



Scheduling

Multilevel Feedback Algo [Becchetti, Leonardi, Marchetti-Spaccamela, Schäfer, Vredeveld (FOCS 2003)]

Local Search Algos [Brunsch, R., Rutten, Vredeveld (ESA 2011)]



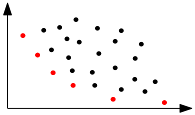
Multiobjective Optimization

Number of Pareto optima

[Brunsch, R. (STOC 2012)]

Knapsack Problem [Beier, Vöcking (STOC 2003)]

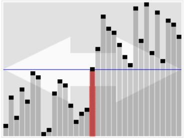
Overview of Results on Smoothed Analsyis



Multiobjective Optimization

Number of Pareto optima
[Brunsch, R. (STOC 2012)]

Knapsack Problem [Beier, Vöcking (STOC 2003)]

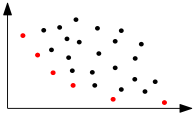


Classical Algorithms and Data Structures

Quicksort [Fouz, Kufleitner, Manthey, Zeini Jahromi (COCOON 2009)]

Binary Search Trees
[Manthey, Tantau (MFCS 2008)]

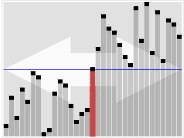
Gaussian Elimination [Sankar, Spielman, Teng (SIAM. J. Matrix Anal. 2006)]



Multiobjective Optimization

Number of Pareto optima
[Brunsch, R. (STOC 2012)]

Knapsack Problem [Beier, Vöcking (STOC 2003)]



Classical Algorithms and Data Structures

Quicksort [Fouz, Kufleitner, Manthey, Zeini Jahromi (COCOON 2009)]

Binary Search Trees
[Manthey, Tantau (MFCS 2008)]

Gaussian Elimination [Sankar, Spielman, Teng (SIAM. J. Matrix Anal. 2006)]

Many more results...