CSCI 361 Lecture 20:

NP Complete Problems

Shikha Singh

Announcements & Logistics

- Hand **reading assignment # 14**
- Pick up **reading assignment # 15**
  - Partner and topic due by Thursday
  - Will discuss more today
- HW 6 grading feedback returned
  - Ask questions if the feedback doesn't make sense
- **HW due 8 tomorrow** 10 pm
  - Can work together **in pairs**
  - Turn in a joint write up on Gradescope

Last Time

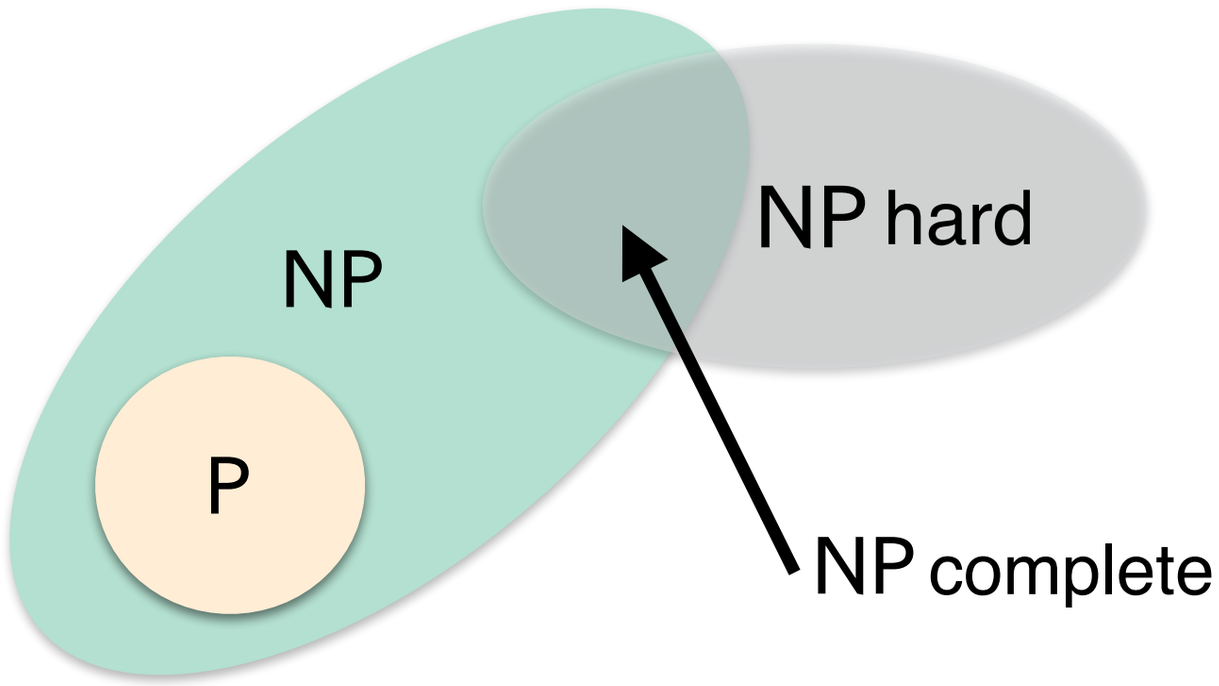
- Definitions:
  - Discussed P vs NP problem
  - Polynomial-time reductions and reduction from 3SAT
    - 3SAT $\leq_p$ CLIQUE

Today

- More practice with NP completeness reductions
  - 3SAT $\leq_p$ VertexCover
  - 3SAT $\leq_p$ HamiltonianCycle
  - 3SAT $\leq_p$ 3COLOR
- Won't get to all of these but reductions are available in the book and slides if you want to refer to them
  - Review of CSCI 256 reductions

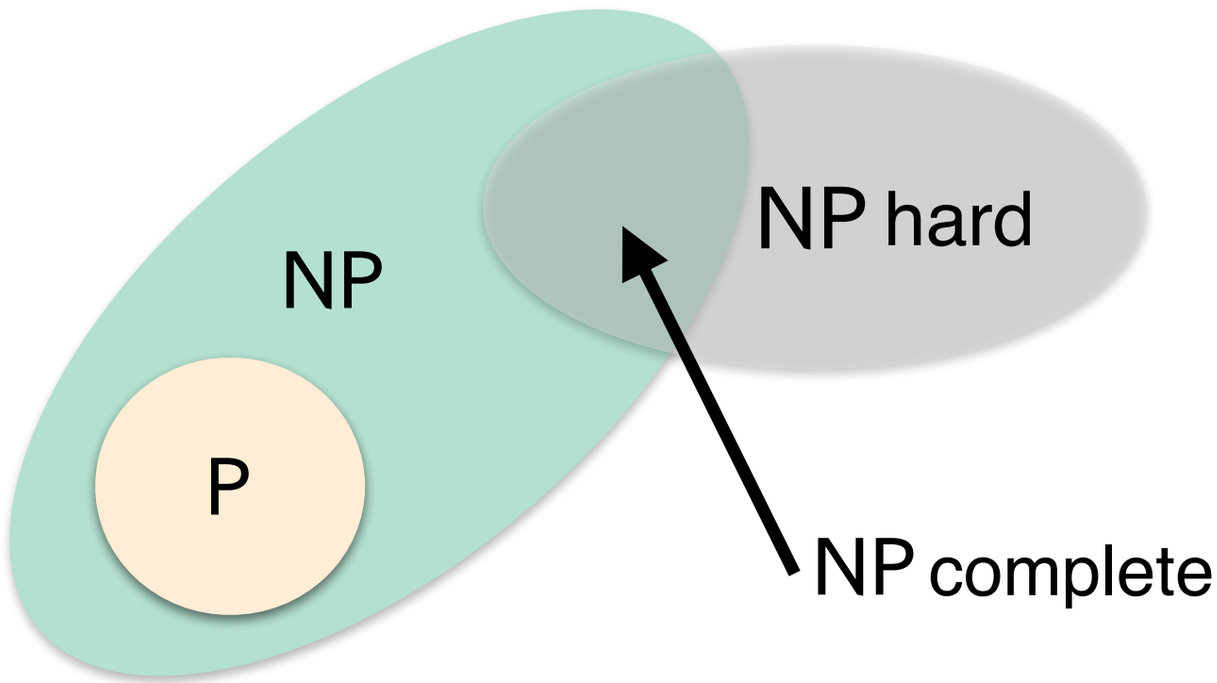
NP Hard and NP Complete

- **Definition. (NP Hard).** A language B is **NP hard** if every language in NP is polynomial-time reducible to B.
- **Definition. (NP Complete).** A language B is **NP complete** if $B \in$ NP **and** B is NP hard.



NP Hard and NP Complete

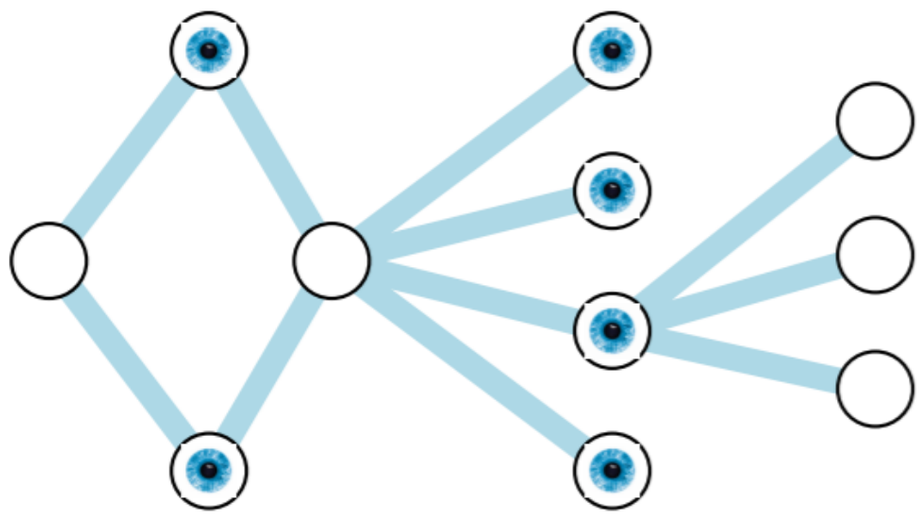
- **Cook-Levin Theorem** says **SAT/ 3SAT** is NP Complete.
    - Will discuss proof next lecture
- As 3SAT $\leq_p$ CLIQUE and CLIQUE is in NP, CLIQUE is also NP complete



Vertex Cover is NP Complete

- **Vertex cover** of a graph $G = (V, E)$ is a subset $C \subseteq V$ such that each edge $e \in E$ has at least one end point in C

VERTEX-COVER $= \{\langle G, k\rangle \mid G$ has a vertex cover of size $k\}$



A vertex cover of size 6

Vertex Cover is NP Complete

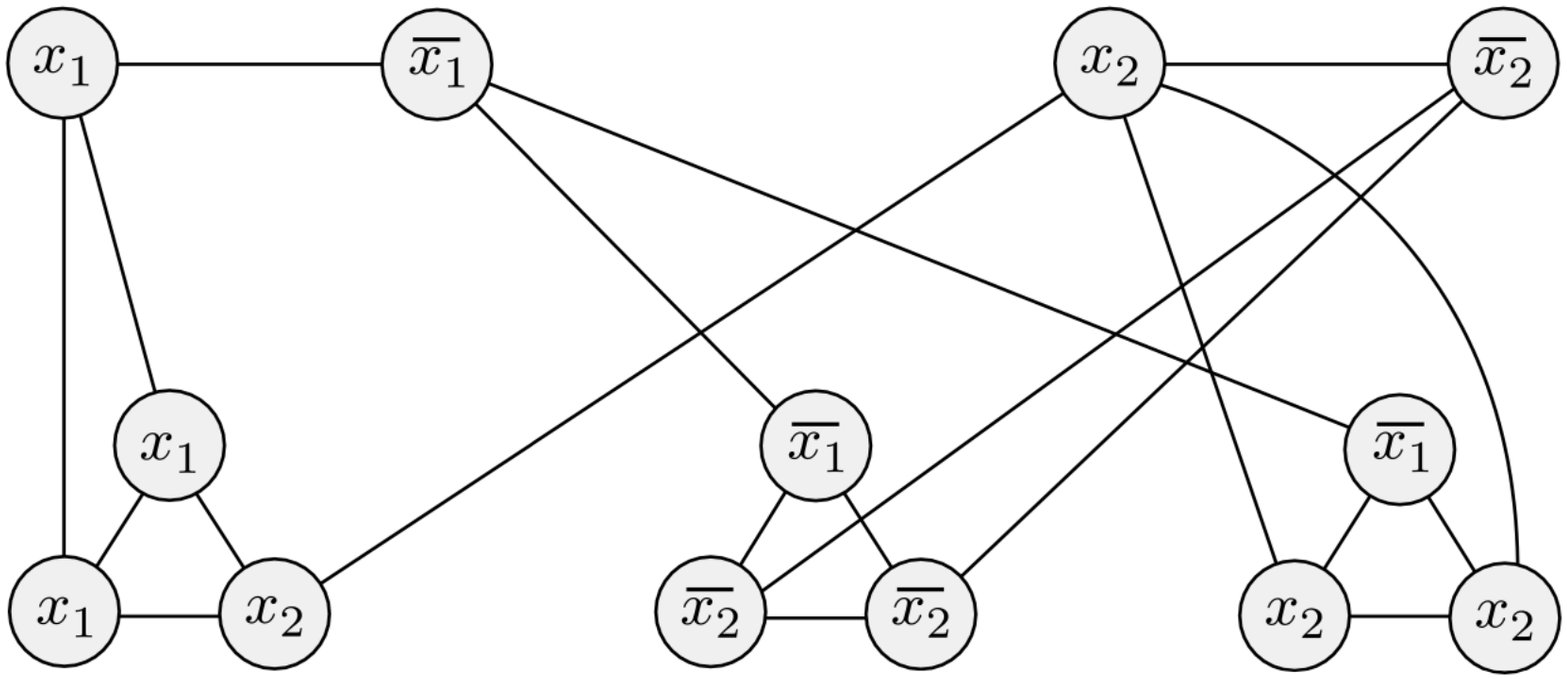
- To show vertex cover is NP complete, need to show:
    - Vertex Cover is in NP, why?
    - Show that a **known** NP hard problem reduces to it via a polynomial-time reduction
- **Strategy**: reduce from 3SAT using **variable and clause gadgets**

3SAT to Vertex Cover

- Given a 3SAT instance ϕ, create a graph $G = (V, E)$ as follows:
  - For each variable x in ϕ, create a node x and $\overline{x}$ and connect them with an edge (**variable gadget**)
  - For each clause c in ϕ, create a "triangle" of 3 nodes, each labeled with the corresponding literal and connected to each other and to the nodes in the variable gadget with the identical label (**clause gadget**)
- Instance of VC is $\langle G, k \rangle$ so we need to set k
  - Easier to "backward engineer" value of k after thinking through the reduction

3SAT to Vertex Cover

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



- If ϕ has m variables and ℓ clauses, what is the # of vertices & edges in G?

Correctness of Reduction

- Set $k = m + 2\ell$
- If the instance ϕ is satisfiable, what is the k-vertex cover in G?
- If there is a k-vertex cover in G, then why is ϕ satisfiable?

- **Takeaway.** Vertex Cover is also NP complete.

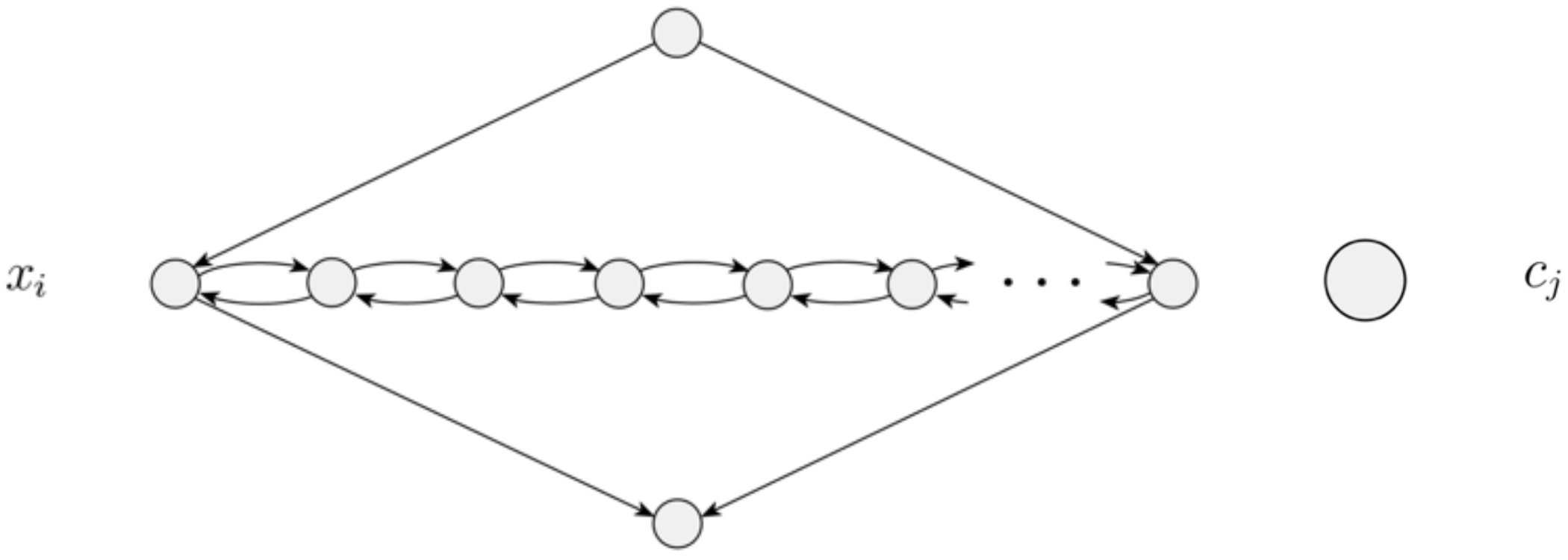
3SAT $\leq_p$ HAMPATH

3SAT $\leq_p$ HAMPATH

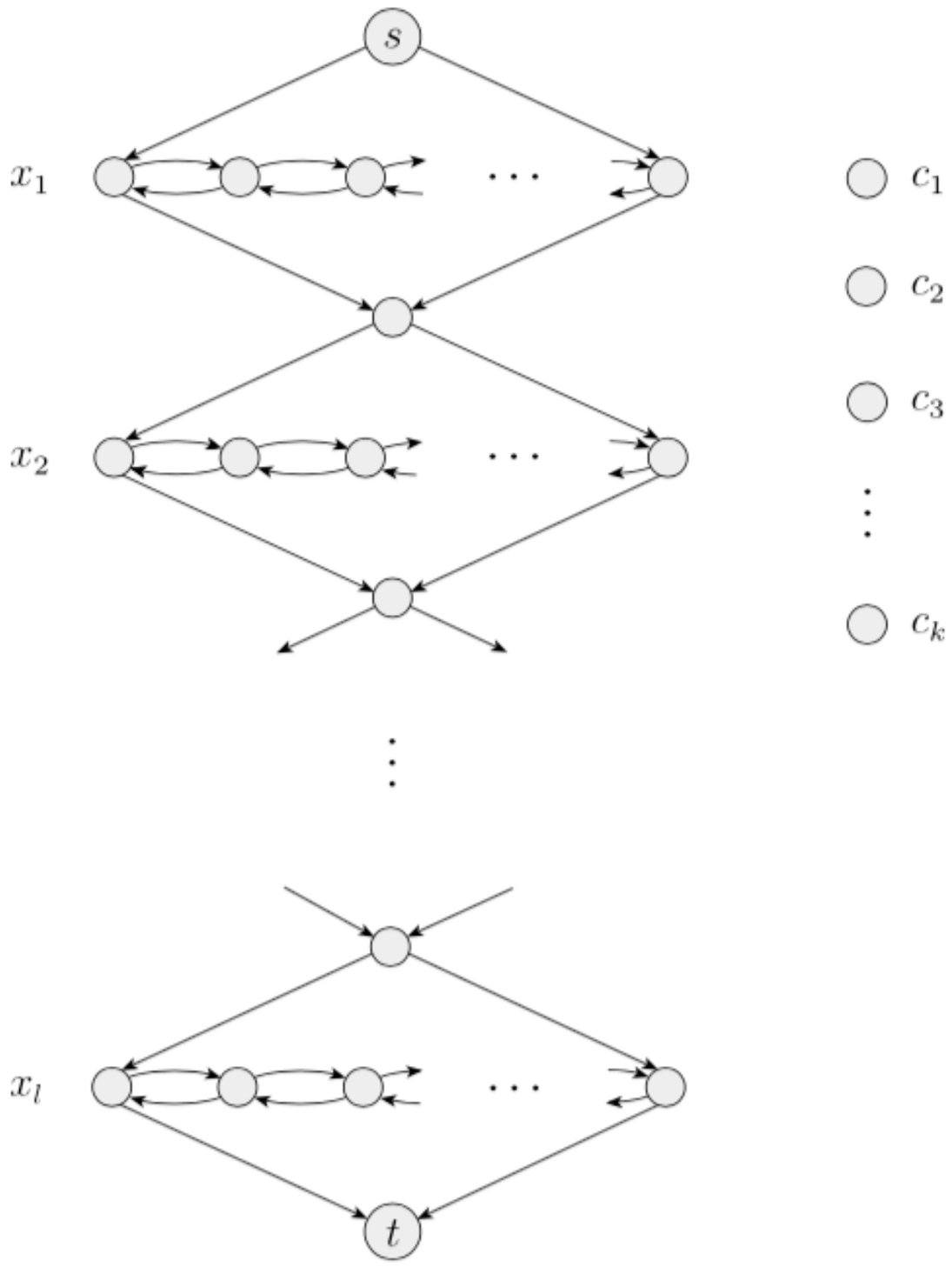
- Given 3SAT instance Φ, transform it to directed graph G with nodes s, t such that Φ is satisfiable iff G has a hamiltonian path from s to t (a path that visits each node exactly once)
- Essential ingredients of a input assignments of Φ
  - Each variable can be set to true or false (need to encode these settings in the graph in our variable gadget)
  - For a clause to be satisfied at least one literal is set to true
  - Need to hook up clause and variable gadgets to ensure iff correctness

3SAT $\leq_p$ HAMPATH

- Let Φ contain k clauses and ℓ variables
- Let $x_1, \ldots, x_\ell$ denote the ℓ variable in Φ
- Variable gadget: for each variable x_i create a diamond shape structure with a horizontal row of nodes
- Clause gadget: for each clause c_j we create a single node



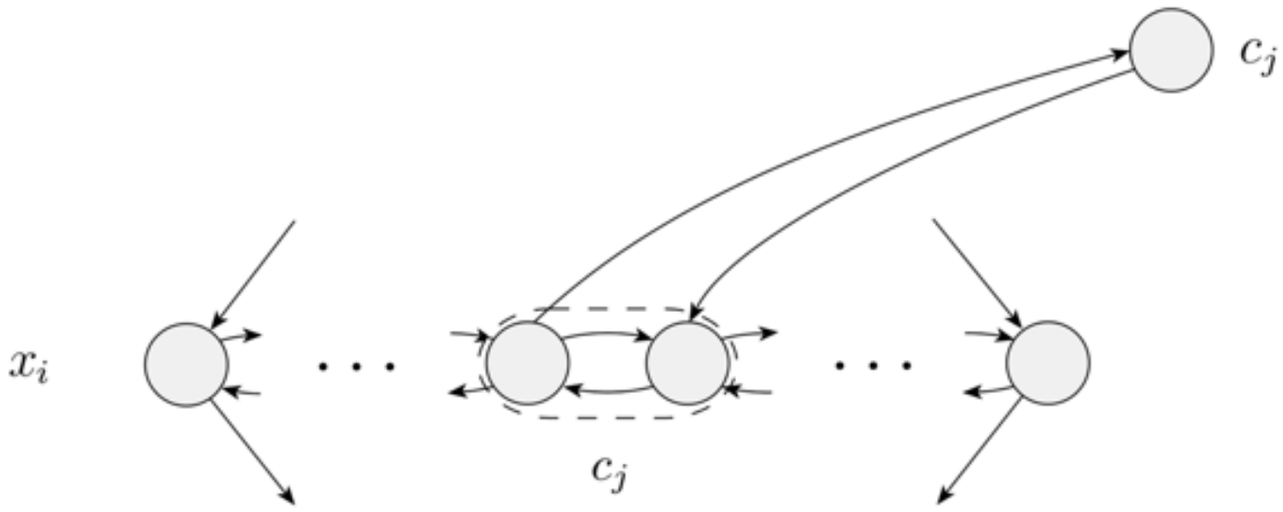
3SAT $\leq_p$ HAMPATH



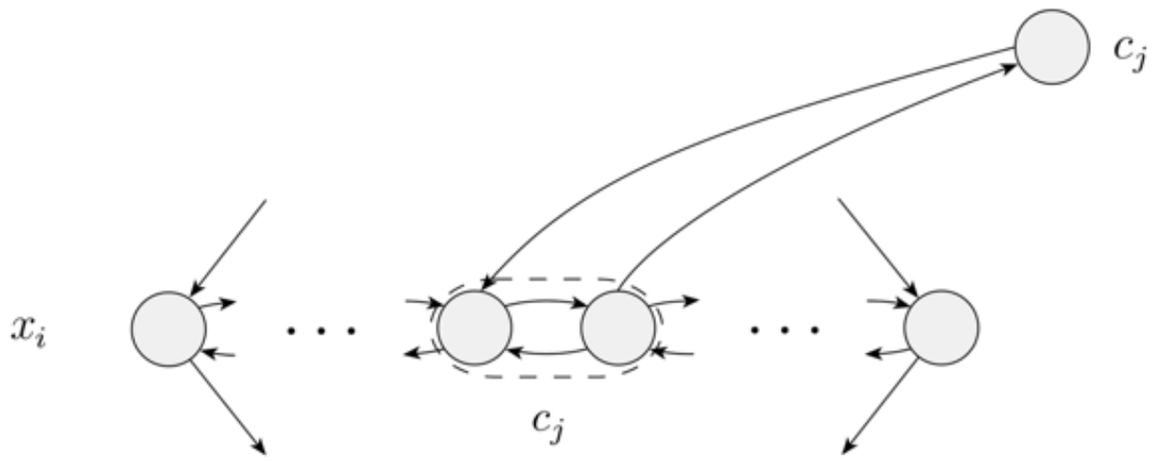
3SAT $\leq_p$ HAMPATH

Connecting the variable and clause gadgets.

- If x_i appears in c_j, connect jth pair in the ith diamond to the jth clause: connect in a zig-zag fashion (left)
- If $\overline{x_i}$ appears in c_j, connect it in a zag-zig fashion (left)



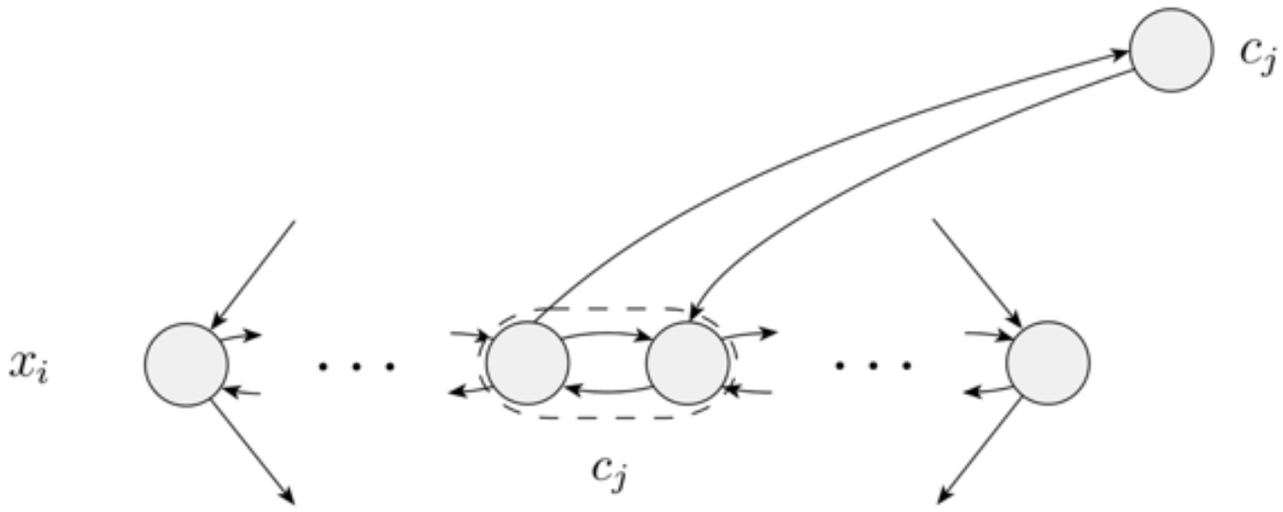
Zig-Zag



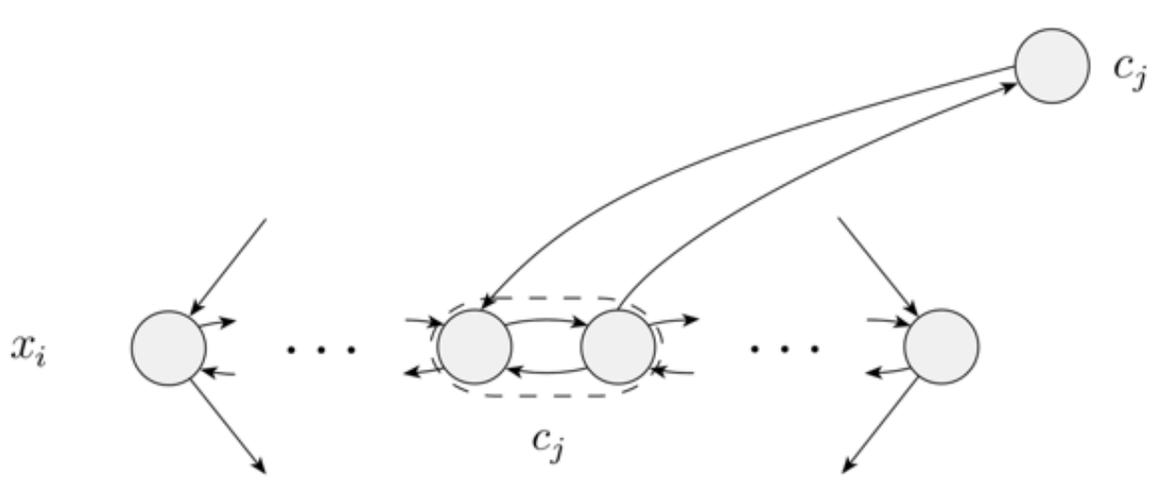
Zag-Zig

3SAT $\leq_p$ HAMPATH

- If x_i is true, the Hamiltonian cycle will visit c_j in a zig-zag way
- Else if $\overline{x_i}$ is true, the Hamiltonian cycle will visit c_j in a zag-zig way
- Let's us map cycle traversal order to true/false assignments



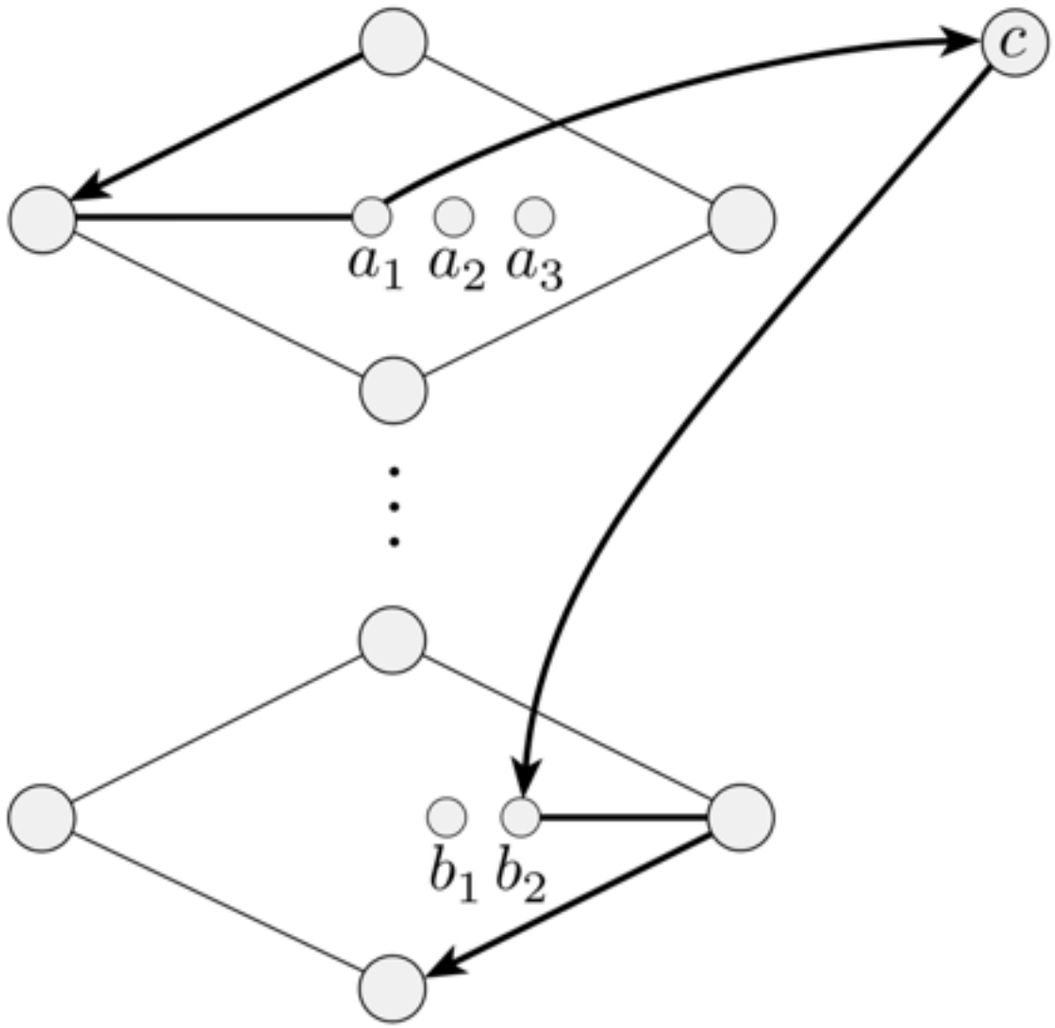
Zig-Zag



Zag-Zig

3SAT $\leq_p$ HAMPATH

- Situation that cannot occur in a Hamiltonian cycle of G: clause entered from one diamond but exited to a different



Such a cycle would never visit node a_2

Survey Paper Discussion

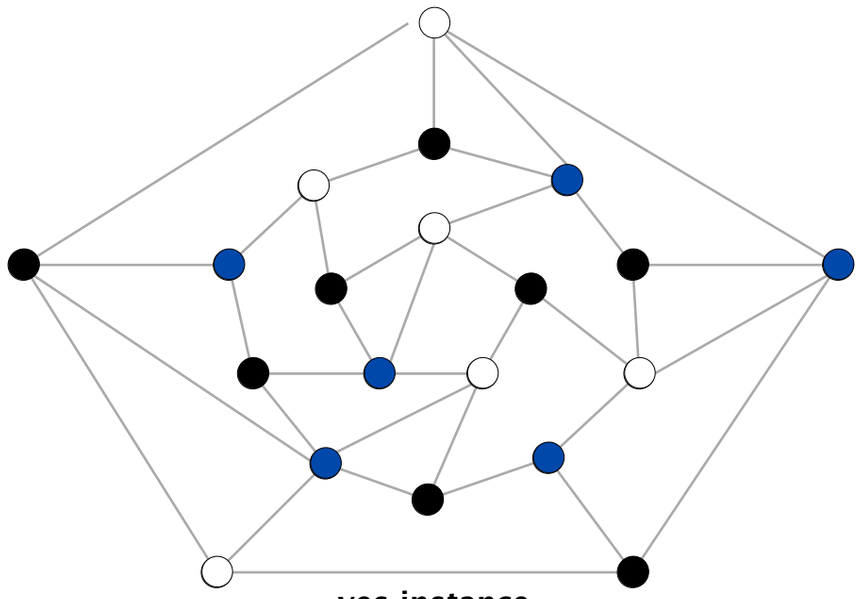
- Read this guide
- Hand in partner and ***topic***: **Nov 21 (Thurs) in class**
- 1 page draft of background due **Nov 26 (Tues) in class**
- Class presentation: Dec 5 and short paper (3 pages) due **Dec 6**

Graph-3-Color is NP Complete:

3-SAT $\leq_p$ Graph 3-Color

Graph 3-Color Problem

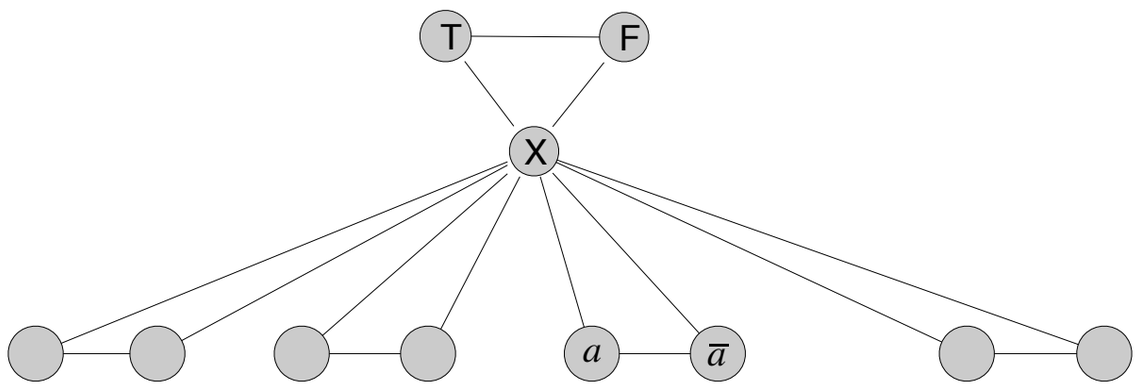
- 3-COLOR. Given an undirected graph $G = (V, E)$, is it possible to color the vertices with 3 colors s.t. no adjacent nodes have the same color.
- We have shown in the past that 3-COLOR $\in$ **NP**.



yes instance

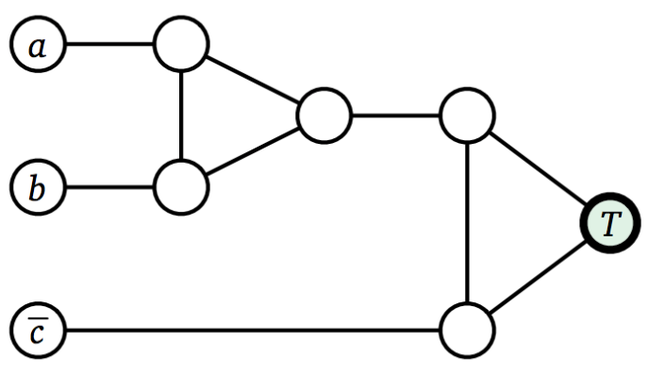
3-SAT $\leq_p$ 3-COLOR

- **Proof.** Given a 3-SAT instance Φ, define G as follows
  - Truth gadget: a triangle with three nodes $T, F,$ and X (for true, false and other) — they must get different colors (say true, false, other)
  - Variable gadget: a triangle made up of variable a, its negation $\overline{a}$ and the X node of the truth gadget — enforces $a, \overline{a}$ are colored true/false



3-SAT $\leq_p$ 3-COLOR

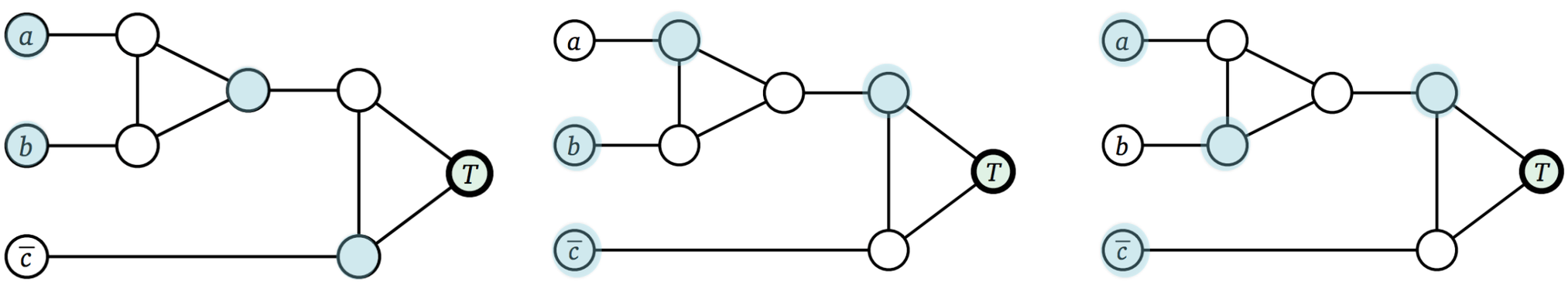
- Given a **3-SAT** instance Φ, define G as follows
  - Truth gadget: a triangle with three nodes $T, F,$ and X (for true, false and other) — they must get different colors (say true, false, other)
  - Variable gadget: triangle made up of variable a, its negation $\overline{a}$ and the X node of the truth gadget — enforces $a, \overline{a}$ are colored true/ false
  - Clause gadget: joins three literal nodes (from the variable gadget) to node T in the truth gadget using a subgraph as shown below



$(a \vee b \vee \overline{c})$

3-SAT $\leq_p$ 3-COLOR

- Observation. A clause gadget has a valid 3-coloring, if and only if, at least one literal is colored True
    - If a, b (or $b, \overline{c}$) or ($a, \overline{c}$) get the same color (say, FALSE) then the right-end-point of the triangle must be colored the same (shown in blue)
    - The remaining literal must be colored True!

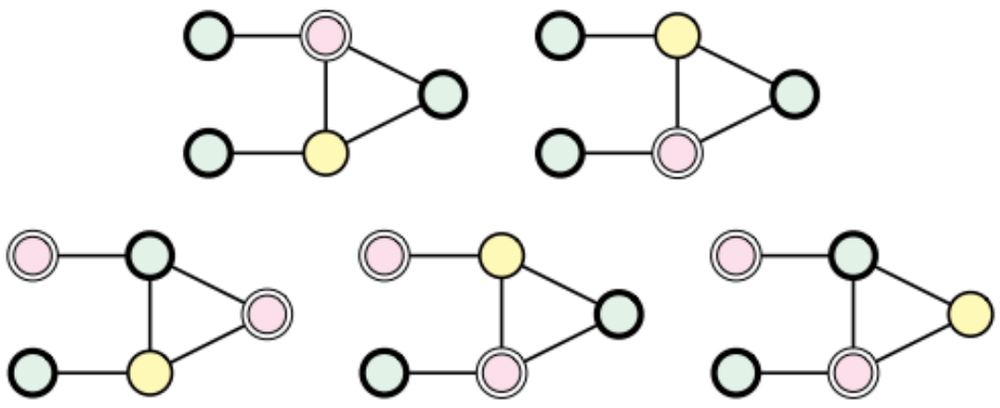


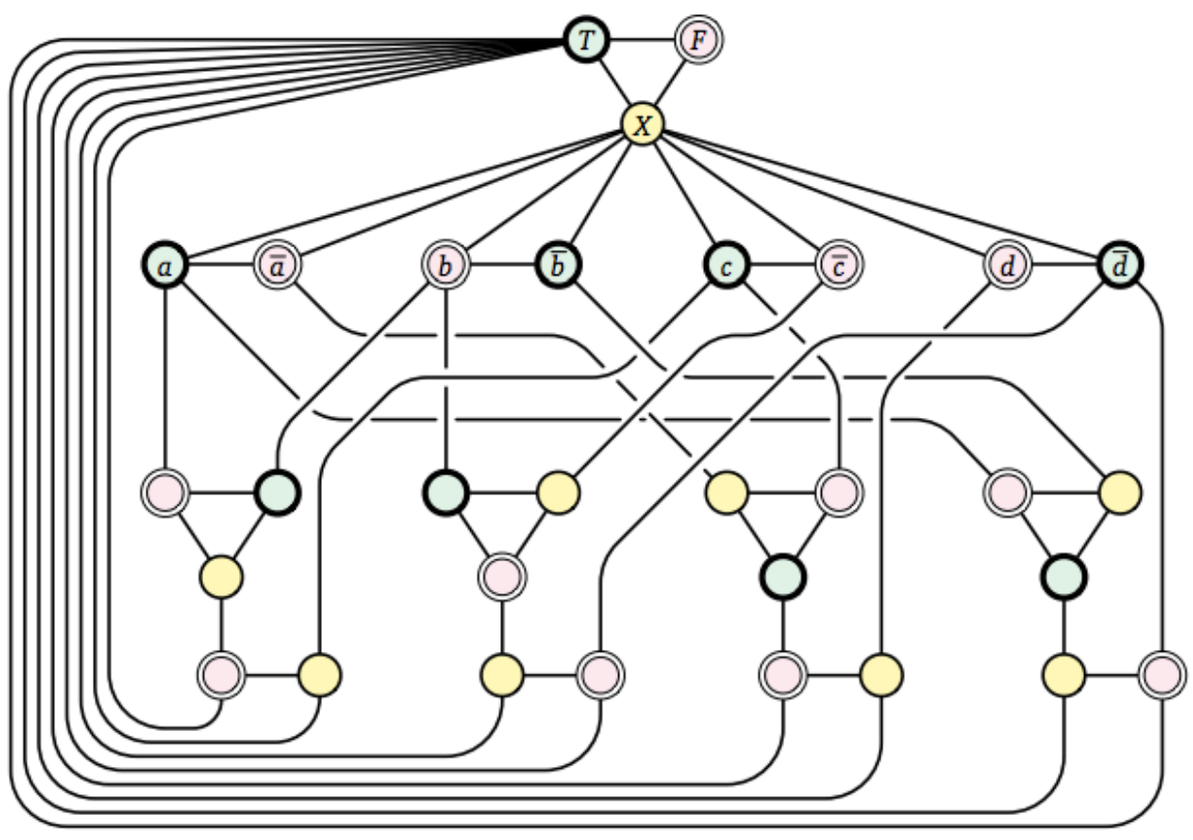
$(a \vee b \vee \overline{c})$

3-SAT $\leq_p$ 3-COLOR

- Example. All valid 3-colorings of the "half-gadget" of the clause on the left
- Overall G for example instance on the right

$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$





3-SAT $\leq_p$ 3-COLOR

Correctness of reduction

- ($\Rightarrow$) If Φ is satisfiable, color the variable gadget based on the satisfying assignment
- Why can we extend this coloring to a valid 3-coloring of the clause gadgets?
    - Because at least one literal in each clause must be True
- ($\Leftarrow$) If G is 3-colorable, then the variable nodes must be colored T or F (because of the variable gadget), we can assign truth values based on the colors
- Why is this a satisfying assignment?
    - At least one of the literals in each clause must be colored true and thus the resulting assignment must satisfy Φ

3-SAT $\leq_p$ 3-COLOR

Running time of reduction

- If ϕ has n variables and k clauses, then our resulting graph G has at most $2n + 5k + 3$ nodes
- Thus, we can construct G in $O(n + k)$ time
- Reduction is polynomial time!

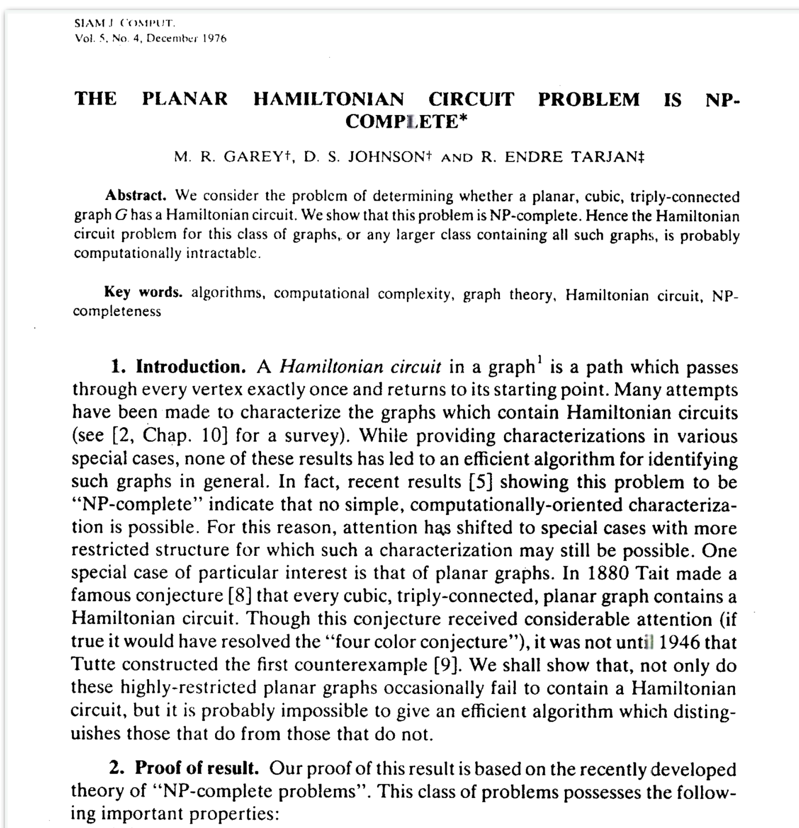
List of NPC Problems

- Satisfiability: SAT/ 3-SAT
- INDEPENDENT SET and CLIQUE
- Covering problems: VERTEX COVER, SET COVER
- Coloring problem: 3-COLOR
- Sequencing problems:
  - Traveling salesman problem
  - Hamiltonian cycle / path
- Packing problems: Subset-Sum, Knapsack (next time)

Have not shown all but similar reductions

Fun Facts

- Hamiltonian path problem says NP complete even on very simple graph: two connected, cubic and planar graphs!
- Still NP complete on general grid graphs, but poly-time solvable on “solid grid graphs” (a Williams undergrad thesis by Chris Umans)



SIAM J. COMPUT.
Vol. 5, No. 4, December 1976

THE PLANAR HAMILTONIAN CIRCUIT PROBLEM IS NP-COMPLETE*

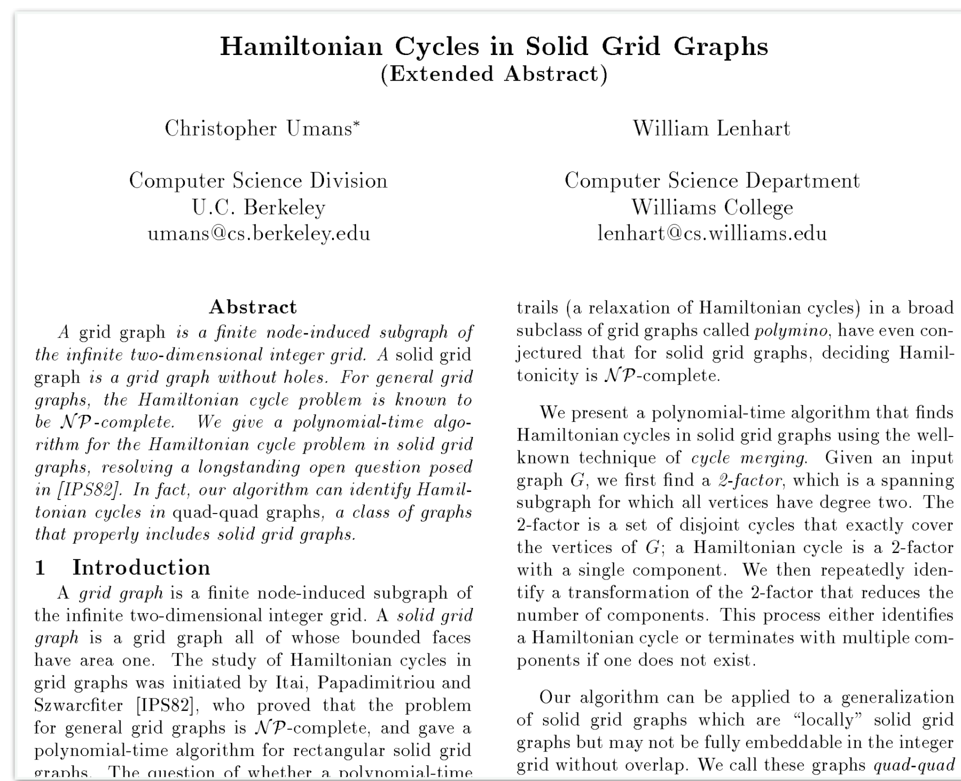
M. R. GAREY†, D. S. JOHNSON† AND R. ENDRE TARJAN‡

Abstract. We consider the problem of determining whether a planar, cubic, triply-connected graph *G* has a Hamiltonian circuit. We show that this problem is NP-complete. Hence the Hamiltonian circuit problem for this class of graphs, or any larger class containing all such graphs, is probably computationally intractable.

Key words. algorithms, computational complexity, graph theory, Hamiltonian circuit, NP-completeness

1. Introduction. A *Hamiltonian circuit* in a graph[1] is a path which passes through every vertex exactly once and returns to its starting point. Many attempts have been made to characterize the graphs which contain Hamiltonian circuits (see [2, Chap. 10] for a survey). While providing characterizations in various special cases, none of these results has led to an efficient algorithm for identifying such graphs in general. In fact, recent results [5] showing this problem to be “NP-complete” indicate that no simple, computationally-oriented characterization is possible. For this reason, attention has shifted to special cases with more restricted structure for which such a characterization may still be possible. One special case of particular interest is that of planar graphs. In 1880 Tait made a famous conjecture [8] that every cubic, triply-connected, planar graph contains a Hamiltonian circuit. Though this conjecture received considerable attention (if true it would have resolved the “four color conjecture”), it was not until 1946 that Tutte constructed the first counterexample [9]. We shall show that, not only do these highly-restricted planar graphs occasionally fail to contain a Hamiltonian circuit, but it is probably impossible to give an efficient algorithm which distinguishes those that do from those that do not.

2. Proof of result. Our proof of this result is based on the recently developed theory of “NP-complete problems”. This class of problems possesses the following important properties:



Hamiltonian Cycles in Solid Grid Graphs
(Extended Abstract)

Christopher Umans*
Computer Science Division
U.C. Berkeley
umans@cs.berkeley.edu

William Lenhart
Computer Science Department
Williams College
lenhart@cs.williams.edu

Abstract

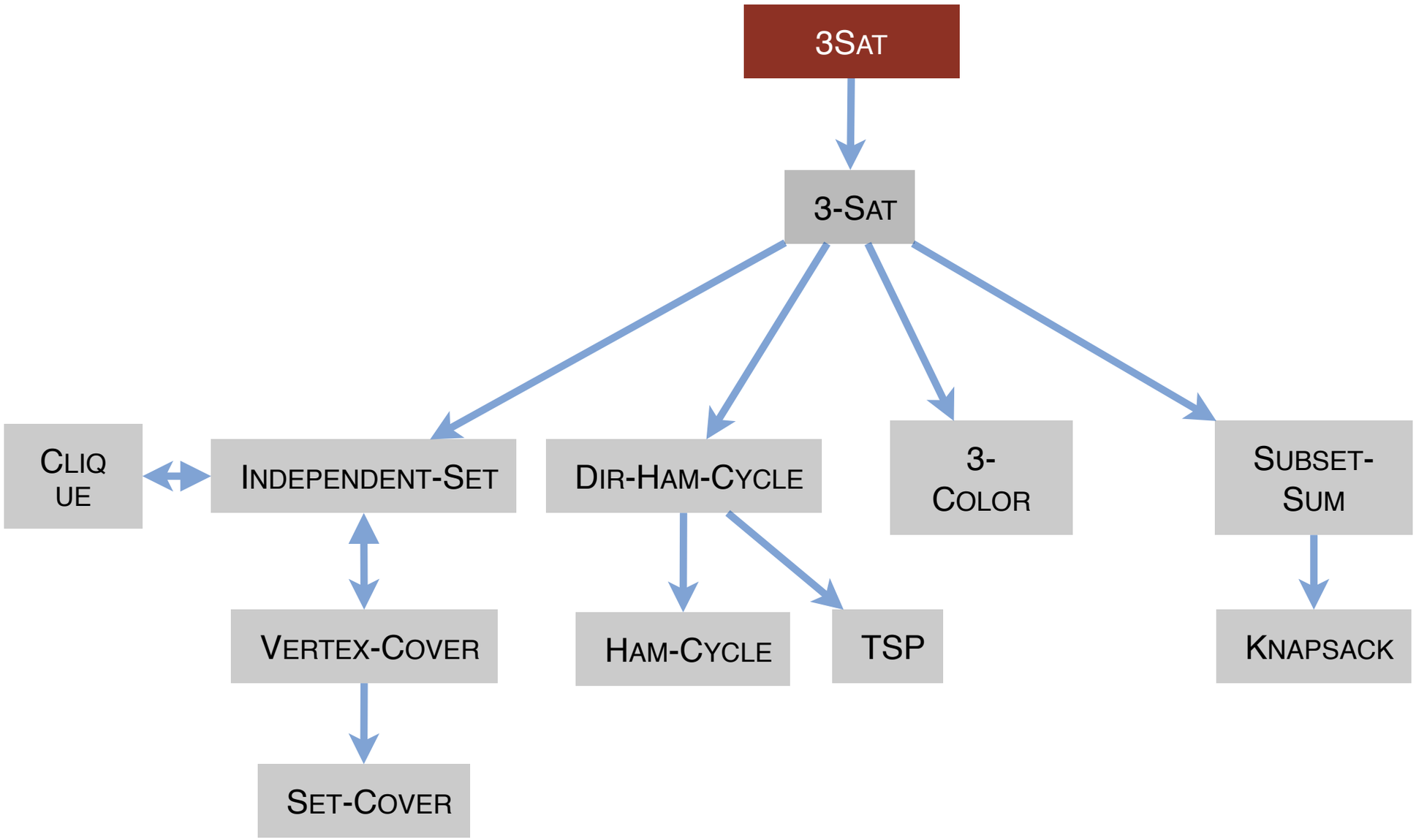
A grid graph is a finite node-induced subgraph of the infinite two-dimensional integer grid. A solid grid graph *is a grid graph without holes. For general grid graphs, the Hamiltonian cycle problem is known to be $\mathcal{NP}$-complete. We give a polynomial-time algorithm for the Hamiltonian cycle problem in solid grid graphs, resolving a longstanding open question posed in [IPS82]. In fact, our algorithm can identify Hamiltonian cycles in* quad-quad graphs, *a class of graphs that properly includes solid grid graphs.*

1 Introduction

A *grid graph* is a finite node-induced subgraph of the infinite two-dimensional integer grid. A *solid grid graph* is a grid graph all of whose bounded faces have area one. The study of Hamiltonian cycles in grid graphs was initiated by Itai, Papadimitriou and Szwarcfiter [IPS82], who proved that the problem for general grid graphs is $\mathcal{NP}$-complete, and gave a polynomial-time algorithm for rectangular solid grid graphs. The question of whether a polynomial-time trails (a relaxation of Hamiltonian cycles) in a broad subclass of grid graphs called *polymino*, have even conjectured that for solid grid graphs, deciding Hamiltonicity is $\mathcal{NP}$-complete.

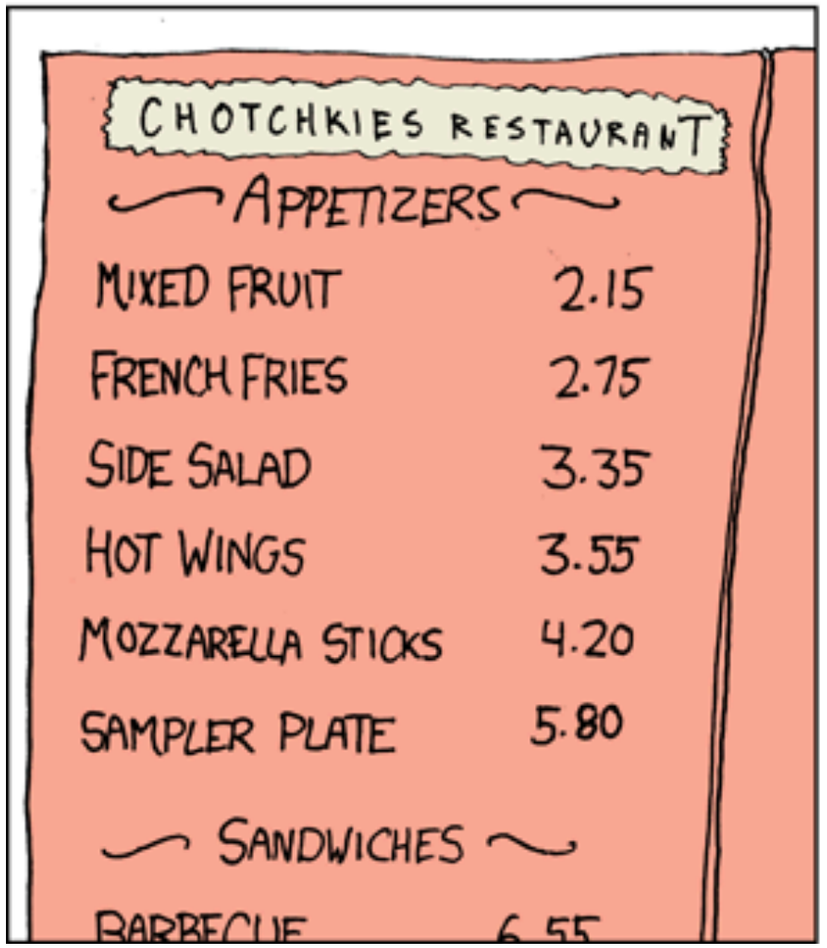
We present a polynomial-time algorithm that finds Hamiltonian cycles in solid grid graphs using the well-known technique of *cycle merging*. Given an input graph G, we first find a *2-factor*, which is a spanning subgraph for which all vertices have degree two. The 2-factor is a set of disjoint cycles that exactly cover the vertices of G; a Hamiltonian cycle is a 2-factor with a single component. We then repeatedly identify a transformation of the 2-factor that reduces the number of components. This process either identifies a Hamiltonian cycle or terminates with multiple components if one does not exist.

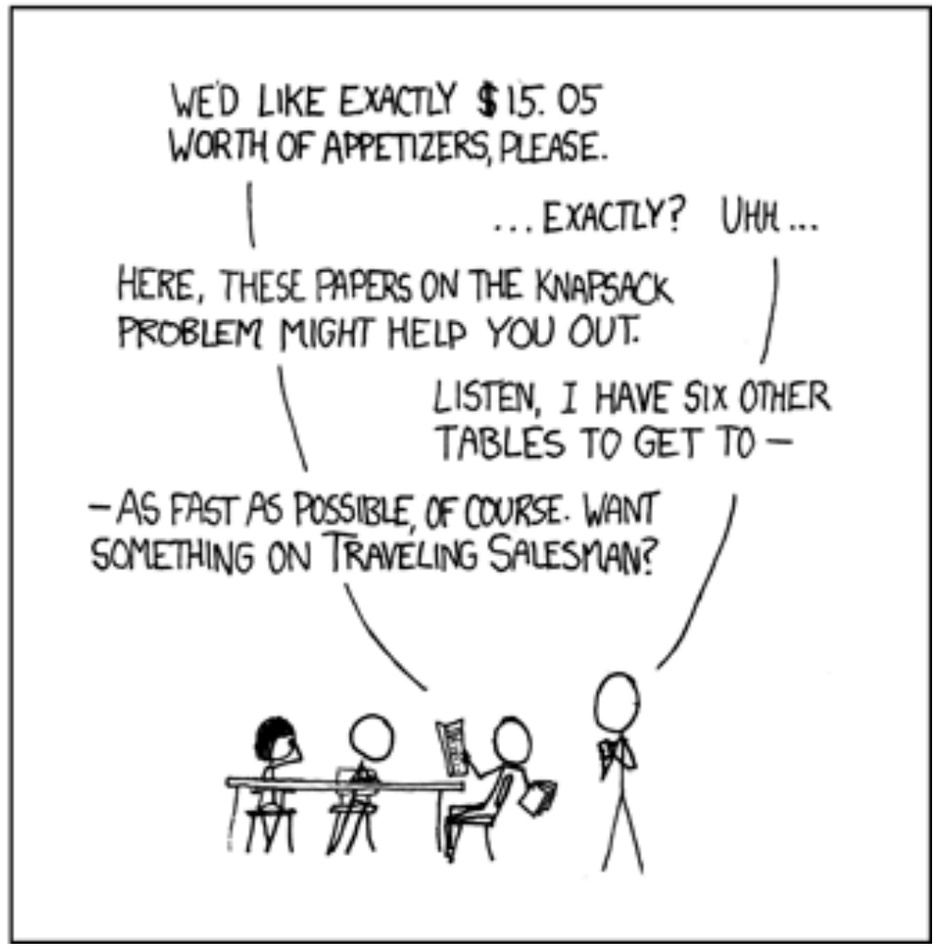
Our algorithm can be applied to a generalization of solid grid graphs which are “locally” solid grid graphs but may not be fully embeddable in the integer grid without overlap. We call these graphs *quad-quad*


3SAT
3-SAT
CLIQUE
INDEPENDENT-SET
DIR-HAM-CYCLE
3-COLOR
SUBSET-SUM
VERTEX-COVER
HAM-CYCLE
TSP
KNAPSACK
SET-COVER

MY HOBBY:

EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS





From: https://xkcd.com/287/

Useful NP-hard Problems

- **BIN-PACKING.** Given a set of items $I = \{1, \ldots, n\}$ where item i has size $s_i \in (0,1]$, bins of capacity c, find an assignment of items to bins that minimizes the number of bins used?
- **PARTITION.** Given a set S of n integers, are there subsets A and B such that $A \cup B = S$, $A \cap B = \emptyset$ and $\sum_{a \in A} a = \sum_{b \in B} b$

- **MAXCUT.** Given an undirected graph $G = (V, E)$, find a subset $S \subset V$ that maximizes the number of edges with exactly one endpoint in S.
- **MAX-2-SAT.** Given a Boolean formula in CNF, with exactly two literals per clause, find a variable assignment that maximizes the number of clauses with at least one true literal. (**2-SAT** on the other hand is in **P**)
- **3D-MATCHING**. Given n instructors, n courses, and n times, and a list of the possible courses and times each instructor is willing to teach, is it possible to make an assignment so that all courses are taught at different times?

Many More hard computational problems

Aerospace engineering. Optimal mesh partitioning for finite elements.

Biology. Phylogeny reconstruction.

Chemical engineering. Heat exchanger network synthesis.

Chemistry. Protein folding.

Civil engineering. Equilibrium of urban traffic flow.

Economics. Computation of arbitrage in financial markets with friction.

Electrical engineering. VLSI layout.

Environmental engineering. Optimal placement of contaminant sensors.

Financial engineering. Minimum risk portfolio of given return.

Game theory. Nash equilibrium that maximizes social welfare.

Mathematics. Given integer $a_1, \ldots, a_n$, compute

Mechanical engineering. Structure of turbulence in sheared flows.

Medicine. Reconstructing 3d shape from biplane angiocardiogram.

Operations research. Traveling salesperson problem.

Physics. Partition function of 3d Ising model.

Politics. Shapley–Shubik voting power.

Recreation. Versions of Sudoku, Checkers, Minesweeper, Tetris, Rubik's Cube.

Fun NP-hard Games

- **MINESWEEPER** (from CIRCUIT-SAT)
- **SODUKO** (from 3-SAT)
- **TETRIS** (from 3PARTITION)
- **SOLITAIRE** (from 3PARTITION)
- **SUPER MARIO BROTHERS** (from 3-SAT)
- **CANDY CRUSH SAGA** (from 3-SAT variant)
- **PAC-MAN** (from Hamiltonian Cycle)
- **RUBIC's CUBE** (recent 2017 result, from Hamiltonian Cycle
- **TRAINYARD** (from Dominating Set)