OSE SEMINAR 2013

Complexity theory for the global optimizer

Anders Skjäl

CENTER OF EXCELLENCE IN
OPTIMIZATION AND SYSTEMS ENGINEERING
ÅBO AKADEMI UNIVERSITY

ÅBO, NOVEMBER 15 2013

What is Global Optimization?

Several interpretations:

- Many trials increase the chance of catching the global optimum
  - Multistart, scatter search
  - Metaheuristics: genetic algorithms, simulated annealing, particle swarm, ant colony...

What is Global Optimization?

Several interpretations:

- Many trials increase the chance of catching the global optimum
  - Multistart, scatter search
  - Metaheuristics: genetic algorithms, simulated annealing, particle swarm, ant colony...
- Many of the algorithms converge asymptotically to a global optimum with probability 1
  - No way to know when it happens

What is Global Optimization?

Several interpretations:

- Many trials increase the chance of catching the global optimum
  - Multistart, scatter search
  - Metaheuristics: genetic algorithms, simulated annealing, particle swarm, ant colony...
- Many of the algorithms converge asymptotically to a global optimum with probability 1
  - No way to know when it happens
- A method classification by Neumaier (2004)
  - **Incomplete**: clever heuristics
  - **Asymptotically complete**: converges eventually
  - **Complete**: converges and knows when prescribed tolerance is reached
  - **Rigorous**: converges despite rounding errors (floating point arithmetic)
- "Deterministic global optimization"

What is Complexity Theory?

- Aims to establish absolute limits for how fast a problem can be solved
- Algorithm-independent

What is Complexity Theory?

- Aims to establish absolute limits for how fast a problem can be solved
- Algorithm-independent
- Results are sometimes negative
  - $n \times n$ chess/checkers/go cannot be solved in polynomial time
  - Conjecture A implies that problem B is C-hard

What is Complexity Theory?

- Aims to establish absolute limits for how fast a problem can be solved
- Algorithm-independent
- Results are sometimes negative
  - $n \times n$ chess/checkers/go cannot be solved in polynomial time
  - Conjecture A implies that problem B is C-hard
- The terminology is well-suited for studying global optimization (in the strict sense)

I will discuss some concepts in complexity theory and their implications for our field

- The NP class, approximation complexity, randomized algorithms

Algorithm Analysis

- First contact with runtimes
- Big O notation
  - $f(n) = O(g(n))$ if f is asymptotically bounded from above by a constant times g

Algorithm Analysis

- First contact with runtimes
- Big O notation
  - $f(n) = O(g(n))$ if f is asymptotically bounded from above by a constant times g
- Also Ω (asymptotically from below) and Θ (asymptotically from above and below)

Algorithm Analysis

- First contact with runtimes
- Big O notation
  - $f(n) = O(g(n))$ if f is asymptotically bounded from above by a constant times g
- Also Ω (asymptotically from below) and Θ (asymptotically from above and below)
- Sorting algorithms
  - Naïve sort: $O(n^2)$ operations
  - Quicksort (1960): $O(n \log n)$ operations on average
  - Merge sort (1945): $O(n \log n)$ operations in worst case

Improved Runtimes

- Asymptotic improvements may have practical significance
  - Fast Fourier transform, $O(n \log n)$
  - Matrix multiplication
    - Trivial bounds: $O(n^3), \Omega(n^2)$
    - Strassen algorithm (1969): $O(n^{2.81})$

Improved Runtimes

- Asymptotic improvements may have practical significance
  - Fast Fourier transform, $O(n \log n)$
  - Matrix multiplication
    - Trivial bounds: $O(n^3), \Omega(n^2)$
    - Strassen algorithm (1969): $O(n^{2.81})$
- ...or not
  - Coppersmith-Winograd algorithm (1987), $O(n^{2.38})$ more efficient only for enormous matrices

Tractability

Polynomial runtime is sometimes considered synonymous with "reasonable algorithm"

- Define P
  - the class of decision problems with a $O(n^k)$ algorithm

Tractability

Polynomial runtime is sometimes considered synonymous with "reasonable algorithm"

- Define P
  - the class of decision problems with a $O(n^k)$ algorithm
- Objections:
  - $O(2^n)$ is ok for small n

Tractability

Polynomial runtime is sometimes considered synonymous with "reasonable algorithm"

- Define P
  - the class of decision problems with a $O(n^k)$ algorithm
- Objections:
  - $O(2^n)$ is ok for small n
  - $O(2^{0.000001n})$ is better than $O(n^{100})$ for moderate n (artificial)

Tractability

Polynomial runtime is sometimes considered synonymous with "reasonable algorithm"

- Define P
  - the class of decision problems with a $O(n^k)$ algorithm
- Objections:
  - $O(2^n)$ is ok for small n
  - $O(2^{0.000001n})$ is better than $O(n^{100})$ for moderate n (artificial)
- Polynomial on what machine?
  - Any classical computer

Tractability

Polynomial runtime is sometimes considered synonymous with "reasonable algorithm"

- Define P
  - the class of decision problems with a $O(n^k)$ algorithm
- Objections:
  - $O(2^n)$ is ok for small n
  - $O(2^{0.000001n})$ is better than $O(n^{100})$ for moderate n (artificial)
- Polynomial on what machine?
  - Any classical computer
  - Turing machines, random-access machines and other theoretical machines can simulate each other with only polynomial slow-down
  - Not true for quantum computers as far as we know

Example: Satisfiability

Does any truth assignment of the Boolean variables x, y, z satisfy the expression:

$$(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y \vee \bar{z})$$

Example: Satisfiability

Does any truth assignment of the Boolean variables x, y, z satisfy the expression:

$$(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (x \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee y \vee \bar{z})$$

- A satisfiability problem, 3-SAT
- Naïve algorithm: $O(2^n)$
- Best known: $O(1.439^n)$

“Easy to verify”

- Many difficult problems have this in common:
  - A ′yes′ answer can be verified quickly by checking a candidate (proof/certificate/witness)
- Define the class NP of problems for which any ′yes′ instance has a proof which can be checked in polynomial time

"Easy to verify"

- Many difficult problems have this in common:
  - A 'yes' answer can be verified quickly by checking a candidate (proof/certificate/witness)
- Define the class NP of problems for which any 'yes' instance has a proof which can be checked in polynomial time
  - Formal definitions with a Turing machine (checking the proof) or a non-deterministic Turing machine (guessing the proof)
  - P $\subset$ NP

"Easy to verify"

- Many difficult problems have this in common:
  - A 'yes' answer can be verified quickly by checking a candidate (proof/certificate/witness)
- Define the class NP of problems for which any 'yes' instance has a proof which can be checked in polynomial time
  - Formal definitions with a Turing machine (checking the proof) or a non-deterministic Turing machine (guessing the proof)
  - $P \subset NP$
- Examples
  - Decision versions of many optimization problems, discrete and continuous
  - Is $\min_{x \in X} f(x) < M$? If it is, then a point $x_0 \in X, f(x_0) < M$ is a proof
  - Graph isomorphism, traveling salesman, quadratic assignment, longest path, bin packing, knapsack, ...

"Easy to verify"

- Many difficult problems have this in common:
  - A ′yes′ answer can be verified quickly by checking a candidate (proof/certificate/witness)
- Define the class NP of problems for which any ′yes′ instance has a proof which can be checked in polynomial time
  - Formal definitions with a Turing machine (checking the proof) or a non-deterministic Turing machine (guessing the proof)
  - $P \subset NP$
- Examples
  - Decision versions of many optimization problems, discrete and continuous
  - Is $\min_{x \in X} f(x) < M$? If it is, then a point $x_0 \in X, f(x_0) < M$ is a proof
  - Graph isomorphism, traveling salesman, quadratic assignment, longest path, bin packing, knapsack, ...
  - Games like Battleships, Mastermind, Minesweeper, ...

Complete Problems

Some problems in NP are *as hard as any other* NP *problem*

- If one of these problems can be solved in polynomial time, then so can any NP problem
- Cook and Levin proved that any NP problem can be *reduced* to (reformulated as) a satisfiability problem
- Karp (1972) listed 21 problems with the property

Complete Problems

Some problems in NP are *as hard as any other* NP *problem*

- If one of these problems can be solved in polynomial time, then so can any NP problem
- Cook and Levin proved that any NP problem can be *reduced* to (reformulated as) a satisfiability problem
- Karp (1972) listed 21 problems with the property
- Definition: C belongs to the class of NP-*hard* problems if
  - every problem in NP can be reduced to C in polynomial time
- Definition: C belongs to the class of NP-*complete* problems if
  - C is NP-hard, and
  - $C \in$ NP

Proving NP-hardness

If B is NP-complete and B reduces to C (in polynomial time), then C is NP-hard

- Three common techniques: restriction, local replacement, component design

Proving NP-hardness

If B is NP-complete and B reduces to C (in polynomial time), then C is NP-hard

- Three common techniques: restriction, local replacement, component design
- Restriction: 0-1 Linear Programming is NP-complete
  $\Rightarrow$ MINLP is NP-hard

Proving NP-hardness

If B is NP-complete and B reduces to C (in polynomial time), then C is NP-hard

- Three common techniques: restriction, local replacement, component design
- Restriction: 0-1 Linear Programming is NP-complete
  $\Rightarrow$ MINLP is NP-hard
- Local replacement: Satisfiability to 3-SAT
  - A local replacement of long clauses by clauses with three literals

$$x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$$
$$\downarrow$$
$$(x_1 \vee x_2 \vee y_1) \wedge (\bar{y_1} \vee x_3 \vee y_2) \wedge (\bar{y_2} \vee x_4 \vee x_5)$$

A recreational example: Lemmings is NP-complete (Cormode 2004)

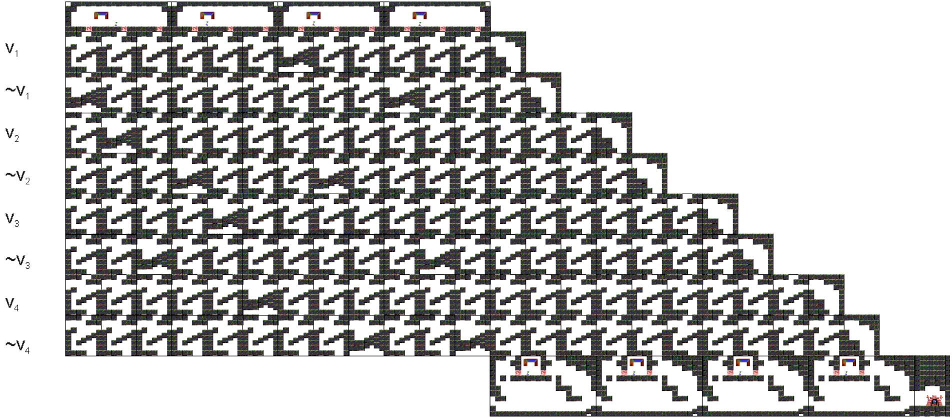


Figure 5: Lemmings level encoding the formula $(\bar{v_1} \vee v_2 \vee \bar{v_3}) \wedge (\bar{v_2} \vee v_3 \vee v_4) \wedge (v_1 \vee \bar{v_2} \vee \bar{v_4}) \wedge (\bar{v_1} \vee \bar{v_3} \vee \bar{v_4})$

Is $P \neq NP$**?**

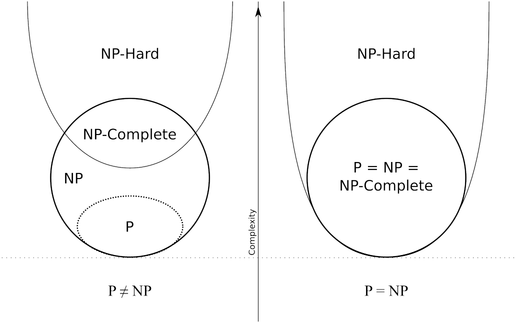
- Maybe all problems in NP have an efficient algorithm?

Is $P \neq NP$**?**

- Maybe all problems in NP have an efficient algorithm?
- Why didn't we find one yet?

Is P ≠ NP?

- Maybe all problems in NP have an efficient algorithm?
- Why didn't we find one yet?
- Often stated as "P versus NP", one of the Millennium Prize problems



Decision Problems and Optimization

- Optimization problem:

  What is the minimum value of $f(x), x \in X$?
- Related decision problem:

  Is there a solution $x \in X$ with $f(x) \leq M$?

Decision Problems and Optimization

- Optimization problem:
  What is the minimum value of $f(x), x \in X$?
- Related decision problem:
  Is there a solution $x \in X$ with $f(x) \leq M$?
- The optimization version is at least as hard as the decision version
- In practice we rarely need the exact optimum

Decision Problems and Optimization

- Optimization problem:

  What is the minimum value of $f(x), x \in X$?

- Related decision problem:

  Is there a solution $x \in X$ with $f(x) \leq M$?

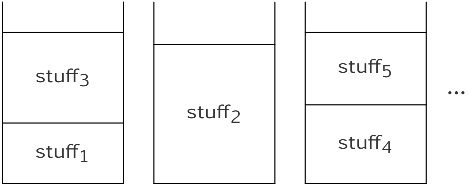
- The optimization version is at least as hard as the decision version
- In practice we rarely need the exact optimum
- Is approximation any easier?

Approximative Solutions

A solution x to a minimization problem is ε-optimal if:

$$f(x) \leq (1+\varepsilon)\, f(x^*)$$

- Example: Bin packing, first-fit

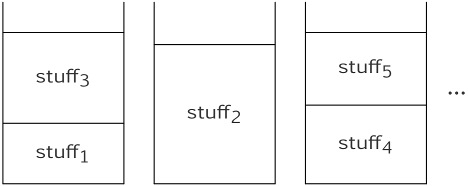


Approximative Solutions

A solution x to a minimization problem is ε-optimal if:

$$f(x) \leq (1+\varepsilon)\, f(x^*)$$

- Example: Bin packing, first-fit



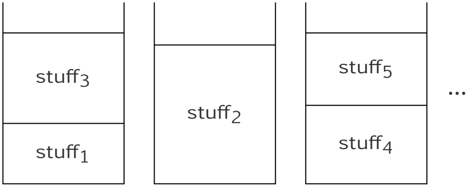
- The optimal number of bins $N^* \geq \lceil \sum \mathrm{stuff}_i \rceil$
- The first-fit solution $N_{FF} < \lceil 2 \sum \mathrm{stuff}_i \rceil$

Approximative Solutions

A solution x to a minimization problem is ε-optimal if:

$$f(x) \leq (1+\varepsilon)\, f(x^*)$$

- Example: Bin packing, first-fit



- The optimal number of bins $N^* \geq \lceil \sum \mathrm{stuff}_i \rceil$
- The first-fit solution $N_{FF} < \lceil 2 \sum \mathrm{stuff}_i \rceil$
- $N_{FF} < 2N^*$ (an improved analysis gives $N_{FF} \leq 1.7N^* + 2$)

Approximation Complexity

By reducing approximation problems to NP-complete decision problems they are shown to be hard

- The gap technique
  - Objective range $\subset (0, a] \cup [b, +\infty)$
  - If it is *NP*-hard to decide if the minimum belongs to (0,a]
  - ... then approximation within $\varepsilon = (b - a)/a$ is NP-hard

Approximation Complexity

By reducing approximation problems to NP-complete decision problems they are shown to be hard

- The gap technique
  - Objective range $\subset (0, a] \cup [b, +\infty)$
  - If it is *NP*-hard to decide if the minimum belongs to (0,a]
  - ... then approximation within $\varepsilon = (b - a)/a$ is NP-hard
- A simple application shows that approximation of general Traveling Salesperson problems is *NP*-hard for any constant ε
- Many hardness results followed on the PCP Theorem (Arora et al. 1990)
  - Probabilistically Checkable Proofs

Approximation Complexity

By reducing approximation problems to NP-complete decision problems they are shown to be hard

- The gap technique
  - Objective range $\subset (0, a] \cup [b, +\infty)$
  - If it is *NP*-hard to decide if the minimum belongs to (0,a]
  - ... then approximation within $\varepsilon = (b - a)/a$ is NP-hard
- A simple application shows that approximation of general Traveling Salesperson problems is *NP*-hard for any constant ε
- Many hardness results followed on the PCP Theorem (Arora et al. 1990)
  - Probabilistically Checkable Proofs
- A hierarchy of complexity classes emerges: APX $\supset$ PTAS $\supset$ FPTAS
- The classes are not equal unless P $=$ NP

APX - efficient approximation within constant ε

- Polynomial time approximation algorithms for some constant ε

APX - **efficient approximation within constant ε**

- Polynomial time approximation algorithms for some constant ε
- Metric Traveling Salesperson (symmetric distances, triangle inequality)
  - Christofides' algorithm, $\varepsilon = \frac{1}{2}$, $O(n^3)$
  - Approximation with $\varepsilon < \frac{1}{219}$ is NP-hard (Papadimitriou and Vempala 2000)

APX - **efficient approximation within constant ε**

- Polynomial time approximation algorithms for some constant ε
- Metric Traveling Salesperson (symmetric distances, triangle inequality)
  - Christofides' algorithm, $\varepsilon = \frac{1}{2}$, $O(n^3)$
  - Approximation with $\varepsilon < \frac{1}{219}$ is NP-hard (Papadimitriou and Vempala 2000)
- $\notin$ APX: Linear Integer Programming, general TSP, and Quadratic Assignment have no efficient approximation algorithms for any constant ε (unless $\mathrm{P} = \mathrm{NP}$)

Polynomial Time Approximation Schemes

PTAS: problems with polynomial time approximation algorithms for any ε

- Geometric Traveling Salesperson
  - Euclidean distances or $l_p, p \geq 1$ norm
  - Dimension d

Polynomial Time Approximation Schemes

PTAS: problems with polynomial time approximation algorithms for any ε

- Geometric Traveling Salesperson
  - Euclidean distances or $l_p, p \geq 1$ norm
  - Dimension d
- $O\left(n^{d+1}(\log n)^{(O(\sqrt{d}/\varepsilon))^{d-1}}\right)$ (Arora 1998)
- Two dimensions: $O\left(n^3(\log n)^{O(1/\varepsilon)}\right)$

Polynomial Time Approximation Schemes

PTAS: problems with polynomial time approximation algorithms for any ε

- Geometric Traveling Salesperson
  - Euclidean distances or $l_p, p \geq 1$ norm
  - Dimension d
- $O\left(n^{d+1}(\log n)^{(O(\sqrt{d}/\varepsilon))^{d-1}}\right)$ (Arora 1998)
- Two dimensions: $O\left(n^3(\log n)^{O(1/\varepsilon)}\right)$
- Grows exponentially with $1/\varepsilon$

Fully Polynomial Time Approximation Schemes

FPTAS: problems with approximation schemes that are polynomial in both n and $1/\varepsilon$

Fully Polynomial Time Approximation Schemes

FPTAS: problems with approximation schemes that are polynomial in both n and $1/\varepsilon$

- Knapsack Problem (see Vazirani 2001)
  - $O(n^3/\varepsilon)$

Fully Polynomial Time Approximation Schemes

FPTAS: problems with approximation schemes that are polynomial in both n and $1/\varepsilon$

- Knapsack Problem (see Vazirani 2001)
  - $O(n^3/\varepsilon)$
- Quadratic Programming
  - Compact polytope, t negative eigenvalues
  - L = complexity of solving a convex QP of the same size
  - (Vavasis 1992)
$$O\left(\left\lceil n(n+1)/\sqrt{\varepsilon}\right\rceil^t L\right)$$
  - Fully polynomial if t is bounded

So which Problems are Hard?

- Not always obvious for discrete problems
- Plenty of references
  - Garey & Johnson: Computers and Intractability (1979)
  - Ausiello et al.: Complexity and Approximation (1999)

So which Problems are Hard?

- Not always obvious for discrete problems
- Plenty of references
  - Garey & Johnson: Computers and Intractability (1979)
  - Ausiello et al.: Complexity and Approximation (1999)
- Continuous problems: "convex easy, nonconvex hard"
  - Polynomial-time interior-point methods for convex programming, Nesterov (1988)
  - Self-concordant barrier functions exist for all closed convex solids

So which Problems are Hard?

- Not always obvious for discrete problems
- Plenty of references
  - Garey & Johnson: Computers and Intractability (1979)
  - Ausiello et al.: Complexity and Approximation (1999)
- Continuous problems: "convex easy, nonconvex hard"
  - Polynomial-time interior-point methods for convex programming, Nesterov (1988)
  - Self-concordant barrier functions exist for all closed convex solids
- Some exceptions
  - Geometric programming: posynomials $cx_1^{p_1} \cdots x_n^{p_n}, c > 0$
  - Linear fractional programming: $(p^T x + \alpha)/(q^T x + \beta)$
  - ...

Will Randomization Help?

Some tasks seem to benefit from using random numbers

- Complexity theory provides frameworks for analysis

Will Randomization Help?

Some tasks seem to benefit from using random numbers

- Complexity theory provides frameworks for analysis
- Genuine randomness versus pseudorandomness

Will Randomization Help?

Some tasks seem to benefit from using random numbers

- Complexity theory provides frameworks for analysis
- Genuine randomness versus pseudorandomness
- ZPP - zero-error probabilistic polynomial-time
  - correct answers in polynomial time, but...

Will Randomization Help?

Some tasks seem to benefit from using random numbers

- Complexity theory provides frameworks for analysis
- Genuine randomness versus pseudorandomness
- ZPP - zero-error probabilistic polynomial-time
  - correct answers in polynomial time, but...
  - returns no answer with probability $\leq 1/2$

Will Randomization Help?

Some tasks seem to benefit from using random numbers

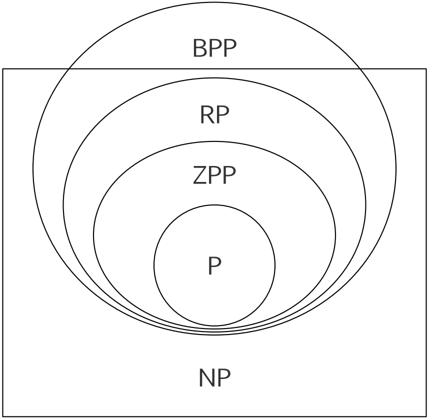
- Complexity theory provides frameworks for analysis
- Genuine randomness versus pseudorandomness
- ZPP - zero-error probabilistic polynomial-time
  - correct answers in polynomial time, but...
  - returns no answer with probability $\leq 1/2$
- BPP - bounded-error probabilistic polynomial-time
  - wrong answer with probability $\leq 1/3$

Will Randomization Help?

Some tasks seem to benefit from using random numbers

- Complexity theory provides frameworks for analysis
- Genuine randomness versus pseudorandomness
- ZPP - zero-error probabilistic polynomial-time
  - correct answers in polynomial time, but...
  - returns no answer with probability $\leq 1/2$
- BPP - bounded-error probabilistic polynomial-time
  - wrong answer with probability $\leq 1/3$
- RP - randomized polynomial-time
  - outputs ′no′ if the correct answer is ′no′
  - outputs ′no′ if the correct answer is ′yes′ with probability $\leq 1/2$

Randomized Decision Classes



Example: MAX-3-SAT

$$\bigwedge_{k=1}^{K} (x_{k_1} \vee x_{k_2} \vee x_{k_3})$$

- A clause (three different literals) is satisfied by a random assignment with probability

$$1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8}$$

- The expected number of satisfied clauses is $\frac{7}{8}K$

Example: MAX-3-SAT

$$\bigwedge_{k=1}^{K} (x_{k_1} \vee x_{k_2} \vee x_{k_3})$$

- A clause (three different literals) is satisfied by a random assignment with probability

$$1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8}$$

- The expected number of satisfied clauses is $\frac{7}{8}K$
- There is always an assignment satisfying $\geq \frac{7}{8}K$ clauses

Example: MAX-3-SAT

$$\bigwedge_{k=1}^{K} (x_{k_1} \vee x_{k_2} \vee x_{k_3})$$

- A clause (three different literals) is satisfied by a random assignment with probability

$$1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8}$$

- The expected number of satisfied clauses is $\frac{7}{8}K$
- There is always an assignment satisfying $\geq \frac{7}{8}K$ clauses
- The fraction of such assignments is $\Omega(1/K)$
- Approximation within $\varepsilon = 1/7$ ($r = 8/7$ in CS texts) is in ZPP

Example: MAX-3-SAT

$$\bigwedge_{k=1}^{K} (x_{k_1} \vee x_{k_2} \vee x_{k_3})$$

- A clause (three different literals) is satisfied by a random assignment with probability

$$1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8}$$

- The expected number of satisfied clauses is $\frac{7}{8}K$
- There is always an assignment satisfying $\geq \frac{7}{8}K$ clauses
- The fraction of such assignments is $\Omega(1/K)$
- Approximation within $\varepsilon = 1/7$ ($r = 8/7$ in CS texts) is in ZPP
- Can be derandomized to give deterministic algorithm

Example: Primality Testing

Is n a prime number?

- Let $D = \log n$, the number of digits in n
- Adleman-Pomerance-Rumely (Jacobi sums), $O(D^{c \log\log D})$
  - Deterministic, not polynomial-time

Example: Primality Testing

Is n a prime number?

- Let $D = \log n$, the number of digits in n
- Adleman-Pomerance-Rumely (Jacobi sums), $O(D^{c \log\log D})$
  - Deterministic, not polynomial-time
- Miller-Rabin, $O(D^2 \log D \log\log D) = \tilde{O}(D^2)$
  - Wrong answer for composite numbers with probability $< 1/4$
  - Primality testing $\in$ coRP

Example: Primality Testing

Is n a prime number?

- Let $D = \log n$, the number of digits in n
- Adleman-Pomerance-Rumely (Jacobi sums), $O(D^{c \log\log D})$
  - Deterministic, not polynomial-time
- Miller-Rabin, $O(D^2 \log D \log\log D) = \tilde{O}(D^2)$
  - Wrong answer for composite numbers with probability $< 1/4$
  - Primality testing $\in$ coRP
- Elliptic Curve Primality Proving
  - Expected runtime $\tilde{O}(D^4)$
  - Primality testing $\in$ ZPP

Example: Primality Testing

Is n a prime number?

- Let $D = \log n$, the number of digits in n
- Adleman-Pomerance-Rumely (Jacobi sums), $O(D^{c \log\log D})$
  - Deterministic, not polynomial-time
- Miller-Rabin, $O(D^2 \log D \log\log D) = \tilde{O}(D^2)$
  - Wrong answer for composite numbers with probability $< 1/4$
  - Primality testing $\in$ coRP
- Elliptic Curve Primality Proving
  - Expected runtime $\tilde{O}(D^4)$
  - Primality testing $\in$ ZPP
- Agrawal-Kayal-Saxena (2002), $\tilde{O}(D^6)$
  - Deterministic
  - Primality testing $\in$ P

Primality testing was known to be in BPP, and now in P

- It has been conjectured that $P = BPP$
- Randomized algorithms might not be fundamentally stronger
- But they may have advantages
  - Lower degree runtimes
  - Sometimes conceptually easier, faster to program
- A probability of errors may be tolerable if it can be bounded
  - Example: “industrial strength primes”

References

G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamala, and M. Protasi.
Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.
Springer, 1999.

G. Cormode.
The hardness of the Lemmings game, or Oh no, more NP-completeness proofs.
In *Proceedings of Third International Conference on Fun with Algorithms*, pages 65–76, 2004.

Michael R. Garey and David S. Johnson.
Computers and Intractability: A Guide to the Theory of NP-Completeness.
W.H. Freeman and Company, 1979.

Arnold Neumaier.
Complete search in continuous global optimization and constraint satisfaction.
Acta Numerica, 13:271–369, 5 2004.

Vijay V. Vazirani.
Approximation Algorithms.
Springer, 2001.

Thank you for listening!

Thank you for listening!

Questions?