Complexity and Algorithms

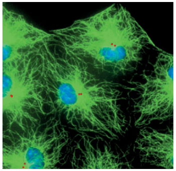
J. Díaz

The pervasive nature of Computation

Computation: any process consisting of a sequence of local steps that we want to perform, or to understand.

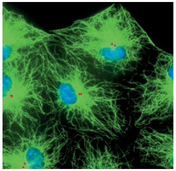
Computation is ubiquitous

Nature



Computation is ubiquitous

Nature

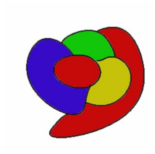


Physics

String Theory

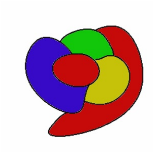
Mathematics

The 4 colour theorem Given any separation of a plane into contiguous regions, the regions can be legally coloured using ≤ 4 colours.



Mathematics

The 4 colour theorem Given any separation of a plane into contiguous regions, the regions can be legally coloured using ≤ 4 colours.



Theorem proving There are not integers $a, b, c \neq 0$ s.t. for any $n > 2$, $a^n + b^n = c^n$.

Economics

If your laptop can't find it, neither can the market Kamail Jain

Economics

If your laptop can't find it, neither can the market Kamail Jain

Understanding the properties of markets that make them actually reach equilibrium. Compute the rate *to achieve equilibrium. Predict the future behaviour of the market, with today data.*

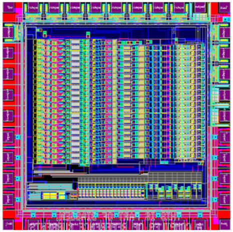
Electrical Engineering

Design reliable communication in cloud computing

Electrical Engineering

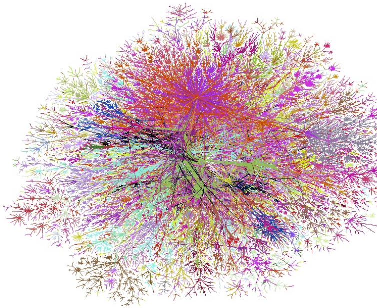
Design reliable communication in cloud computing

Optimise the design of a VLSI



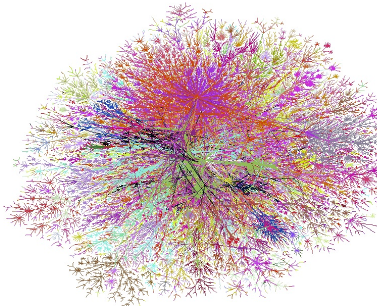
Information retrieval and manipulation

Find *efficiently* a document in the web



Information retrieval and manipulation

Find *efficiently* a document in the web

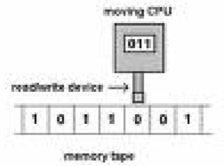


Find *efficiently* two similar documents in the web

The algorithm

The algorithm is the language to describe computation: step by step, local, mechanical procedure, which works for every input in finite number of steps.

Formal definition of the algorithm:
Turing Machine



The algorithm coming of-age as the new language of science, promise to be the most disrupting scientific development since quantum mechanics.

B. Chazelle, The algorithm: Idiom of modern science (2001)

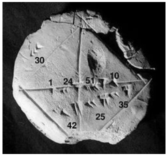
Computer Science is no more about computers than astronomy is about telescopes.

Edsger W. Dijkstra, 1992

Algorithmics are known for a long time.

How to compute $\sqrt{n}$, for $n \in \mathbb{N}$ Baylonian ($\sim$ XVI BC)

```
Given $n$, choose $x_0 \sim \sqrt{n}$
from $i = 0$ to $k$ do
    $x_{i+1} = (x_i + \frac{n}{x_i})/2$
```



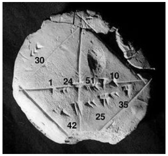
Then $\sqrt{n} \sim \lim_{i \to \infty} x_i$

Algorithmics are known for a long time.

How to compute $\sqrt{n}$, for $n \in \mathbb{N}$ Baylonian ($\sim$ XVI BC)

```
Given $n$, choose $x_0 \sim \sqrt{n}$
from $i = 0$ to $k$ do
    $x_{i+1} = (x_i + \frac{n}{x_i})/2$
```

Then $\sqrt{n} \sim \lim_{i\to\infty} x_i$



The word Algorithms comes from
Muhamad ibn Musa al-Khwarizmi (VIII AC)
The Compendious Book on Calculation by Completion and Balancing
(*Liber algebrae et almucabala*)

Do we have algorithms for all problems?

There are UDECIDIBLE problems.

Do we have algorithms for all problems?

There are UDECIDIBLE problems.

Real example from project proposal as final project for CS in a university X:

The purpose of this project is to create a debugger program. This program will take as input the source code another program, and will analyze that other program and determine if it will run to completion, or have an error, or go into an infinite loop.

Do we have algorithms for all problems?

There are UDECIDIBLE problems.

Real example from project proposal as final project for CS in a university X:
The purpose of this project is to create a debugger program. This program will take as input the source code another program, and will analyze that other program and determine if it will run to completion, or have an error, or go into an infinite loop.

Halting problem: Given a computer program, decide if it always halts.

The halting problem is undecidible

Some consequences of the halting problem

Goldbach's conjecture (1742): *Every even integer ≥ 2 can be written as the sum of two primes.*

$6 = 3 + 3$, $14 = 3 + 11$, $60 = 7 + 53$
If the halting problem was decidable then the Goldbach conjecture would be solvable:

```
from $i = 2$ to $\infty$ do
    if $2 * i$ is not the sum of two primes, HALT.
```

Some consequences of the halting problem

Goldbach's conjecture (1742): *Every even integer ≥ 2 can be written as the sum of two primes.*

$6 = 3 + 3$, $14 = 3 + 11$, $60 = 7 + 53$
If the halting problem was decidable then the Goldbach conjecture would be solvable:

```
from $i = 2$ to $\infty$ do
    if $2 * i$ is not the sum of two primes, HALT.
```

The halting problem!
Experimentally, the Golbach conjecture has been verified up to $n \leq 10^{18}$ by T. Olivera (2008)

DECIDIBLE PROBLEMS

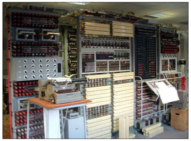
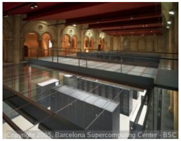
There is an algorithm to solve the problem for any input size.

Efficiency of an algorithm: asymptotic number of steps as function of input size,

TIME: number of steps

SPACE: size of storage

Complexity measures must be independent of existing technology

DECIDIBLE=FEASIBLE?

Find the non-trivial factors of n

```
factor($n, i$)
from $k = 2$ to $\sqrt{n}$ do
    if $k \mid n$ return $k$
       factor($n/k, k$)

factor($n, 1$)
```

DECIDIBLE=FEASIBLE?

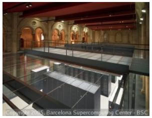
Find the non-trivial factors of n

```
factor($n, i$)
from $k = 2$ to $\sqrt{n}$ do
    if $k \mid n$ return $k$
       factor($n/k, k$)
```

factor$(n, 1)$

Input: $N = \lg n$.
The above algorithm must test $2^{\sqrt{N}}$ and each iteration at a cost N^3.

With current technology, for $N = 2000$ it would take $>$ life span



Feasible Problems

P: *class of problems having an efficient algorithm to find solutions.*

Edmonds (1956), ..., Karp (1972)

efficient: polynomial number of steps, in the worst case.

Is $10^{10^{10^{10}}} n \lg n$ efficient?

Feasible Problems

P: *class of problems having an efficient algorithm to find solutions.*

Edmonds (1956), ..., Karp (1972)

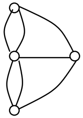
efficient: polynomial number of steps, in the worst case.

Is $10^{10^{10^{10}}} n \lg n$ efficient?
YES (for sufficiently large input size)

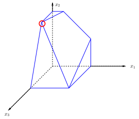
Examples of problems in P

- *Integer multiplication:* $n \times m$. If $N = \max(n, m)$ then $T = N^2$ ($T = N \lg N$ with clever trick)

- *Eulerian tour:* Given $G = (V, E)$ find a tour that traverses all edges exactly once.

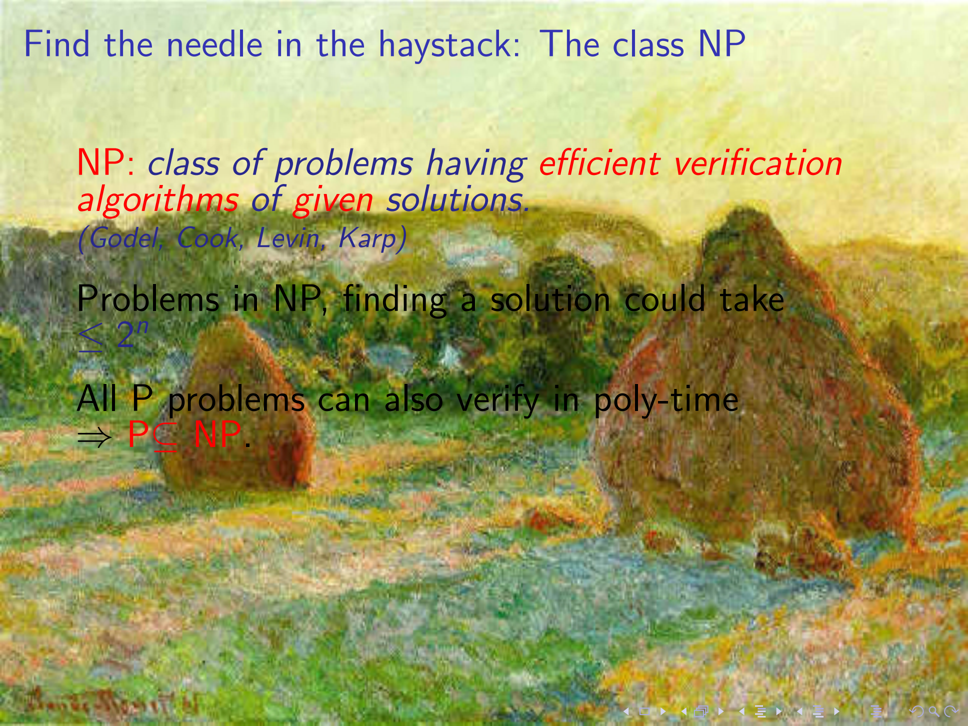


- *Linear Programming:* optimize linear function $f(\vec{x})$, subject to constrains $A\vec{x} \leq \vec{b}$.
Ellipsoid algorithm (Khachiyan), Interior Methods (Karmarkar). Dantzing's Simplex is not always polynomial!



- *Primality test:* Is $2^{57} - 2$ prime? (Agrawal-Kayal-Saxene).

Find the needle in the haystack: The class NP

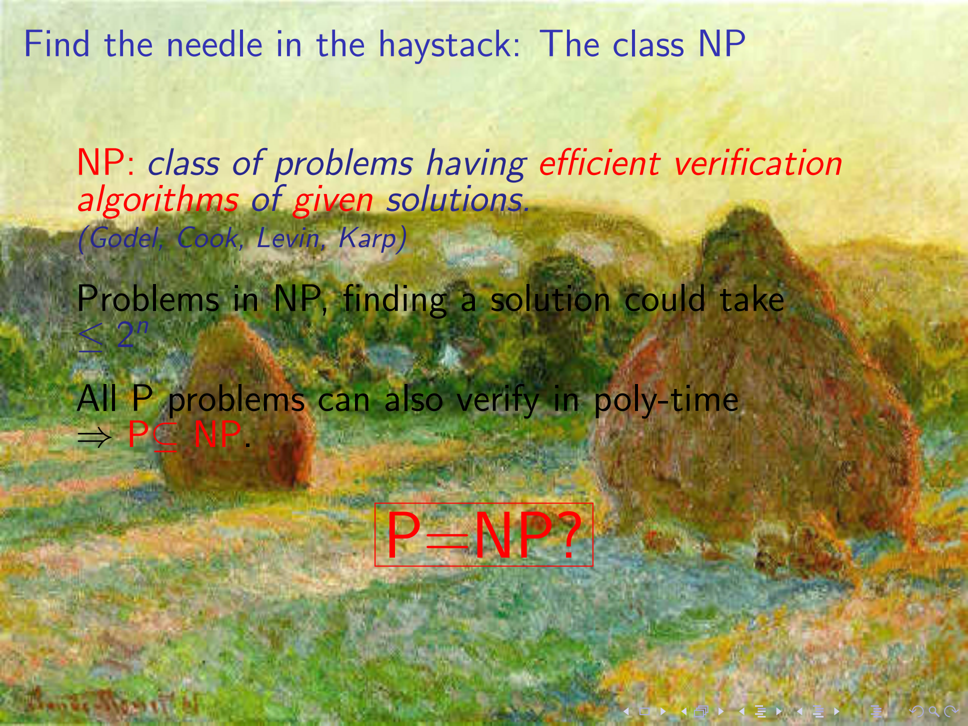


NP: *class of problems having efficient verification algorithms of given solutions.*
(Godel, Cook, Levin, Karp)

Problems in NP, finding a solution could take $\leq 2^n$

All P problems can also verify in poly-time
$\Rightarrow$ P$\subseteq$ NP.

Find the needle in the haystack: The class NP



NP: *class of problems having efficient verification algorithms of given solutions.*

(Godel, Cook, Levin, Karp)

Problems in NP, finding a solution could take $\leq 2^n$

All P problems can also verify in poly-time $\Rightarrow$ P$\subseteq$ NP.

$$\boxed{P{=}NP?}$$

P and NP

P: we can find a solution efficiently

NP: we can *verify* a solution efficiently

Conjecture: Finding is much difficult than verifying.

$$\boxed{\text{P} \neq \text{NP}}$$

NP Problems

- *Factoring:* Find the non-trivial factors of $2^{57} - 2$.

Given (5)(156288505774077994650069306918569)(2017) (939691588217) prove in poly-time that their product $= 2^{57} - 2$.

- *3-SAT:* Given a set of Boolean variables $X = \{x_1, \ldots, x_n\}$ and a Bolean CNF $\phi = C_1 \vee \cdots \vee C_m$ on X, where each C_i is the disjunction of exactly 3 literals on X, is there a truth assignment $A : X \to \{T, F\}$ s.t. $A(\phi) = T$?

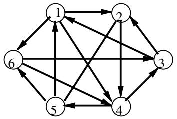
GIven:

$\phi = (x_1 \vee \bar{x_2} \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3})$.

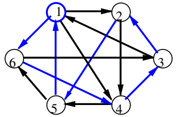
and truth assignment $A(x_1) = A(x_4) = T$ and $A(x_2) = A(x_3) = F$, prove in $3m$ steps that $A(\phi) = T$.

NP Problems

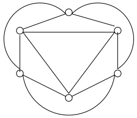
- *Hamiltonian tour:* Given $G = (V, E)$ find a tour that traverses all vertices exactly once. Rudrata (IXc), Hamilton (XIXc)



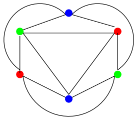
Given the solution prove in n^2 that indeed is a Hamiltonian tour.



- *3-colorability of planar graph:*



Given the solution, prove in n^2 that indeed a legal coloring.

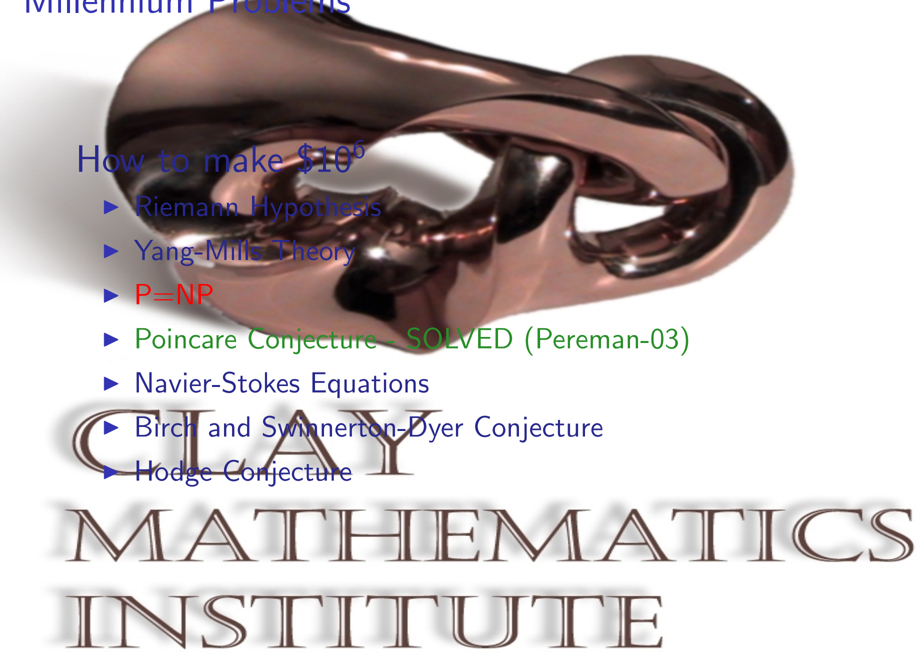


Millennium Problems

How to make $\$10^6$

- Riemann Hypothesis
- Yang-Mills Theory
- P=NP
- Poincare Conjecture
- Navier-Stokes Equations
- Birch and Swinnerton-Dyer Conjecture
- Hodge Conjecture

Millennium Problems



How to make \$$10^6$

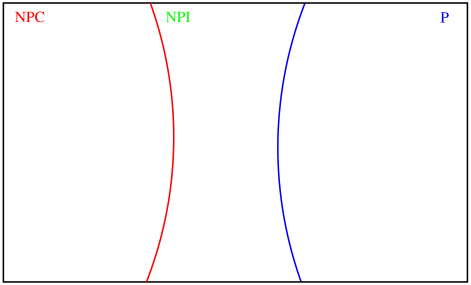
- Riemann Hypothesis
- Yang-Mills Theory
- P=NP
- Poincare Conjecture - SOLVED (Pereman-03)
- Navier-Stokes Equations
- Birch and Swinnerton-Dyer Conjecture
- Hodge Conjecture

NP-Complete Problems

The most difficult subclass of problems in NP

All problems in the class NPC are computationally equivalent in the sense that, if one problem is easy, all the problems in the class are easy.

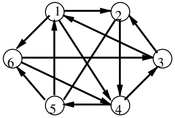
Therefore, if one problem in NPC proves to be in P ⇒ P=NP.



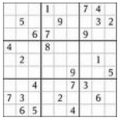
The world of NP

NPC Problems

- *3-SAT:*
  $\phi = (x_1 \vee \bar{x_2} \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3})$.
- *Hamiltonian tour:*

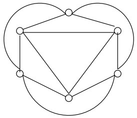


- *Sudoku:*



			1			7	4	
	5			9			3	2
		6	7			9		
4			8					
	2						1	
					9			5
		4			7	3		
7	3			2			6	
	6	5			4			

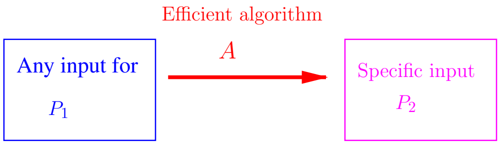
- *3-colorability of map:*



All NPC have equivalent difficulty

To prove that a problem $P_2 \in$ NPC:

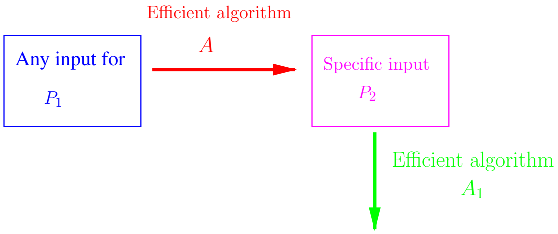
- $P_2 \in$ NP
- For a $P_1 \in$ NPC, P_1 has to be reducible to P_2.



All NPC have equivalent difficulty

To prove that a problem $P_2 \in$ NPC:

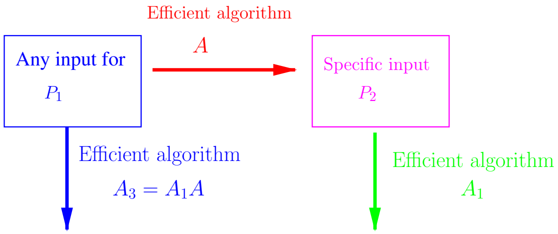
- $P_2 \in$ NP
- For any $P_1 \in$ NPC, P_1 has to be reducible to P_2.



All NPC have equivalent difficulty

To prove that a problem $P_2 \in$ NPC:

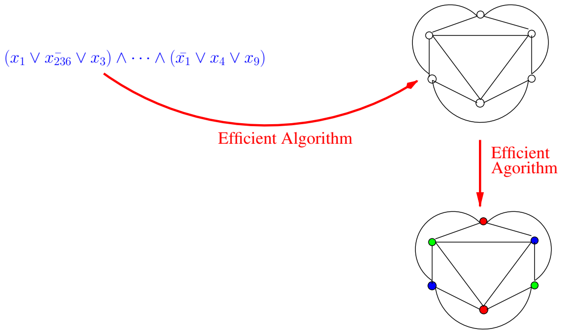
- $P_2 \in$ NP
- For any $P_1 \in$ NPC, P_1 has to be reducible to P_2.



All NPC have equivalent difficulty

An *efficient algorithm* for one of the NPC problems, would yield an efficient algorithm for any NP problem.

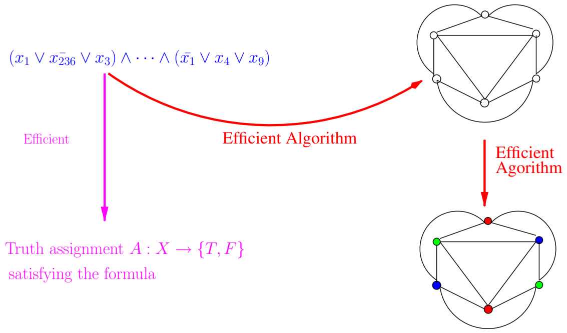
If we had an efficient algorithm for 3-col. of map, it could be use as a solver for any other NP problem. an efficient algorithm


$(x_1 \vee x_{\bar{2}36} \vee x_3) \wedge \cdots \wedge (\bar{x_1} \vee x_4 \vee x_9)$
Efficient Algorithm
Efficient Agorithm

The 3-col. problem as a solver for NPC problems

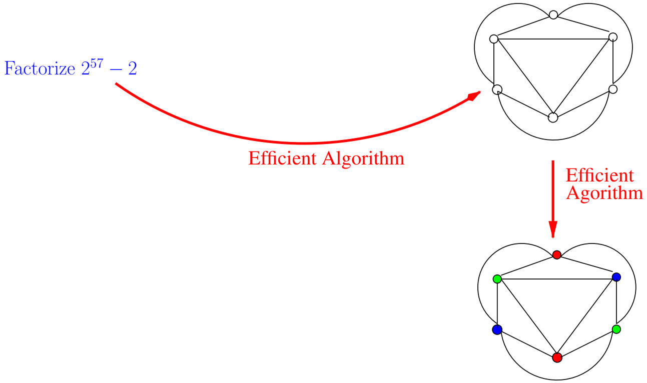
An *efficient algorithm* for one of the NPC problems, would yield an efficient algorithm for any NP problem.

If we had an efficient algorithm for 3-col. of map, it could be use as a solver for any other NP problem. an efficient algorithm


$(x_1 \vee \bar{x}_{236} \vee x_3) \wedge \cdots \wedge (\bar{x}_1 \vee x_4 \vee x_9)$
Efficient
Efficient Algorithm
Efficient Agorithm
Truth assignment $A : X \to \{T, F\}$ satisfying the formula

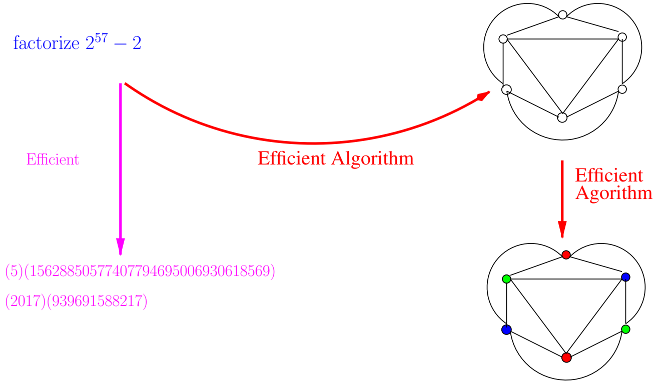
The 3-col. problem as a solver for NPI problems

If there is an *efficient algorithm* for one NPC problem, it would yield an efficient algorithm for factorize (which is NP but is not known to be NPC).

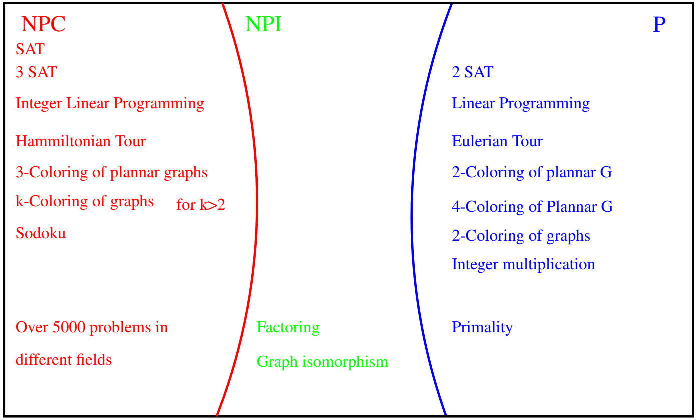


Sovers for NPI

If there is an *efficient algorithm* for one NPC problem, it would yield an efficient algorithm for factorize (which is NP but is not known to be NPC).

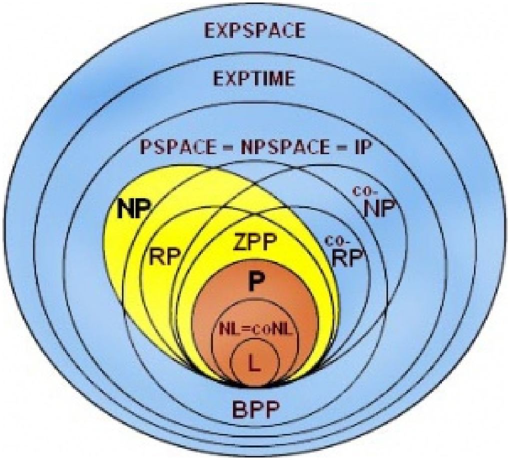


Some Problems in NP (under P≠ NP



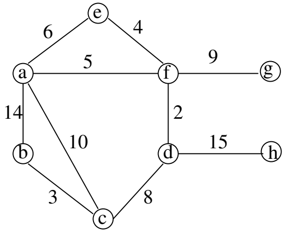
The complexity zoo

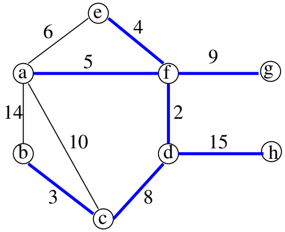
About 500 complexity classes and their relation.



Exercise

Given a graph $G = (V, E)$ a spanning tree is a tree using a subset of E and spanning all the nodes in V. If the graph has weight on the edges, the greedy Jarnik, Prim algorithms computes in poly-time the spanning tree with min sum weight of the edges.





Exercise

Given a graph $G = (V, E)$ (without weights) determine which of the following problems are NP-complete and which are solvable in polynomial time:

- Given a set of nodes $L \subset V$, find a spanning tree s.t. its sets of leaves include L.
- Given a set of nodes $L \subset V$, find a spanning tree s.t. its sets of leaves are exactly L.
- Given a set of nodes $L \subset V$, find a spanning tree s.t. its sets of leaves is included in L.
- Given an integer $k > 0$, find a spanning tree s.t. it has at most k leaves.

Exercise. Solution

Given a graph $G = (V, E)$ (without weights) determine which of the following problems are NP-complete and which are solvable in polynomial time:

- Given a set of nodes $L \subset V$, find a spanning tree s.t. its sets of leaves include L. P
- Given a set of nodes $L \subset V$, find a spanning tree s.t. its sets of leaves are exactly L. NP-complete
- Given a set of nodes $L \subset V$, find a spanning tree s.t. its sets of leaves is included in L. NP-complete
- Given an integer $k > 0$, find a spanning tree s.t. it has at most k leaves. NP-complete.

Proving P vs. NP

At least there are 53 existing proofs of P=NP or P≠NP!
26 "proving" P=NP; 24 "proving" P≠NP; 3 "proving" the problem is undecidible
(see *http://www.win.tue.nl/ gwoegi/P-versus-NP.htm*)

Proving P vs. NP

At least there are 53 existing proofs of P=NP or P≠NP!
26 "proving" P=NP; 24 "proving" P≠NP; 3 "proving" the problem is undecidible
(see *http://www.win.tue.nl/ gwoegi/P-versus-NP.htm*)

• 1987: T. Swart gave linear programming formulations of polynomial size for the Hamiltonian cycle problem. Since LP is P and Hamiltonian cycle is NP-hard ⇒ P=NP.

Proving P vs. NP

At least there are 53 existing proofs of P=NP or P≠NP!
26 "proving" P=NP; 24 "proving" P≠NP; 3 "proving" the problem is undecidible
(see *http://www.win.tue.nl/ gwoegi/P-versus-NP.htm*)

• 1987: T. Swart gave linear programming formulations of polynomial size for the Hamiltonian cycle problem. Since LP is P and Hamiltonian cycle is NP-hard ⇒ P=NP.
(1988) Mihalis Yannakakis proved that expressing the Hamiltonian cycle problem by a symmetric LP requires exponential size. (STOC-88, JCSS-91)

In the year 2009:

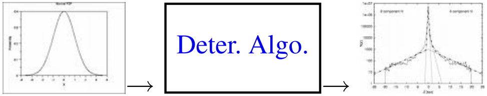
- (March) R. Valls Hidalgo-Gato P=NP.
- (April) Xinwen Jiang P=NP. *http://xinwenjiang.googlepages.com/*
- (June) Arto Annila P$\neq$NP. *http://arxiv.org/abs/0906.1084.*
- (July) Andre L. Barbosa P$\neq$NP. *http://arxiv.org/abs/0907.3965.*
- (Sept.) Yann Dujardin P=NP *http://arxiv.org/abs/0909.3466.*
- (Sept.) Luigi Salemi P=NP *http://arxiv.org/abs/0909.3868.*
- (Dec.) Ari Blinder P$\neq$NP. *http://sites.google.com/site/ariblindercswork/.*

Why are some problems qualitatively harder than others?

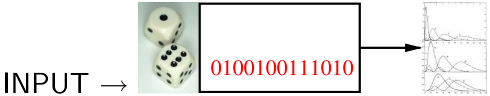
For many problems, there are a few instances that make the problem hard. (That is why Local Search techniques work well). However, the adversary could design problems that are as hard as possible, forcing us to solve them in the worst case.

The power of random choices

Average Analysis
Deterministic
Algo.

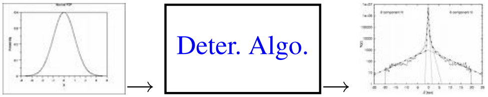


Randomized
Algorithm

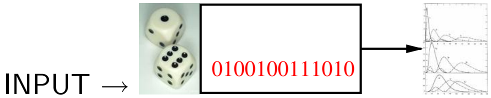


The power of random choices

Average Analysis
Deterministic
Algo.



Randomized
Algorithm



For example, if we choose uniformly at random one of the $n!$ possible inputs to sort a table with n keys, using quicksort, the expected number of steps will be $1.7n \lg n + O(n)$, while it is known the worst case could take $\Theta(n^2)$ steps (ordered input)

An example: primality testing of an integer

Given an integer n, to decide if it is a prime is in P, but the best current algorithm is too slow $O((\lg n)^7)$.

An example: primality testing of an integer

Given an integer n, to decide if it is a prime is in P, but the best current algorithm is too slow $O((\lg n)^7)$.
Recall *Fermat's little theorem:* For any n prime and for all $a \in \mathbb{Z}_n^+$,

$$a^{n-1} \equiv 1 \mod n,$$

where $\mathbb{Z}_N^+ = \{a | a \in \{1, 2, \ldots, n-1\}\}$.

An example: primality testing of an integer

Given an integer n, to decide if it is a prime is in P, but the best current algorithm is too slow $O((\lg n)^7)$.
Recall *Fermat's little theorem:* For any n prime and for all $a \in \mathbb{Z}_n^+$,

$$a^{n-1} \equiv 1 \mod n,$$

where $\mathbb{Z}_N^+ = \{a | a \in \{1, 2, \ldots, n-1\}\}$.

Algorithm to test if a given $n \in \mathbb{N}$ is prime

```
$a :=$ random $(1, n-1)$
**if** $a^{n-1} \equiv 1 \mod n$
  **return** prime
  **else return** not-prime
```

An example: primality testing of an integer

Given an integer n, to decide if it is a prime is in P, but the best current algorithm is too slow $O((\lg n)^7)$.
Recall *Fermat's little theorem:* For any n prime and for all $a \in \mathbb{Z}_n^+$,

$$a^{n-1} \equiv 1 \mod n,$$

where $\mathbb{Z}_N^+ = \{a | a \in \{1, 2, \ldots, n-1\}\}$.

Algorithm to test if a given $n \in \mathbb{N}$ is prime

```
$a :=$ random $(1, n-1)$
**if** $a^{n-1} \equiv 1 \mod n$
  **return** prime
  **else return** not-prime
```

The complexity of the Monte-Carlo algorithm is $O(\lg n)^3$ steps.

Correctness

The following result is a easy consequence of Lagrange's Theorem (if S is a subgroup of the abelian group A, $\Rightarrow |S| \mid |A|$):
If $a^{n-1} \not\equiv 1 \mod n$ for some $a \in \mathbb{Z}_n^*$, then this also happens with at least half of the $a \in \mathbb{Z}_n^*$.
Therefore the probability of mistake is $\leq 1/2$.

Correctness

The following result is a easy consequence of Lagrange's Theorem (if S is a subgroup of the abelian group A, $\Rightarrow |S| \mid |A|$):
If $a^{n-1} \not\equiv 1 \mod n$ for some $a \in \mathbb{Z}_n^*$, then this also happens with at least half of the $a \in \mathbb{Z}_n^*$.
Therefore the probability of mistake is $\leq 1/2$.

If we repeated the algorithm k-times, the probability of mistake is $\leq 1/2^k$, which for instance $k = 4$ could be quite small!.

Unfortunately,

Fermat little theorem IS NOT iff:
$\exists n : a^{n-1} \equiv 1 (\mod n)$ with n NOT a prime.

The Carmichael numbers, which are very rare (255 with value < 100000000) 561, 1105, 1729, $\cdots$ but fool the above primality test.

The previous theorem can be easily adapted to take into consideration the Carmichael numbers, with the same complexity.

For difficult problems randomness does not help:

Theorem (Yao, Impagliazo-Wigderson)
If P$\neq$NP, randomness adds no power to NPC problems.

However, randomness speeds up problems in P:
Primality, sorting, order statistics, min-cut, ...

The difficulty of problems: A paradigmatic example

Recall the *3-Satisfiability Problem (3SAT)* is: given a formula $\phi = C_1 \wedge \cdots \wedge C_m$, where each C_i contais 3 literals, decide if there is an assignment of the boolean variables such that it makes ϕ satisfiable.

$$\phi = (x_1 \vee \bar{x_2} \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3}).$$

The difficulty of problems: A paradigmatic example

Recall the *3-Satisfiability Problem (3SAT)* is: given a formula $\phi = C_1 \wedge \cdots \wedge C_m$, where each C_i contais 3 literals, decide if there is an assignment of the boolean variables such that it makes ϕ satisfiable.

$\phi = (x_1 \vee \bar{x_2} \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3})$.

It is NP-complete.
Does it mean we can forget about solving most of the instances of it?

The Davis-Putnam procedure

Backtracking algorithm (could be exponential time)

Prune the formulae by assigning truth values to literal

Example

$\phi = (x_1 \vee \bar{x_2} \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3})$.

Making $x_1 = 1$ yields $(\bar{x_2} \vee \bar{x_4}) \wedge (\bar{x_2} \vee \bar{x_4})$
Now make $x_4 = 0$ and we get a sat truth assignment $A(x_1) = 1$ and $A(x_4) = 0$

The Davis-Putnam : Example 2

$\phi =$
$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x_2}) \wedge (x_2 \vee \bar{x_3}) \wedge (x_2 \vee \bar{x_3}) \wedge (x_3 \vee \bar{x_1}) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$.

$A(x_1) = 1 \Rightarrow (x_2 \vee \bar{x_3}) \wedge (x_3) \wedge (\bar{x_2} \vee \bar{x_3})$.
$A(x_3) = 1 \Rightarrow (x_2) \wedge (\bar{x_2})$ Backtrack

The Davis-Putnam : Example 2

$\phi =$
$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x_2}) \wedge (x_2 \vee \bar{x_3}) \wedge (x_2 \vee \bar{x_3}) \wedge (x_3 \vee \bar{x_1}) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$.

$A(x_1) = 1 \Rightarrow (x_2 \vee \bar{x_3}) \wedge (x_3) \wedge (\bar{x_2} \vee \bar{x_3})$.
$A(x_3) = 1 \Rightarrow (x_2) \wedge (\bar{x_2})$ Backtrack

$A(x_1) = 0 \Rightarrow (x_2 \vee x_3) \wedge (\bar{x_2}) \wedge (x_2 \vee \bar{x_3})$.
$A(x_2) = 0 \Rightarrow (x_3) \wedge (\bar{x_3})$ So ϕ does not have a sat assignment!

The Davis-Putnam : Example 2

$\phi =$
$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x_2}) \wedge (x_2 \vee \bar{x_3}) \wedge (x_2 \vee \bar{x_3}) \wedge (x_3 \vee \bar{x_1}) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$.

$A(x_1) = 1 \Rightarrow (x_2 \vee \bar{x_3}) \wedge (x_3) \wedge (\bar{x_2} \vee \bar{x_3})$.
$A(x_3) = 1 \Rightarrow (x_2) \wedge (\bar{x_2})$ Backtrack

$A(x_1) = 0 \Rightarrow (x_2 \vee x_3) \wedge (\bar{x_2}) \wedge (x_2 \vee \bar{x_3})$.
$A(x_2) = 0 \Rightarrow (x_3) \wedge (\bar{x_3})$ So ϕ does not have a sat assignment!

For many SAT inputs, (variations of) Davis-Putnam yield a solution quite fast.

The Davis-Putnam : Example 2

$\phi =$
$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x_2}) \wedge (x_2 \vee \bar{x_3}) \wedge (x_2 \vee \bar{x_3}) \wedge (x_3 \vee \bar{x_1}) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$.

$A(x_1) = 1 \Rightarrow (x_2 \vee \bar{x_3}) \wedge (x_3) \wedge (\bar{x_2} \vee \bar{x_3})$.
$A(x_3) = 1 \Rightarrow (x_2) \wedge (\bar{x_2})$ Backtrack

$A(x_1) = 0 \Rightarrow (x_2 \vee x_3) \wedge (\bar{x_2}) \wedge (x_2 \vee \bar{x_3})$.
$A(x_2) = 0 \Rightarrow (x_3) \wedge (\bar{x_3})$ So ϕ does not have a sat assignment!

For many SAT inputs, (variations of) Davis-Putnam yield a solution quite fast.

Goldberg, Purdom, Brown, 1982 DP can solved a random instance of SAT with length m in $O(m^2)$.

Random Inputs for 3-SAT

Given n variables, the set of possible clauses is $2^3\binom{n}{3}$.

To generate a random instance for 3-SAT on n variables and with $m = rn$ clauses:

• Choose uar $m = rn$ clauses, each with probability $\frac{1}{\binom{n}{3}}$

• Go over the variables in the selected m clauses, negate each variable with probability $= 1/2$.

$\mathcal{F}_{n,m}$ = set of formulas generated in this way.

Density of a random formulae

Density of $\phi \in \mathcal{F}_{n,m}$ is defined by $r = \frac{n}{m}$.

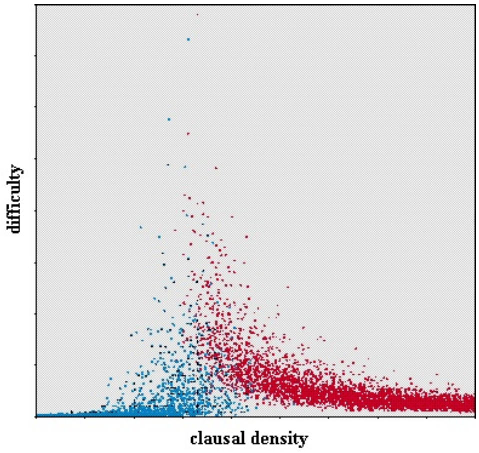
Examples:

$\phi_1 = (x_1 \vee \bar{x_2} \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3})$.
has $r = 1$

$\phi_2 =$
$(x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \bar{x_2} \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3}) \wedge$
$(\bar{x_2} \vee x_3 \vee x_4) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3}) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee \bar{x_3} \vee \bar{x_4}) \wedge (\bar{x_2} \vee \bar{x_3} \vee \bar{x_4})$
has $r = 0.4$

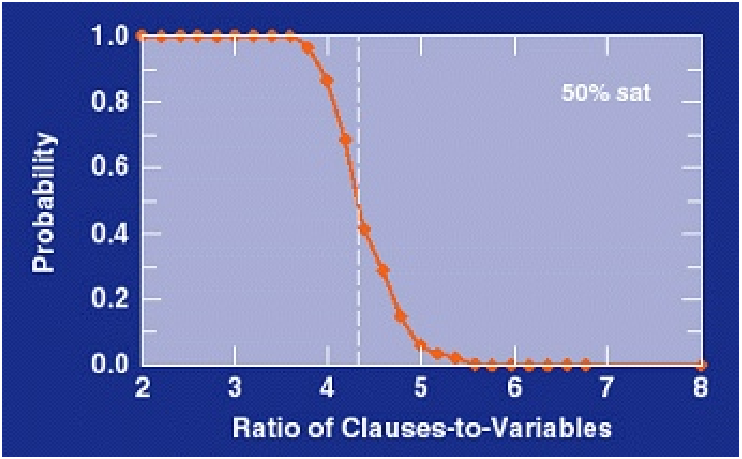
Phase transition 3-SAT

Experimentally: Mitchell, Selman, Levesque (1991) for 3-SAT
For the Davis-Putnam backtrack algorithm:



Phase transition 3-SAT

For $r < 4.2$ whp formulae are SAT
For $r > 4.2$ whp formulae are SAT



Phase transition 3SAT: The physics approach

Using techniques from statistical physics on very large instances of 3SAT, physics people where able to give **theoretical non-rigorous evidence** that the threshold for 3SAT occurs at

$$r_c = 4.27$$

Mézard, Parisi, Zecchina (2002), Mézard, Zecchina (2002), ...

Rigorous approach

Consider a random 3SAT formula $\phi \in \mathcal{F}_{n,rn}$

Upper bound: $r > r_c = 4.27$ Get a value as low as possible of r (≥ 4.27) such that whp ϕ is not SAT. (Variations on the first moment)

Lower bound: $r < r_c = 4.27$ Consider an easy to analyze algorithm. Get a value as large as possible of r ($r \leq 4.27$) such that whp the algorithm produces a satisfying assignment for ϕ.

Status on Formal bounds to 3-SAT. Upper-bound:

$r = 5.1909$ (1983) Franco, Paull (and others)
$r = 5.19 - 10^{-7}$ (1992) Frieze and Suen
$r = 4.758$ (1994) Kamath, Motwani, Palem, Spirakis
$r = 4.667$ (1996) Kirousis, Kranakis, Krizanc.
$r = 4.642$ (1996) Dubois, Boufkhad
$r = 4.602$ (1998) Kirousis, Kranakis, Krizac, Stamatiou
$r = 4.596$ (1999) Janson, Stamatiou, Vamvakari (1999)
$r = 4.571$ (2007) Kaporis, Kirousis, Stamatiou, Vamvakari
$r = 4.506$ (1999) Dubois, Boukhand, Mandler
$r = 4.4907$ (2008) Díaz, Kirousis, Mitsche, Pérez
$r_c = 4.27$ Experimental threshold (Replica Method)

Status on Formal bounds to 3-SAT. Lower bound:

$r_c = 4.27$Experimental threshold (Replica Method)
$r > 3.52$ Kaporis, Kirousis, Lalas (2003)
$r > 3.52$ Hajiaghayi-Sorkin (2003)
$r > 3.42$ Kaporis, Kirousis, Lalas (2002).
$r > 3.26$ Achlioptas and Sorkin (2001).
$r > 3.145$, Achlioptas (2000).
$r > 3.003$, Frieze, Suen (1992).
$r > 2.99$ Chao, Franco (1986).
$r > 2.66$ Chao, Franco (1986).

First moment: Basic technique for upper bounds

Let ϕ be a random formula and $S(\phi)$ the set of its satisfying truth assignments. Using Markov inequality

$\mathbf{Pr}_{m*}[\phi \text{ is sat}] = \mathbf{Pr}[|S(\phi)| \geq 1] \leq \mathbf{E}[|S(\phi)|]$.

Must compute $\mathbf{E}[|S(\phi)|]$

Notice that given a truth assignment A and 3 variables x_i, x_j, x_k then *there is only one clause on x_i, x_j, x_k which is not SAT by A*. Therefore, out of the $8\binom{n}{3}$ clauses only $\binom{n}{3}$ evaluate to 0 under any given A.

$\mathbf{E}[|S(\phi)|] = \sum_{A \in S(A)} \mathbf{Pr}[A \vDash \phi] = \frac{|\{<A,\phi> \mid A \vDash \phi\}|}{|\{\phi\}|}$

$\mathbf{E}[|S(\phi)|] = (2(7/8)^r)^n$ to make it < 1 we need

$$r \geq 5.1909$$

5.12 is far above the experimental 4.27, because there could be a few formulas with many sat. truth assignment which contribute too much to $\mathbf{E}[|S(\phi)|]$.

General methods for lower bounds to 3SAT threshold

Given a random ϕ in $\mathcal{F}_{n,m}$, $m = rn$ consider an *easy to analyze* heuristic, to find a $A \vDash \phi$,

Let r_l denote the lower bound for the density that we try to compute. Prove that for all $r < r_l$, the heuristic succeeds whp.

General methods for lower bounds to 3SAT threshold

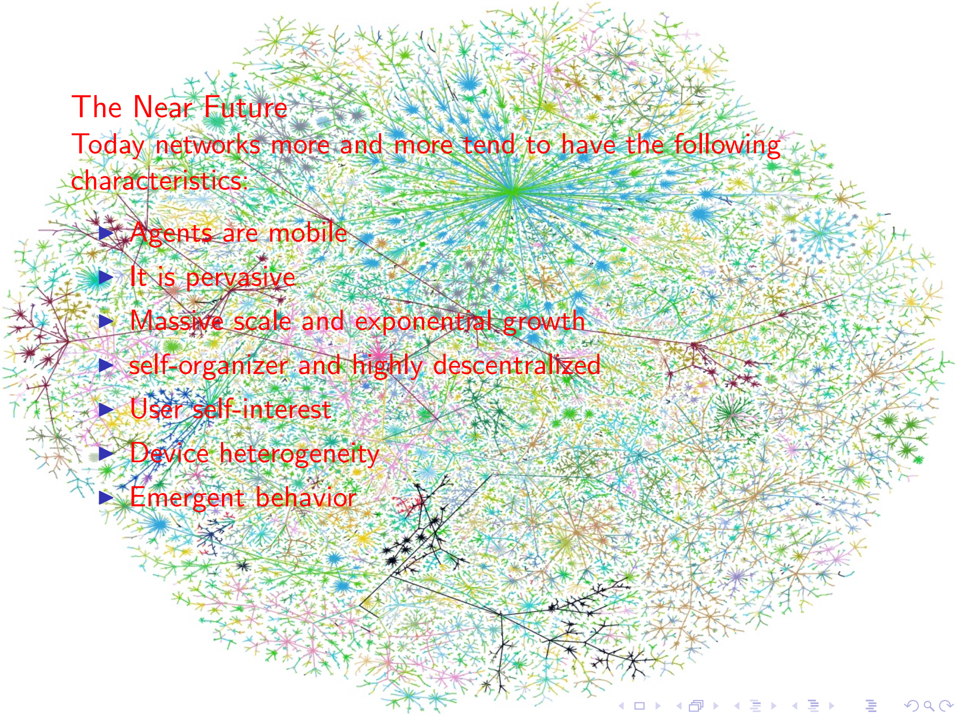
Given a random ϕ in $\mathcal{F}_{n,m}$, $m = rn$ consider an *easy to analyze* heuristic, to find a $A \vDash \phi$,

Let r_l denote the lower bound for the density that we try to compute. Prove that for all $r < r_l$, the heuristic succeeds whp.

The Unit Clause algorithm Chao, Franco (1986).

```
**UC** $\phi$
**if** there is a 1- clause **then**
  **select** u.a.r. one 1-clause and satisfy it (forced step)
**else**  select u.a.r $x_i$ and assign u.a.r. 0 or 1 (free step)
```

Chao and Franco got $r_l = 2.66$.



The Near Future

Today networks more and more tend to have the following characteristics:

- Agents are mobile
- It is pervasive
- Massive scale and exponential growth
- self-organizer and highly descentralized
- User self-interest
- Device heterogeneity
- Emergent behavior

The Near Future

Today networks more and more tend to have the following characteristics:

- Agents are mobile
- It is pervasive
- Massive scale and exponential growth
- self-organizer and highly descentralized
- User self-interest
- Device heterogeneity
- Emergent behavior

Old Example: Internet and the WWW

Today large-scale networks present an *emergent* behavior, which makes it very difficult to predict future behavior.

The emergent system is much more complex and powerful that the components. But it depends of single behaviour of individuals.

The sharing of utilities creates serious problems of security and privacy

The massive growth of social networks are changing social behaviour.

Need new paradigms for complexity and algorithms

Some hot tasks:

T1 Develop probabilistic topological models for MANETS (Mobile Ad-hoc NETworkS) and other social networks, and establish mechanisms for proving that the model is consistent with observed data or that a model fits better than other.

The goal of a network modeling and analysis is the ability to understand network behavior so we can make more informed decisions at future junctures. How better to search the network and get the maximum performance of it.

Some hot tasks:

T1 Develop probabilistic topological models for MANETS (Mobile Ad-hoc NETworkS) and other social networks, and establish mechanisms for proving that the model is consistent with observed data or that a model fits better than other.

The goal of a network modeling and analysis is the ability to understand network behavior so we can make more informed decisions at future junctures. How better to search the network and get the maximum performance of it.

T2 How topology can be exploited in wireless networks? How to deal with high level of failures? What are the alternatives to classical routing? (Routing with mobility should be done on the fly.)

Search in networks

Today most of the search engines are keyword-based, which limits ability of finding non textual data, and semantic search. Algorithmic community has played a key role in the development of the actual search engines:

- HITS algorithm [Kleinberg 1999],
- PageRank algorithm [Page, Brin, et al. 1999]

Search in networks

Today most of the search engines are keyword-based, which limits ability of finding non textual data, and semantic search.
Algorithmic community has played a key role in the development of the actual search engines:

- HITS algorithm [Kleinberg 1999],
- PageRank algorithm [Page, Brin, et al. 1999]

T3 Use the global characteristics of new networks to design efficient searching, mining and retrieval of information

Agent's incentive.

In a network of agents, when agents cooperate to compute, they tend to behave *differently*.

In wireless and ad-hoc networks, bandwidth is controlled by individual nodes. Network performance suffers dramatically if one (key) node fails.

In the last years, quite a few progress has been made in the design of incentive-compatible protocols for some internet problems P2P file distribution, internet based auctions, etc.)
But with pervasive networks, new ideas are needed:
T4 Can one agent determine whether another agent is responding to incentives?

Agent's incentive.

In a network of agents, when agents cooperate to compute, they tend to behave *differently*.

In wireless and ad-hoc networks, bandwidth is controlled by individual nodes. Network performance suffers dramatically if one (key) node fails.

In the last years, quite a few progress has been made in the design of incentive-compatible protocols for some internet problems P2P file distribution, internet based auctions, etc.)
But with pervasive networks, new ideas are needed:
T4 Can one agent determine whether another agent is responding to incentives?
T5 Develop new concepts of rationality and stability, beyond the known ones (Nash equilibria, etc.) sufficient for the analysis of the forthcoming networks?

Algorithmic Mechanism Design.

In the future, internet-based payment mechanisms will probably be the most used way of economical transaction.
In Economics, *mechanism design* deals with how to incentivize individual selfish agents so that their actions result in a globally desirable outcome.
Nisan and Ronen [2001] created the new field of *algorithmic mechanism design*, by considering the computational efficiency in the economical concept. The following years saw an array of interesting algorithmic results dealing with problems in internet as auctions (e-Bay) and min-cost routing.

T6 Design, analyze and deploy distributed algorithmic mechanisms for networked-computational problems like internet auctions, P2P file management, multicast cost sharing. Develop new notions of complexity and equilibrium for distributed mechanisms as they are needed in the new computing-network setting.

Set a new computational theory for the new systems

T7 Give the definition(s) that a computational system must satisfy to be a network:

- Should there be several models or a generic one with few parameters?
- How are the networks created? How do they evolve?
- Need a new notion of *efficiency*, taking into consideration new parameters, communication, bandwidth used, etc.
- Which critical resources should be considered?
- Which bounds on these resources must be satisfied for the computation to be considered efficient.

T8 Formulate a new *complexity theory* for computing on networks:

- The concept of equivalently powerful networks, according to the computational tasks that they can and cannot do. This may need the concept of reduction among networked-computational problems.
- Is there a universal network?

T8 Formulate a new *complexity theory* for computing on networks:

- The concept of equivalently powerful networks, according to the computational tasks that they can and cannot do. This may need the concept of reduction among networked-computational problems.
- Is there a universal network?

The new complexity theory of networking will need new radical ideas, starting from the concept of *efficiency*.

The general theory of networking to be developed, should include the pervasive network models for biological and social systems.

In future networks, *mechanism design* could play an important role to understanding the behavior of the highly decentralized, networks, full of selfish users.

In future networks, *mechanism design* could play an important role to understanding the behavior of the highly decentralized, networks, full of selfish users.
T9 Develop methods for the identification of market structures in which particular network protocols work well, in spite of the existence of conditions in which these protocols are know to work poorly.

In future networks, *mechanism design* could play an important role to understanding the behavior of the highly decentralized, networks, full of selfish users.

T9 Develop methods for the identification of market structures in which particular network protocols work well, in spite of the existence of conditions in which these protocols are know to work poorly.

T10 What types of coalitions should network designer of pervasive networks be concerned about? What type of coalitions would be a natural part of pervasive network development?

References: complexity

C. Papadimitriou: *Computational Complexity* Addison-Wesley, 1994

A. Doxiadis, C. Papadimitriou, A. Papadatos, A. Donna: *LOGICOMIX*. Bloomsbury. 2009.
Lance Fortnow: *The status of the P vs NP Problem* CACM, Nov. 2009, 78-86.

Richard Lipton: *http://rjlipton.wordpress.com/*

Lance Fortnow: *http://weblog.fortnow.com/*

Luca Trevisan: *http://lucatrevisan.wordpress.com/*

References: algorithms

Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani: *Algorithms*, McGraw Hill, 2007.

Jon Kleinberg, Eva Tardos: *Algorithm Design*, 2006

T. Cormen, C. Leiserson, R. Rivest, and C. Stein,*Introduction to Algorithms, 3rd Ed.* MIT Press, 2009.

Steven Skiena: *The Algorithm Design Manual. 2nd Ed* Springer 2008.

Michael Mitzenmacher, Eli Upfal: *Probobility and Computing*. CUP, 2005.

Michael Mitzenmacher: *http://mybiasedcoin.blogspot.com/*
Daniel Lemire: *http://www.daniel-lemire.com/blog/*