ORF 522

Linear Programming and Convex Analysis

Efficiency of the Simplex

Marco Cuturi

Reminder: Initial Solutions and Particular Cases

- **M-method**: use a big M to get an initial BFS directly to the **modified** LP

$$\begin{array}{ll} \text{maximize} & x_0 = \mathbf{c}^T\mathbf{x} - M\mathbb{1}^T\mathbf{u} \\ \text{subject to} & \left\{ \begin{array}{rcl} A\mathbf{x} + I_m\mathbf{u} & = & \mathbf{b} \\ \mathbf{x}, \mathbf{u} & \geq & 0 \end{array} \right. \end{array}$$

- **Two-phase method**:
  - bring x_0 to zero in phase 1 to get a correct BFS for phase 2.

$$\begin{array}{ll} \text{maximize} & x_0 = -\mathbb{1}^T\mathbf{u} \\ \text{subject to} & \left\{ \begin{array}{rcl} A\mathbf{x} + I_m\mathbf{u} & = & \mathbf{b} \\ \mathbf{x}, \mathbf{u} & \geq & 0 \end{array} \right. \end{array}$$

- **Cycling** never happens but... we can solve it.
  - Perturbations,
  - Lexicographic pivot rule,
  - Bland's pivot rule.

Today

- Efficiency
- Kleen-Minty counter-example
- Average performance of the simplex in practice
- Randomized rules
- Smoothed Analysis

Measures of Efficiency

Simplex: efficiency

- We have examined the simplex algorithm completely.
- Works in every situation: unbounded, infeasible, degenerate, etc
- The question is now: how **fast**?

- Quite important one: the simplex is a **discrete** and **combinatorial** algorithm.
- The **combinatorial** makes it a suspect for being quite time consuming.

Simplex: efficiency

- Efficiency measures:
  - Should be a function of the problem size, characteristics
  - Should be easy to compute
  - Should work for entire classes of problems

- For linear programming, two classic answers:
  - **Worst case**: Time it takes the simplex to find a solution to the hardest problem in a class
  - **Average case**: Time it takes the simplex to finish, averaged over random problems in a class

Simplex: efficiency

- **Worst case** analysis:
  - Most common measure
  - More tractable
  - Does not reflect **practical** performance
- **Average case** analysis:
  - Hard to evaluate explicitly
  - Equally difficult to define the priors for the problem
  - Mostly empirical results
- Why: it's easier to measure the complexity of only **one bad** program, it also produces an **upper bound** on the computing time.

Simplex: efficiency

- Measures of problem **size**:
  - Number of constraints m, number of variables d
  - We assume canonical forms usually, with obvious "standardizations" if necessary.
  - Number of parameters $(d+1)(m+1)$
- Measures of **computing time**:
  - Total number of iterations
  - CPU time of each iteration (in flops, or floating point operations)
- Up to a multiplicative constant: written $O(n^2)$ for example if require time is proportional to n^2.. . .

Simplex: efficiency

- Problems are usually classified according to their worst-case complexity:
  - **Polynomial** problems: the worst-case total CPU time is a polynomial function of the problem size
  - **Non polynomial** problems: the worst-case total CPU time grows faster than **all** polynomial functions of the problem size (very often: exponential)
- Examples:
  - Computing a matrix times vector product is $O(d^2)$ in $\mathbf{R}^d$
  - Combinatorial problems are usually exponential (sparse linear programs, integer programs, etc)

Simplex: efficiency

- Verdict on the **simplex method** :
  - For **all** known deterministic pivot rules, there are problems for which the **simplex** method takes an exponential (λ^m) number of pivots.
  - However, **good** performance in practice.
  - In applications, the convergence only takes **a few times** m steps.

What about Linear Programming?

- However, **linear programming** is (relatively) **easy**:
  - Can prove **theoretically** that linear programming is polynomial
  - This is also true in **practice**: interior point algorithms produce a solution in $O(d^{3.5})$.
  - Interesting contrast: bounds for the **simplex** are usually in the **number of constraints**, **IPM** to the **number of variables**.

  - Finding a pivot rule that makes the simplex polynomial in the worst case is still an open problem: we do not know whether such a rule exists.

- any intuition why?

Polyhedra, number of vertices

Simplex: more accurate numbers

- Consider the vertices of the feasible set.

- 2 scenarios:
  - **the best one, the oracle path**:
    Hirsch conjecture $\rightarrow$ any two vertices of a **bounded polyhedron** in $\mathbf{R}^d$ defined by m linear inequalities $(m > d)$ may be connected to each other by a path of at most $m - d$ edges.

  - **the bad one, visiting all vertices**: tight upperbound (McMullen 1970) for the number of vertices of $\{A\mathbf{x} \leq \mathbf{b}\}$, $A \in \mathbf{R}^{m\times d}$:

$$\binom{m - \lfloor \frac{d+1}{2} \rfloor}{m-d} + \binom{m - \lfloor \frac{d+2}{2} \rfloor}{m-d} = O(m^{\lfloor \frac{d}{2} \rfloor}) \qquad (1)$$

Simplex: Intuitions on the Number of Vertices

- Feasible region is $\{A\mathbf{x} \leq \mathbf{b}\} \cap \{\mathbf{x} \geq \mathbf{0}\}$.
- We assume nonnegativity constraints are in the m inequalities $\Rightarrow m > d$.
- Any vertex is at the intersections of at least d hyperplanes of those described in m.
- A very loose upper-bound would be $\binom{m}{d}$ vertices.
- For instance, using Stirling's approximations and assuming $m \gg d$,

$$\binom{m}{d} \approx \frac{m^d}{d!}$$

- Order of m^d...

Simplex: Number of Vertices

- Remember there is a finer bound

$$\binom{m - \lfloor\frac{d+1}{2}\rfloor}{m-d} + \binom{m - \lfloor\frac{d+2}{2}\rfloor}{m-d} = O(m^{\lfloor\frac{d}{2}\rfloor})$$

  the bound is similar, except we divided d by 2!

- $m = 16, d = 8$, the loose upperbound gives 12870, the McMullen upper-bound gives 660.
- For symmetric (around zero) probability densities for the a_{ij} and b_i, the expected number of vertices is exactly...$\frac{\binom{m}{d}}{2^{m-d}}$(Prekopa 72).
- On the other hand the Hirsch conjecture gives an idea of a lower bound: **m-d** steps given a BFS.
- The simplex **could** converge in a multiple of m iterations.
- Is this guaranteed?

Klee Minty counterexample

No. Real issues for some pathological cases.

Klee Minty counterexample

First such example by Klee and Minty in 1972:

$$\begin{array}{lrll} \text{maximize} & \displaystyle\sum_{j=1}^{d} 10^{d-j} x_j & & \\ \text{subject to} & \displaystyle 2\sum_{j=1}^{i-1} 10^{i-j} x_j + x_i & \leq 100^{i-1} & i = 1, 2, \ldots, m \\ & x_j & \geq 0 & j = 1, 2, \ldots, d. \end{array}$$

In practice, this looks like:

$$\begin{array}{rcrcrcr} x_1 & & & & & \leq & 1 \\ 20x_1 & + & x_2 & & & \leq & 100 \\ 200x_1 & + & 20x_2 & + & x_3 & \leq & 10000. \end{array}$$

Simplex: efficiency

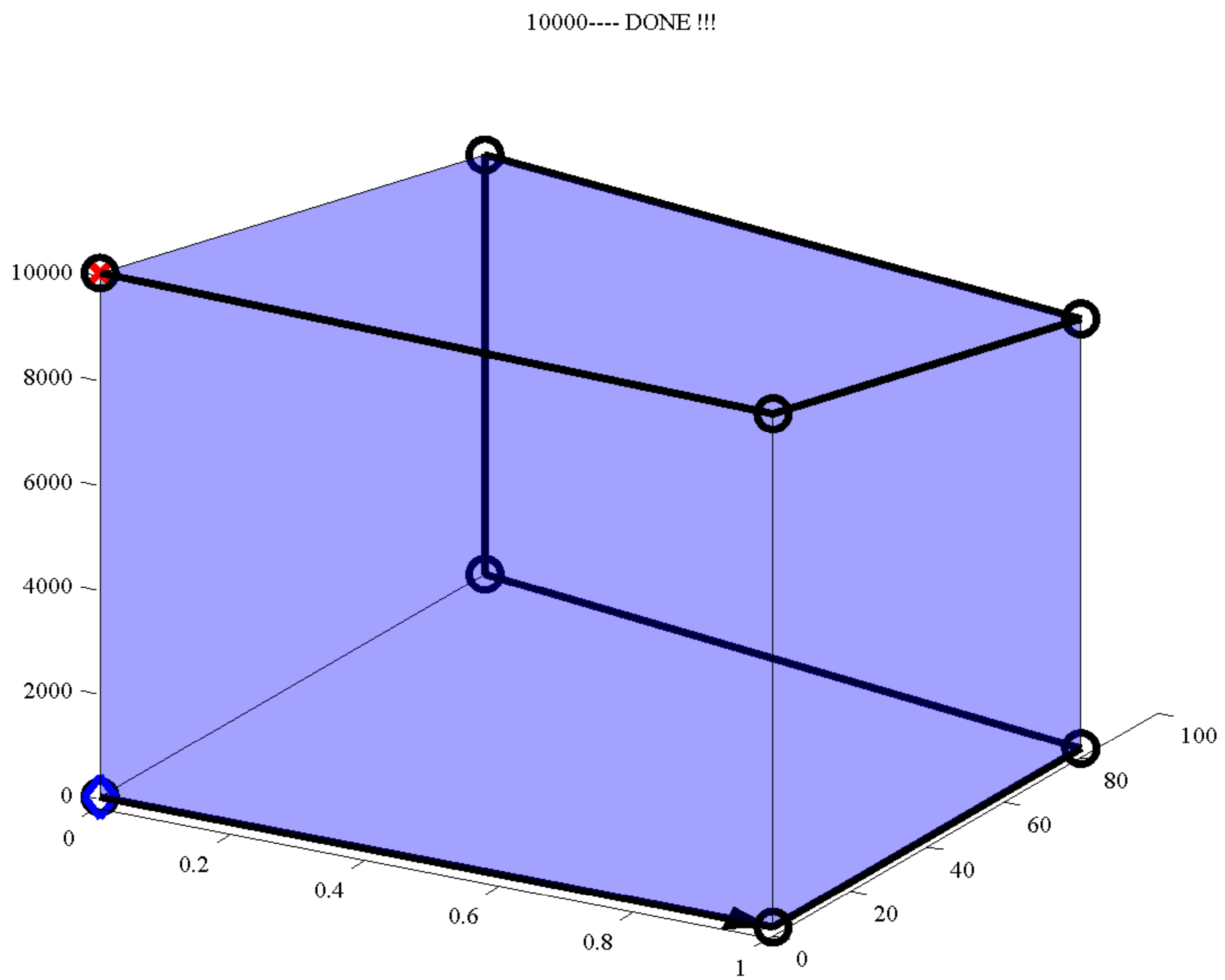
Intuition behind this problem:

- A hypercube in dimension m has 2^m vertices
- The constraints in the K&M problem are *roughly* equivalent to:

$$
\begin{aligned}
0 \leq \quad & x_1 & \leq 1 \\
0 \leq \quad & x_2 & \leq 100 \\
& \vdots & \\
0 \leq \quad & x_d & \leq 100^{d-1}.
\end{aligned}
$$

- The pivot rule choosing the largest **reduced cost coefficient** will visit every vertex of that box before reaching the solution

Klee-Minty: Matlab demo



Tableaux for Klee Minty

- Let us check the corresponding tableaux.

1	0	0	1	0	0	1
20	1	0	0	1	0	100
200	20	1	0	0	1	10000
100	10	1	0	0	0	0

1	0	0	1	0	0	1
0	**1**	0	-20	1	0	80
0	20	1	-200	0	1	9800
0	**10**	1	-100	0	0	-100

1	0	0	**1**	0	0	1
0	1	0	-20	1	0	80
0	0	1	200	-20	1	8200
0	0	1	**100**	-10	0	-900

1	0	0	1	0	0	1
20	1	0	0	1	0	100
-200	0	**1**	0	-20	1	8000
-100	0	**1**	0	-10	0	-1000

Tableaux for Klee Minty

1	0	0	1	0	0	1
20	1	0	0	1	0	100
−200	0	1	0	−20	1	8000
100	0	0	0	10	−1	−9000

1	0	0	1	0	0	1
0	1	0	−20	1	0	80
0	0	1	200	−20	1	8200
0	0	0	−100	**10**	−1	−9100

1	0	0	1	0	0	1
0	1	0	−20	1	0	80
0	20	1	−200	0	1	9800
0	−10	0	**100**	0	−1	−9900

1	0	0	1	0	0	1
20	1	0	0	1	0	100
200	20	1	0	0	1	10000
−100	**−10**	0	0	0	**−1**	−10000

Simplex: efficiency

- Suppose now that we do a simple change of variables:

$$u_j = 100^{1-j} x_j$$

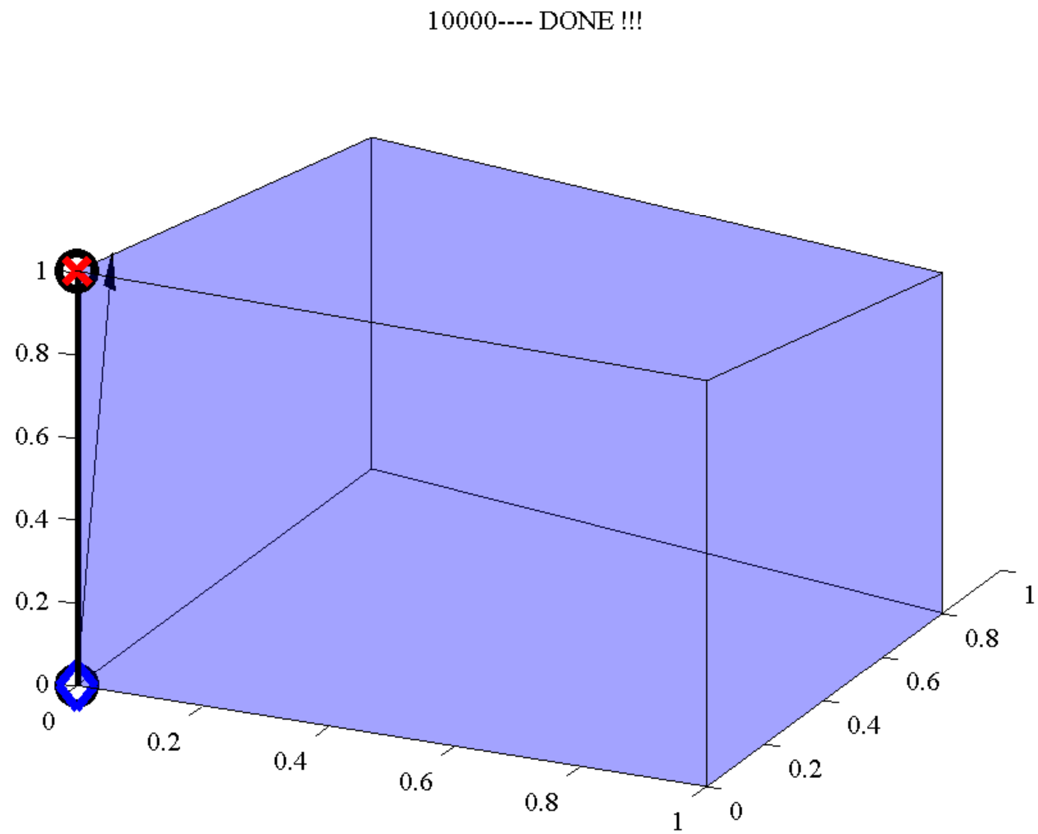
- that is $u_1 = x_1, 100u_2 = x_2$ and $10000u_3 = x_3$
- This is just a scaling of the variables and should not (ideally) affect the complexity of the problem
- The constraint become:

$$\begin{array}{rcrcrcr} u_1 & & & & & \leq & 1 \\ 20u_1 & + & 100u_2 & & & \leq & 100 \\ 200u_1 & + & 2000u_2 & + & 10000u_3 & \leq & 10000. \end{array}$$

- The objective: maximize $100u_1 + 1000u_2 + 10000u_3$.

Simplex: efficiency

- everything should be the same yet...



Tableaux for Klee Minty

- Only one pivot

1	0	0	1	0	0	1
20	100	0	0	1	0	100
200	2000	**10000**	0	0	1	10000
100	1000	**10000**	0	0	0	0

1	0	0	1	0	0	1
20	100	0	0	1	0	100
0	0	1	0	0	0	1
−100	**−1000**	0	0	0	**−1**	−10000

- Embarassing!

Simplex: efficiency

- After the change of variable, the simplex method performs much better...
- This means that the **largest reduced cost coefficient** rule is probably not the most reasonable choice.
- There exist pivot rules for the simplex that are **scale invariant**
- However: **K&M examples have also been found** for most of these rules

Simplex: efficiency

- Klee and Minty show that the largest coefficient rule takes $2^m - 1$ pivots to solve a given problem with m variables and constraints.
- For $m = 70$, this means
$$2^m = 1.2\ 10^{21} \text{ pivots}$$
- At 1000 iterations per second, it will take 40 billion years to solve the problem. (The age of the universe is estimated at 15 billion years)
- On the other hand, very large problems are solved routinely with $m = 10,000$.
- Conclusion here: simplex can take an exponential amount of time on pathological problems.

Simplex: efficiency

n	n^2	n^3	2^n
1	1	1	2
2	4	8	4
3	9	27	8
4	16	64	16
5	25	125	32
6	36	216	64
7	49	343	128
8	64	512	256
9	81	729	512
10	100	1000	1024
12	144	1728	4096
14	196	2744	16384
16	256	4096	65536
18	324	5832	262144
20	400	8000	1048576
22	484	10648	4194304
24	576	13824	16777216
26	676	17576	67108864
28	784	21952	268435456

Simplex: efficiency

Complexity: a few examples for comparison. . .

- Sorting: fast algorithm $O(n \log n)$, simple one $O(n^2)$
- Matrix - matrix product: $O(n^3)$
- Matrix inverse: $O(n^3)$
- Linear Programming with Simplex, worst case: $O(n^2 2^n)$
- Linear Programming with Simplex, average case: $O(n^3)$
- Linear Programming with interior point methods: $O(n^{3.5})$

Simplex: empirical efficiency

Simplex: Complexity History

- Monte-Carlo simulations were pioneered in the 63 (Kuhn& Quandt)
- Objective $\mathbf{c} = \mathbf{1}$, $\mathbf{b} = 10000 \cdot \mathbf{1}$ and and each entry of A selected uniformly between 1 and 1000.
- Limited computational powers: dimensions 5 to 25.
- Computations made on a super-computer in the E-quad (Von Neumann Hall).
- 9 different pivot rules.
- Very successful: again, convergence below $3 \cdot m$ pivots.

- Ironically, this success *might have slowed down* research on other methodologies.

Simplex: Complexity History

- Researchers spent years trying to prove that the simplex worst-case complexity was polynomial.
- The '72 Klee-Minty counter-example killed such hopes.
- For **most advanced pivot rules** there has been a KM type counterexample.

- No pivot rule guaranteed to yield worst-case polynomial time yet.
- Yet practical performance definitely competitive...
- Spurred alternative ways to analyze the simplex and propose pivots.

Simplex: Complexity History

3 more precise questions

1. Given **random problems**, what are the **average** finishing times for a deterministic pivot rule?

2. Given **random pivot rules**, what is the **worst** average finishing time?

3. Given a problem that is randomly **perturbed**, what is the finishing time when **averaged over all perturbations**?

we only give results here, for proofs you should check the original papers.

1. Random Problems, Deterministic Rules, Upper Bound on Average Time

- Topic of intense study in 70' and 80's.
- Borgwardt pioneered such studies in 82, followed by Smale.
- All consider $\mathbf{b}, \mathbf{a}_i$ **i.i. distributed** with a **rotationally symmetric distribution** (density $p(\mathbf{x}) = p(O\mathbf{x})$ where O orthonormal, $e.g.$ centered isotrope Gaussian) and $\mathbf{c} = \mathbf{1}$.
- Simple pivot rules are considered.
- Borgwardt obtains a polynomial upper-bound of $O(m^3 d^{\frac{1}{m-1}})$.
- Mainly theoretical: real life problems **do not satisfy i.i.d assumptions** on coefficients...

2. Random Rules, Upper Bound on Average Worst Time

- Fostered by Kalai's search, closer to us: 90's.
- Randomization is natural. Consider the largest reduced cost coefficient. If tie, choose randomly.
- Some vocabulary first...

Faces

Definition 1. *Let K be a closed convex set. A set $F \subset K$ is called a* **face** *of K if there exists an affine hyperplane H which isolates K and such that $F = K \cap H$.*

Definition 2. *The* **dimension** *of a convex set $K \subset \mathbf{R}^d$ is the dimension of the smallest affine subspace that contains K*

- **remark**
  1. A face K of dimension 1 is an **exposed point**.
  2. A face K of dimension 2 is an **edge**.
  3. A face K of dimension $d-2$ is called a **ridge**.
  4. A face K of dimension $d-1$ is called a **facet**.
- A facet F of P is active w.r.t $\mathbf{v}$ if $c^T\mathbf{v} < \max\{c^T\mathbf{x}, \mathbf{x} \in F\}$.

2. Random Rules, Upper Bound on Worst Average Time

- Kalai in 92 and 97 proposed the following (recursive) random pivot Rule.
- Given a vertex in a polyhedron P, the following algorithm finds the vertex $\mathbf{w}$ that maximizes $c^T\mathbf{w}$:
- Algorithm I:
  - Given a vertex $\mathbf{v}$, of all **active facets** $F_1, F_2, \cdots, F_k$ that contain $\mathbf{v}$, choose one, F_i, randomly with uniform probability.
  - Apply the algorithm recursively (lowering the dimension) to find the best vertex $\mathbf{u}$ in F_i.
  - If $\mathbf{u} = \mathbf{v}$ terminate. Otherwise apply the algorithm to $\mathbf{u}$.
- The algorithm is a naive random exploration where the exploration goes from a facet in a given dimension to a facet in a lower dimension.

2. Random Rules, Upper Bound on Worst Average Time

- For a linear problem $U = (A, \mathbf{b}, \mathbf{c})$, starting with a vertex $\mathbf{v}$ in the feasible set with $r \leq d$ active facets, let $f(U, \mathbf{v})$ the expected number of pivots of the algorithm above.
- $f(d, r)$ is the maximal value (worst average) of $f(U, \mathbf{v})$ over all problems U and vertices $\mathbf{v}$ with r active facets.
- Kalai shows that $f(d, r) \leq \exp^{C\sqrt{r \log d}}$ that is more generally a $\exp^{C\sqrt{d \log d}}$ bound.
- Hence the name of a **subexponential** rule.

2. Random Rules, Upper Bound on Worst Average Time

- **NO** practical interest. Randomized algorithms are **not competitive**.
- Only useful to circumvent the issue of having a deterministic strategy "*attacked*" by a counterexample.
- A random strategy performs badly on average, but its weaknesses cannot be exploited to yield pathological scenarios in KM style.

- Also useful as **proof strategies for the Hirsch conjecture**.

3. Perturbations of the original problem

- In smoothed analysis, the LP $(A, \mathbf{b}, \mathbf{c})$ is seen as the realization of a random problem.
- Parameters $A, \mathbf{b}$ are centered around $\bar{A}, \bar{\mathbf{b}}$ with a variance width σ.
- Spielman and Teng prove that the average complexity of solving such an LP is at most a polynomial of $d, n, \frac{1}{\sigma}$.
- Namely that there exists a polynomial P and a constant σ_0 such that for every $n, d \geq 3$,
$$E_{A,\mathbf{b}}[C(A, \mathbf{b}, \mathbf{c})] \leq P(d, n, \tfrac{1}{\min(\sigma,\sigma_0)}).$$
- Polynomial expected complexity around any arbitrary problem.
- Again... purely theoretical

Next time

Duality