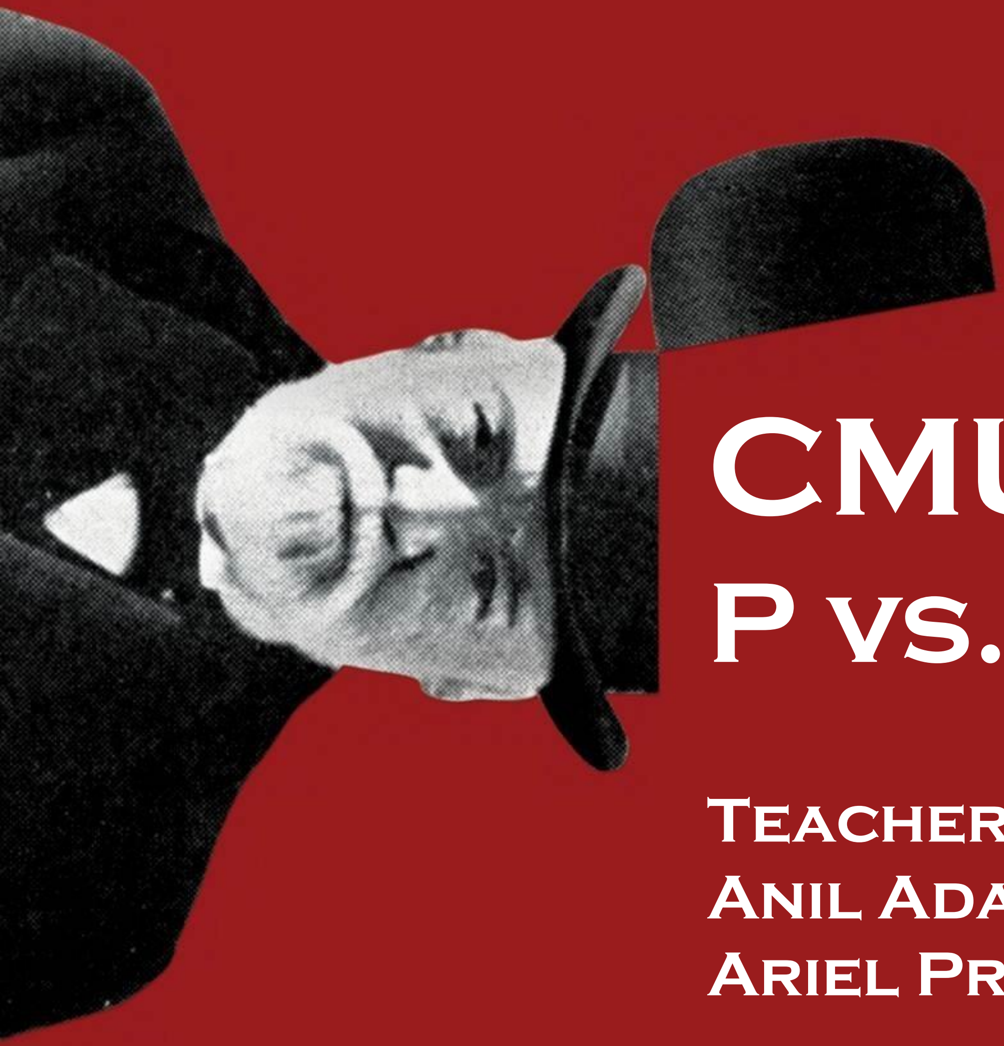


CMU 15-251

P vs. NP

Teachers:
Anil Ada
Ariel Procaccia (this time)

Millennium Prize Problems

- Seven famous problems in math stated in 2000 by the Clay Foundation
- \$1,000,000 prize for solving any of them
- One of the problems: **P** vs. **NP**

Millennium Prize Problems

Keith Devlin

If one is solved in the next few years, it'll probably be **P** vs. **NP**

If, in the year 3000, one of them is unsolved, it will be **P** vs. **NP**

Laszlo Lovasz

Millennium Prize Problems

- The **P** vs. **NP** problem is the only Millennium Prize problem that has the potential to change the world
- So what is it?

Sudoku

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

$$3 \times 3 \times 3 \times 3$$

SUDOKU

	F		2						6			C	B	3	
	C				4	8	E	A			0		D		
D	A	8			3		2	7	F			6		5	
6			E	D	F		C		8						7
	9	3		7					A						2
E						6	F	5		8	4		3		1
C	8		1	3	9	D		0	2		E				
	D		6		5	E	B		1					0	4
9	6					1		F	3	2		0		A	
				4		A	8		D	0	9	B		2	5
2		A		0	D		5	6	C						F
5						2					A		4	8	
B						4		1		A	2	F			0
	0		7			F	3	C		D			2	9	B
		5		1			A	9	0	B				D	
	2	D	A			9						1		4	

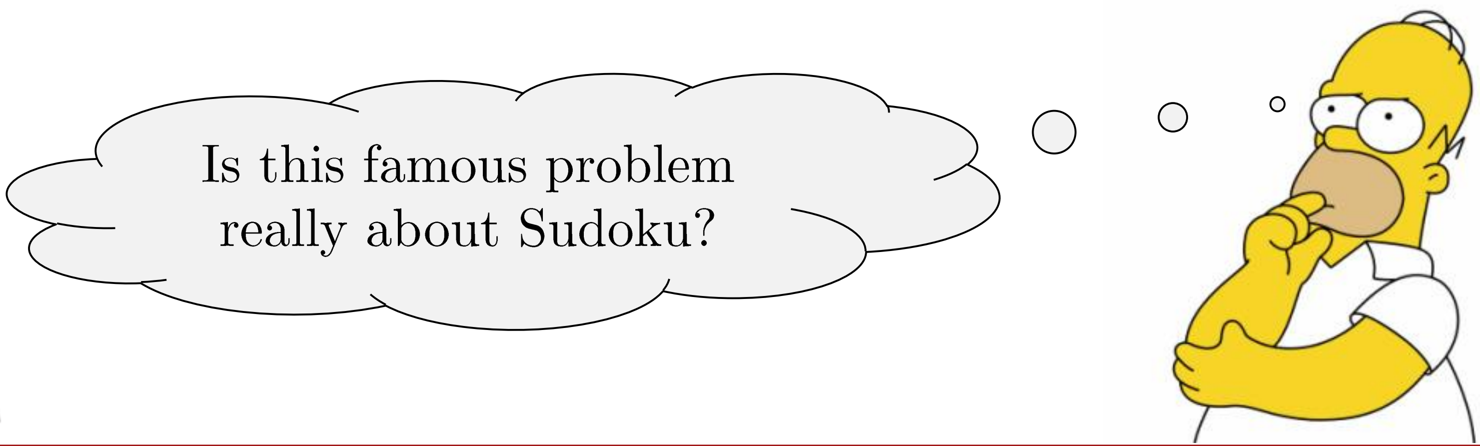
$$4 \times 4 \times 4 \times 4$$

Sudoku

- Sudoku: Given a partially filled $n \times n \times n \times n$ Sudoku board, can it be filled?
- Naive decision algorithm: Check all possibilities, in time $O\left(n^{2n^4}\right)$
- Verifying a solution: $O(n^4)$
- For $n = 100$
  - Verifying a solution: 100M steps
  - Deciding YES/NO: Number with 400M digits!

SUDOKU

- Question: Is there a polynomial-time algorithm that can solve SUDOKU?
- This is equivalent to the **P** vs. **NP** problem!



P vs. NP

- Informal formulation of **P** vs. **NP**:
  - Let L be an algorithmic task
  - Suppose there is an efficient algorithm for verifying solutions to L ($L \in NP$)
  - Is there an efficient algorithm for finding solutions to L? ($L \in P$)

SUDOKU is not just one instance of this problem; if the answer is "yes" for SUDOKU, it is "yes" in general!

Efficiency

- Efficient = polynomial time
- Given a decision problem L, $x \in L$ means that x is a YES instance of L; $|x|$ is its size
- **P** = Decision problems L such that there exists a constant c and an algorithm A such that A runs in time $|x|^c$ and $A(x) =$ YES if and only if $x \in L$
- We saw last time that 2-Coloring is in **P**

Verifying solutions

- In problems like SUDOKU, verifying the solution can be done efficiently
- **NP** = Decision problems whose solutions can be verified in polynomial time in their input size

The N in **NP** stands for "nondeterministic"

NP: SEMI-FORMAL DEFINITION

- $L \in \mathbf{NP}$ if and only if there are constants c, d and an algorithm V called the verifier such that:
  - V takes two inputs, x and y, where $|y| \leq |x|^c$; x is called the instance and y is called the certificate
  - $V(x, y)$ runs in time $O\left((|x| + |y|)^d\right)$
  - If $x \in L$, $\exists y$ such that $V(x, y) =$ YES
  - If $x \notin L$, $\forall y$, $V(x, y) =$ NO

Examples

- Sudoku: Given a partially filled $n \times n \times n \times n$ Sudoku board, can it be completed?
- Input size: n
- Certificate: board filled with numbers
- Verifier: Check that each square, row, and column contain all numbers

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

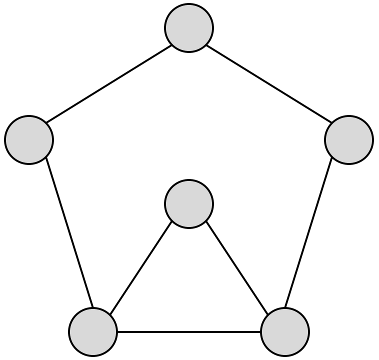
Instance

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

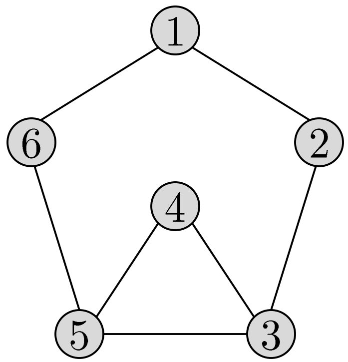
Certificate

Examples

- Hamiltonian-cycle: Given a graph $G = (V, E)$, does it contain a Hamiltonian cycle?
- Input size: $n = |V|$
- Certificate: A permutation of the n vertices
- Verifier: Check that the permutation contains each vertex exactly once, and there is an edge between adjacent vertices



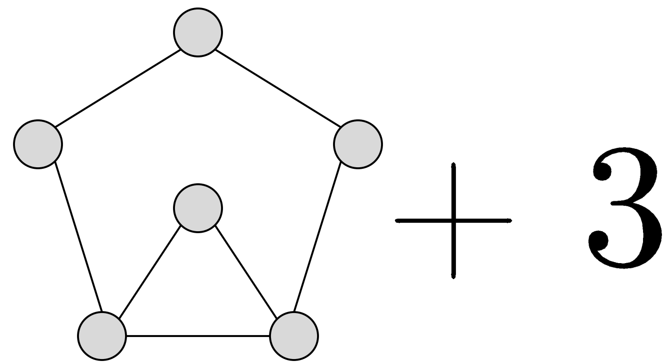
Instance



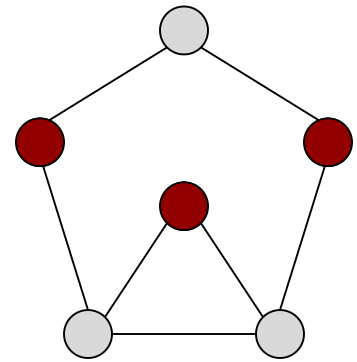
Certificate

Examples

- Independent-set: Given a graph $G = (V, E)$ and $k \in \mathbb{N}$, does G contain an independent set of size k?
- Input size: $n = |V|$
- Certificate: k vertices
- Verifier: Check that there are no edges between pairs of vertices



Instance



Certificate

Examples

- Poll 1: Which of the following two problems is in **NP**?
  1. Given numbers $a_1, \ldots, a_n$ and $k \in \mathbb{N}$, is there a subset S such that $\sum_{i \in S} a_i = k$?
  2. Given a graph G and $k \in \mathbb{N}$, is the largest clique of size at most k?
  3. Both
  4. Neither

Examples

- Poll 2: Which of the following two problems is in **NP**?
  1. Given a graph G, does it not have a 2-coloring?
  2. Given a graph G, does it not have an Eulerian cycle?
  3. Both
  4. Neither

P vs. NP

- **Theorem: P $\subseteq$ NP**
- Proof:
  - Suppose $L \in \mathbf{P}$
  - Let A be a poly-time algorithm that decides L
  - The verifier V takes as input the instance x and an empty certificate y
  - $V(x, y)$ outputs $A(x)$ ■

P vs. NP

- We know that $\mathbf{P} \subseteq \mathbf{NP}$; does $\mathbf{P} = \mathbf{NP}$?
- If $\mathbf{P} = \mathbf{NP}$ then there would be an efficient algorithm for SUDOKU, 3-COLORING, CIRCUIT-SAT… Awesome!
- If $\mathbf{P} \neq \mathbf{NP}$ then there is some particular $L \in \mathbf{NP}$ such that $L \notin \mathbf{P}$; but maybe it is an obscure L?

The Cook-Levin Theorem

- Theorem (Cook 71, Levin 73): $\mathbf{P} = \mathbf{NP}$ if and only if Circuit-Sat $\in \mathbf{P}$
- In particular, if $\mathbf{P} \neq \mathbf{NP}$ then Circuit-Sat $\notin \mathbf{P}$
- In a sense, Circuit-Sat is the hardest problem in $\mathbf{NP}$

Reductions, revisited

- L has a polynomial-time reduction to L', denoted $L \leq_T^P L'$, if and only if it is possible to solve L in polynomial time using a polynomial-time algorithm for L'
- If $L \leq_T^P L'$ then:
  1. $L' \in \mathbf{P} \Rightarrow L \in \mathbf{P}$
  2. $L \notin \mathbf{P} \Rightarrow L' \notin \mathbf{P}$

The hardest problem(s)

- If Circuit-Sat is in **P** then all of **NP** is in **P**
- Last lecture: there is a poly-time reduction from Circuit-Sat to 3-Coloring

$\Rightarrow$ If 3-coloring is in **P** then Circuit-Sat is in **P**, and hence all of **NP** is in **P**

$\Rightarrow$ **P** = **NP** if and only if 3-coloring $\in$ **P**

The hardest problem(s)

- **Theorem (Yato-Seta 2002):** There is a poly-time reduction from 3-COLORING to SUDOKU

  ⇒ If SUDOKU is in **P** then 3-COLORING is in **P**, and hence all of **NP** is in **P**

  ⇒ **P** = **NP** if and only if SUDOKU ∈ **P**

Cook-Levin, revisited

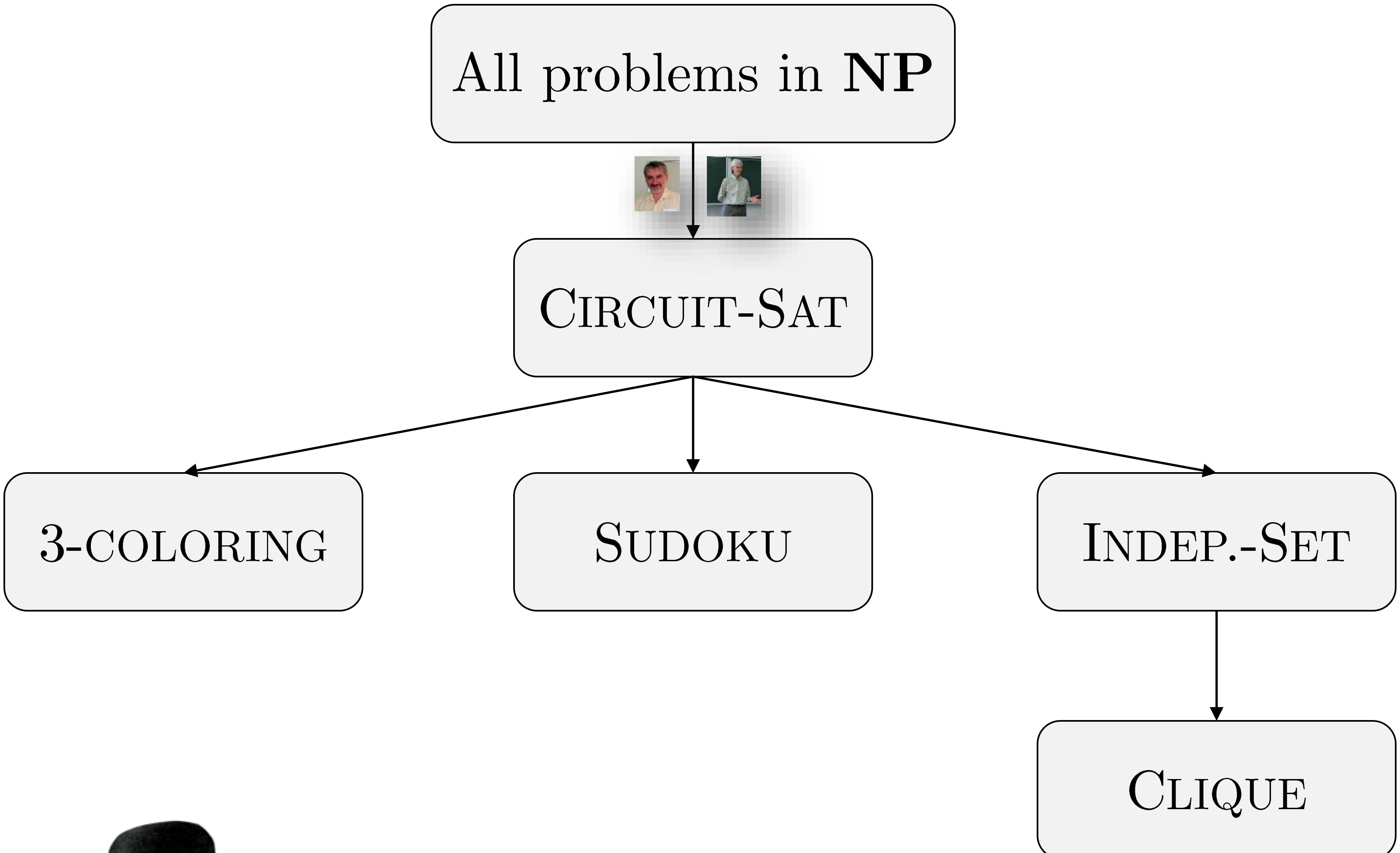
- Actual statement of Cook-Levin: Let $L \in$ **NP**, then there is a poly-time reduction from L to Circuit-Sat

$$\text{Circuit-Sat} \in \mathbf{P} \Rightarrow \mathbf{P} = \mathbf{NP}$$
$$\mathbf{P} = \mathbf{NP} \Rightarrow \text{Circuit-Sat} \in \mathbf{P}$$

NP-completeness

- L is **NP**-hard if every problem in **NP** has a polynomial time reduction to L
- L is **NP**-complete if $L \in$ **NP** and L is **NP**-hard
- To show that a problem is **NP**-complete:
  - Show that it is in **NP**
  - Show that a known **NP**-hard problem reduces to it

NP-COMPLETENESS

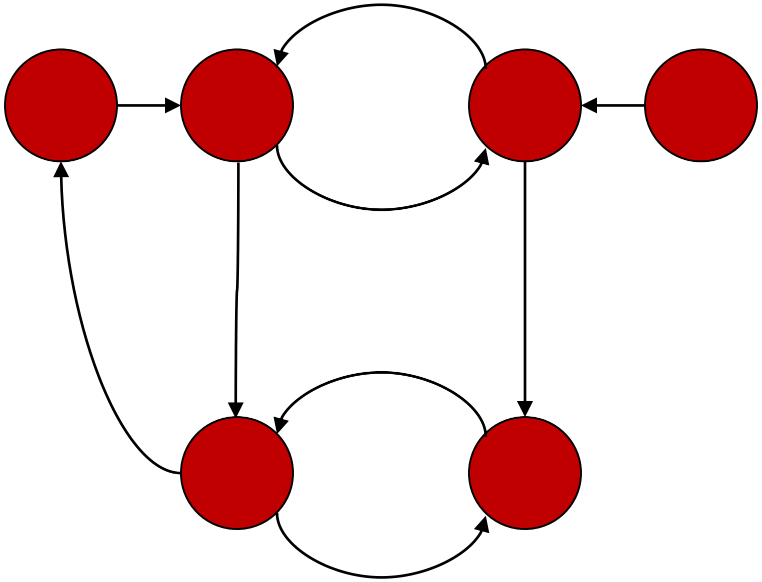


NP-complete problems

- Tens of thousands of problems are known to be **NP**-complete
- If even one of them has a poly-time algorithm then all of them are in **P**

NP-complete problems

- CYCLE-COVER: Given a directed graph and $L \in \mathbb{N}$, is there a collection of disjoint cycles of length $\leq L$ that covers $\geq k$ vertices?
- **Theorem:** CYCLE-COVER is **NP**-complete
- Relevant to kidney exchange



NP-complete problems

Artificial Intelligence 187–188 (2012) 31–51

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

On the approximability of Dodgson and Young elections ☆

Ioannis Caragiannis [a], Jason A. Covey [b], Michal Feldman [c,d], Christopher M. Homan [b], Christos Kaklamanis [a], Nikos Karanikolas [a], Ariel D. Procaccia [e,*], Jeffrey S. Rosenschein [f]

[a] Computer Technology Institute and Department of Computer Engineering and Informatics, University of Patras, 26504 Rio, Greece
[b] Department of Computer Science, Rochester Institute of Technology, 102 Lomb Memorial Drive, Rochester, NY 14623-5603, USA
[c] School of Business Administration and Center for the Study of Rationality, The Hebrew University of Jerusalem, Jerusalem 91904, Israel
[d] Microsoft Israel R&D Center, Israel
[e] School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA
[f] School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem 91904, Israel

ARTICLE INFO

Article history:
Received 18 January 2011
Received in revised form 8 April 2012
Accepted 10 April 2012
Available online 16 April 2012

Keywords:
Computational social choice
Approximation algorithms

ABSTRACT

The voting rules proposed by Dodgson and Young are both designed to find an alternative closest to being a Condorcet winner, according to two different notions of proximity; the score of a given alternative is known to be hard to compute under either rule. In this paper, we put forward two algorithms for approximating the Dodgson score: a combinatorial, greedy algorithm and an LP-based algorithm, both of which yield an approximation ratio of H_{m-1}, where m is the number of alternatives and H_{m-1} is the $(m-1)$st harmonic number. We also prove that our algorithms are optimal within a factor of 2, unless problems in $\mathcal{NP}$ have quasi-polynomial-time algorithms. Despite the intuitive appeal of the greedy algorithm, we argue that the LP-based algorithm has an advantage from a social choice point of view. Further, we demonstrate that computing any reasonable approximation of the ranking produced by Dodgson's rule is $\mathcal{NP}$-hard. This result provides a complexity-theoretic explanation of sharp discrepancies that have been observed in the social choice theory literature when comparing Dodgson elections with simpler voting rules. Finally, we show that the problem of calculating the Young score is $\mathcal{NP}$-hard to approximate by any factor. This leads to an inapproximability result for the Young ranking.

1. Introduction

The discipline of voting theory deals with the following setting: there is a group of n agents and each of them ranks a set of m alternatives; one alternative is to be elected. The big question is: which alternative best reflects the social good?

This question is fundamental to the study of multiagent systems, because the agents of such a system often need to combine their individual objectives into a single output or decision that best reflects the aggregate needs of all the agents in the system. For instance, web meta-search engines [12] and recommender systems [21] have used methods based on voting theory.

Reflecting on this question, the French philosopher and mathematician Marie Jean Antoine Nicolas de Caritat, marquis de Condorcet, suggested the following intuitive criterion: the winner should be an alternative that beats every other alternative

☆ A preliminary version of the results in this paper appeared in the *Proceedings of the 20th Annual ACM–SIAM Symposium on Discrete Algorithms* (*SODA'09*).

* Corresponding author.
E-mail addresses: caragian@ceid.upatras.gr (I. Caragiannis), jac8687@cs.rit.edu (J.A. Covey), mfeldman@huji.ac.il (M. Feldman), cmh@cs.rit.edu (C.M. Homan), kakl@ceid.upatras.gr (C. Kaklamanis), nkaranik@ceid.upatras.gr (N. Karanikolas), arielpro@seas.harvard.edu (A.D. Procaccia), jeff@cs.huji.ac.il (J.S. Rosenschein).

0004-3702/$ – see front matter
http://dx.doi.org/10.1016/j.artint.2012.04.004

Proof. Let $H \subseteq \mathcal{S}$ be a cover for $(U, \mathcal{S})$ with $|H| = K$. By the definition of a cover, H covers all elements of U. Hence, by pushing a^* to the first position in the preference of the critical agent ℓ^i such that $S_i \in H$, a^* will decrease its deficit with respect to each of the basic alternatives by 1, and hence it will become a Condorcet winner. The total number of positions a^* rises is at most $|H| \cdot (|Z| + n) = (1 + \zeta)nK$. □

Lemma 5.5. *If every cover of $(U, \mathcal{S})$ has size at least $\alpha K \ln n$, then a^* has Dodgson score at least $\alpha \zeta K n \ln n$.*

Proof. We first assume that the minimum number of positions a^* has to rise in order to beat the basic alternatives and become a Condorcet winner includes raising a^* by at least $|F|$ positions in the ranking of some indifferent agent r^i. Hence, a^* rises $|F|$ positions in the preference of r^i in order to reach position $|U \setminus S'_i| + 1$ and at least n additional positions in order to beat the basic alternatives. Its Dodgson score is thus at least $|F| + n \geqslant \alpha \zeta K n \ln n$.

Now, assume that the minimum number of positions a^* has to rise in order to beat the basic alternatives does not include raising a^* by at least $|F|$ positions in the ranking of some indifferent agent. We will show that if the Dodgson score of a^* is less than $\alpha \zeta K n \ln n$, then there exists a cover of $(U, \mathcal{S})$ of size less than $\alpha K \ln n$, contradicting the assumption of the lemma.

Let H be the set of critical agents in whose preferences a^* is pushed at least $|Z|$ positions higher. Over all the preference lists of all the agents in H, a^* rises a total of $|H| \cdot |Z|$ positions in order to reach position $|S_i| + 1$ in each list, plus at least n additional positions in order to decrease by 1 its deficit with respect to each of the alternatives in U. So, recalling $|Z| = \zeta n$, a^* rises at least $\zeta |H| n + n$ positions. Denoting the Dodgson score of a^* by $sc_D(a^*)$, we thus have $|H| \leqslant \frac{1}{\zeta n} sc_D(a^*) - \frac{1}{\zeta} < \alpha K \ln n$. The proof is completed by observing that the union of the sets S_i for each critical agent ℓ^i belonging to H contains all the basic alternatives, i.e., H corresponds to a cover for $(U, \mathcal{S})$ of size less than $\alpha K \ln n$. □

This completes the proof of Theorem 5.1. □

5.2. Inapproximability of Dodgson rankings

A question related to the approximability of Dodgson scores is the approximability of the Dodgson ranking, that is, the ranking of alternatives given by ordering them by nondecreasing Dodgson score. To the best of our knowledge, no rank aggregation function, which maps preference profiles to rankings of the alternatives, is known to provably produce rankings that are close to the Dodgson ranking [38,39,27–29] (see the survey of related work in Section 1).

Our next result establishes that efficient approximation algorithms for Dodgson ranking are unlikely to exist unless $\mathcal{P} = \mathcal{NP}$. It does so by proving that the problem of distinguishing between whether a given alternative is the unique Dodgson winner or in the last $O(\sqrt{m})$ positions is $\mathcal{NP}$-hard. This result provides a complexity-theoretic explanation for the sharp discrepancies observed in the Social Choice Theory literature when comparing Dodgson elections with simpler, efficiently computable, voting rules.

Theorem 5.6. *Given a preference profile with m alternatives and an alternative a^*, it is $\mathcal{NP}$-hard to decide whether a^* is a Dodgson winner or has rank at least $m - 6\sqrt{m}$ in any Dodgson ranking.*

Proof. We use a reduction from Minimum Vertex Cover in 3-regular graphs, and exploit a result concerning its inapproximability that follows from the work of Berman and Karpinski [3]. Our approach is similar to the proof of Theorem 5.1, albeit considerably more involved. We use the following result.

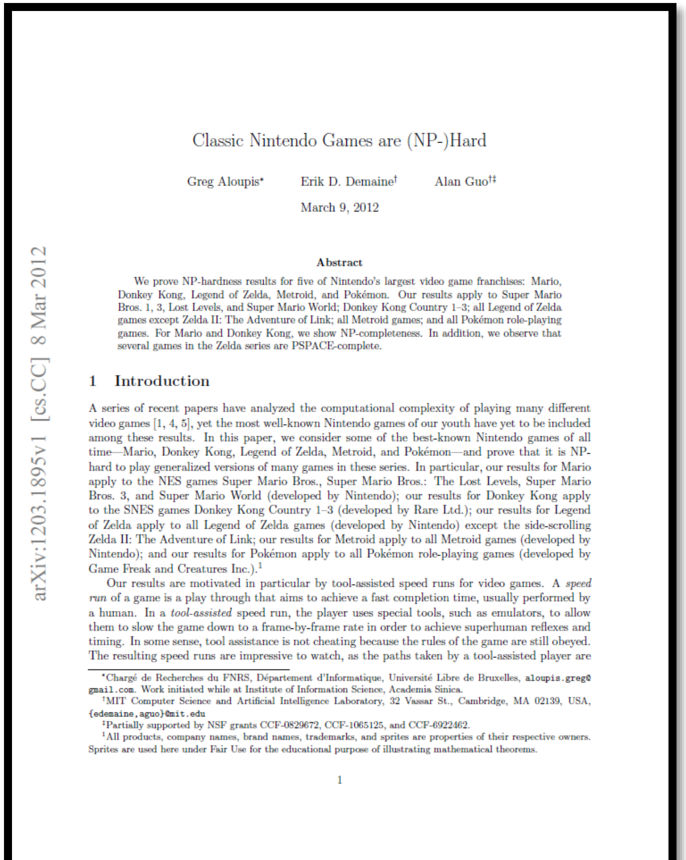
Theorem 5.7. *(See Berman and Karpinski [3], see also [25].) Given a 3-regular graph G with $n = 22t$ nodes for some integer $t > 0$ and an integer K in $[n/2, n-6]$, it is $\mathcal{NP}$-hard to distinguish between the following two cases:*

- *G has a vertex cover of size at most K.*
- *Any vertex cover of G has size at least $K + 6$.*

Given an instance of Minimum Vertex Cover consisting of a 3-regular graph G with $n = 22t$ nodes $v_0, v_1, \ldots, v_{n-1}$ and an integer $K \in [n/2, n-6]$, we construct in polynomial time a preference profile in which if we could distinguish whether a particular alternative is a Dodgson winner or not very far from the last position in any Dodgson ranking, then we could also distinguish between the two cases mentioned in Theorem 5.7 for the original Minimum Vertex Cover instance. See page 46 for an example of the construction. The Dodgson election has the following sets of alternatives:

- A special alternative a^*.
- A set F of $4Kn/11 + 3n/2$ alternatives. These alternatives are partitioned into n disjoint blocks $F_0, F_1, \ldots, F_{n-1}$ so that each block contains either $\lceil 4K/11 + 3/2 \rceil$ or $\lfloor 4K/11 + 3/2 \rfloor$ alternatives.
- A set A of n alternatives $a_0, a_1, \ldots, a_{n-1}$.

NP-complete problems



Classic Nintendo Games are (NP-)Hard

Greg Aloupis* Erik D. Demaine† Alan Guo†‡

March 9, 2012

Abstract

We prove NP-hardness results for five of Nintendo's largest video game franchises: Mario, Donkey Kong, Legend of Zelda, Metroid, and Pokémon. Our results apply to Super Mario Bros. 1, 3, Lost Levels, and Super Mario World; Donkey Kong Country 1–3; all Legend of Zelda games except Zelda II: The Adventure of Link; all Metroid games; and all Pokémon role-playing games. For Mario and Donkey Kong, we show NP-completeness. In addition, we observe that several games in the Zelda series are PSPACE-complete.

1 Introduction

A series of recent papers have analyzed the computational complexity of playing many different video games [1, 4, 5], yet the most well-known Nintendo games of our youth have yet to be included among these results. In this paper, we consider some of the best-known Nintendo games of all time—Mario, Donkey Kong, Legend of Zelda, Metroid, and Pokémon—and prove that it is NP-hard to play generalized versions of many games in these series. In particular, our results for Mario apply to the NES games Super Mario Bros., Super Mario Bros.: The Lost Levels, Super Mario Bros. 3, and Super Mario World (developed by Nintendo); our results for Donkey Kong apply to the SNES games Donkey Kong Country 1–3 (developed by Rare Ltd.); our results for Legend of Zelda apply to all Legend of Zelda games (developed by Nintendo) except the side-scrolling Zelda II: The Adventure of Link; our results for Metroid apply to all Metroid games (developed by Nintendo); and our results for Pokémon apply to all Pokémon role-playing games (developed by Game Freak and Creatures Inc.).[1]

Our results are motivated in particular by tool-assisted speed runs for video games. A *speed run* of a game is a play through that aims to achieve a fast completion time, usually performed by a human. In a *tool-assisted* speed run, the player uses special tools, such as emulators, to allow them to slow the game down to a frame-by-frame rate in order to achieve superhuman reflexes and timing. In some sense, tool assistance is not cheating because the rules of the game are still obeyed. The resulting speed runs are impressive to watch, as the paths taken by a tool-assisted player are

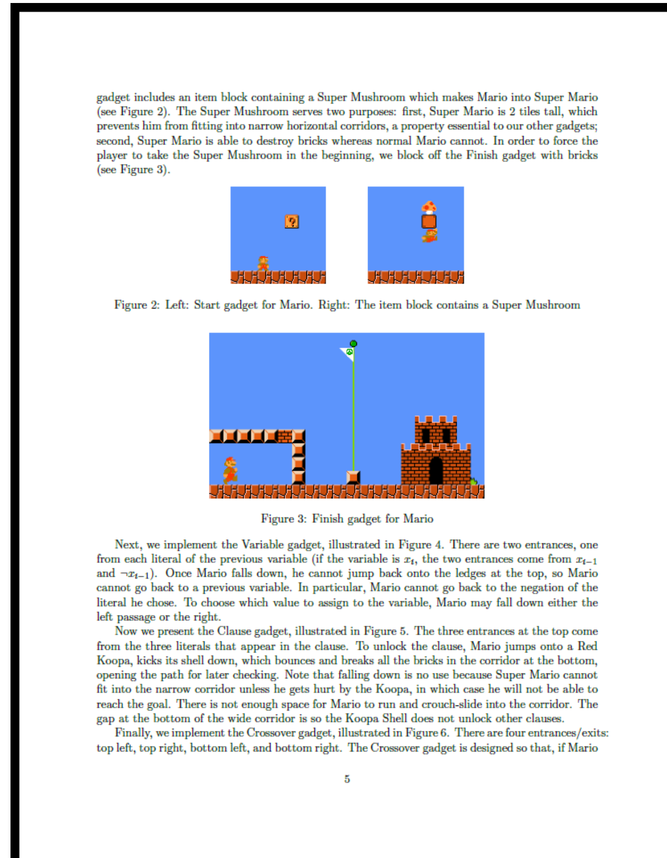
*Chargé de Recherches du FNRS, Département d'Informatique, Université Libre de Bruxelles, aloupis.greg@gmail.com. Work initiated while at Institute of Information Science, Academia Sinica.

†MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA, {edemaine,aguo}@mit.edu

‡Partially supported by NSF grants CCF-0829672, CCF-1065125, and CCF-6922462.

[1]All products, company names, brand names, trademarks, and sprites are properties of their respective owners. Sprites are used here under Fair Use for the educational purpose of illustrating mathematical theorems.

1



gadget includes an item block containing a Super Mushroom which makes Mario into Super Mario (see Figure 2). The Super Mushroom serves two purposes: first, Super Mario is 2 tiles tall, which prevents him from fitting into narrow horizontal corridors, a property essential to our other gadgets; second, Super Mario is able to destroy bricks whereas normal Mario cannot. In order to force the player to take the Super Mushroom in the beginning, we block off the Finish gadget with bricks (see Figure 3).

Figure 2: Left: Start gadget for Mario. Right: The item block contains a Super Mushroom

Figure 3: Finish gadget for Mario

Next, we implement the Variable gadget, illustrated in Figure 4. There are two entrances, one from each literal of the previous variable (if the variable is x_i, the two entrances come from x_{i-1} and $\neg x_{i-1}$). Once Mario falls down, he cannot jump back onto the ledges at the top, so Mario cannot go back to a previous variable. In particular, Mario cannot go back to the negation of the literal he chose. To choose which value to assign to the variable, Mario may fall down either the left passage or the right.

Now we present the Clause gadget, illustrated in Figure 5. The three entrances at the top come from the three literals that appear in the clause. To unlock the clause, Mario jumps onto a Red Koopa, kicks its shell down, which bounces and breaks all the bricks in the corridor at the bottom, opening the path for later checking. Note that falling down is no use because Super Mario cannot fit into the narrow corridor unless he gets hurt by the Koopa, in which case he will not be able to reach the goal. There is not enough space for Mario to run and crouch-slide into the corridor. The gap at the bottom of the wide corridor is so the Koopa Shell does not unlock other clauses.

Finally, we implement the Crossover gadget, illustrated in Figure 6. There are four entrances/exits: top left, top right, bottom left, and bottom right. The Crossover gadget is designed so that, if Mario

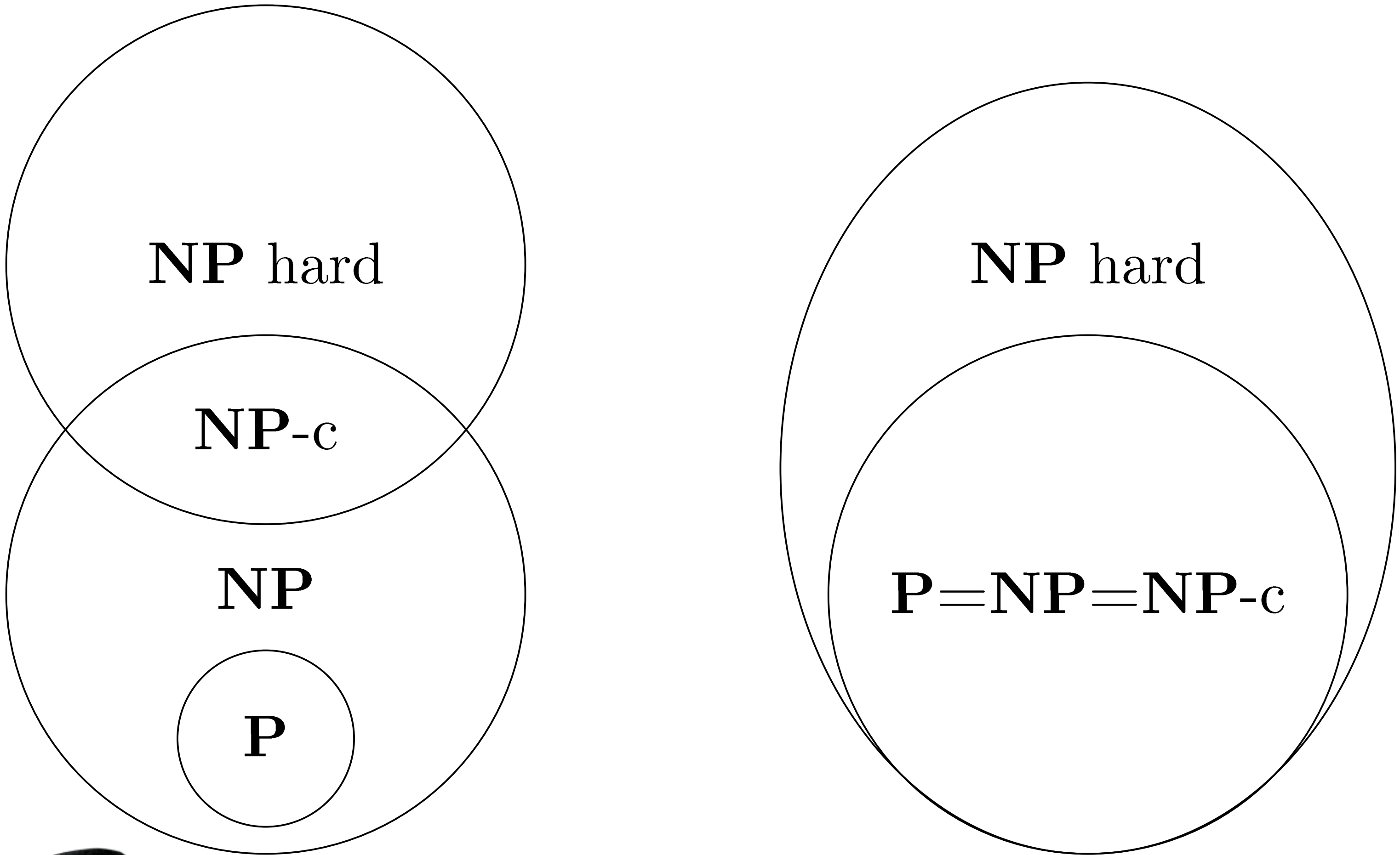
5

P vs. NP

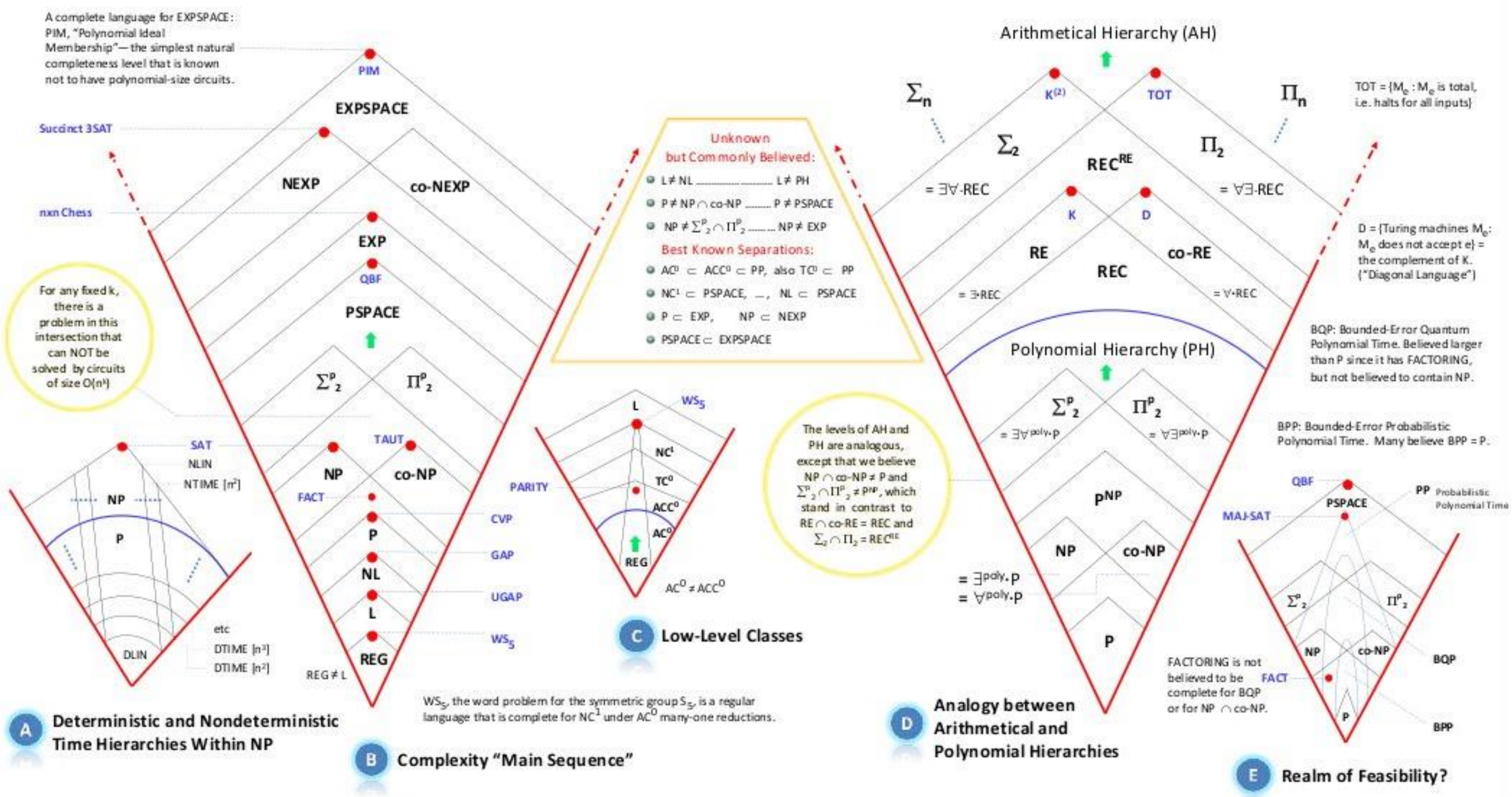
- So what do the experts think about the P vs. NP problem?
- Two polls from 2002 and 2012
  - 100 respondents in 2002
  - 152 respondents in 2012

Year	P≠NP	P=NP	Ind.	DC	BM
2002	61%	9%	4%	1%	22%
2012	83%	9%	3%	3%	1%

The two possible worlds



Complexity universe



What we have learned

- Definitions / facts
  - **P** and **NP**
  - Cook-Levin Theorem
  - **NP**-complete
- Principles:
  - Proving that problems are in **P**, **NP**, or **NP**-complete