Pearls of Algorithms

Part 3: Randomized Algorithms and Probabilistic Analysis

Prof. Dr. Heiko Röglin
Institut für Informatik



Winter 2013/14

Efficient Algorithms

When is an algorithm considered efficient?

Efficient Algorithms

When is an algorithm considered efficient?



Engineer

The algorithm must be efficient in practice, i.e., it must solve practical instances in an appropriate amount of time.

Efficient Algorithms

When is an algorithm considered efficient?



Engineer

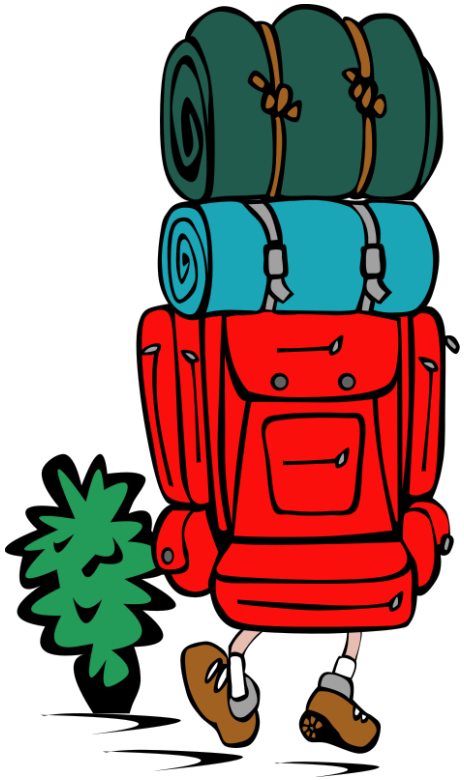
The algorithm must be efficient in practice, i.e., it must solve practical instances in an appropriate amount of time.



Theoretician

The algorithm must be efficient in the worst case, i.e., it must solve all instances in polynomial time.

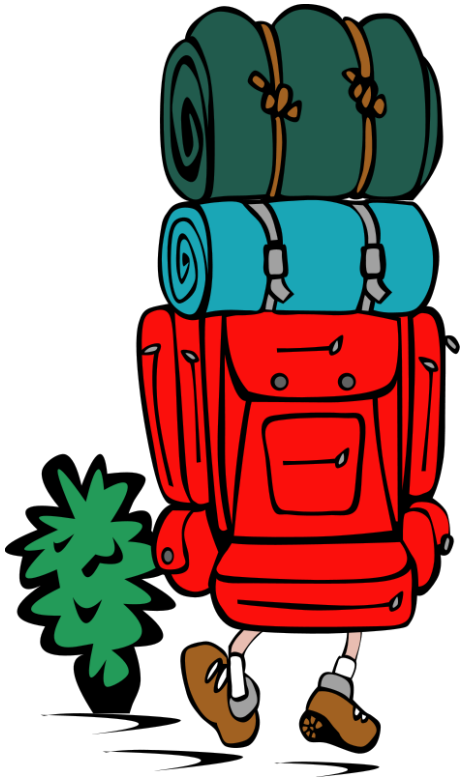
The Knapsack Problem



Knapsack problem (KP)

- **Input**
  - set of items $\{1, \ldots, n\}$
  - profits $p_1, \ldots, p_n$
  - weights $w_1, \ldots, w_n$
  - capacity b

The Knapsack Problem

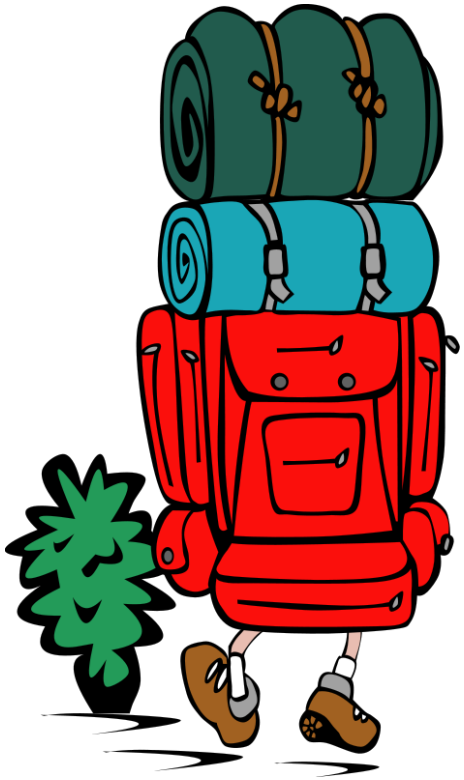


Knapsack problem (KP)

- **Input**
  - set of items $\{1, \ldots, n\}$
  - profits $p_1, \ldots, p_n$
  - weights $w_1, \ldots, w_n$
  - capacity b
- **Goal**

  Find a subset of items that fits into the knapsack and maximizes the profit.

The Knapsack Problem



Knapsack problem (KP)

- **Input**
  - set of items $\{1, \ldots, n\}$
  - profits $p_1, \ldots, p_n$
  - weights $w_1, \ldots, w_n$
  - capacity b
- **Goal**

  Find a subset of items that fits into the knapsack and maximizes the profit.
- **Formal description**

$$\begin{aligned} \max \quad & p_1 x_1 + \cdots + p_n x_n \\ \text{subject to} \quad & w_1 x_1 + \cdots + w_n x_n \leq b \\ & \text{and } x_i \in \{0, 1\} \end{aligned}$$

Different Opinions



Theorists say...

- KP is NP-hard.
- FPTAS exists.

No efficient algorithm for KP, unless $P = NP$.

Different Opinions





Theorists say...

- KP is NP-hard.
- FPTAS exists.

No efficient algorithm for KP, unless $P = NP$.

Engineers say...

- KP is easy to solve!
- Does not even require quadratic time.

There are very good heuristics for practical instances of KP.

Reason for discrepancy

- Worst-case complexity is too pessimistic!
- There are (artificial) worst-case instances for KP on which the heuristics are not efficient. These instances, however, do not occur in practice.
- This phenomenon occurs not only for KP, but also for many other problems.

Reason for discrepancy

- Worst-case complexity is too pessimistic!
- There are (artificial) worst-case instances for KP on which the heuristics are not efficient. These instances, however, do not occur in practice.
- This phenomenon occurs not only for KP, but also for many other problems.

How to make theory more consistent with practice?

Find a more realistic performance measure.

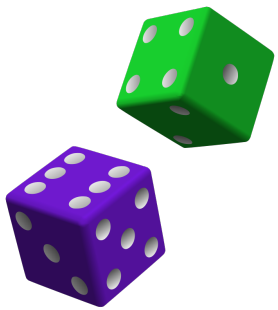
Reason for discrepancy

- Worst-case complexity is too pessimistic!
- There are (artificial) worst-case instances for KP on which the heuristics are not efficient. These instances, however, do not occur in practice.
- This phenomenon occurs not only for KP, but also for many other problems.

How to make theory more consistent with practice?

Find a more realistic performance measure.

Average-case analysis

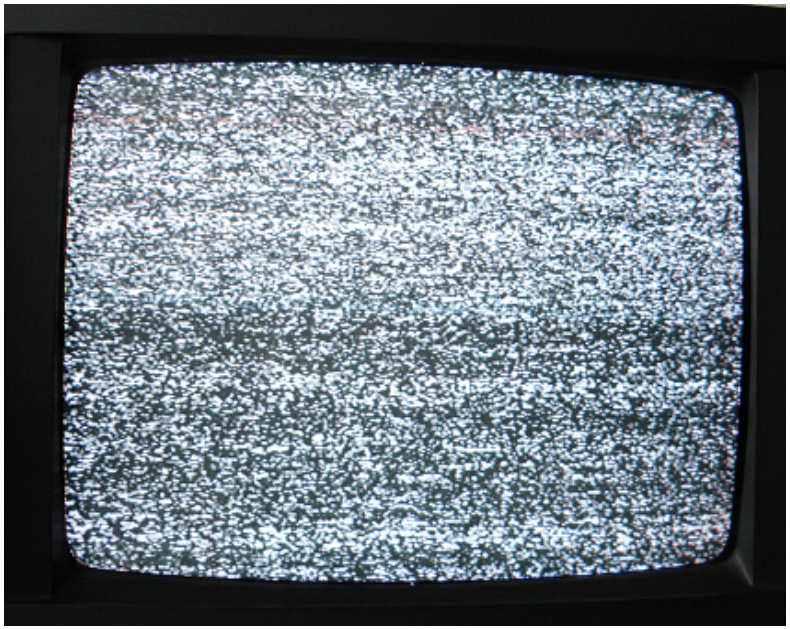


Is it realistic to consider the average case behavior instead of the worst case behavior?

Random Inputs are not Typical

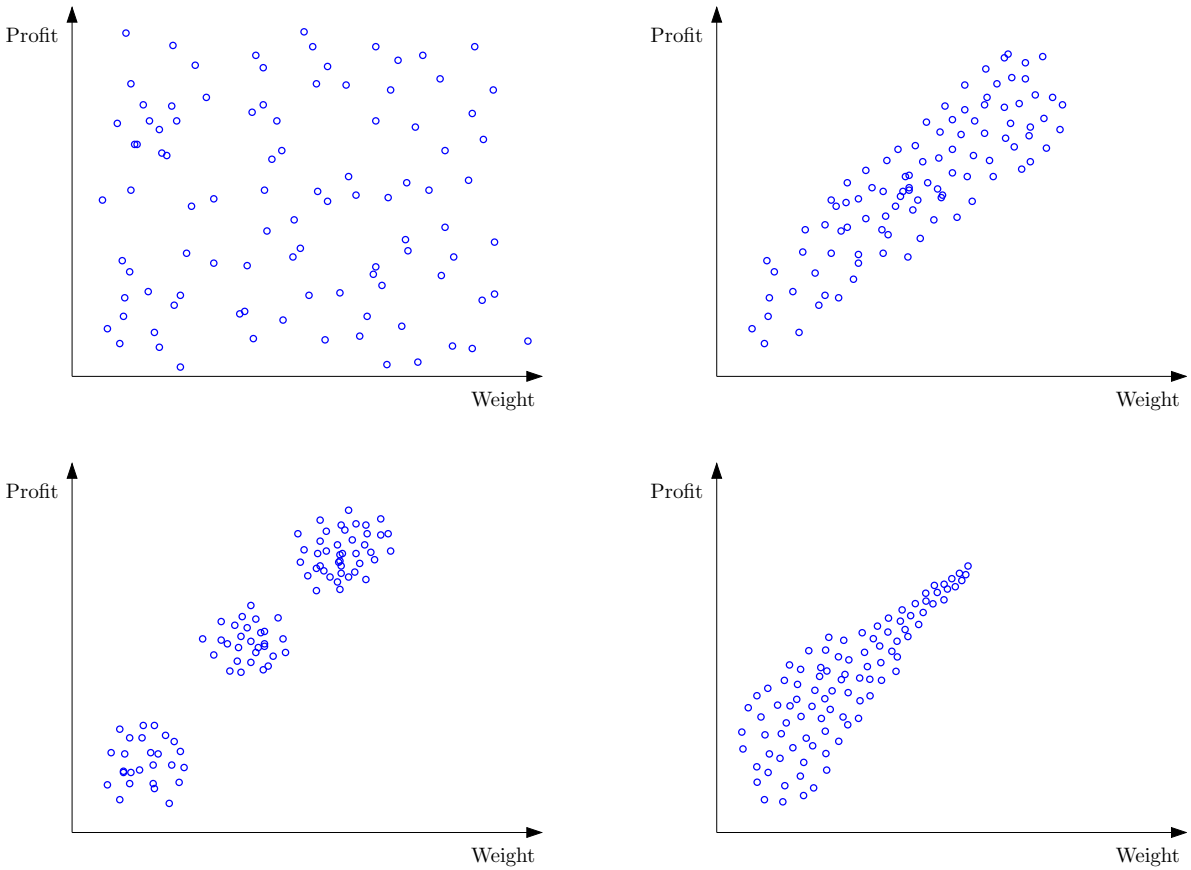
Random inputs are not typical!

If real-world data was random, watching TV would be very boring...





What is a typical instance?



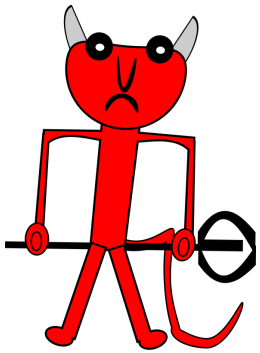
It depends very much on the concrete application. We cannot say in general what a typical instance for KP looks like.

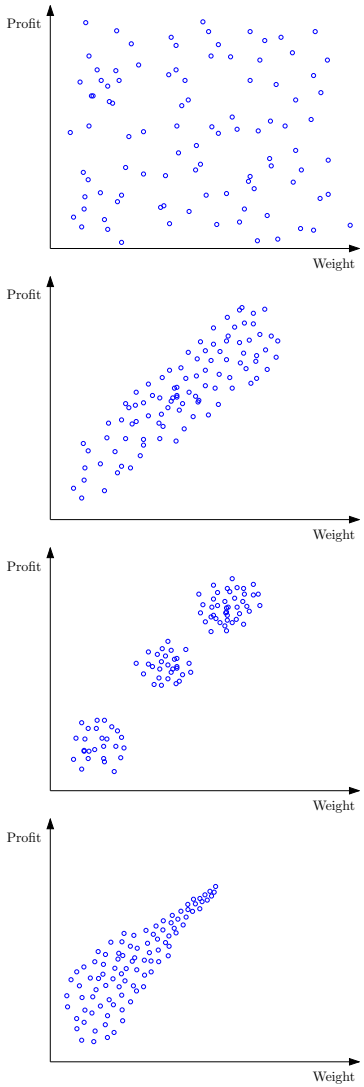
Smoothed Analysis

Step 1:

Adversary

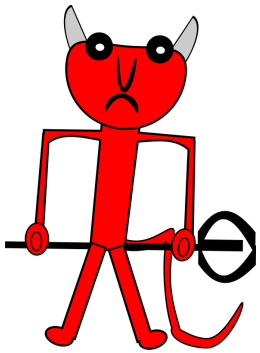
chooses input *I*.

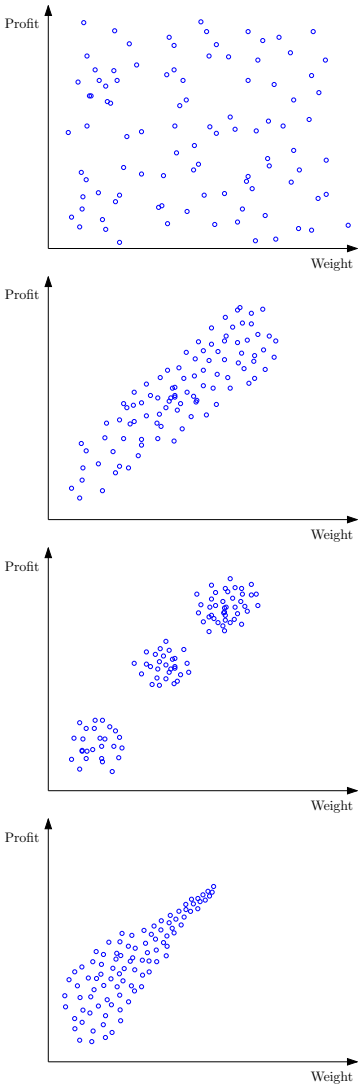




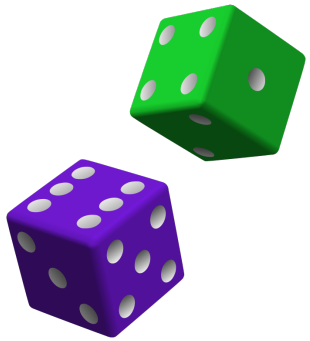
Smoothed Analysis

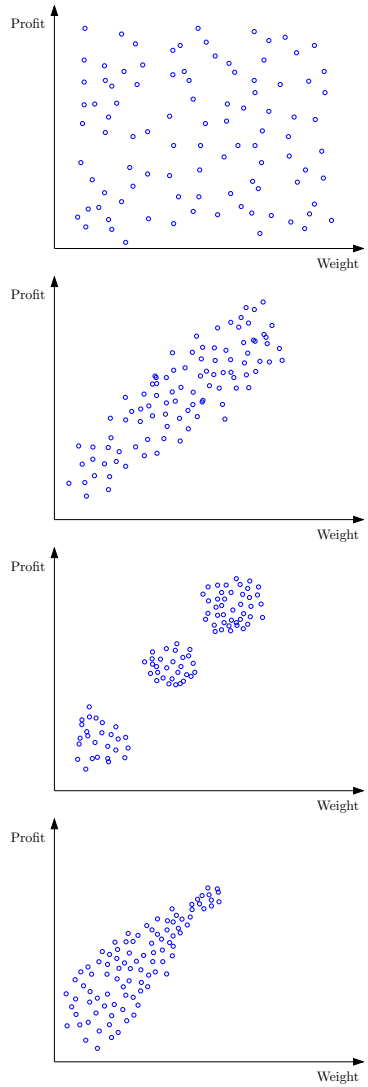
Step 1:
Adversary
chooses input I.



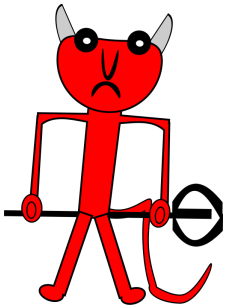


Step 2: Random
perturbation.
$I \to \mathrm{per}(I)$

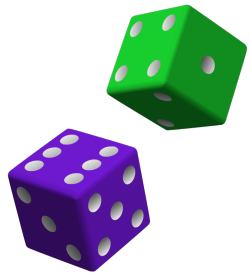




Smoothed Analysis



Step 1:
Adversary
chooses input I.



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

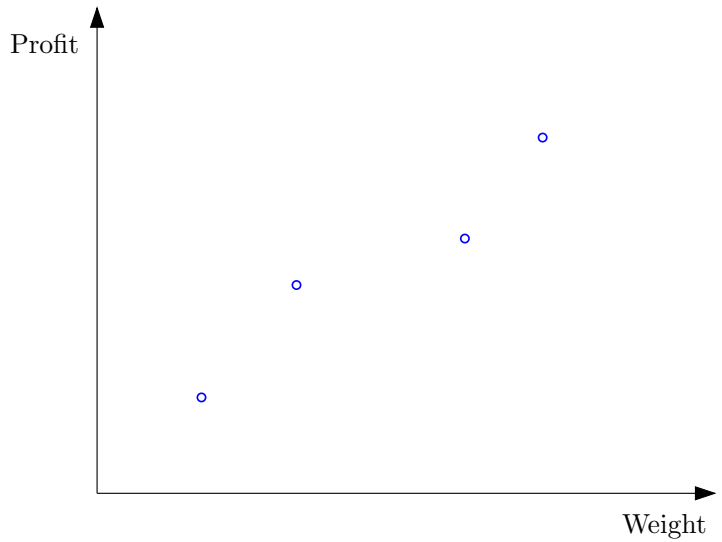
Smoothed Complexity = worst expected running time the adversary can achieve

Why do we consider this model?

- First step alone: worst case analysis.
- Second step models random influences, e.g., measurement errors, numerical imprecision, rounding, . . .
- So we have a combination of both: instances of any structure with some amount of random noise.

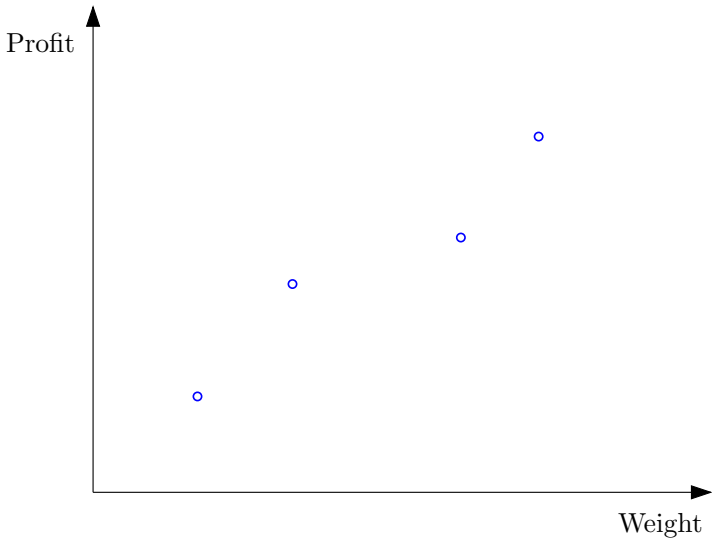
Example

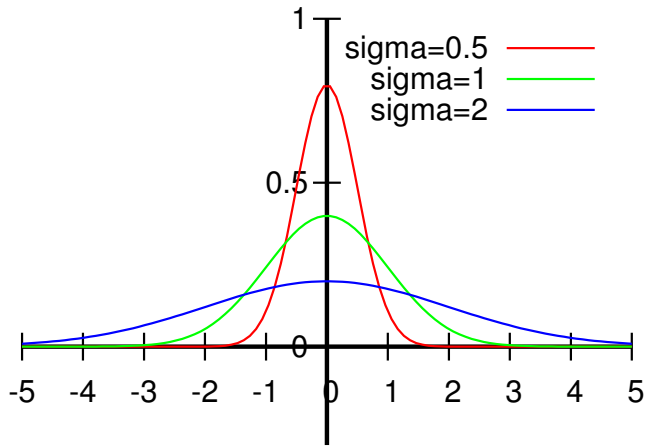
- Step 1: Adversary chooses all $p_i, w_i \in [0, 1]$ arbitrarily.



Example

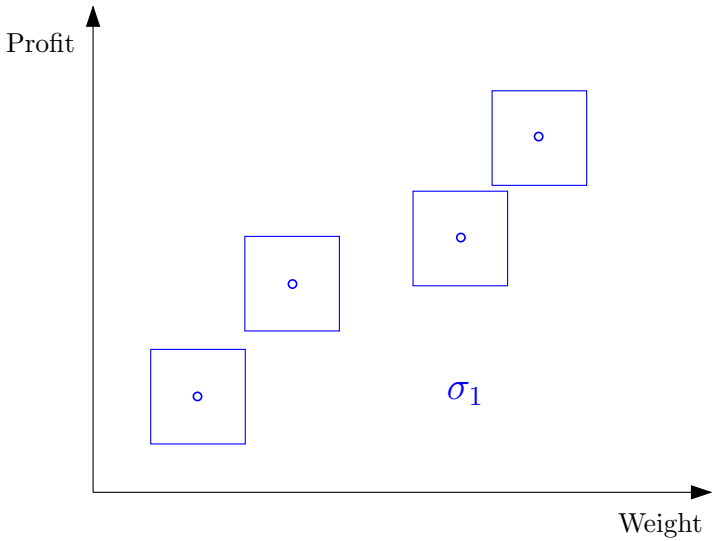
- Step 1: Adversary chooses all $p_i, w_i \in [0, 1]$ arbitrarily.
- Step 2: Add an independent Gaussian random variable to each profit and weight.

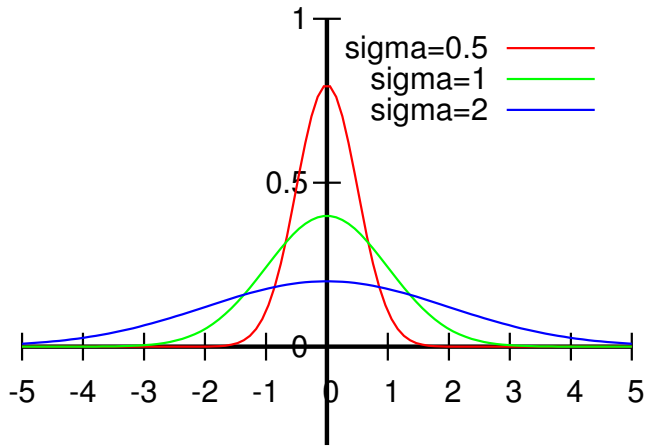




Example

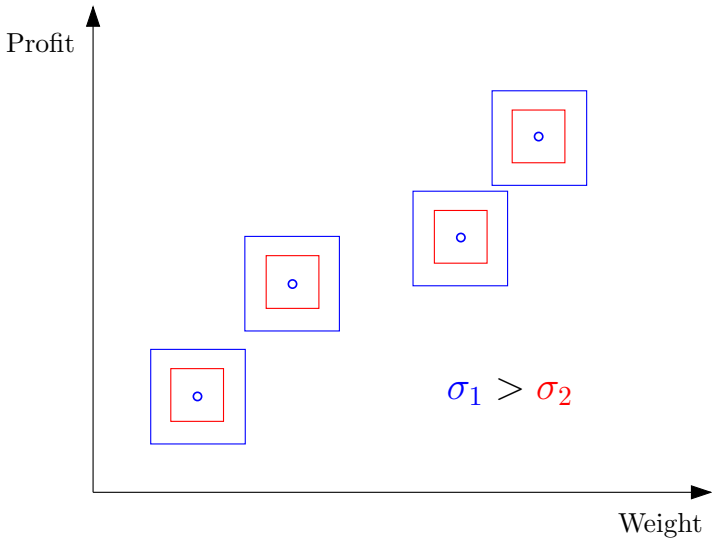
- Step 1: Adversary chooses all $p_i, w_i \in [0, 1]$ arbitrarily.
- Step 2: Add an independent Gaussian random variable to each profit and weight.

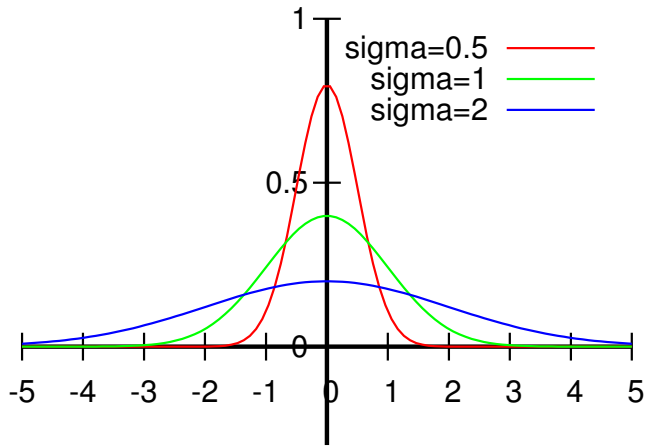




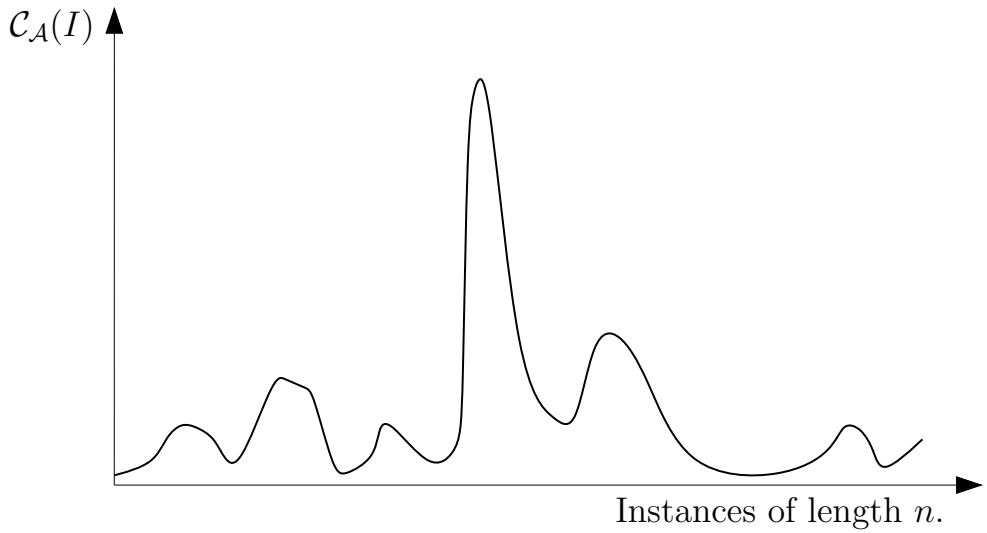
Example

- Step 1: Adversary chooses all $p_i, w_i \in [0, 1]$ arbitrarily.
- Step 2: Add an independent Gaussian random variable to each profit and weight.



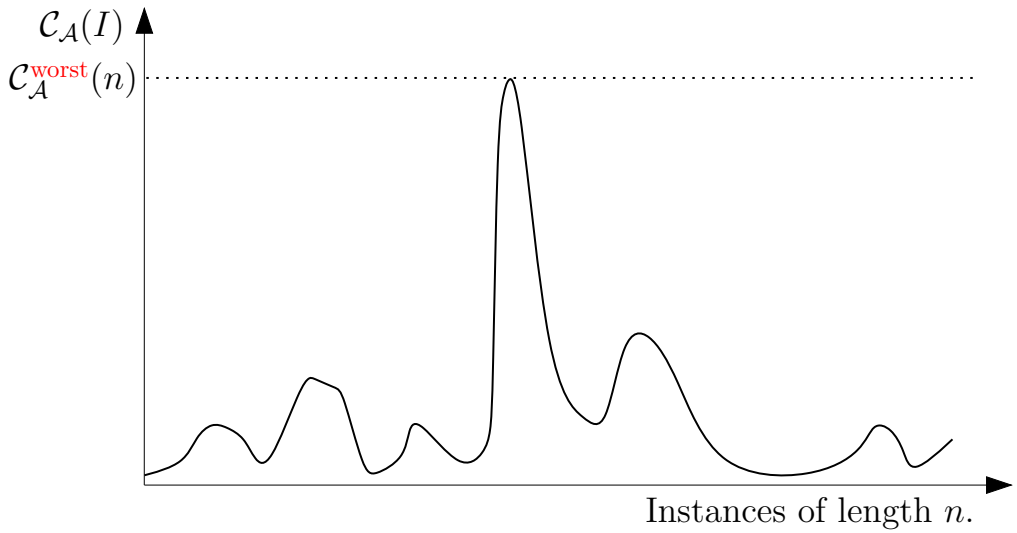


Complexity Measures



$C_{\mathcal{A}}(I) =$ running time of algorithm $\mathcal{A}$ on input I
$X_n =$ set of inputs of length n,

Complexity Measures

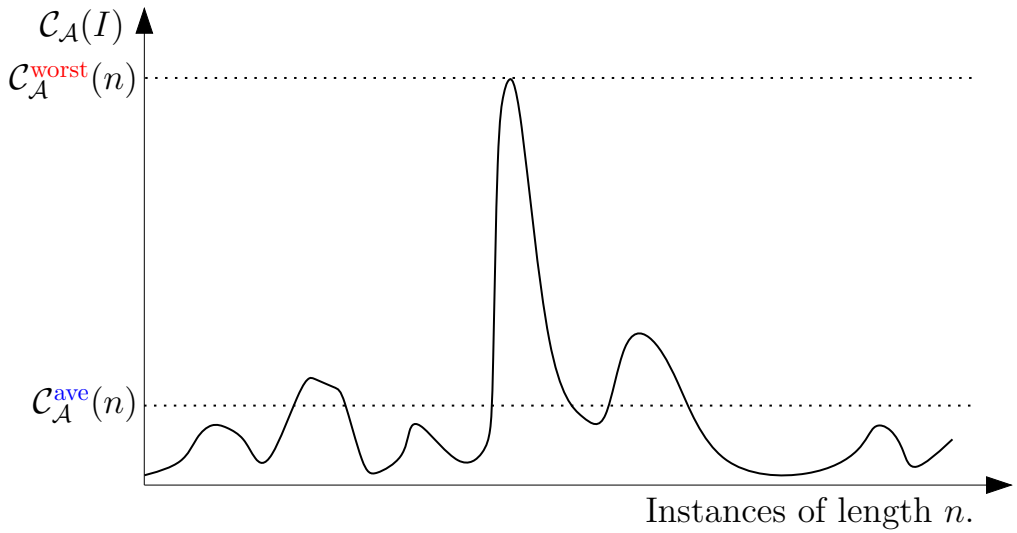


$C_{\mathcal{A}}(I) =$ running time of algorithm $\mathcal{A}$ on input I

$X_n =$ set of inputs of length n,

- $C_{\mathcal{A}}^{\text{worst}}(n) = \max_{I \in X_n} (C_{\mathcal{A}}(I))$

Complexity Measures

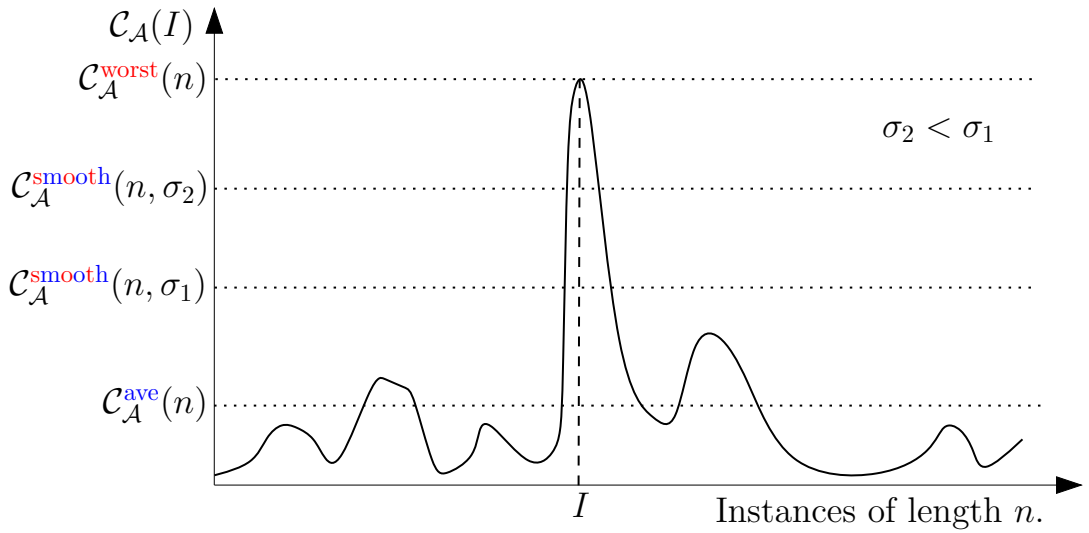


$C_A(I)$ = running time of algorithm A on input I

X_n = set of inputs of length n, μ_n = prob. distribution on X_n

- $C_A^{\text{worst}}(n) = \max_{I \in X_n} (C_A(I))$
- $C_A^{\text{ave}}(n) = \mathbf{E}_{I \overset{\mu_n}{\leftarrow} X_n} [C_A(I)]$

Complexity Measures

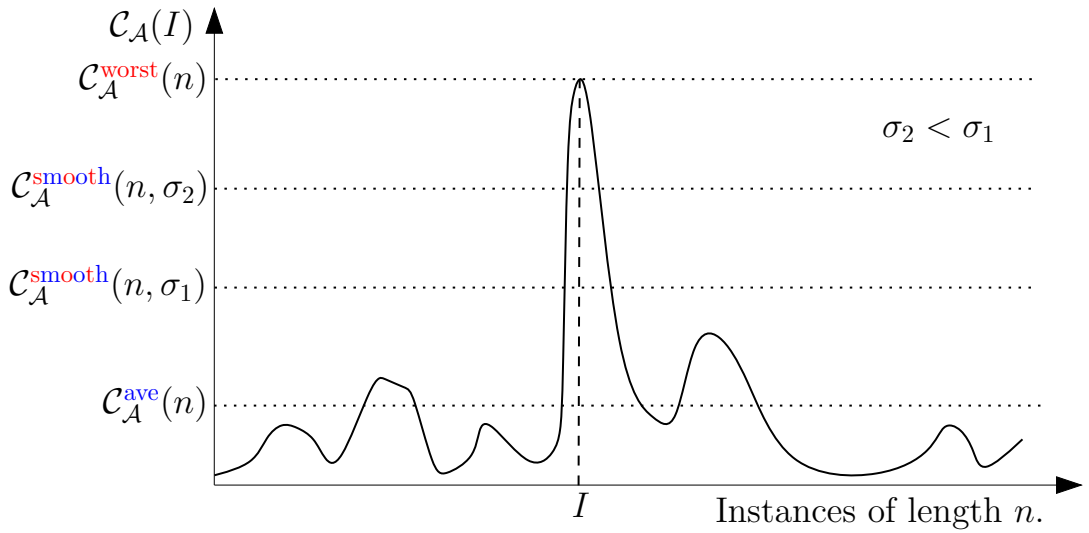


$C_{\mathcal{A}}(I)$ = running time of algorithm $\mathcal{A}$ on input I

X_n = set of inputs of length n, μ_n = prob. distribution on X_n

- $C_{\mathcal{A}}^{\text{worst}}(n) = \max_{I \in X_n} (C_{\mathcal{A}}(I))$
- $C_{\mathcal{A}}^{\text{ave}}(n) = \mathbf{E}_{I \overset{\mu_n}{\leftarrow} X_n} [C_{\mathcal{A}}(I)]$
- $C_{\mathcal{A}}^{\text{smooth}}(n,\sigma) = \max_{I \in X_n} \mathbf{E}[C_{\mathcal{A}}(\text{per}_{\sigma}(I))]$

Complexity Measures



$C_{\mathcal{A}}(I) =$ running time of algorithm $\mathcal{A}$ on input I

$X_n =$ set of inputs of length n, $\mu_n =$ prob. distribution on X_n

- $C_{\mathcal{A}}^{\text{worst}}(n) = \max_{I \in X_n} (C_{\mathcal{A}}(I))$
- $C_{\mathcal{A}}^{\text{ave}}(n) = \mathbf{E}_{I \xleftarrow{\mu_n} X_n} [C_{\mathcal{A}}(I)]$
- $C_{\mathcal{A}}^{\text{smooth}}(n,\sigma) = \max_{I \in X_n} \mathbf{E}[C_{\mathcal{A}}(\text{per}_\sigma(I))]$

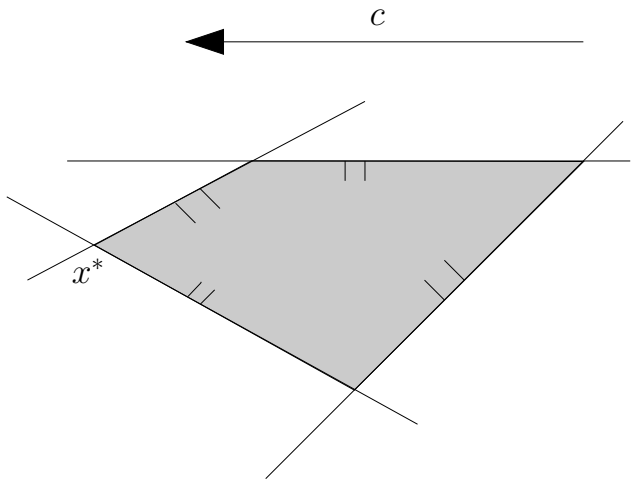
smoothed complexity low $\Rightarrow$ bad instances are isolated peaks

Linear Programming

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:

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

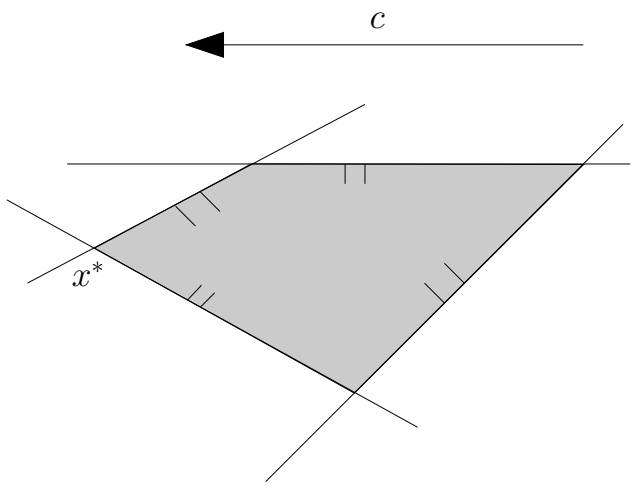


Linear Programming

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:

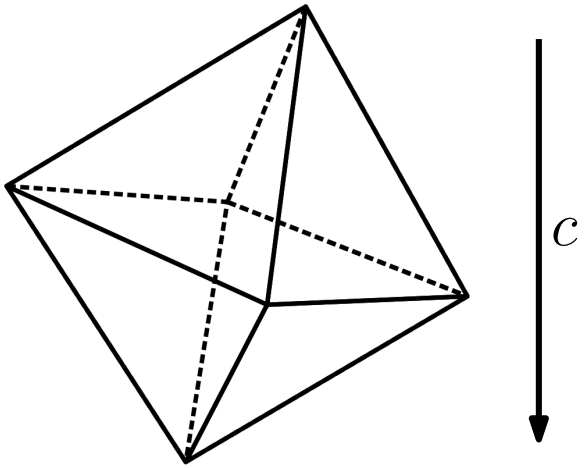
$$\begin{aligned} a_{1,1} x_1 &+ \ldots + a_{1,n} x_n \leq b_1 \\ &\vdots \\ a_{m,1} x_1 &+ \ldots + a_{m,n} x_n \leq b_m \end{aligned}$$



Complexity of LPs

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

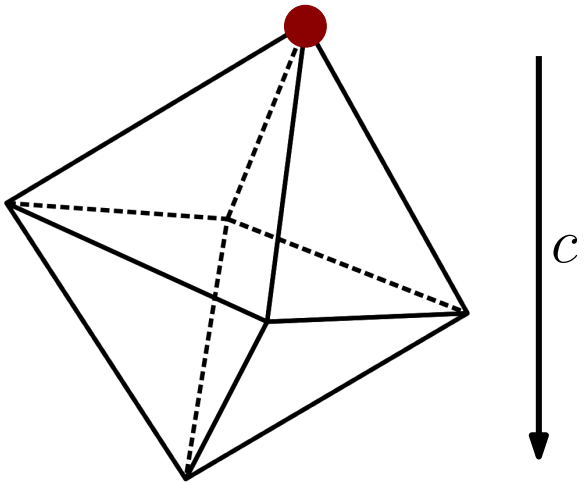
Simplex Algorithm



Simplex Algorithm

- Start at some vertex of the polytope.

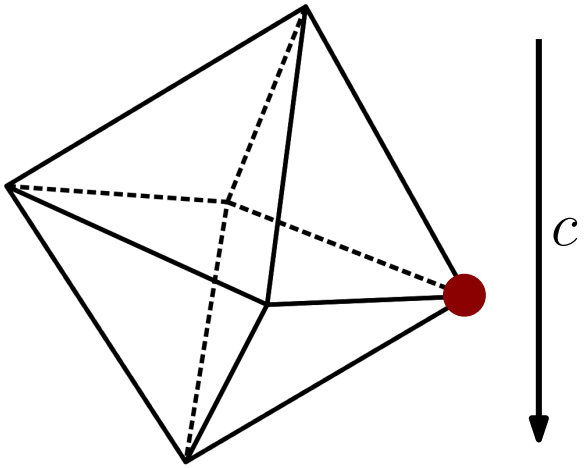
Simplex Algorithm



Simplex Algorithm

- Start at some vertex of the polytope.

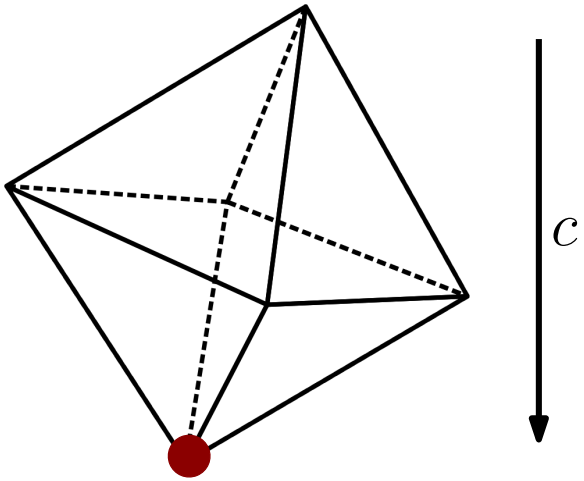
Simplex Algorithm



Simplex Algorithm

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

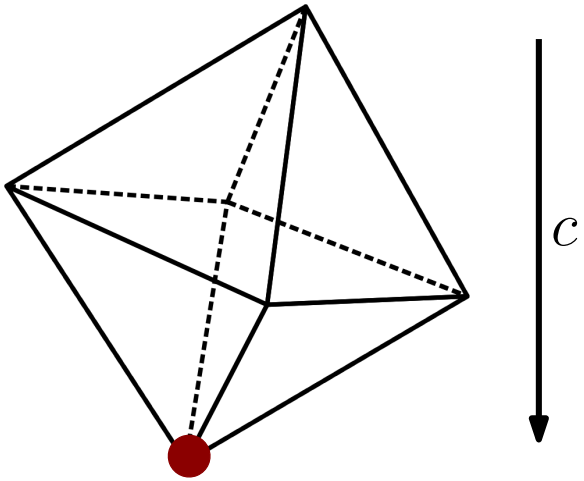
Simplex Algorithm

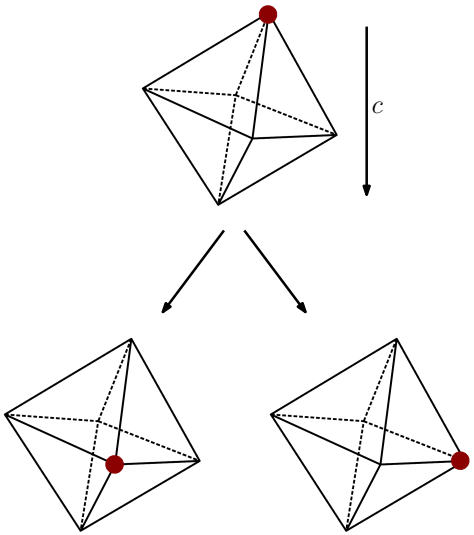


Simplex Algorithm

- 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





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

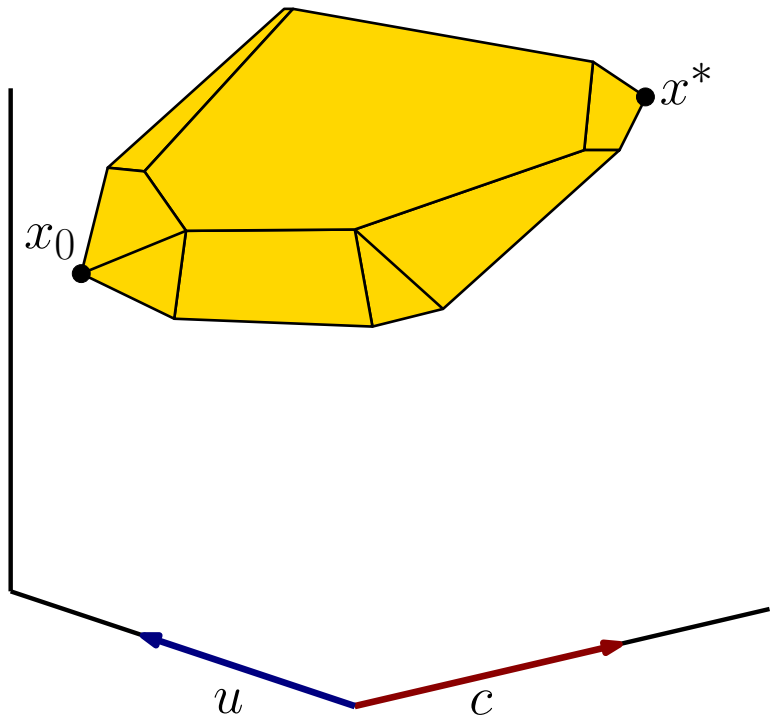
Pivot Rules

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

Simplex Algorithm – Shadow Vertex Pivot Rule

Shadow Vertex Pivot Rule

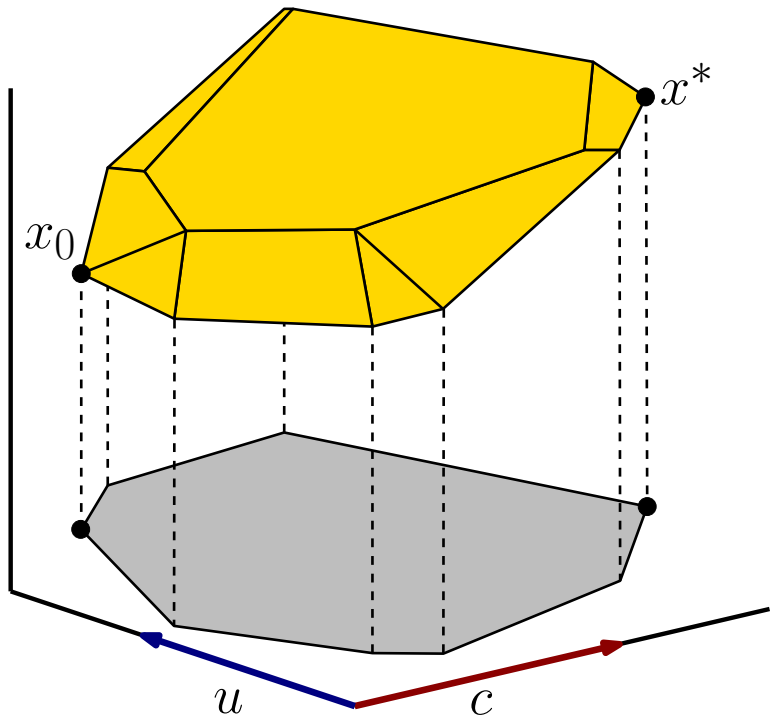
- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.



Simplex Algorithm – Shadow Vertex Pivot Rule

Shadow Vertex Pivot Rule

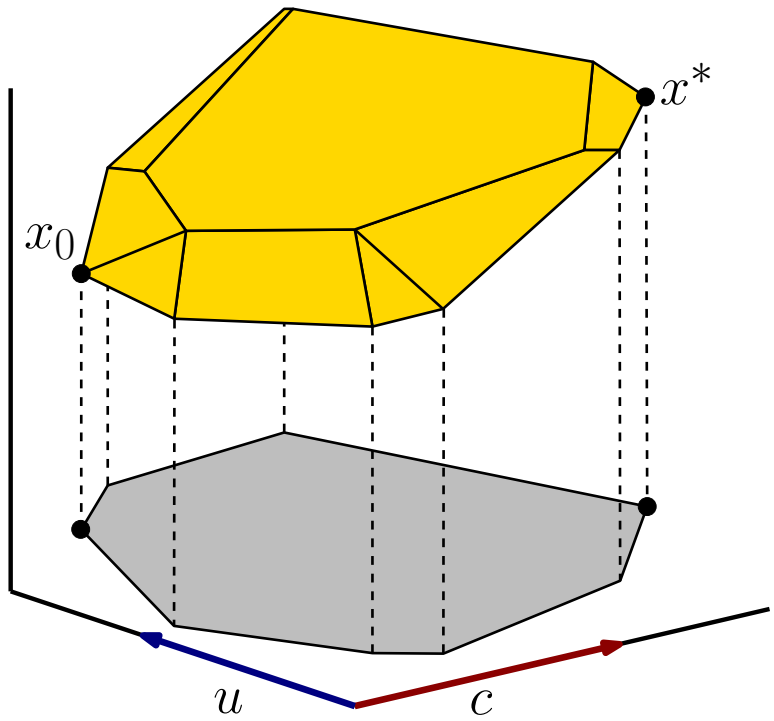
- Let x_0 be some vertex of the polytope.
- Compute $u \in \mathbb{R}^d$ such that x_0 maximizes $u^T x$.
- Project the polytope onto the plane spanned by c and u.



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.

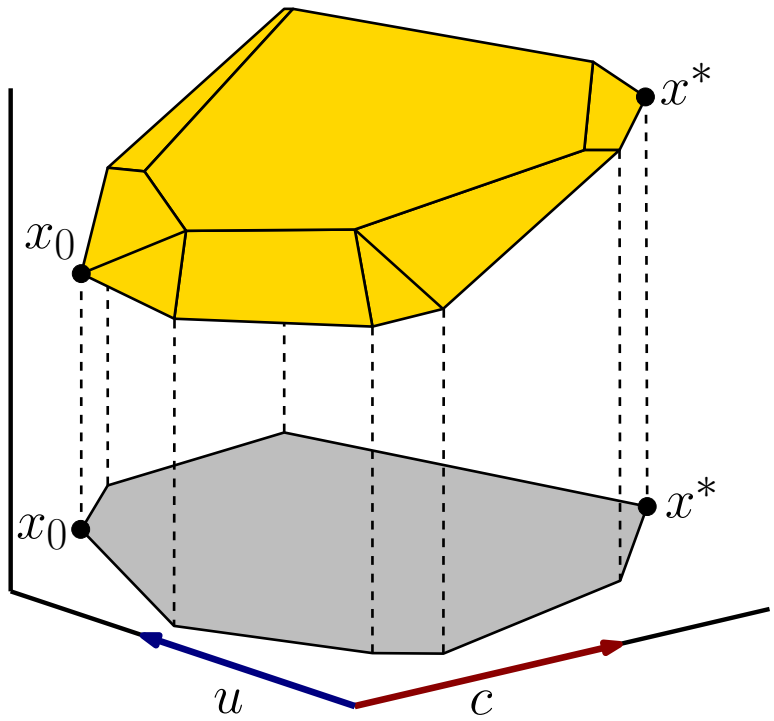


- The shadow is a polygon.

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.

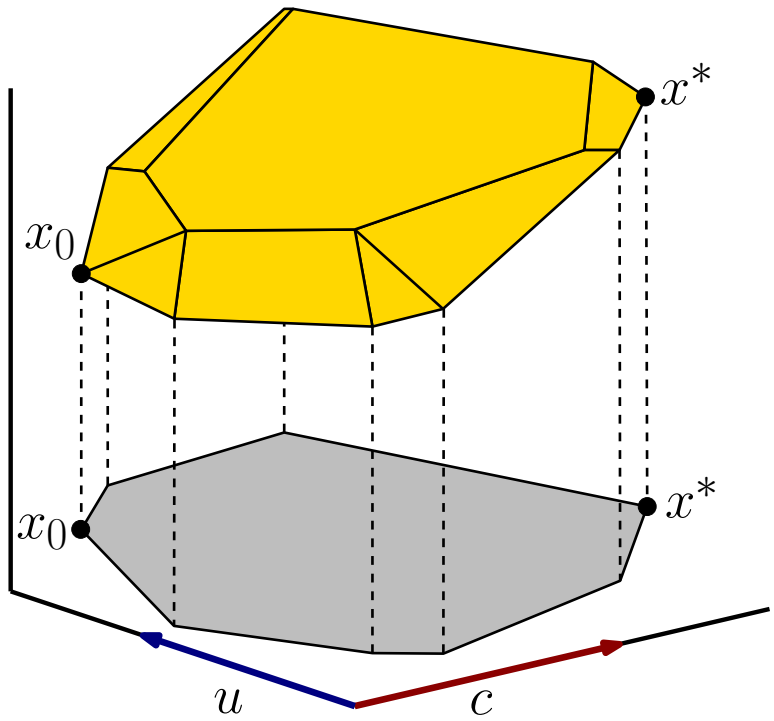


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.

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.

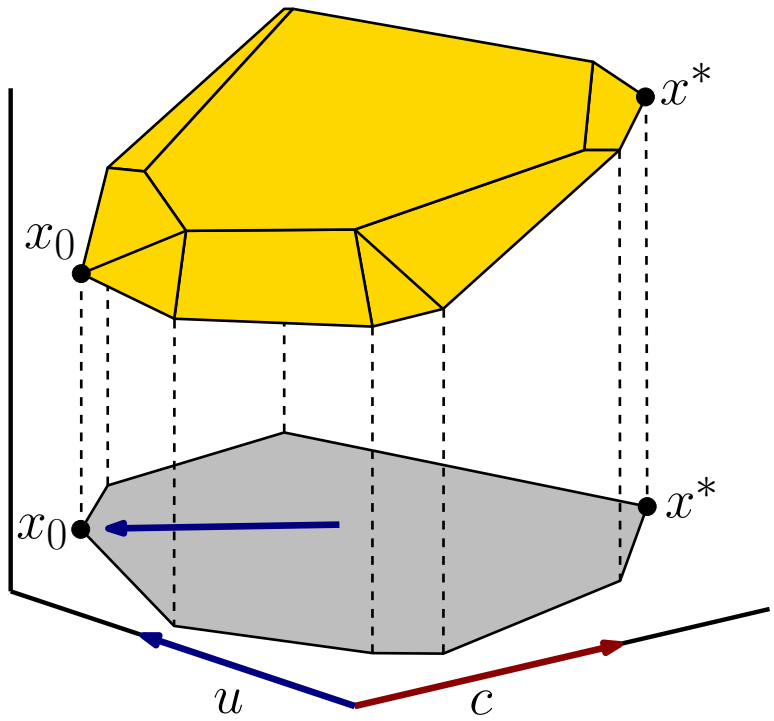


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

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.**

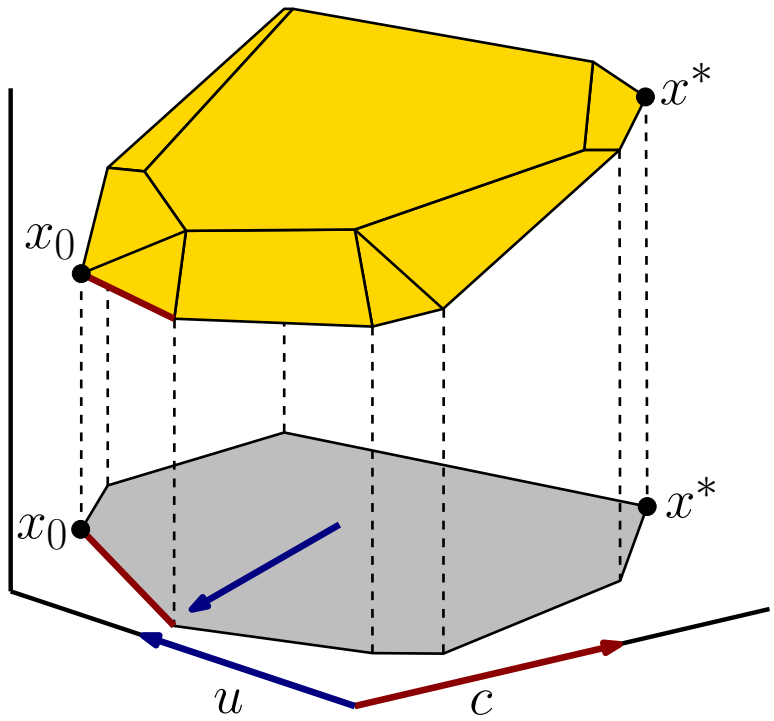


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

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.**

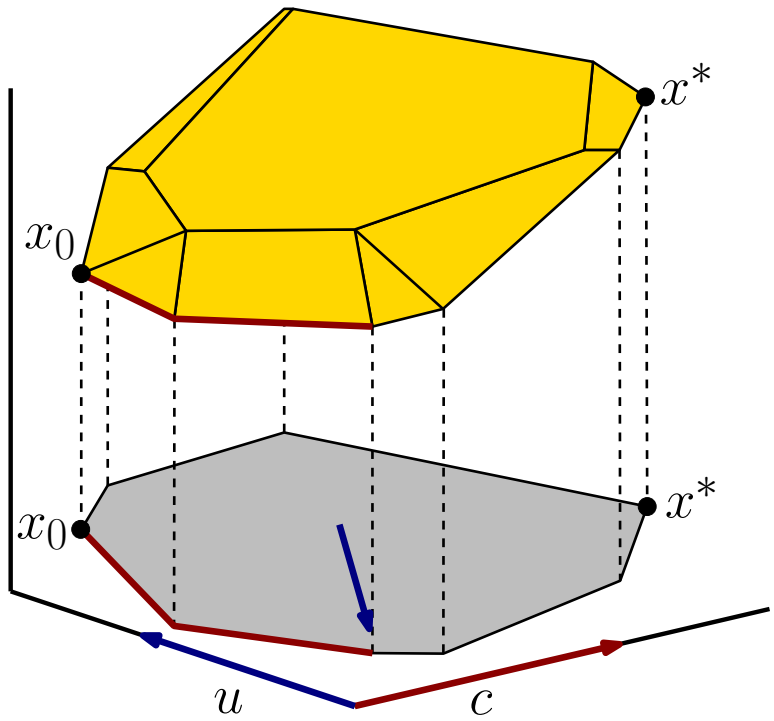


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

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.**

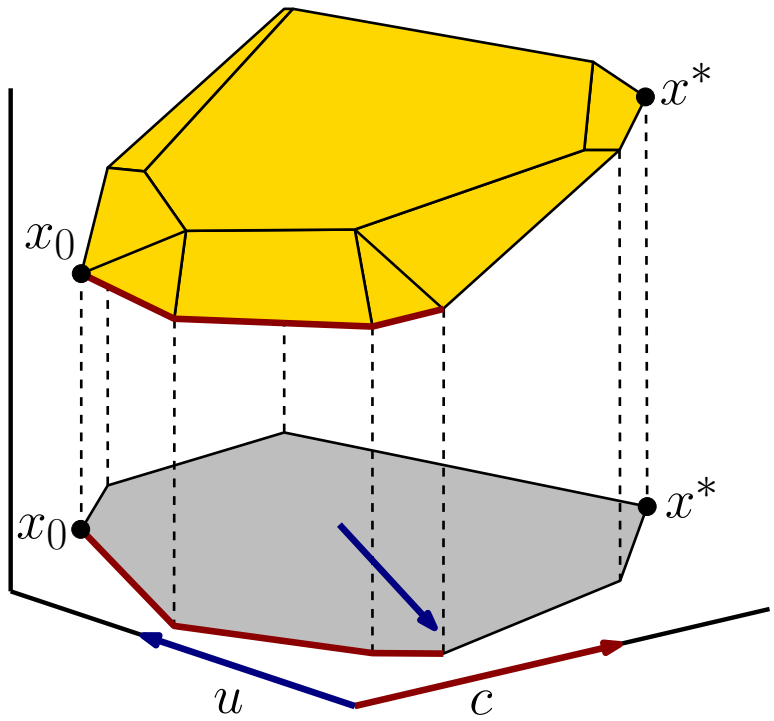


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

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.**

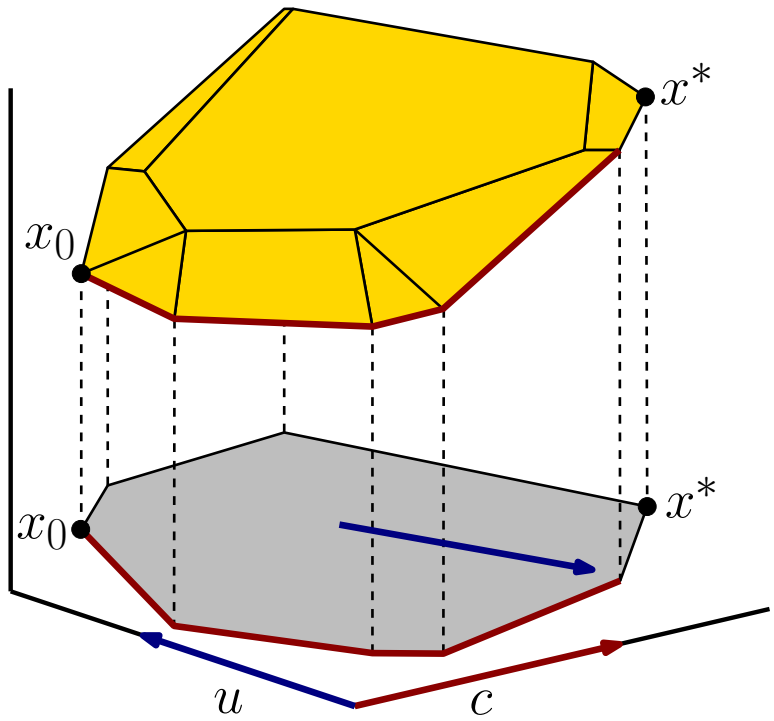


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

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.**

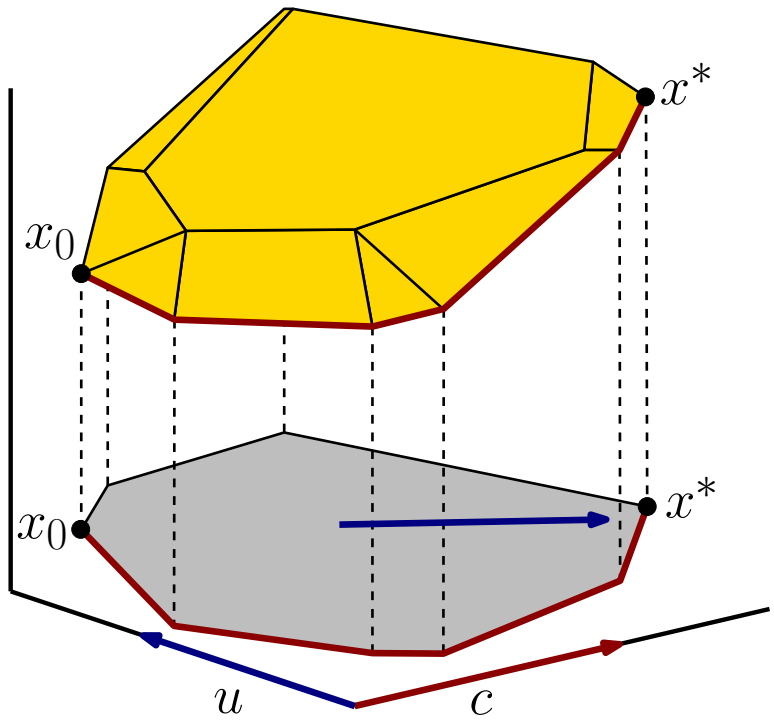


- The shadow is a polygon.
- x_0 is a vertex of the shadow.
- x^* is a vertex of the shadow.
- Edges of the shadow correspond to edges of the polytope.

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.

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

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



Engineers say...

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

Smoothed Analysis

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

Idea: Let's weaken him!

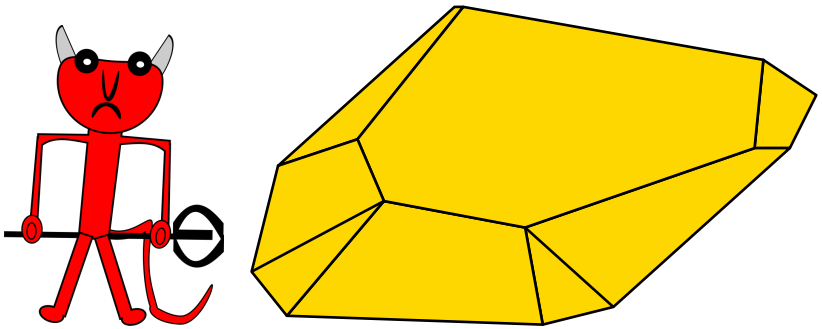
Smoothed Analysis

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. $\|(a_i, b_i)\| = 1$.



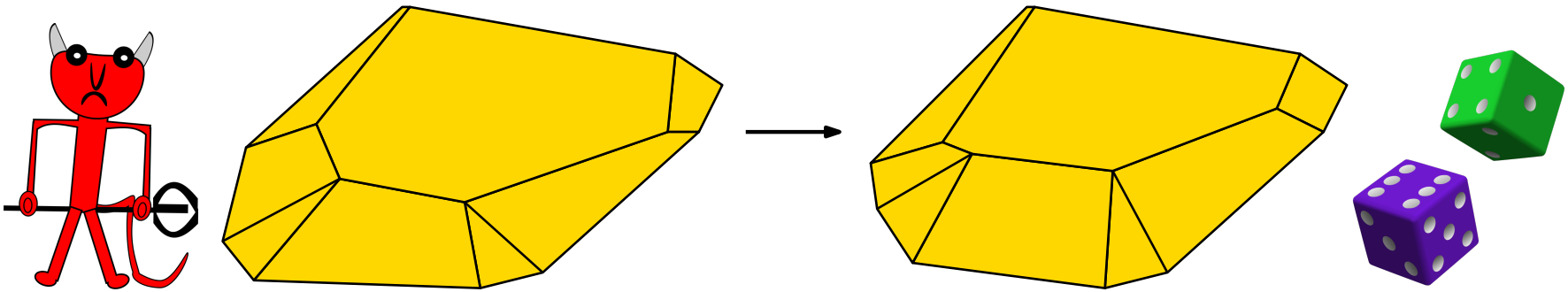
Smoothed Analysis

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.



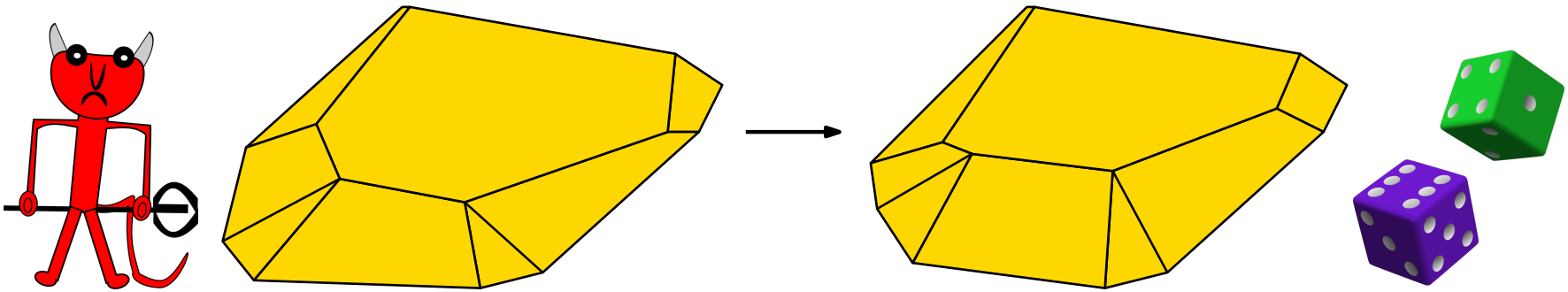
Smoothed Analysis

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

Idea: Let's weaken him!

Perturbed LPs

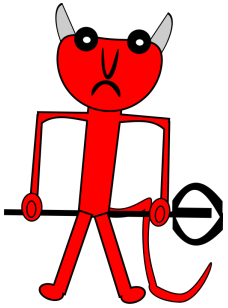
- Step 1: Adversary specifies arbitrary LP:
  $\max c^T x$ subject to $a_1^T x \leq b_1 \ldots a_n^T x \leq b_n$.
  W. l. o. g. $\|(a_i, b_i)\| = 1$.
- Step 2: Add Gaussian random variable with standard deviation σ to each coefficient in the constraints.



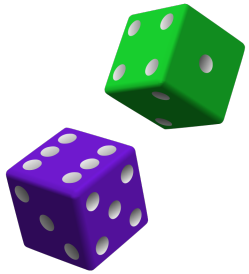
Smoothed Running Time

$=$ worst expected running time the adversary can achieve

Smoothed Analysis



Step 1:
Adversary
chooses input I



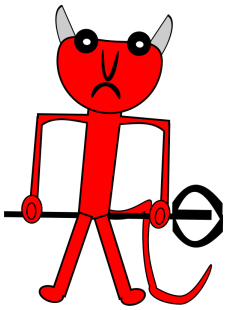
Step 2: Random
perturbation
$I \to \mathrm{per}_\sigma(I)$

Formal Definition:

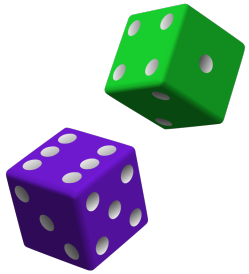
$\mathrm{LP}(n, m) =$ set of LPs with n variables and m constraints

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

Smoothed Analysis



Step 1:
Adversary
chooses input I



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

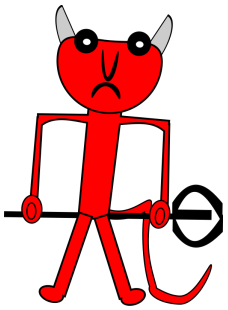
Formal Definition:

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

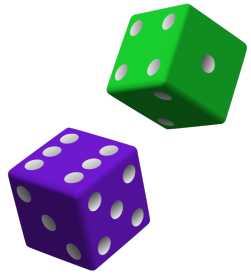
$T(I)$ = number of steps of simplex method on LP I

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

Smoothed Analysis



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

Why do we consider this model?

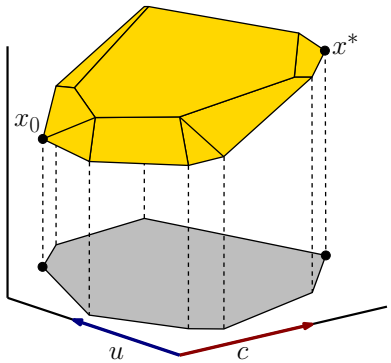
- First step models unknown structure of the input.
- Second step models random influences, e.g., measurement errors, numerical imprecision, rounding, ...
- smoothed running time low $\Rightarrow$ bad instances are unlikely to occur
- σ determines the amount of randomness

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

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

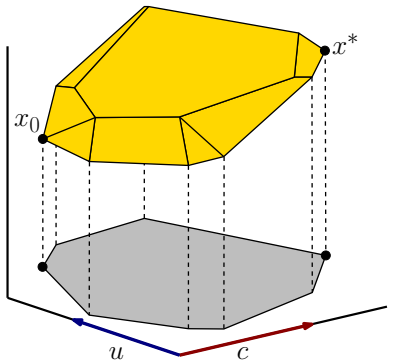


Smoothed Analysis of the Simplex Algorithm

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

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



Theorem [Spielman and Teng (STOC 2001)]

The smoothed running time of the simplex algorithm with shadow vertex pivot rule is $O\left(\text{poly}\left(n, m, \sigma^{-1}\right)\right)$.

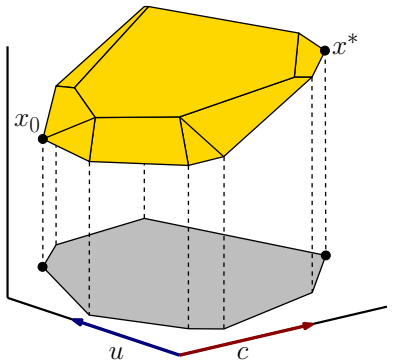
Already for small perturbations exponential running time is unlikely.

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

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



Theorem [Spielman and Teng (STOC 2001)]

The smoothed running time of the simplex algorithm with shadow vertex pivot rule is $O\left(\text{poly}\left(n, m, \sigma^{-1}\right)\right)$.

Already for small perturbations exponential running time is unlikely.

Main Difficulties in Proof of Theorem:

- x_0 is found in phase I $\rightarrow$ 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

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

only polylogarithmic in the number of constraints m

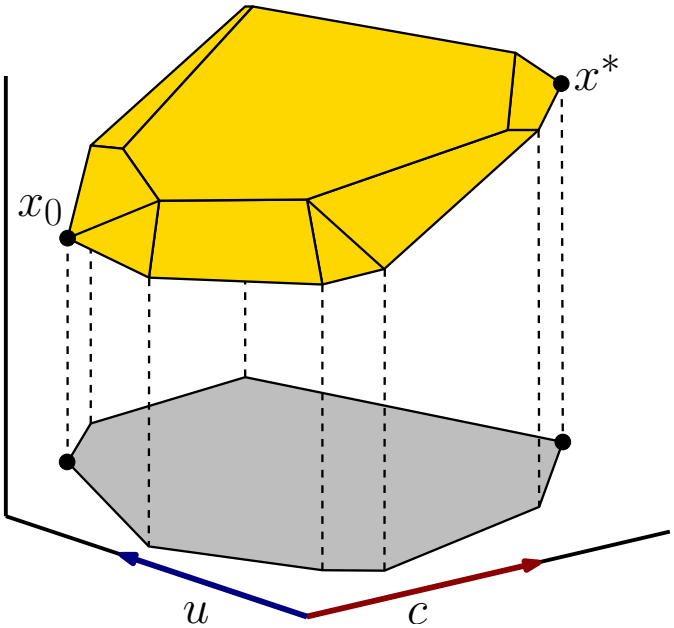
Improved Analysis

Theorem [Vershynin (FOCS 2006)]

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

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

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.

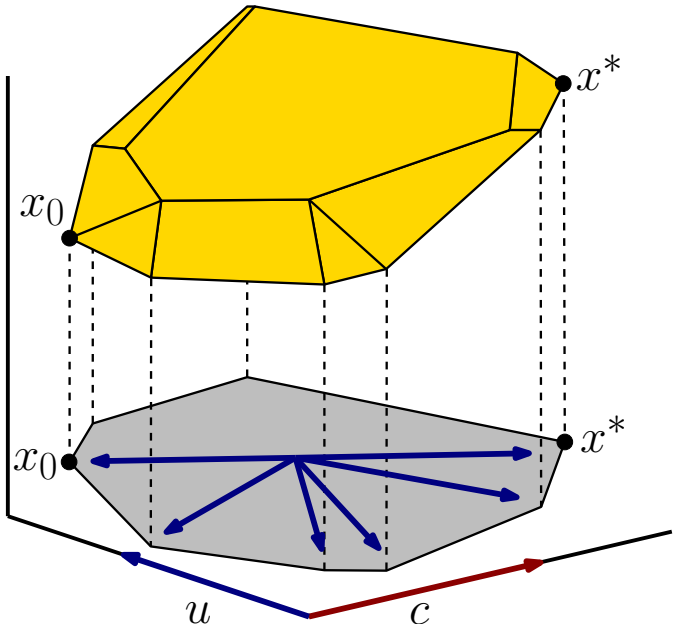
Improved Analysis

Theorem [Vershynin (FOCS 2006)]

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

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

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.

Overview of the coming Lectures

Smoothed Analysis

- 2-Opt heuristic for the traveling salesperson problem
- Nemhauser/Ullmann algorithm for the knapsack problem