Complexity of Computation

Stephan Mertens

Santa Fe Institute

Computational Complexity

No human investigation can be called real science if it can not be demonstrated mathematically.

Leonardo da Vinci (1452–1519)

Computational complexity analyses **intrinsic limits** on what **mathematical problems can be solved**, pretty much like **thermodynamics** analyses intrinsic limits on what **heat engines can do**.

Grand Unified Theory of Computation

D. Hilbert (1862-1943)

Entscheidungsproblem (1928)
Is there an algorithmic procedure which can, in principle, solve all mathematical problems?

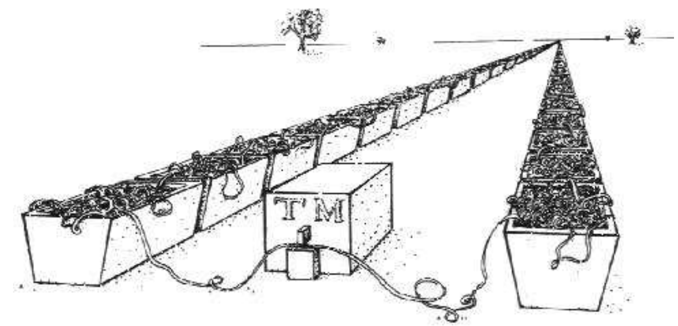
What is an **algorithmic procedure**?

A. Church (1903–1995)

Different answers (1934–1937):

- recursive functions
- λ-calculus
- Turing-machine

all equivalent!



© Roger Penrose, *The Emperor's new mind*

A.Turing (1912–1954)

Church-Turing Hypothesis
Any function that can be computed, can be computed by a Turing machine.

Or (equivalently) by a **program** in C, FORTRAN, . . .

Grand Unified Theory of Computation

Halting Problem
Can we decide whether a program P halts on input i by **inspection** rather than **running** $P(i)$?

Is there a program `halt`(P, i) such that

$$\texttt{halt}(P,i) = \begin{cases} \text{true} & \text{if } P(i) \text{ halts} \\ \text{false} & \text{otherwise} \end{cases}$$

Halting Problem = Entscheidungsproblem

Grand Unified Theory of Computation

Halting Problem
Can we decide whether a program P halts on input i by **inspection** rather than **running** $P(i)$?

Is there a program `halt`(P, i) such that

$$\texttt{halt}(P,i) = \begin{cases} \text{true} & \text{if } P(i) \text{ halts} \\ \text{false} & \text{otherwise} \end{cases}$$

Halting Problem = Entscheidungsproblem

B. Riemann (1826–1866)

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

Riemann Hypothesis
All nontrivial zeros of $\zeta(s)$ are of the form $s = 1/2 + it$, t real.

Grand Unified Theory of Computation

Halting Problem
Can we decide whether a program P halts on input i by **inspection** rather than **running** $P(i)$?

Is there a program `halt`(P,i) such that

$$\texttt{halt}(P,i) = \begin{cases} \text{true} & \text{if } P(i) \text{ halts} \\ \text{false} & \text{otherwise} \end{cases}$$

Halting Problem = Entscheidungsproblem

B. Riemann (1826–1866)

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

Riemann Hypothesis
All nontrivial zeros of $\zeta(s)$ are of the form $s = 1/2 + it$, t real.

```
Riemann(r)
  do
      z := NextZetaZero()
  while (Re(z) ≠ r)
  return z
```

Riemann Hypothesis
`halt(Riemann,1/2)=false`

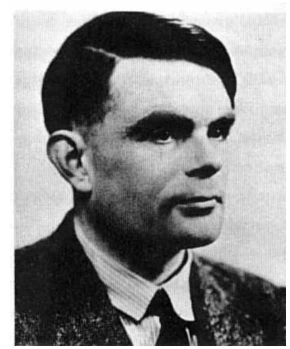
Grand Unified Theory of Computation

Halting Problem
Can we decide whether a program P halts on input i by **inspection** rather than **running** $P(i)$?

Is there a program `halt`(P, i) such that

$$\texttt{halt}(P, i) = \begin{cases} \text{true} & \text{if } P(i) \text{ halts} \\ \text{false} & \text{otherwise} \end{cases}$$

Halting Problem = Entscheidungsproblem



A.Turing (1912–1954)

G. Cantor (1845–1918)

Suppose, `halt` exists. Define

```
function trouble(string s)
  if halt(s, s)
   loop forever
  else
   return true
```

`trouble(trouble)` = ?

Grand Unified Theory of Computation

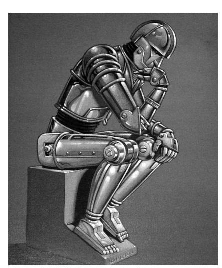
Halting Problem
Can we decide whether a program P halts on input i by **inspection** rather than **running** $P(i)$?

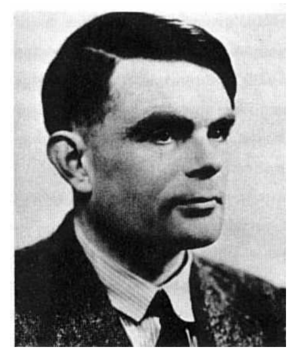
Is there a program `halt`(P, i) such that

$$\texttt{halt}(P,i) = \begin{cases} \text{true} & \text{if } P(i) \text{ halts} \\ \text{false} & \text{otherwise} \end{cases}$$

Halting Problem = Entscheidungsproblem

$\neq$





A.Turing (1912–1954)

G. Cantor (1845–1918)

Suppose, `halt` exists. Define

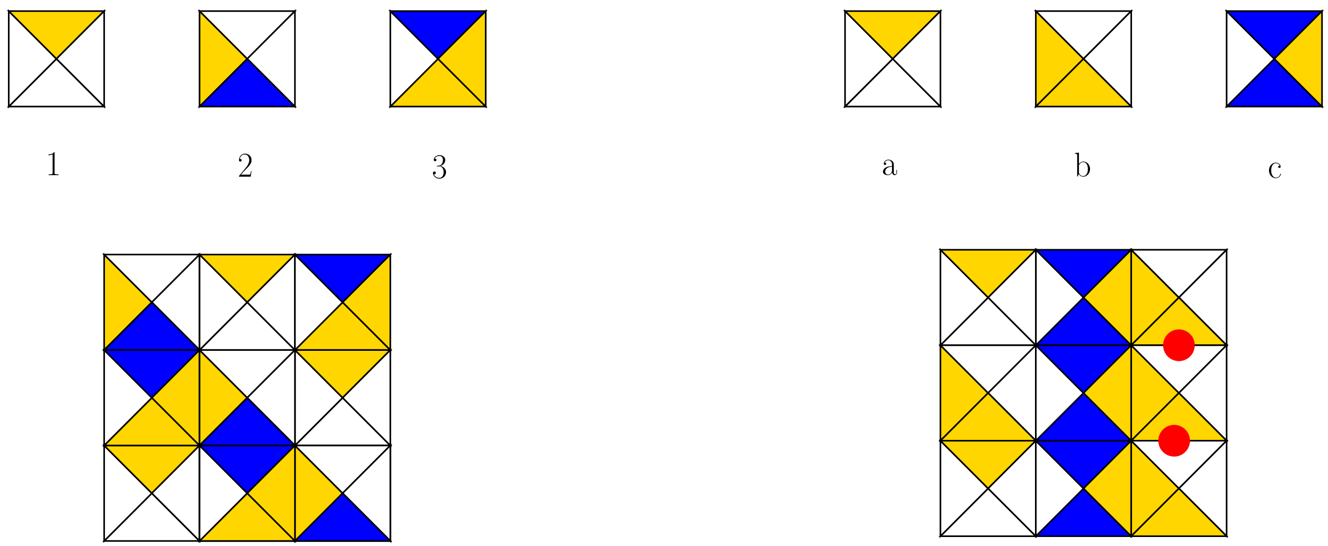
```
function trouble(string s)
  if halt(s, s)
   loop forever
  else
   return true
```

`trouble(trouble)` = ?

`halt` does not exist.

Grand Unified Theory of Computation

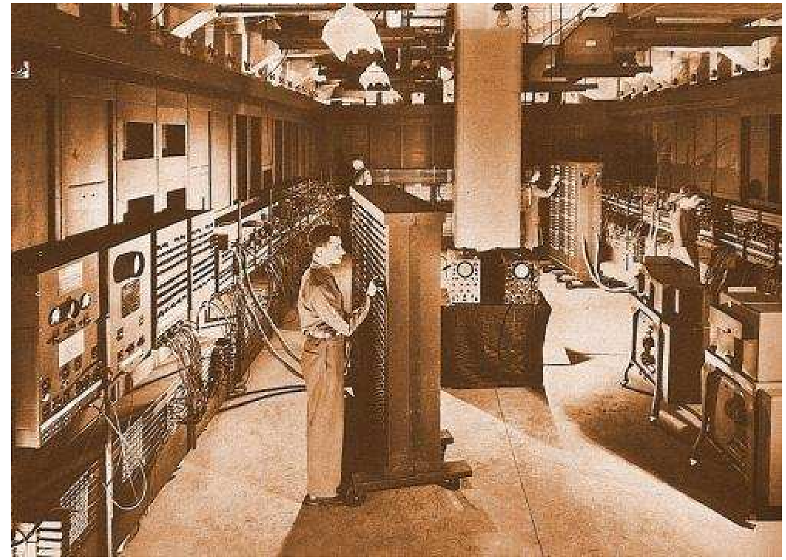
Wang-Tilings: Given a finite set of colored, quadratic tiles. Can we tile the plane with copies from this set so that abutting edges of adjacent tiles have the same color?



This problem is **undecidable**.

Computing on Industrial Scale

“Computers” in the observatory of Hamburg (1920s)



ENIAC (1946), 300 mult. per sec !

Computable ?

Efficiently Computable ?

complexity(problem) = **amount** of **resources** consumed by **solution**

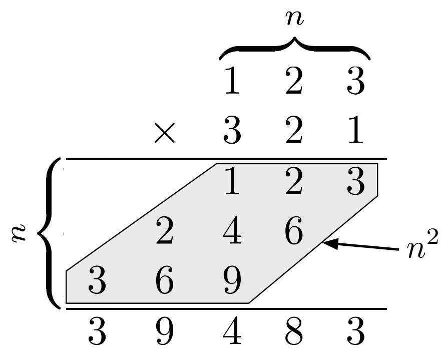
elementary operations
asymptotic scaling
worst case bound

time
space
energy

best algorithm

Multiplication vs. Factoring

$$2^{67} - 1 = 147\,573\,952\,589\,676\,412\,927 = 193\,707\,721 \cdot 761\,838\,257\,287$$



Multiplication:

grade school method: $\mathcal{O}(n^2)$

best known algorithm (FFT): $\mathcal{O}(n \log n \log\log n)$

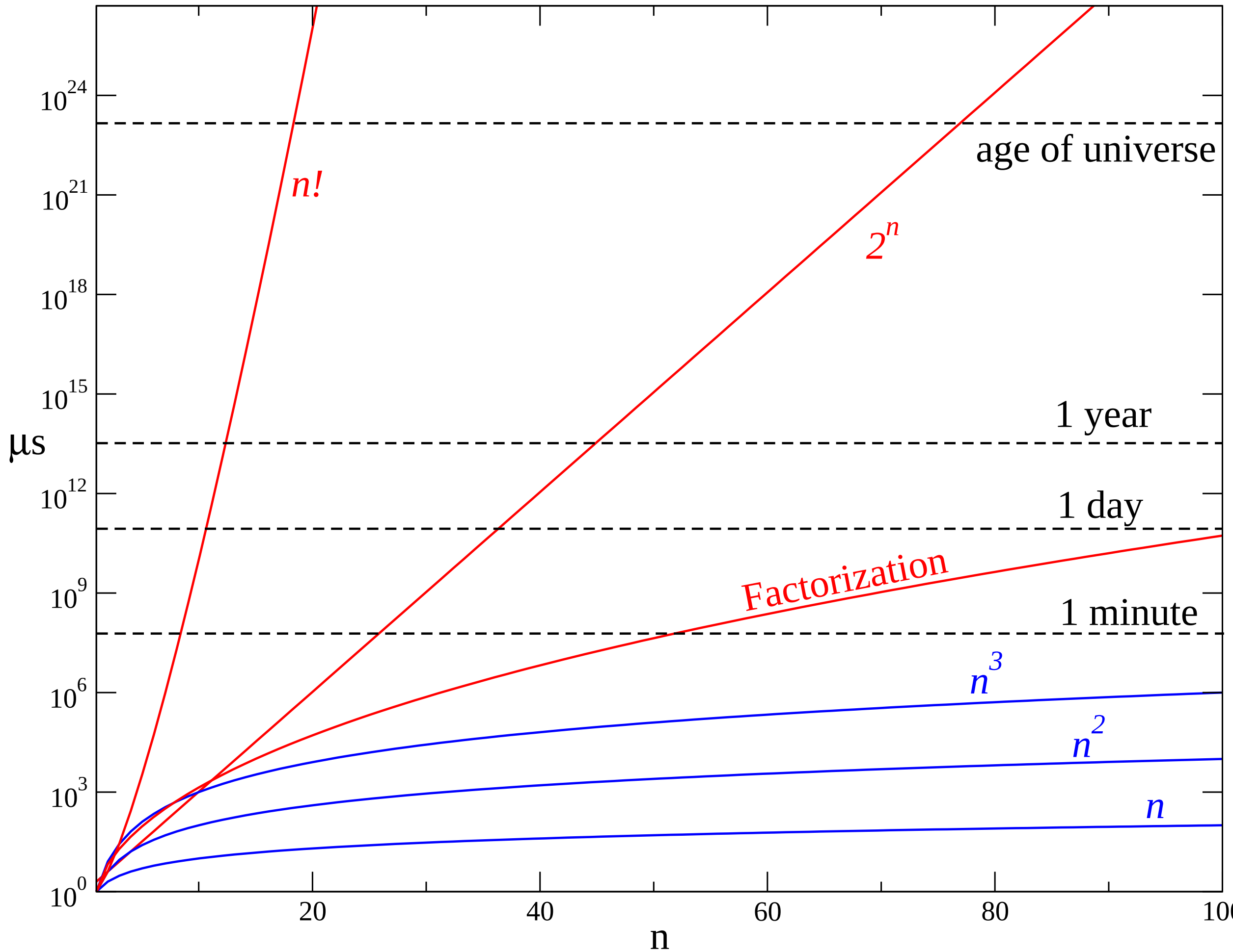
276 BC–194 BC

Factorization:

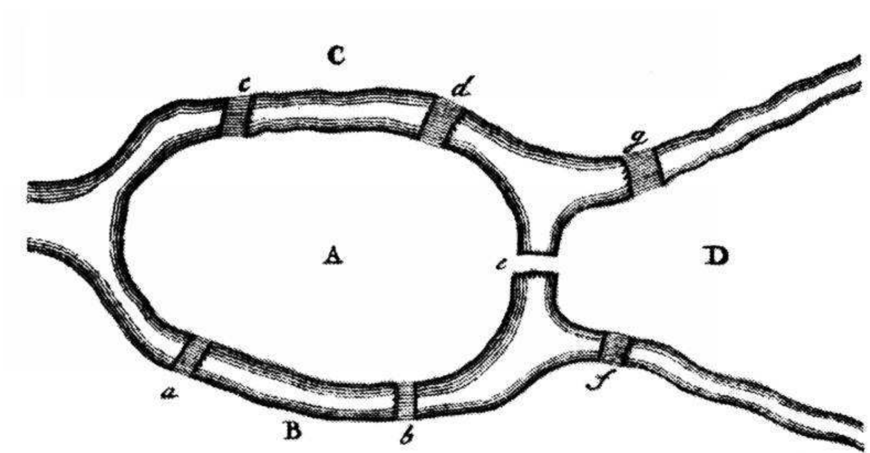
naive (trial division): $\mathcal{O}(n^2 \cdot 2^{n/2})$

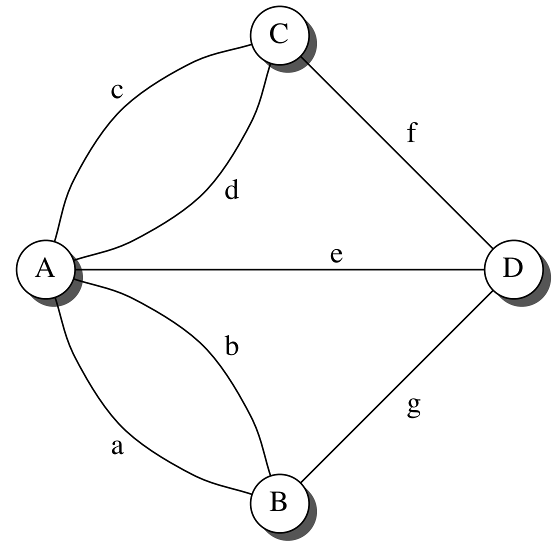
best known algorithm (GNFS): $\mathcal{O}\left(\exp\left((\frac{64}{9}n)^{\frac{1}{3}} (\log n)^{\frac{2}{3}}\right)\right)$

Tractable and Intractable Scalings



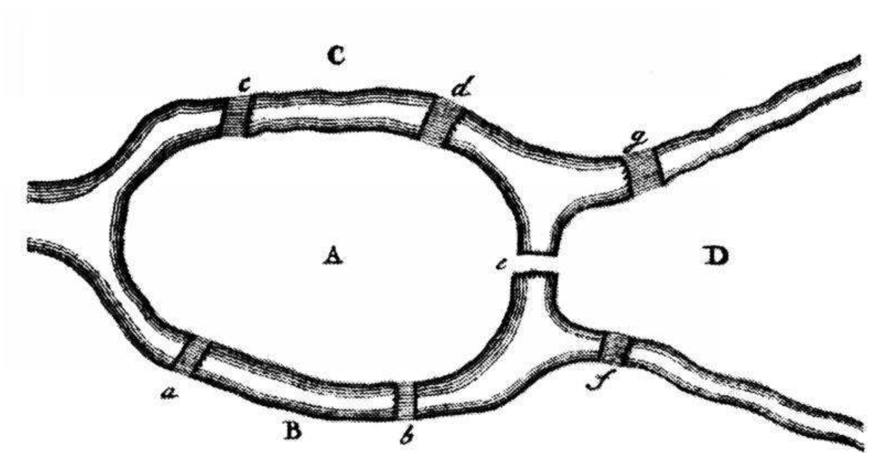
Königsberg Bridges

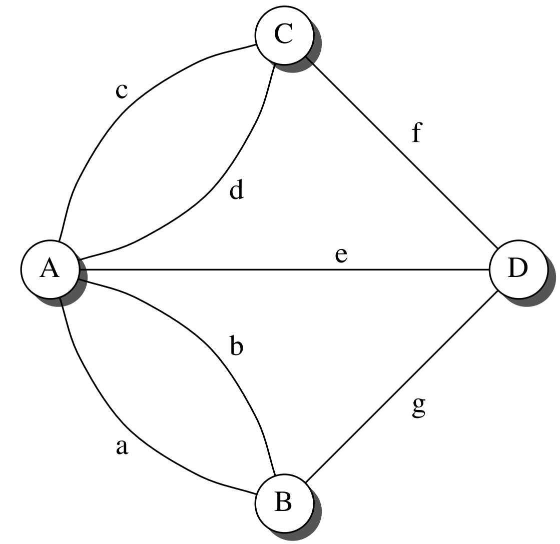




Leonhard Euler (1703–1783)

Königsberg Bridges

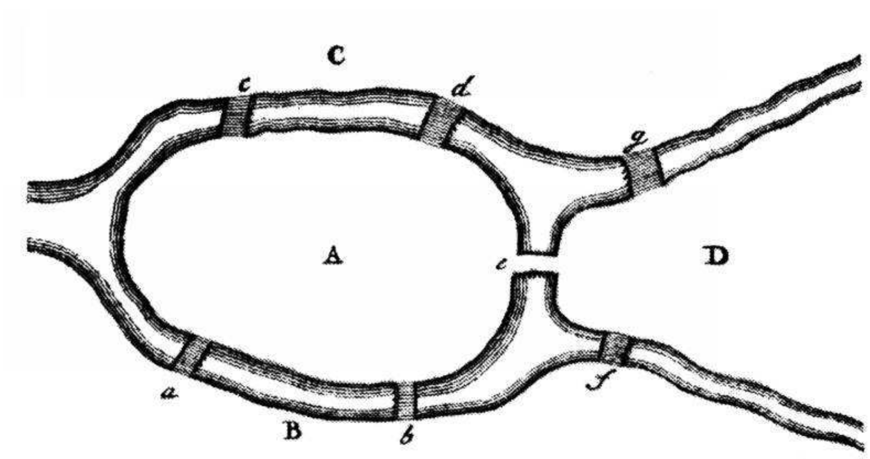


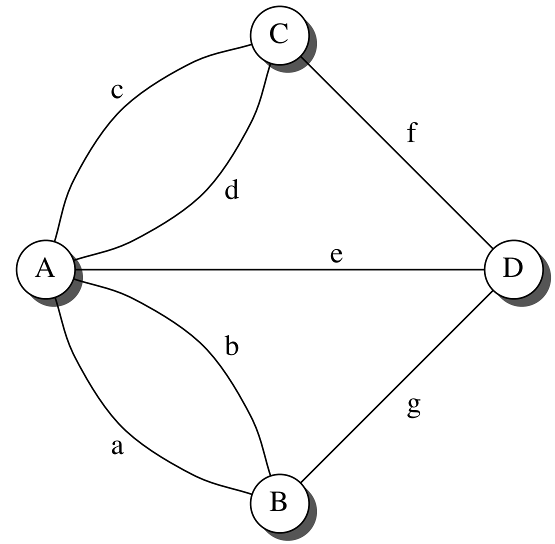


Leonhard Euler (1703–1783)

"As far as the problem of the seven bridges of Königsberg is concerned, it can be solved by making an ***exhaustive list*** *of possible routes, and then finding whether or not any route satisfies the conditions of the problem. Because of the number of possibilities, this method of solutions would be too difficult and laborious, and in other* ***problems with more bridges****, it would be* ***impossible****".*

Königsberg Bridges



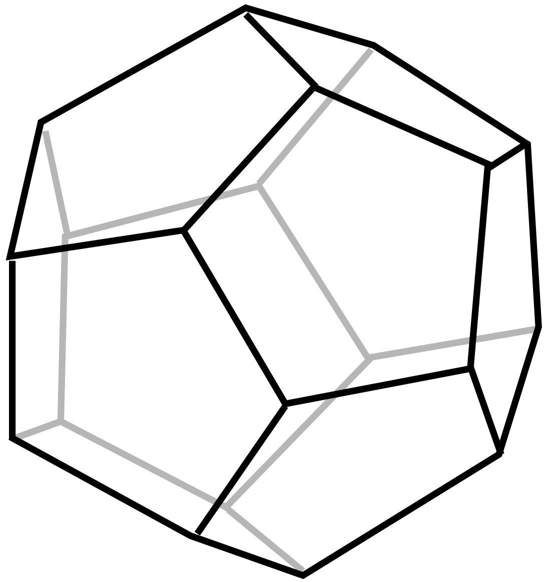


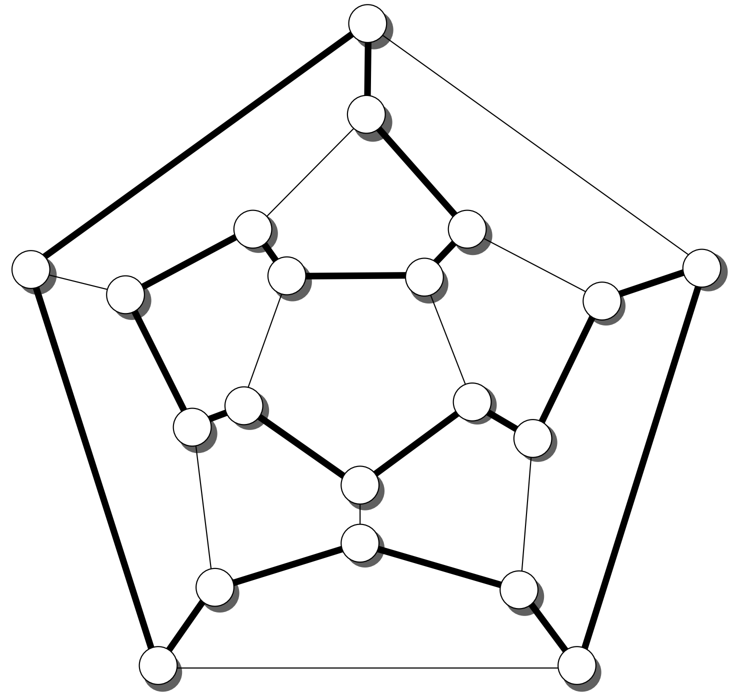
Leonhard Euler (1703–1783)

A cycle that traverses **each edge** of a graph excatly once is called an **Eulerian cycle**.

A connected graph G has an Eulerian cycle if and only if the degree of all vertices is even.

Intractable Itineraries



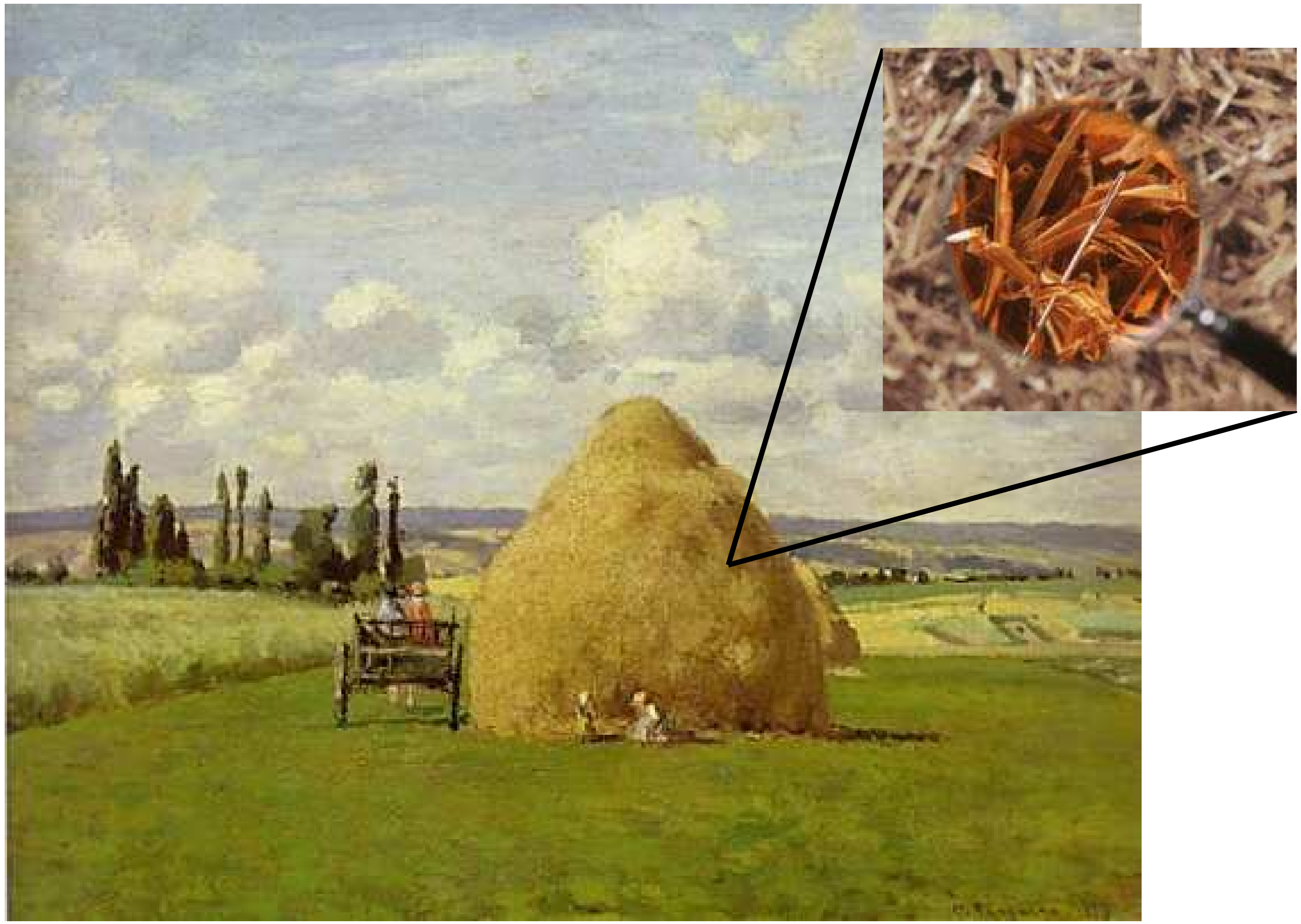


Sir William R. Hamilton (1805–1865)

A cycle that traverses **each vertex** of a graph exactly once is called an **Hamiltonian cycle**.

No insight available. **Exhaustive search** seems to be unavoidable.

Needle Problems

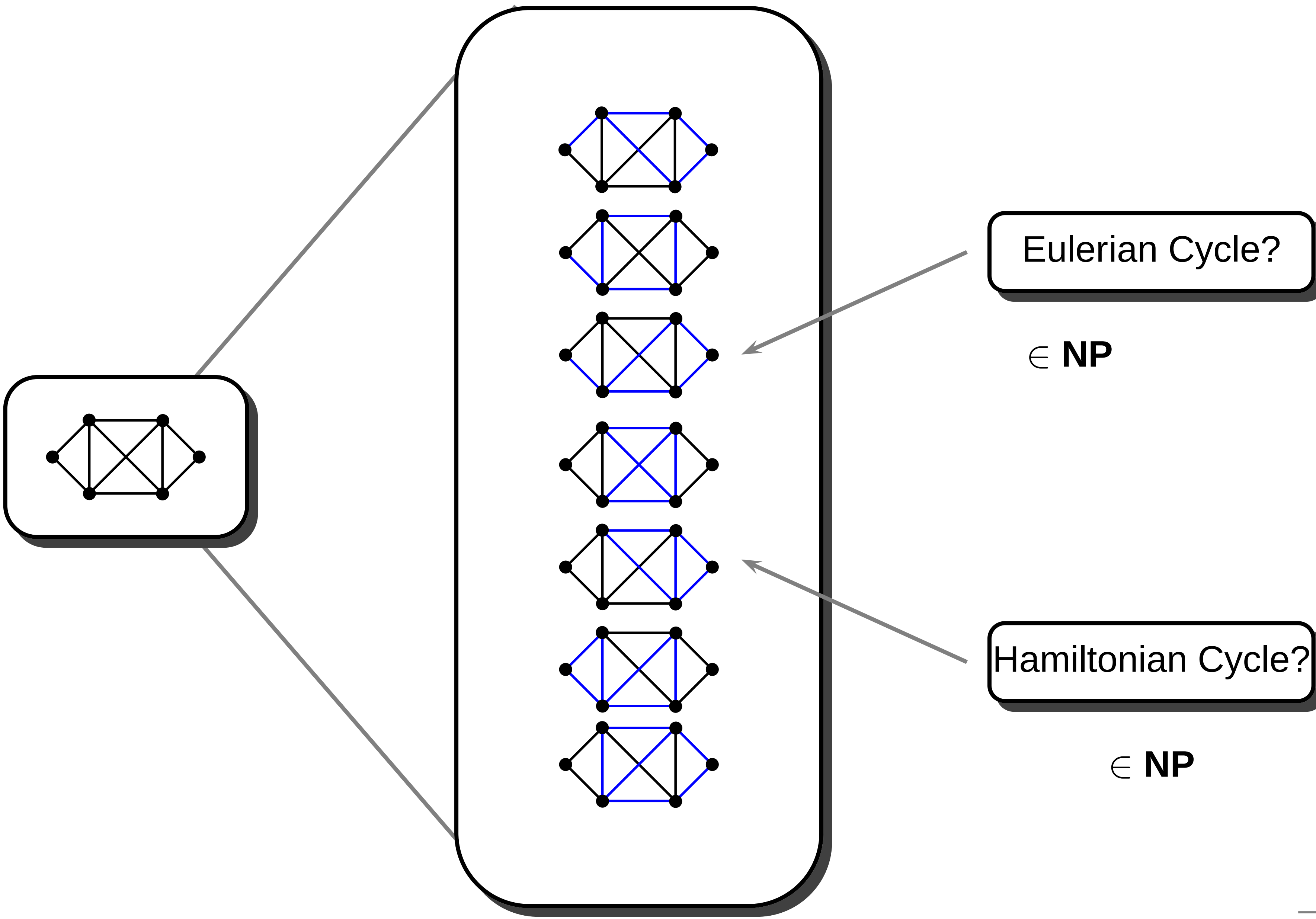


Camille Pissaro, *Haystack* (1873)

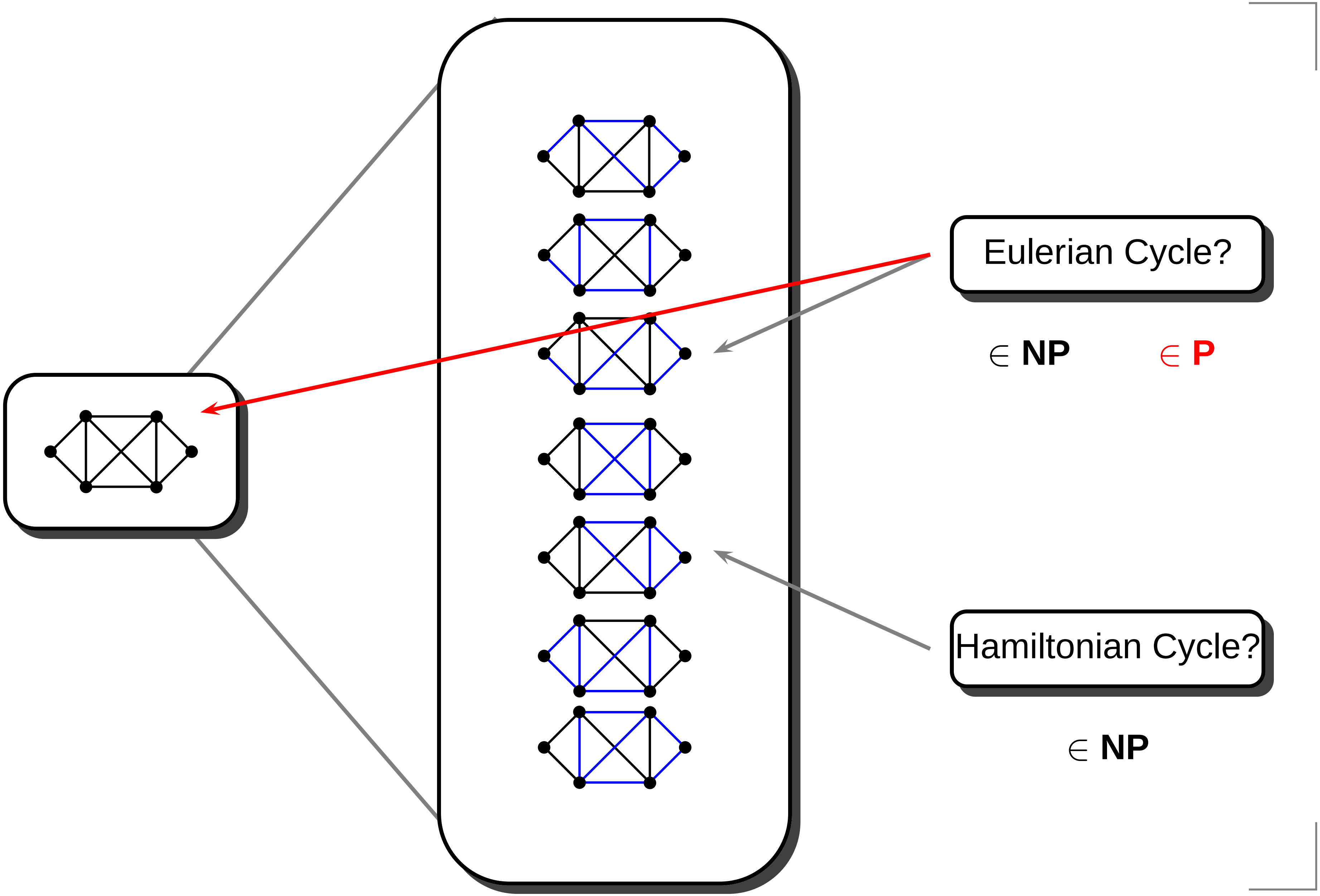
NP: solution can be **verified** in polynomial time

P: solution can be **found** in polynomial time

Mathematical Haystacks

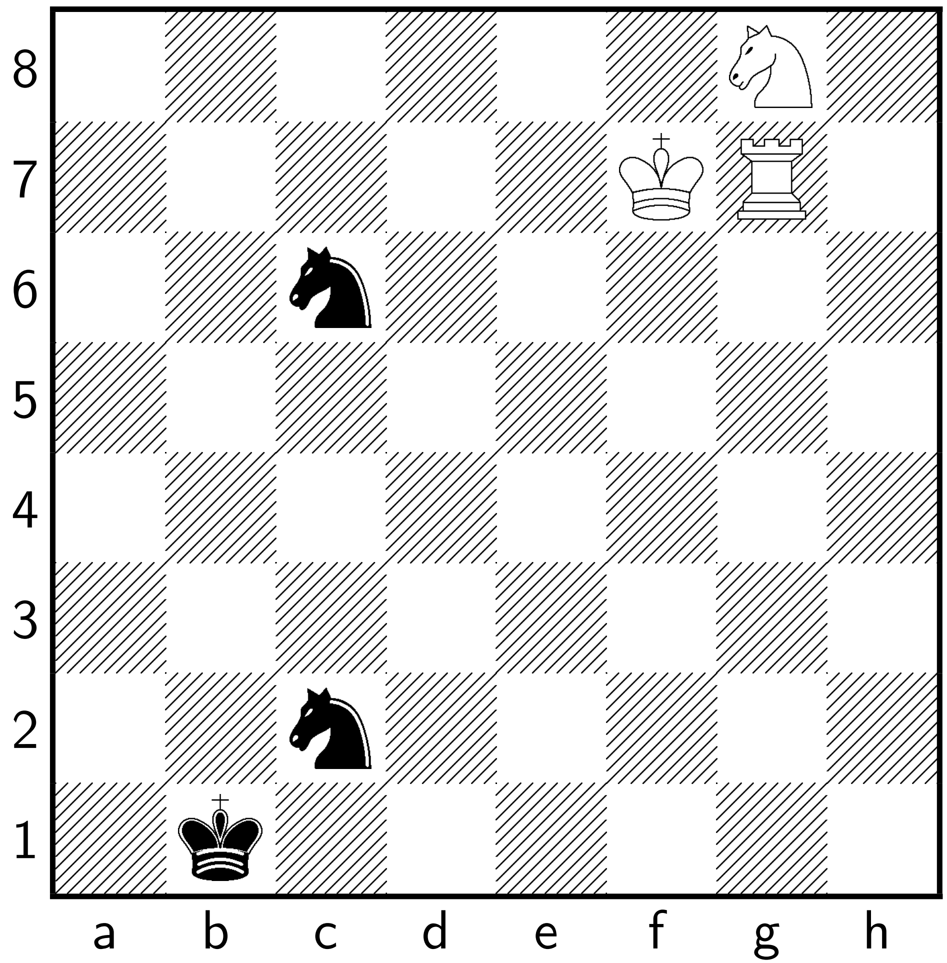


Mathematical Haystacks



A problem not in NP

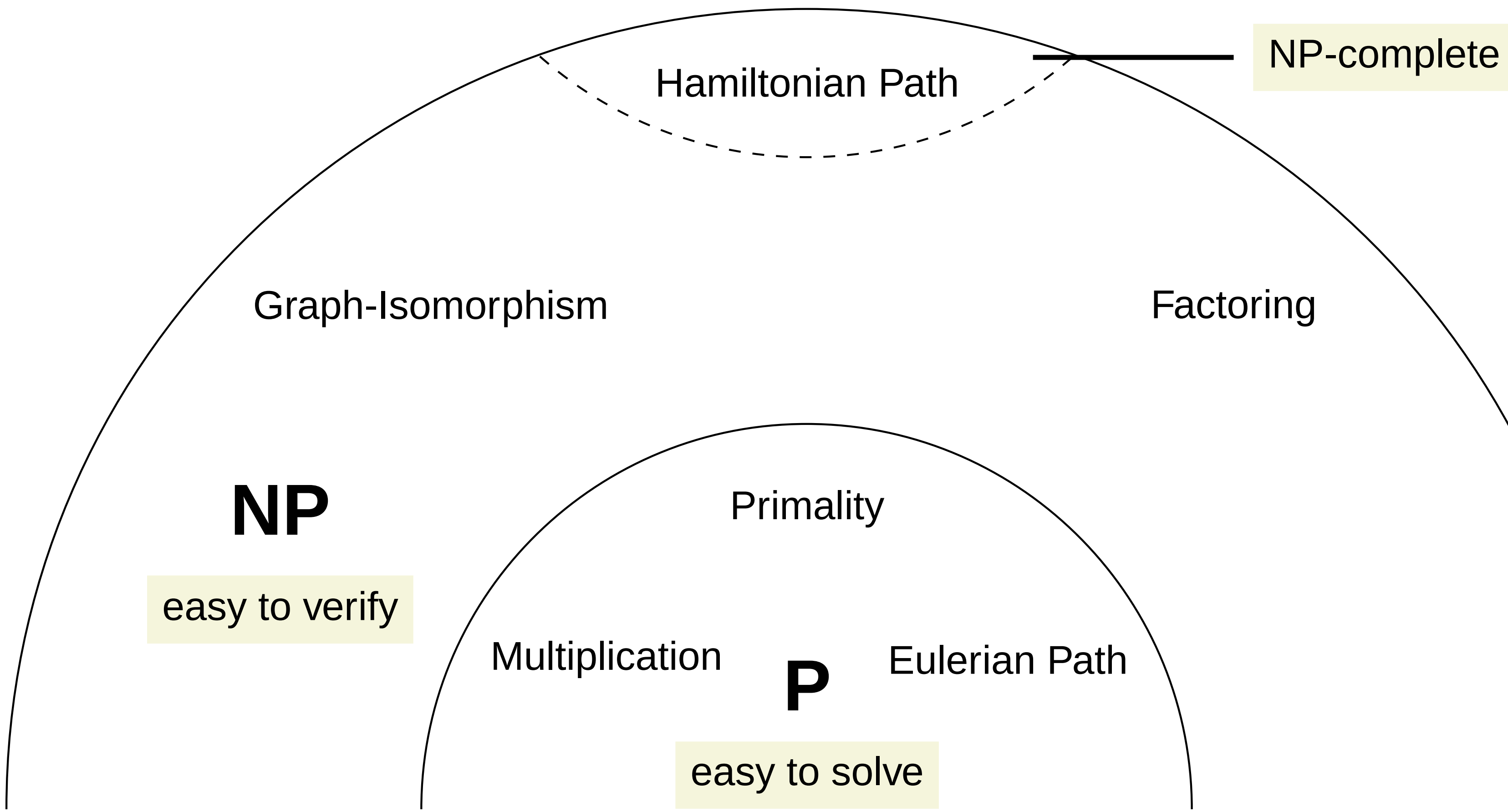
Lewis Stiller (1995)



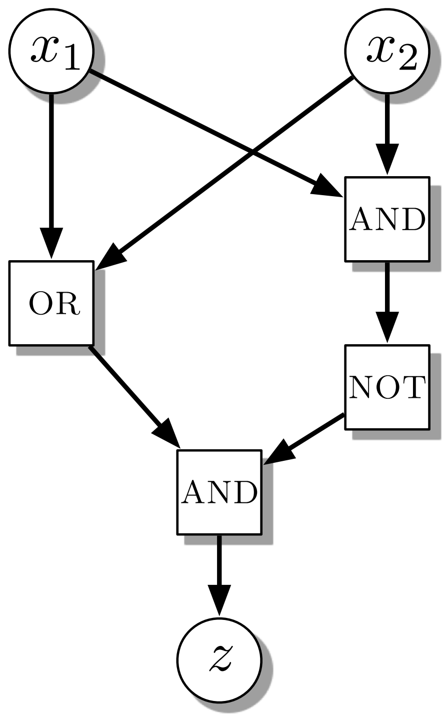
Mate in 262

P and NP

Is **finding** a solution fundamentally harder than **verifying** it? Is **P** $\neq$ **NP**?



NP-completeness



Any program that *verifies* a solution can be "compiled" into a Boolean circuit.

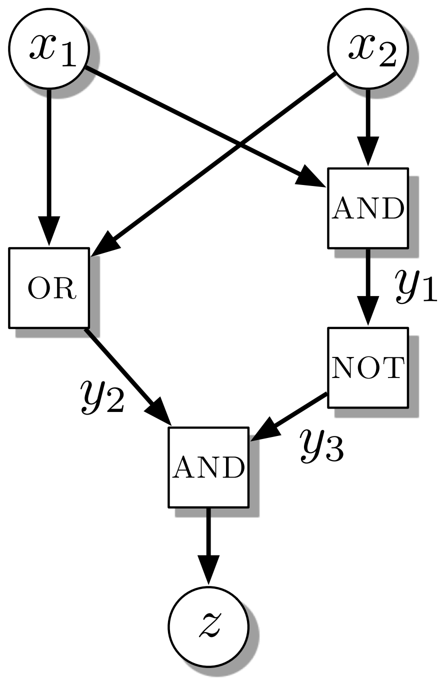
The circuit outputs "true" if an input solution works.

Is there a set of values for the inputs that makes the output true?

Circuit SAT
Given a circuit C.
Is C satisfiable?

Circuit SAT is **NP-complete** because Boolean circuits are powerful enough to carry out any finite computation.

From Circuits to Formulas



AND-gate:

$$y_1 = x_1 \wedge x_2 \iff (x_1 \vee \overline{y}_1) \wedge (x_2 \vee \overline{y}_1) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee y_1)$$

NOT-gate:

$$y_3 = \overline{y}_1 \iff (y_1 \vee y_3) \wedge (\overline{y}_1 \vee \overline{y}_3)$$

The circuit is equivalent to a Boolean formula:

$$\Phi(x_1, \ldots, z) = (x_1 \vee \overline{y}_1) \wedge (x_2 \vee \overline{y}_1) \wedge \ldots \wedge (z)$$

SAT (Satisfiability)
Given a Boolean formula $\Phi(x_1, \ldots, x_n)$.
Are there truth assignments for the x_i such that

$$\Phi(x_1, \ldots, x_n) = \text{true ?}$$

SAT is **NP-complete**.

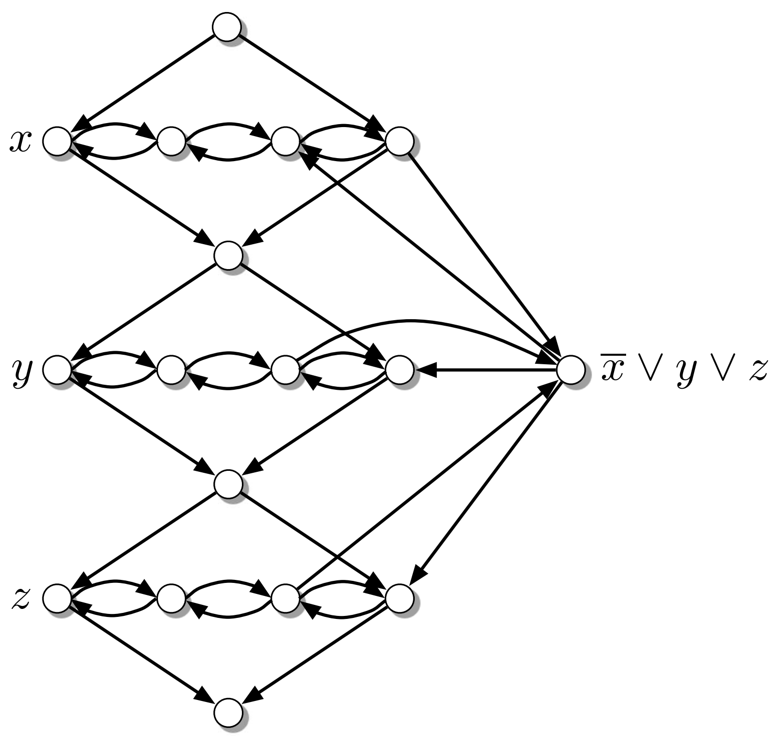
Simpler Formulas and Hamiltonian Paths

3-SAT:
Given a Boolean formula Φ with 3 variables in each clause.
Is Φ satisfiable?

3-SAT is **NP-complete**

$$(x_1 \vee x_2) \iff (x_1 \vee x_2 \vee \mathbf{z_1}) \wedge (\overline{\mathbf{z}}_\mathbf{1} \vee x_1 \vee x_2)$$

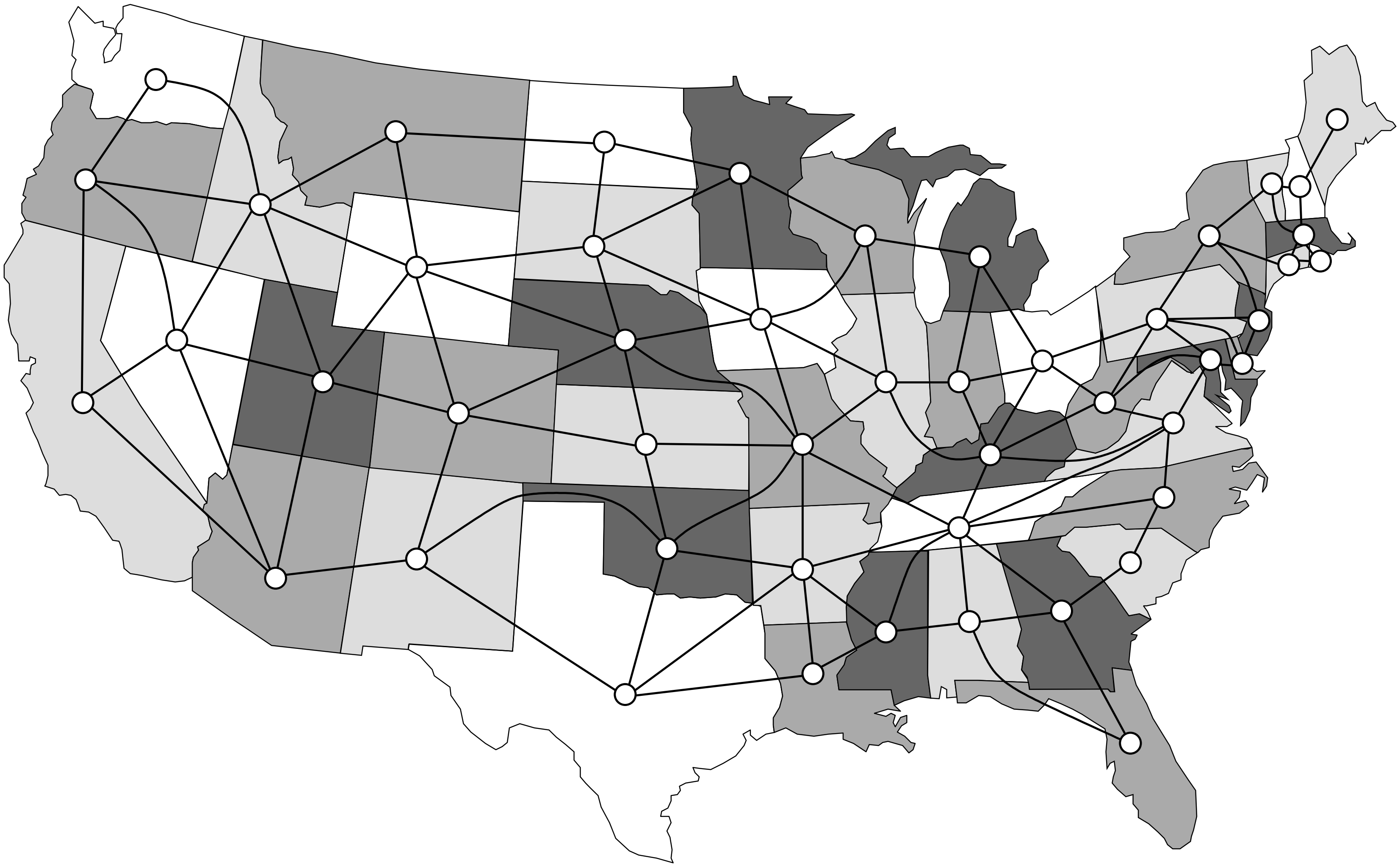
$$(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) \iff (x_1 \vee x_2 \vee \mathbf{z_1}) \wedge (\overline{\mathbf{z}}_\mathbf{1} \vee x_3 \vee \mathbf{z_2}) \wedge (\overline{\mathbf{z}}_\mathbf{2} \vee x_4 \vee x_5)$$



"gadget"

Hamiltonian Path is **NP-complete**.

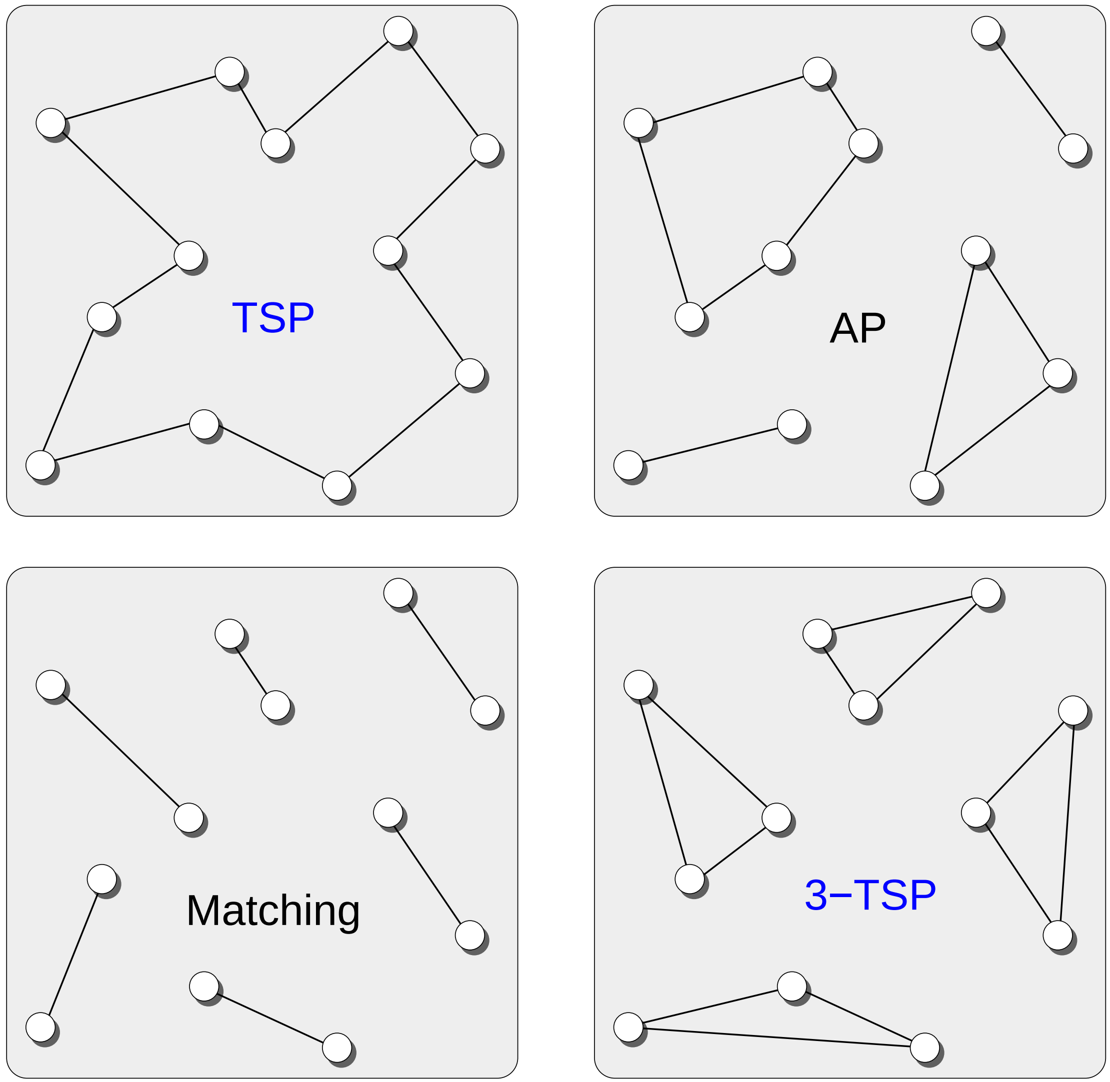
Map Coloring



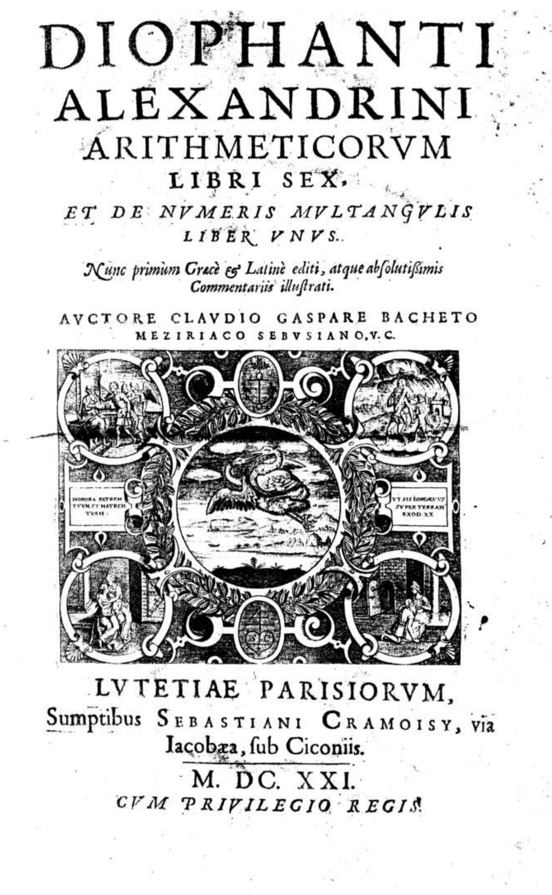
Planar K-Coloring: Can one color a planar graph with at most K colors?

Is in P for $K \neq 3$. Is **NP-complete for** $K = 3$.

Travelling Salesmen & Co



Diophantine Equations


DIOPHANTI
ALEXANDRINI
ARITHMETICORVM
LIBRI SEX.
ET DE NVMERIS MVLTANGVLIS
LIBER VNVS.
Nunc primùm Græcè & Latinè editi, atque absolutissimis
Commentariis illustrati.
AVCTORE CLAVDIO GASPARE BACHETO
MEZIRIACO SEBVSIANO, V.C.
LVTETIAE PARISIORVM,
Sumptibus SEBASTIANI CRAMOISY, via
Iacobæa, sub Ciconiis.
M. DC. XXI.
CVM PRIVILEGIO REGIS.

Given natural numbers a, b, and c.
Do the following equations have a solution x, y in natural numbers?

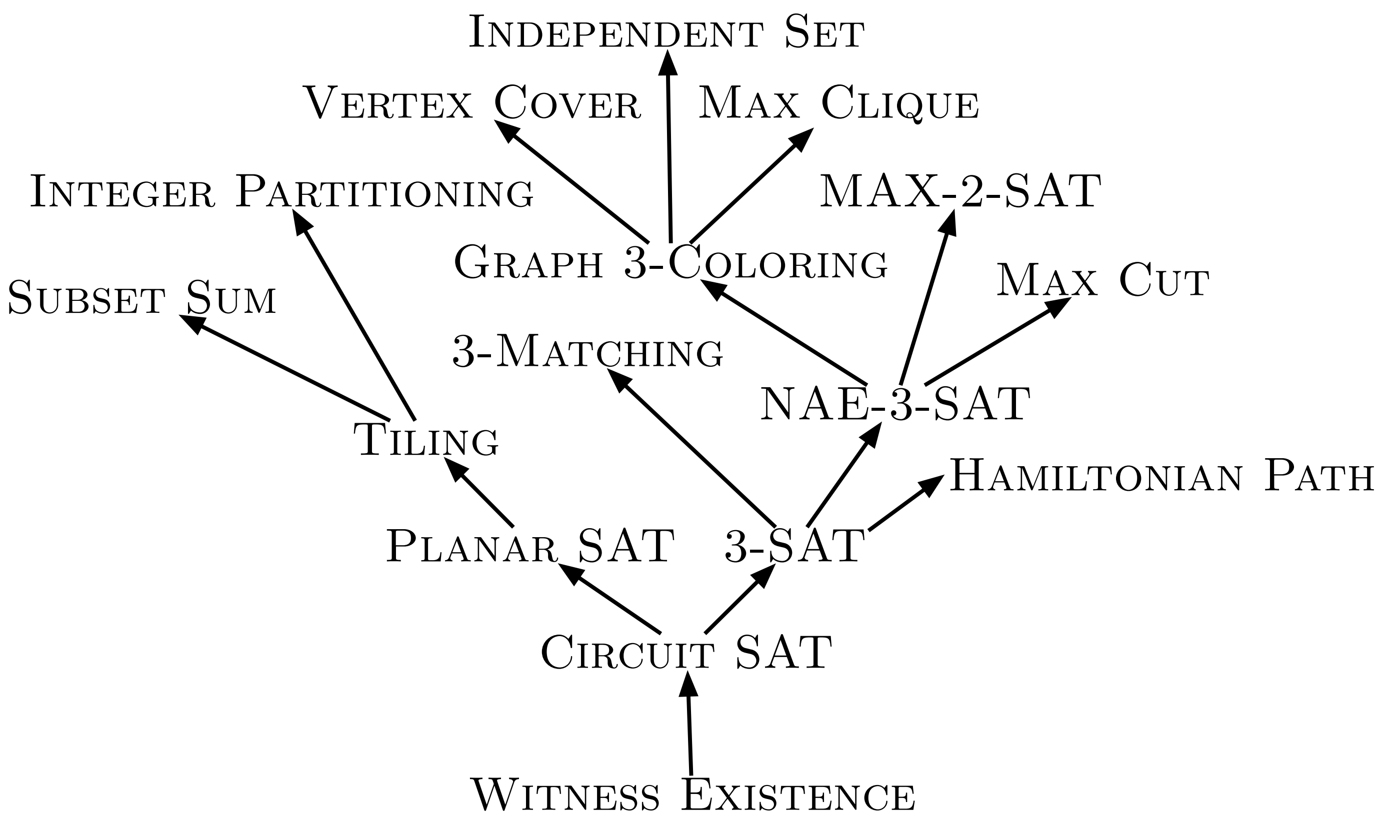
$$a\,x + b\,y = c$$

$$a\,x + b\,y^2 = c$$

Linear Diophantine Equation is in P.

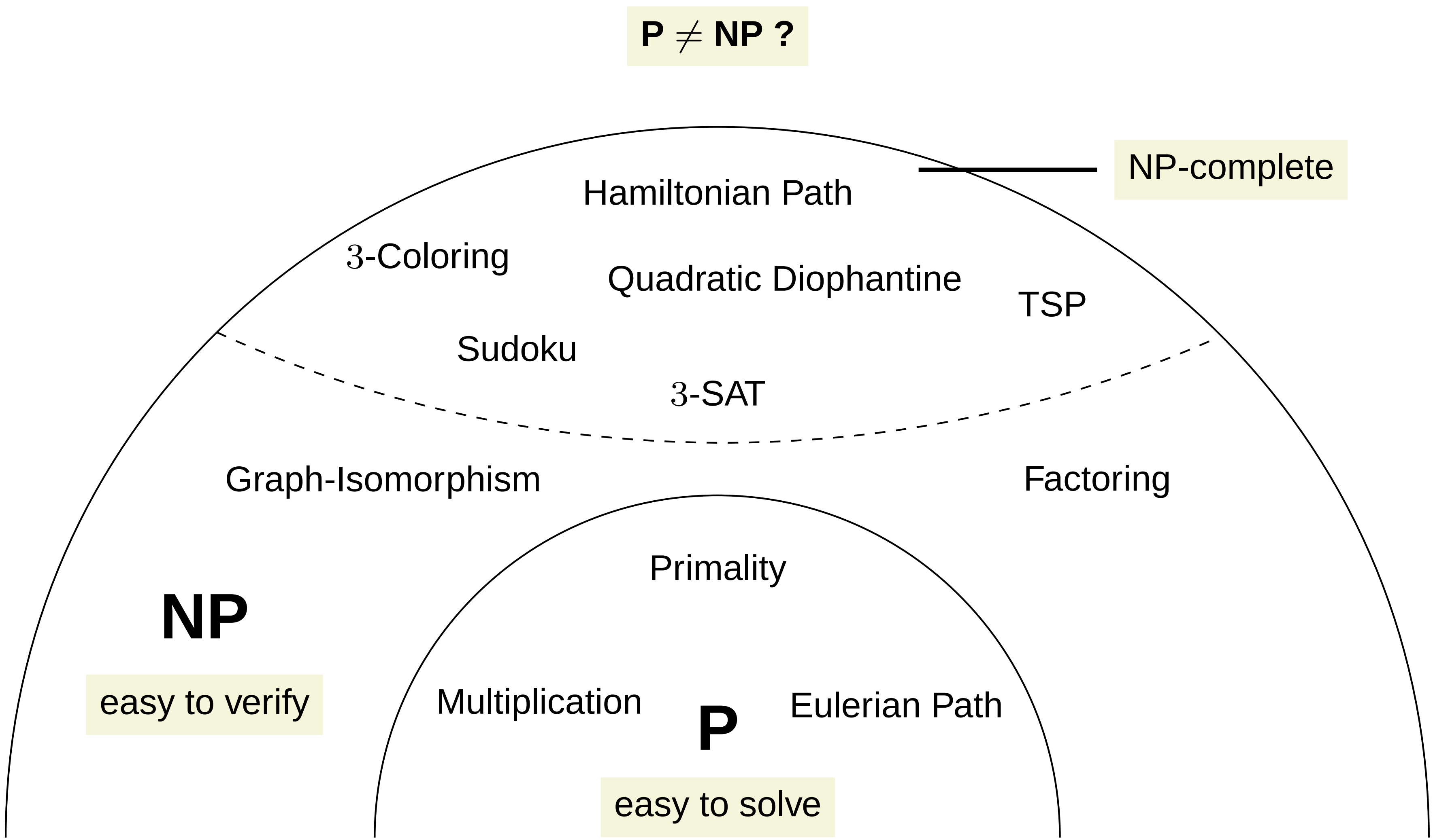
Quadratic Diophantine Equation is **NP-complete**.

NP-complete Family Tree

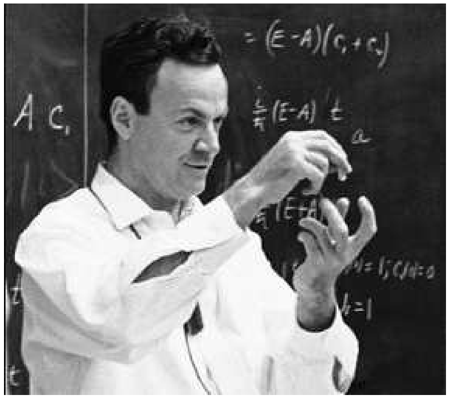


More than 3000 NP-complete problems known

P and NP


P $\neq$ NP ?
NP-complete
Hamiltonian Path
3-Coloring
Quadratic Diophantine
TSP
Sudoku
3-SAT
Graph-Isomorphism
Factoring
Primality
NP
easy to verify
Multiplication
P
Eulerian Path
easy to solve

Quantum Computation



R.P. Feynman (1918–1988)

Classical computers cannot efficiently simulate a quantum mechanical system.

Hilbert space is too big!

qbit: $|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$

n qbits = 2^n probability amplitudes!

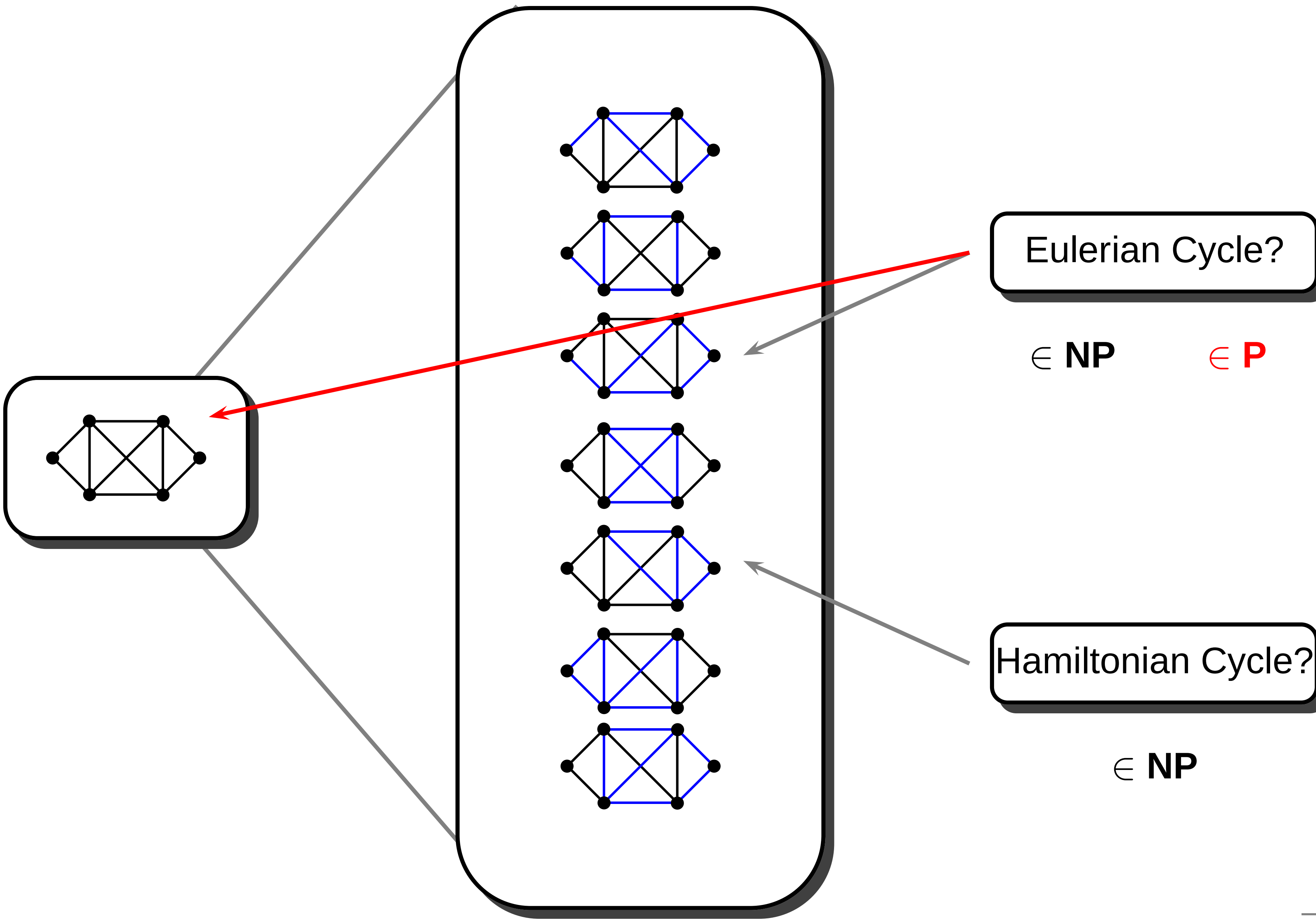
Information processing in quantum mechanics is enormous.

Can we get a ride?

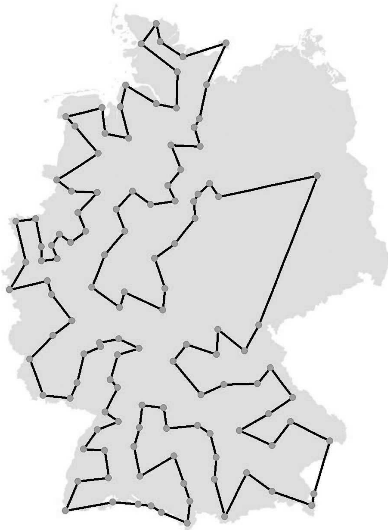
- Shor (1994): Factoring in polytime
- Grover (1995): Searching a list of N entries in time $\mathcal{O}(\sqrt{N})$

Problem: Measurement process

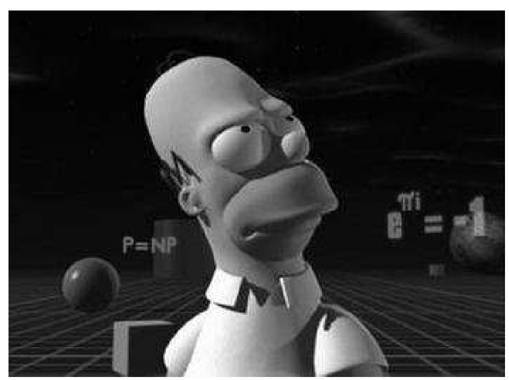
Quantum Search ?



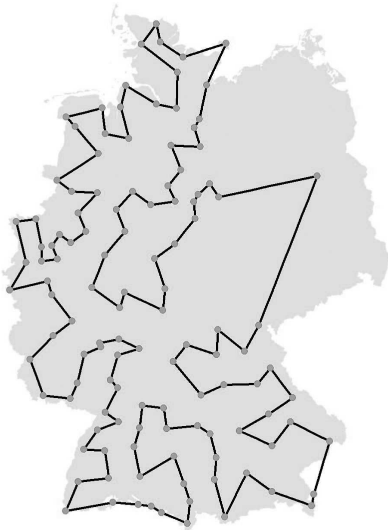
What if P=NP ?



Optimization
shorter tours



What if P=NP ?



Optimization
shorter tours

Cryptography

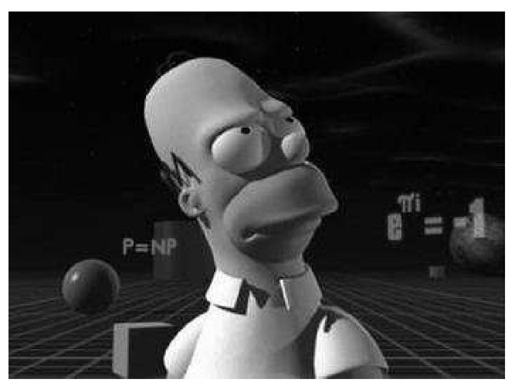
Decrypt: Does encrypted message M correspond to clear text T?

Decrypt $\in$ NP

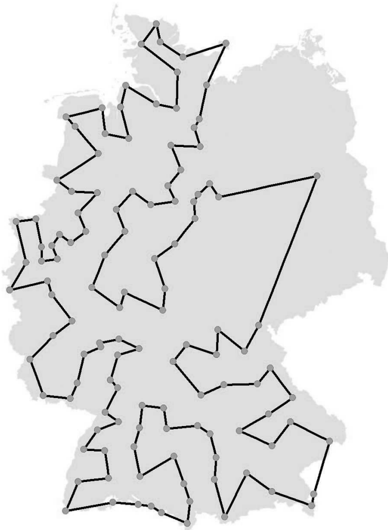
What if P=NP ?

Optimization
shorter tours

Cryptography
disappears



What if P=NP ?



Optimization
shorter tours

Cryptography
disappears

Mathematics

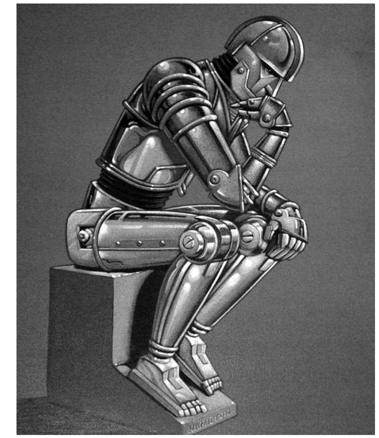
Short-Proof-Existence: Does Theorem T have a proof with less than n lines?

Short-Proof-Existence $\in$ NP

What if P=NP ?

Optimization
shorter tours

Cryptography
disappears



Mathematics
mechanized

The evidence in favor of the P$\neq$NP hypothesis is so overwhelming, and the consequences of its failure are so grotesque, that its status may perhaps be compared to that of physical laws rather than that of ordinary mathematical conjectures.

V. Strassen

A Letter from Gödel

1906–1978

1903–1957

Princeton, 20 March 1956

Dear Mr. von Neumann:

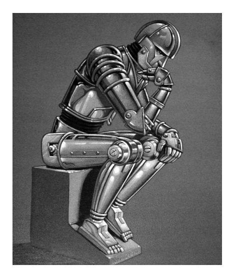
.
.
.

One can obviously easily construct a Turing machine, which for every formula F in first order predicate logic and every natural number n, allows one to decide if there is a proof of F of length n. Let $\varphi(n)$ be the number of steps the machine requires for this. The question is, how fast does $\varphi(n)$ grow for an optimal machine. One can show that $\varphi(n) \geq Kn$. If there actually were a machine with $\varphi(n) \sim Kn$ (or even only $\varphi(n) \sim Kn^2$), this would have consequences of the greatest magnitude. That is to say, it would clearly indicate that, despite the unsolvability of the Entscheidungsproblem, the mental effort of the mathematician in the case of yes-or-no questions could be completely replaced by machines. One would simply have to select an n large enough that, if the machine yields no result, there would then be no reason to think further about the problem.

.
.
.

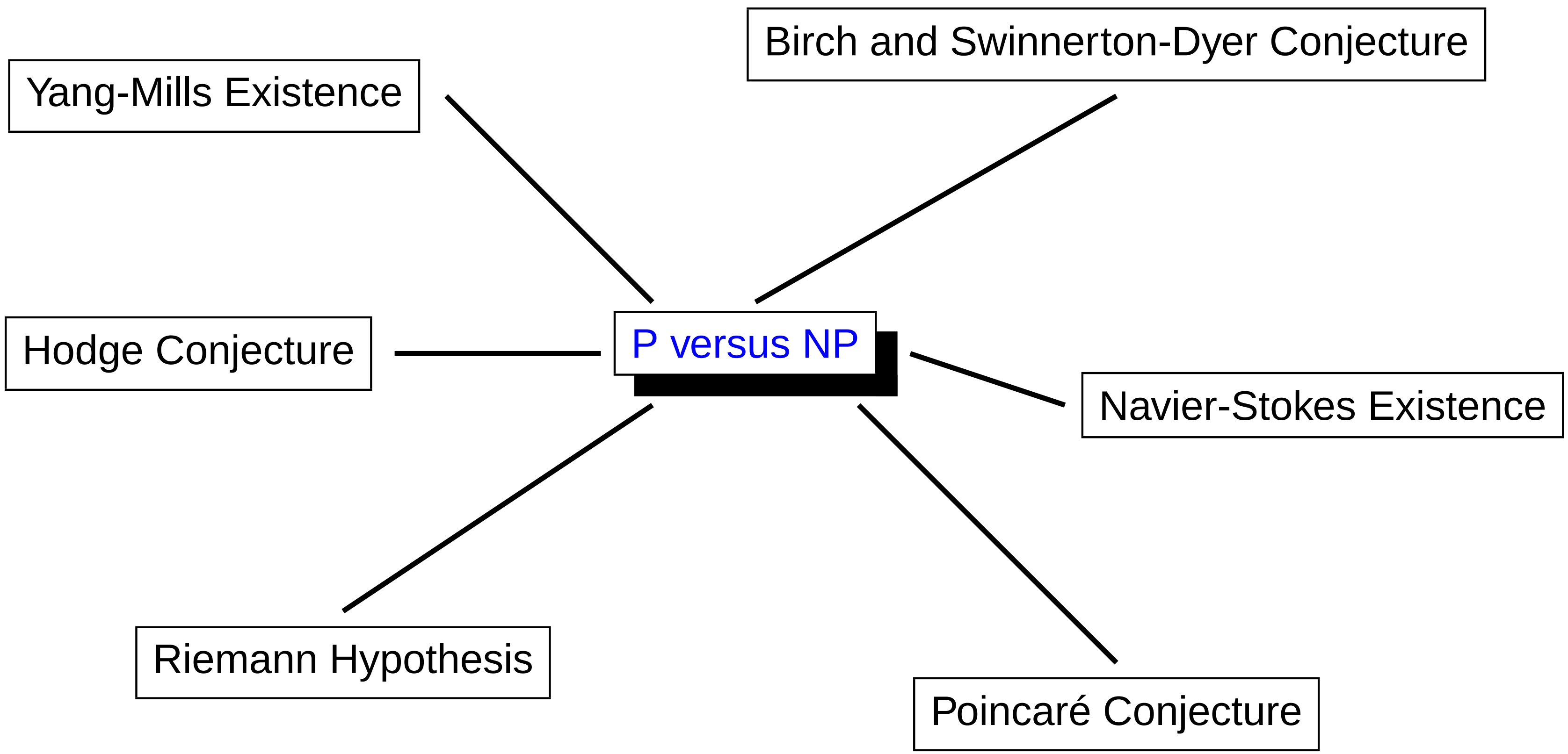
Sincerely yours,
Kurt Gödel

$\xrightarrow{\text{P=NP}}$



Clay Millenium Problems

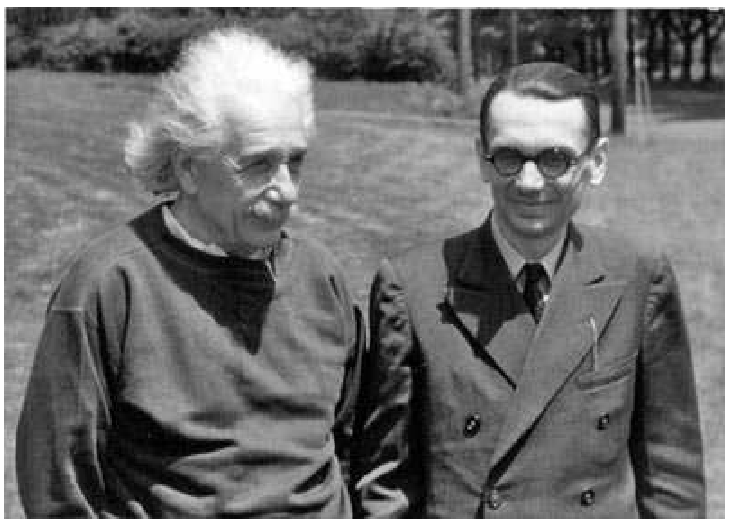
P versus NP—a gift to mathematics from computer science
Steve Smale



The Evil Adversary

- Theory of computational complexity is based on **worst case** analysis
- Benefits:
  - guaranteed bounds
  - powerful tool: reduction
- Drawbacks:
  - worst case can be rather exotic
  - Nature’s not evil!
- Alternative: **average case** complexity
  - Phasetransitions
  - Clustering
  - REM-like scenarios
  - powerful tools: experiments, moment bounds, . . .

Experimental Mathematics



"If mathematics describes an objective world just like physics, there is no reason why inductive methods should not be applied in mathematics just the same as in physics."

Kurt Gödel (1951)

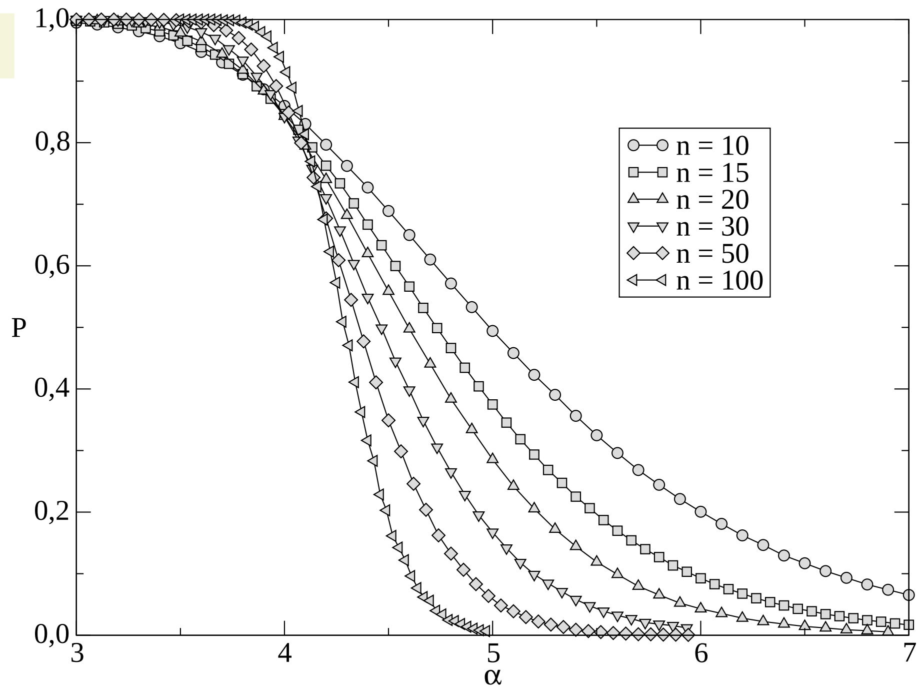
Random 3-SAT

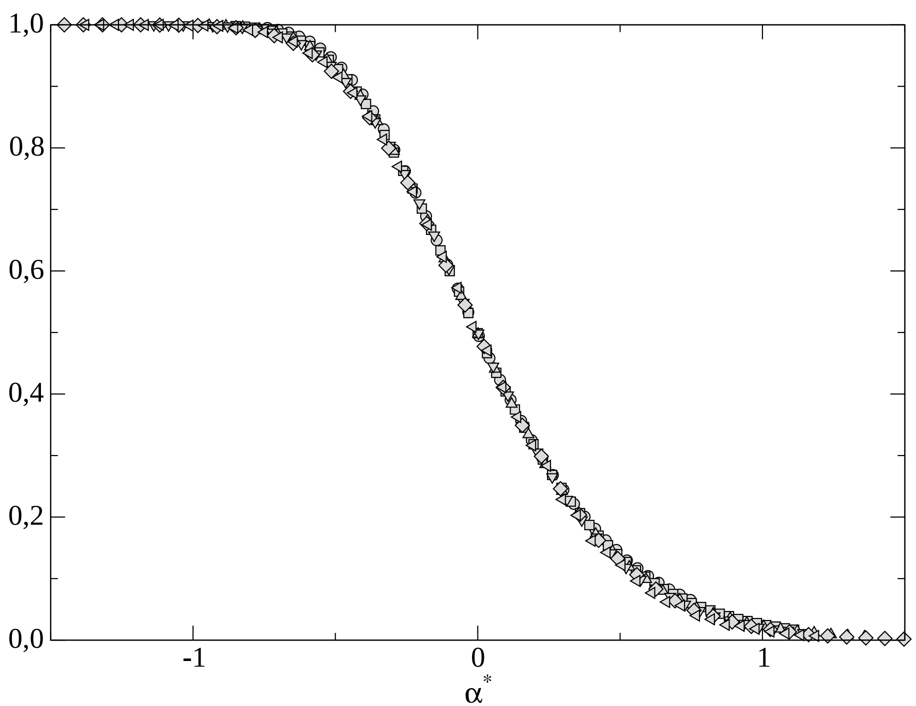
3-SAT formula Φ with n variables and m clauses

Choose each clause randomly among $2^{-3}\binom{n}{3}$.

Sparse case: $m = \alpha n$ for some density α.

$P(\alpha)$ = Probability that Φ is sat.]





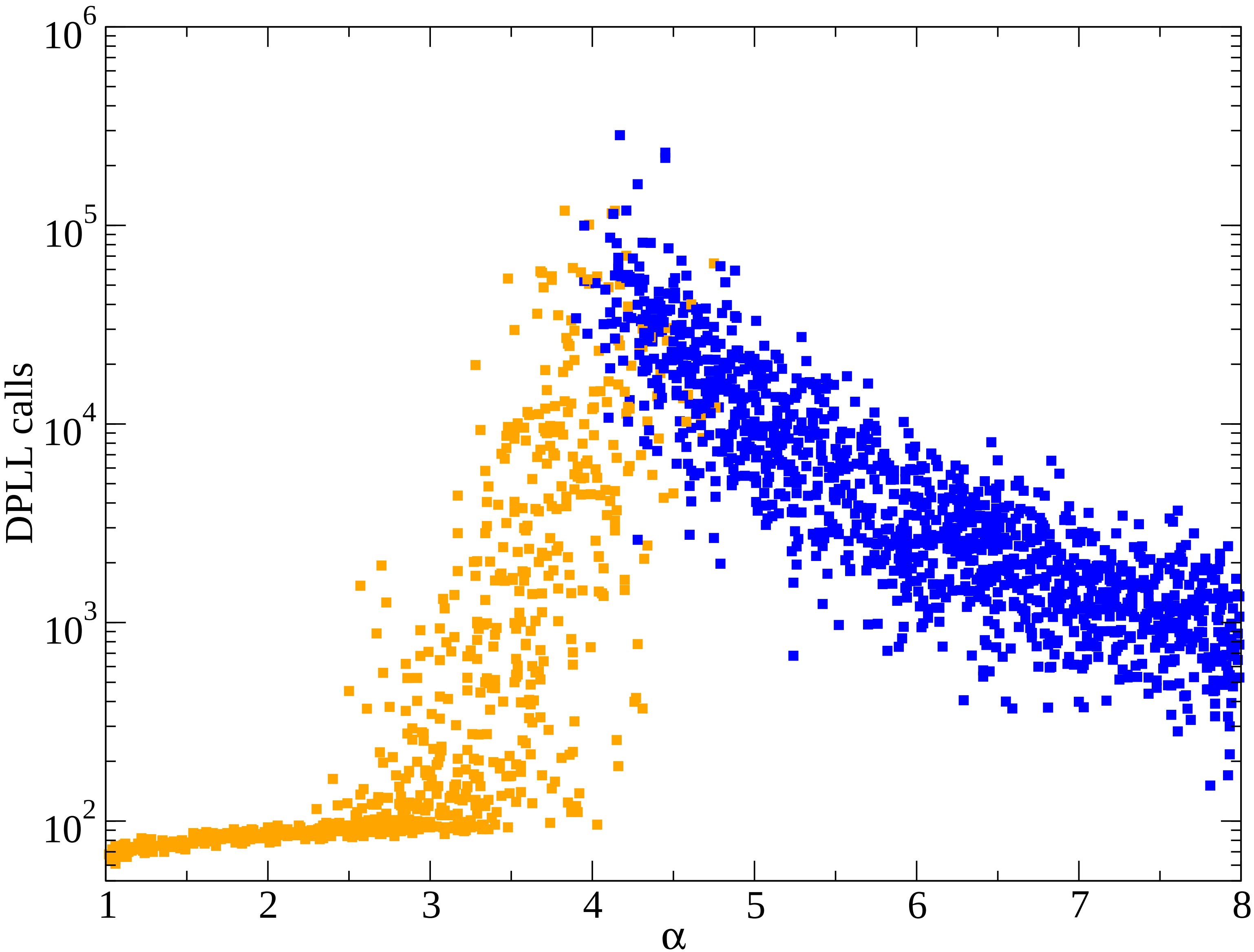
Finite size scaling

$$\alpha^\star = \sigma(n)[\alpha - \alpha_c(n)]$$

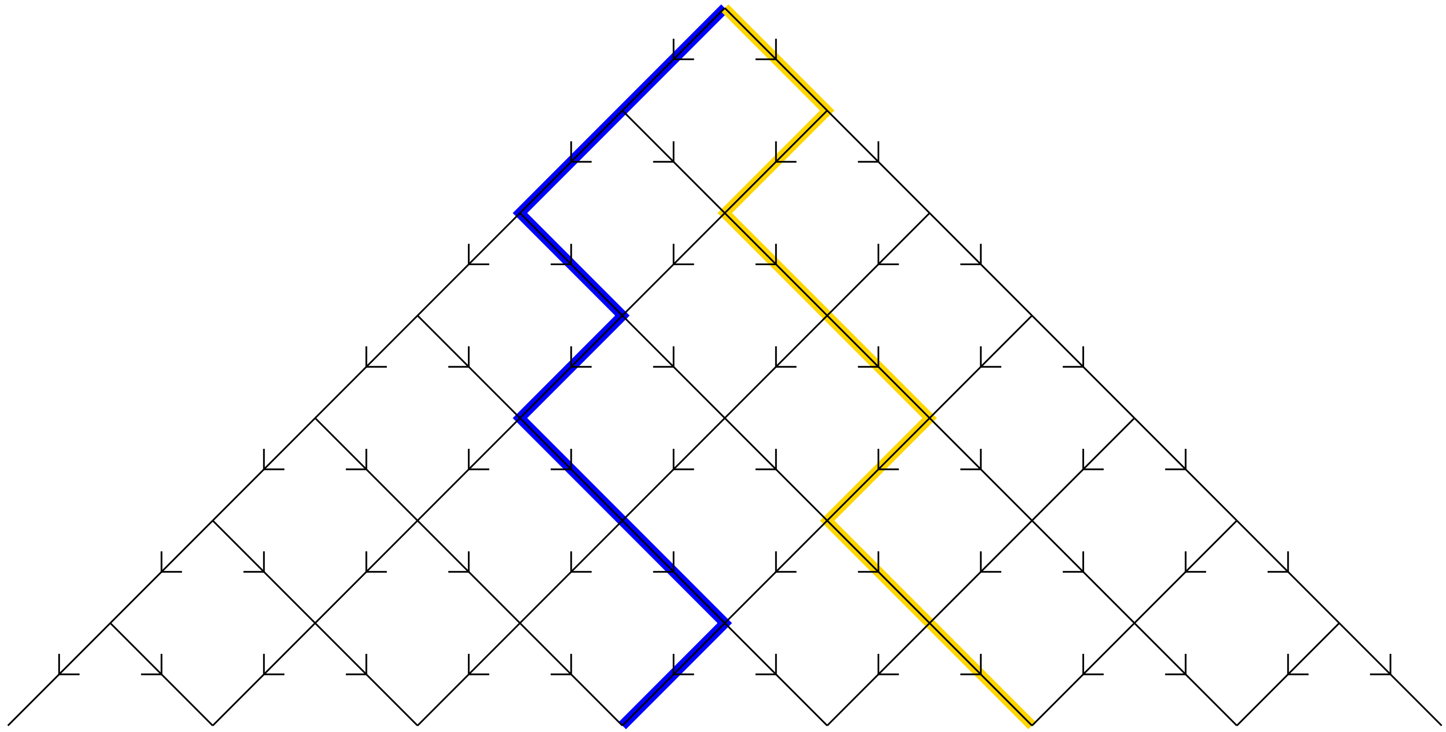
Sharp transition for $n \to \infty$ at

$$\alpha_c \simeq 4.26$$

Easy-Hard Transition



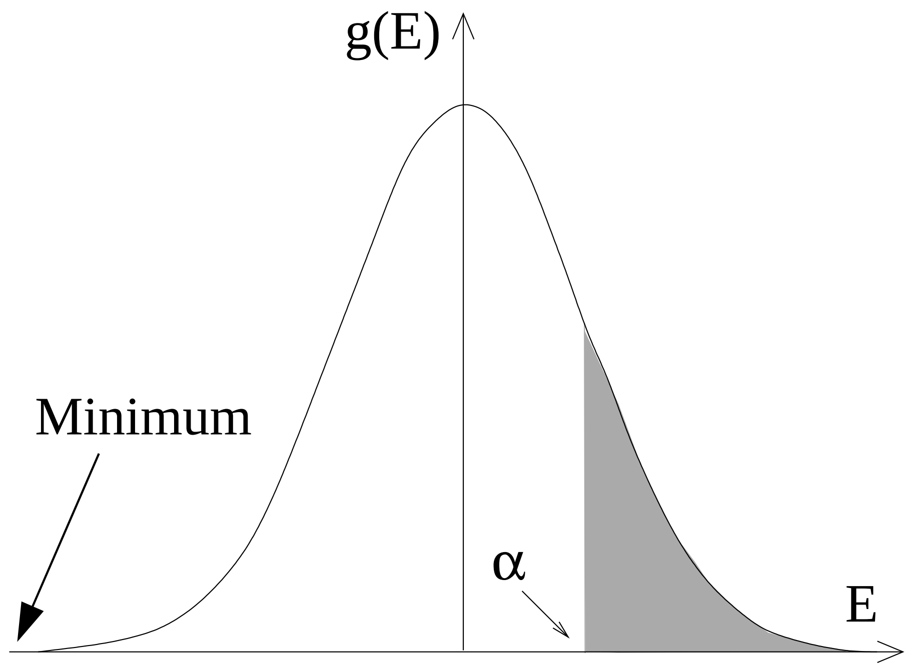
DPRM



Directed **P**olymer in **R**andom **M**edia

- single-source shortest-path problem
- solvable in polynomial time (Bellman-Ford)

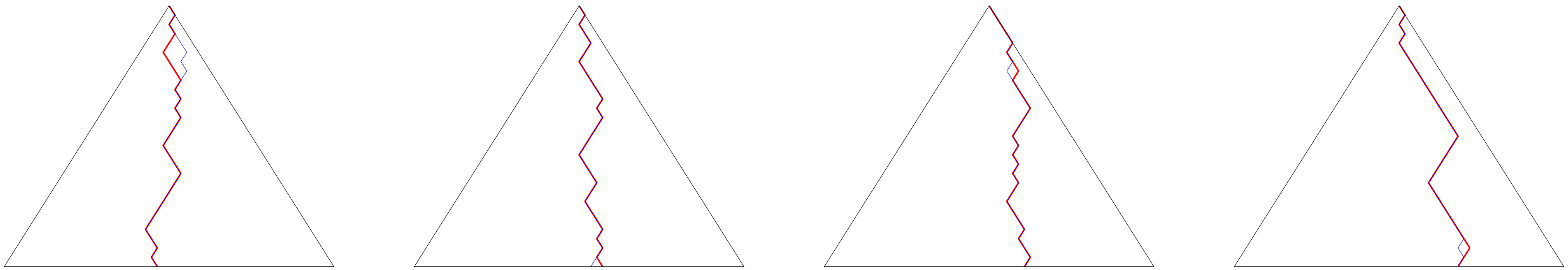
Constrained DPRM



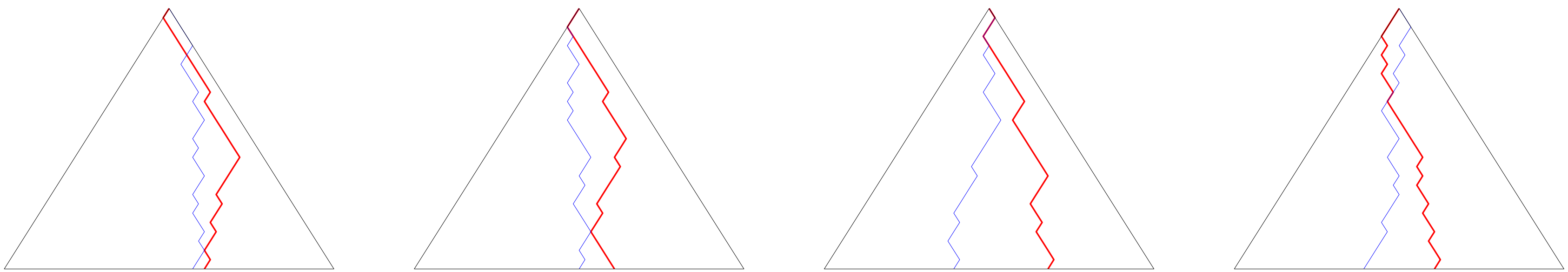
Find shortest path among all paths with **length** $\geq \alpha$.

- cannot be easier than unconstrained case ($\alpha = -\infty$)
- is NP-complete
- has **local REM property**

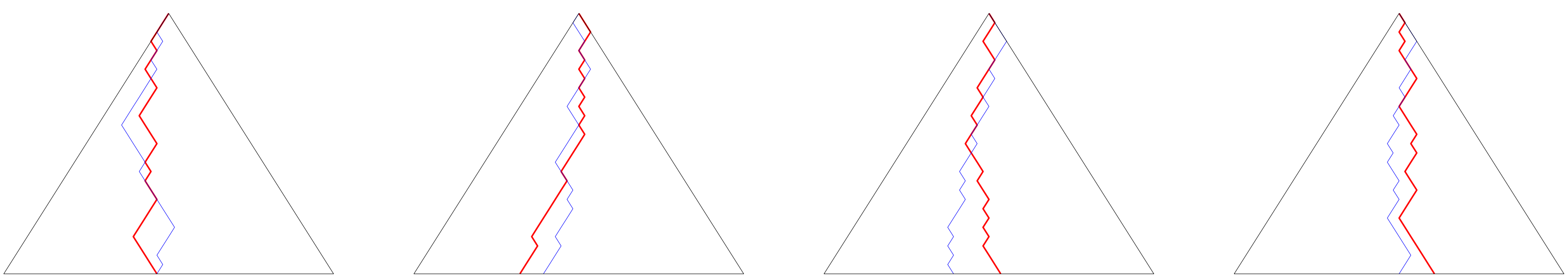
Energetically Adjacent Paths



$\alpha = -\infty$

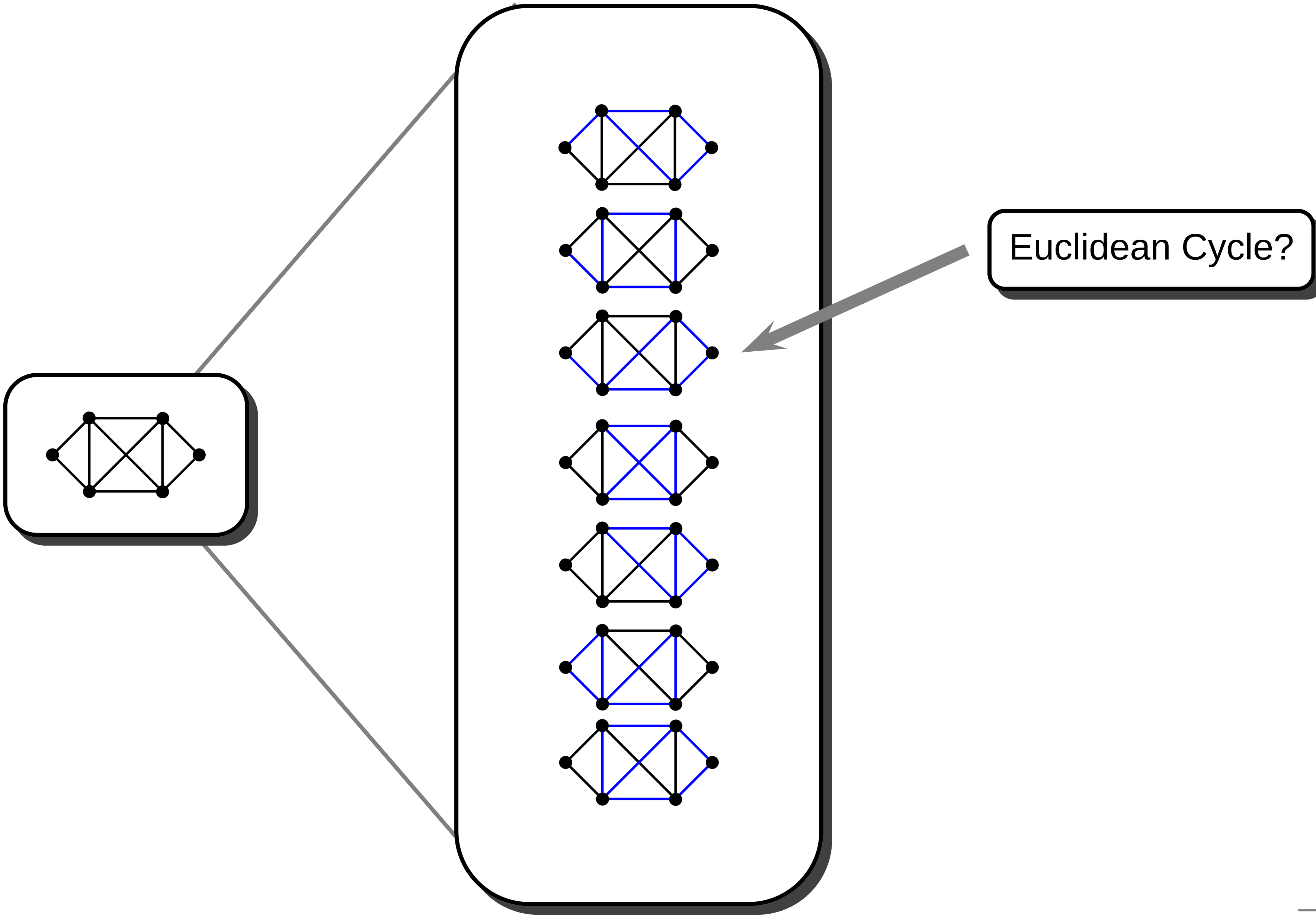


$\alpha = 0$

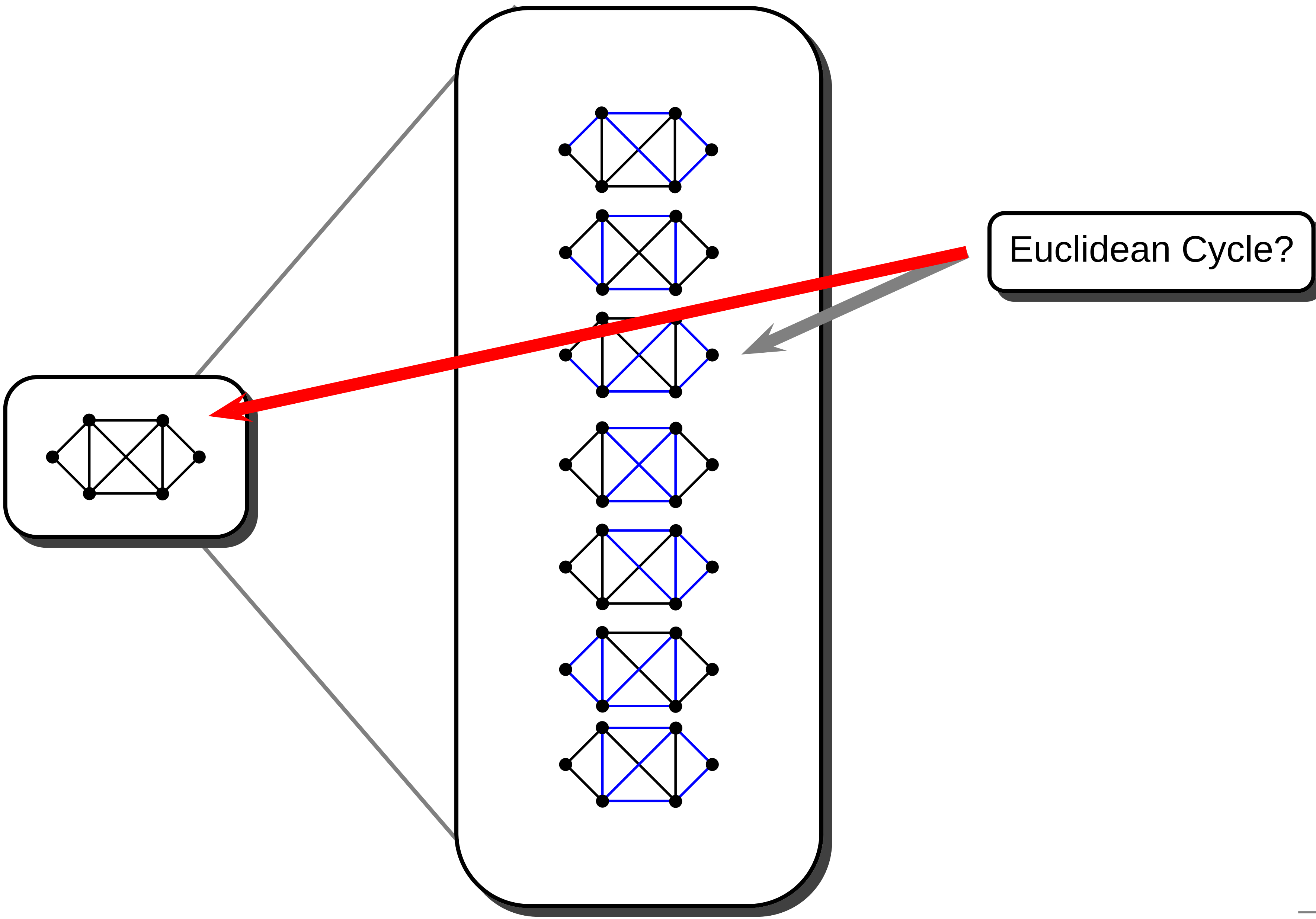


random

Mathematical Haystacks



Mathematical Haystacks



Mathematical Haystacks

74636649
31389813
34562189
73552552
31456321
46372280
20349548
43289010
74093204
31415926
05647308
89745638
80103341
31443277
45632831
21467430

31415926 ?

Further Reading

Oxford Univ. Press (2008)
www.nature-of-computation.org

Trying to understand the nature of computation has its own beauty just like trying to understand the fundamental building blocks of the universe.

Lance Fortnow

- Brian Hayes, The Easiest Hard Problem, *American Scientist* March-April 2002
- S.M., Computational Complexity for Physicists, *Computing in Science and Engineering* **4** (2002) 31–47
- A.G. Percus, G. Istrate and C. Moore, eds., Computational Complexity and Statistical Physics, Oxford University Press, New York, 2006