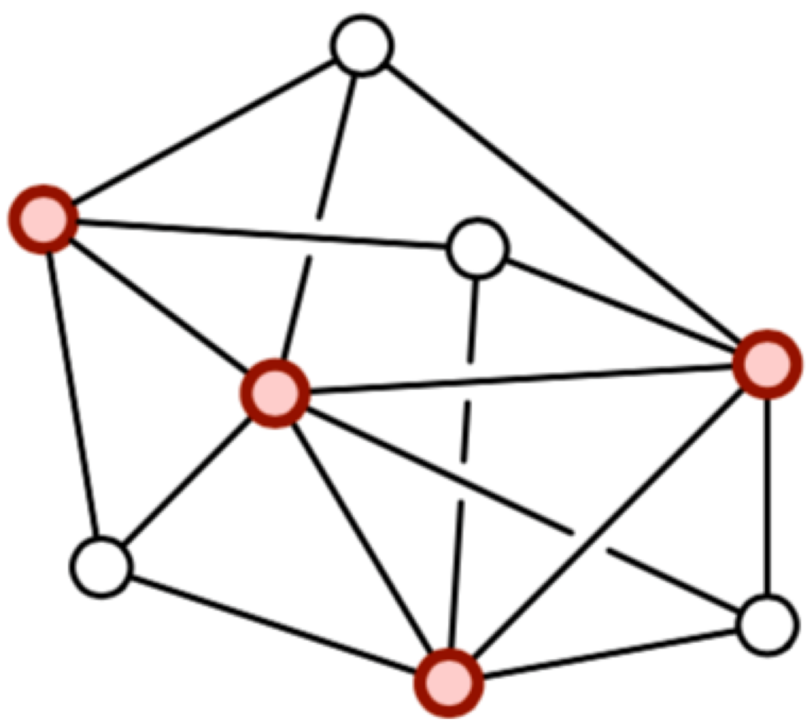
NP hardness Reductions III

Lecture 15

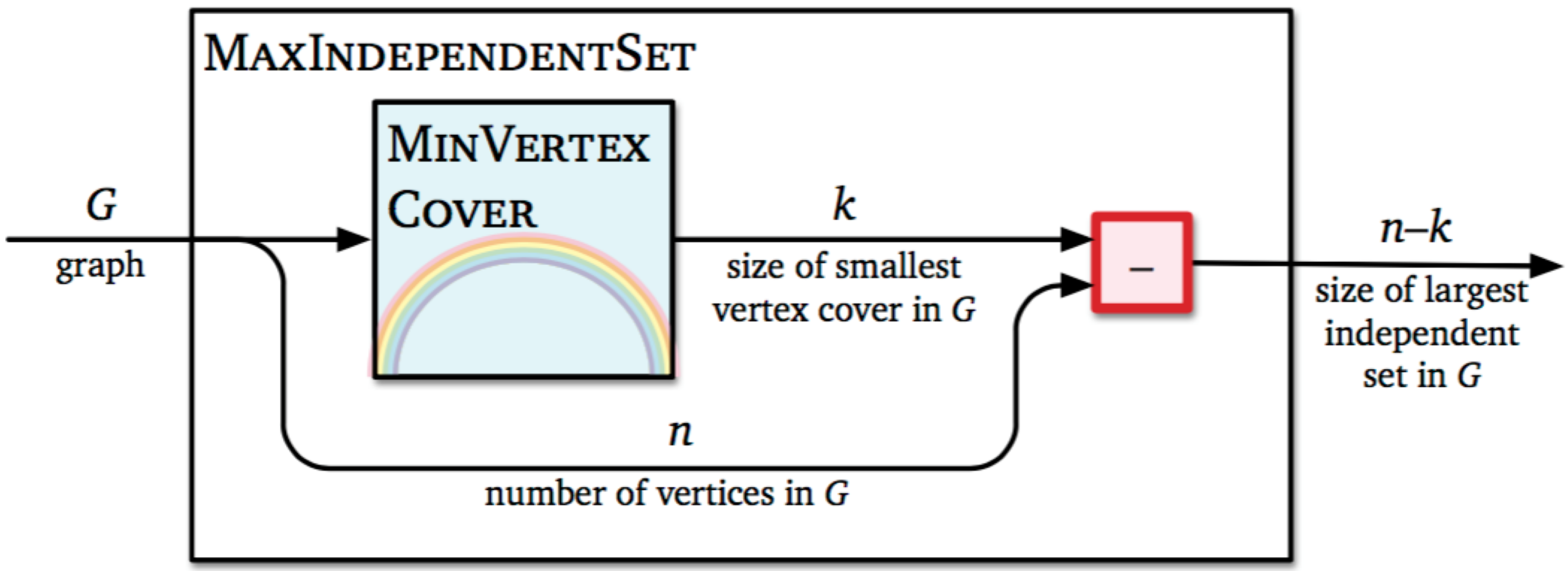
MIN Vertex Cover

- Input: a graph G(V,E)
- Output: Smallest set of vertices that touch every edge



- If I is Independent set in G,
- V\I is vertex cover!

- Largest IS in G is the complement of smallest VC in G



what is G’? same graph as G
Output is different

How to prove NP hardness

- To prove X is NP-hard:

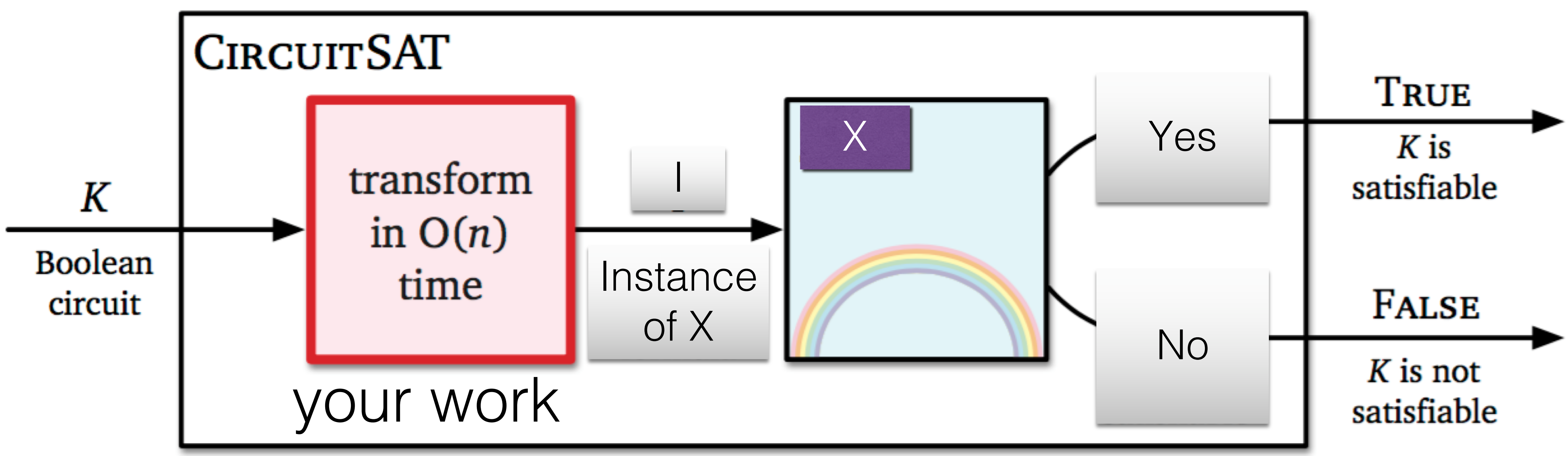
- **Step 1**: Pick a known NP-hard problem Y
- **Step 2:** Assume for the sake of argument, a polynomial time algorithm for X.
- **Step 3**: Derive a polynomial time algorithm for Y, using algorithm for X as subroutine.
- **Step 4**: Contradiction

Reduce Y to X

Reduce FROM the problem
I know about
TO the problem
I am curious about

NP hardness of X

- To show X is NP hard (example):
- Poly time reduction from CircuitSAT.
- If there is a poly time algorithm to solve X, then there is poly time algorithm to solve CircuitSAT



NP hardness

- Library of NP-hard problems

CircuitSAT

SAT

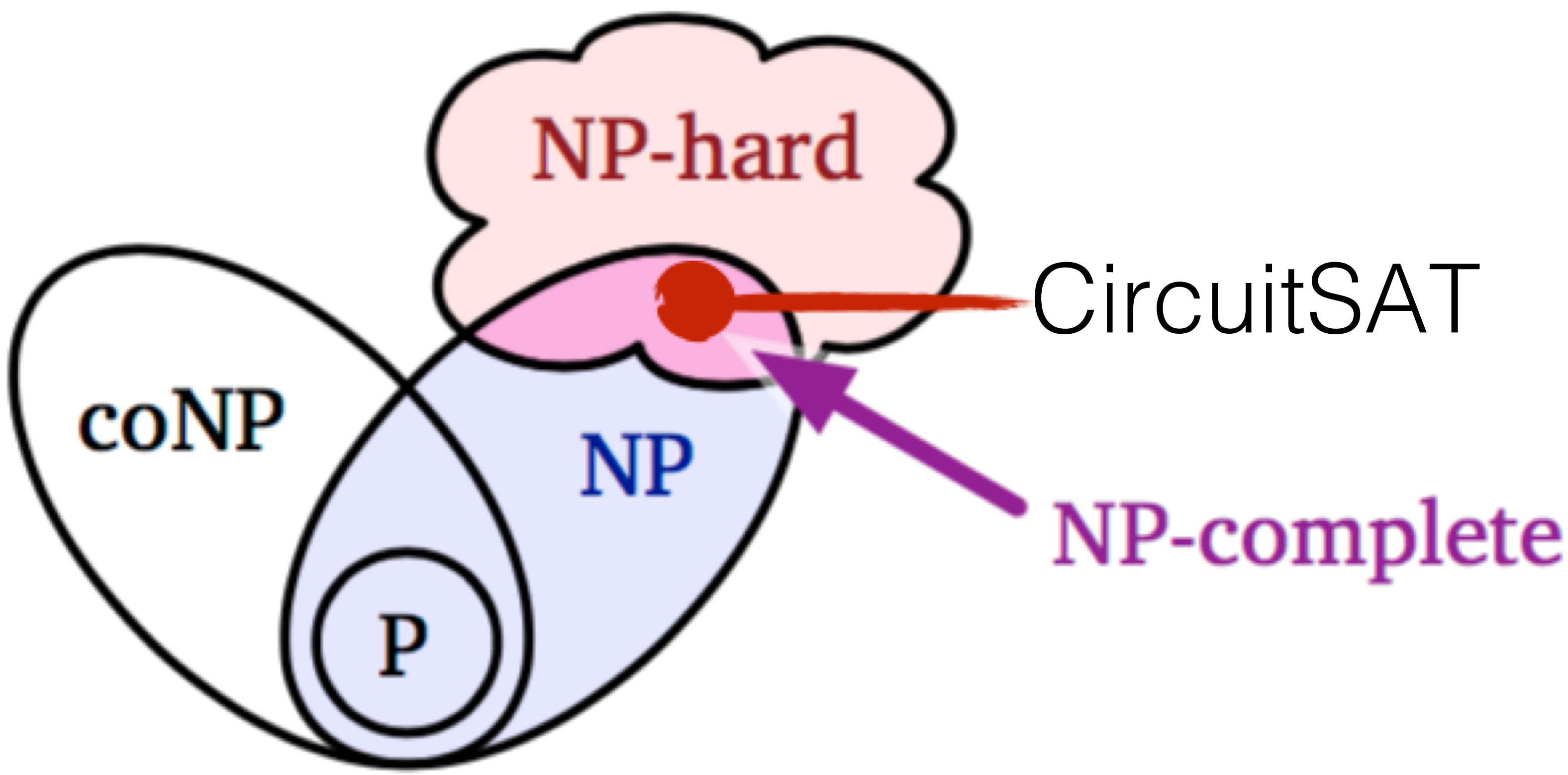
3SAT

MAX IS

MAX Clique

Min Vertex Cover

Mickey Mouse Diagram



SAT

Does a given boolean forumla, in CNF, have a satisfying assignment?

3-SAT

Does a given boolean forumla, in CNF with exactly three literals per clause, have a satisfying assignment?

Min Vertex Cover

In a given undirected graph, what is the (size of the) smallest subset of the vertices covering all of the edges?

Max Independent Set

In a given undirected graph, what is the (size of the) larges subset of the vertices having no edges in common?

Max Clique

What is the (size of the) largest complete subgraph of a given undirected graph?

Min Set Cover

Given a set S and a collection of subsets of S, what is smallest set of these subsets whose union is S?

Min Hitting Set

Given a set S and a collection of subsets of S, what is smallest subset of S containing at least one element from every subset?

Hamilton Path

Does a given graph have a Hamilton Path?

Hamilton Cycle

Does a given graph have a Hamilton Cycle?

Traveling Salesperson

What is the minimum cost Hamilton Cycle in a weighted, complete, graph?

Longest Path

What is the longest path between two given nodes in a weighted, undirected, graph?

Subset Sum

Does a given set of positive integers have a subset with sum k?

Partition

Can a given set of positive integers be partitioned into two subsets each with the same sum?

3-Partition

Can a given set of $3n$ positive integers be partitioned into n 3-element subsets each with the same sum?

Minesweeper

In a given Minesweeper configuration, is it safe to click on a particular square?

Sodoku

Does a given Sodoku puzzle have a solution?

NP hardness

- Library of NP-hard problems

CircuitSAT

SAT

3SAT

MAX IS

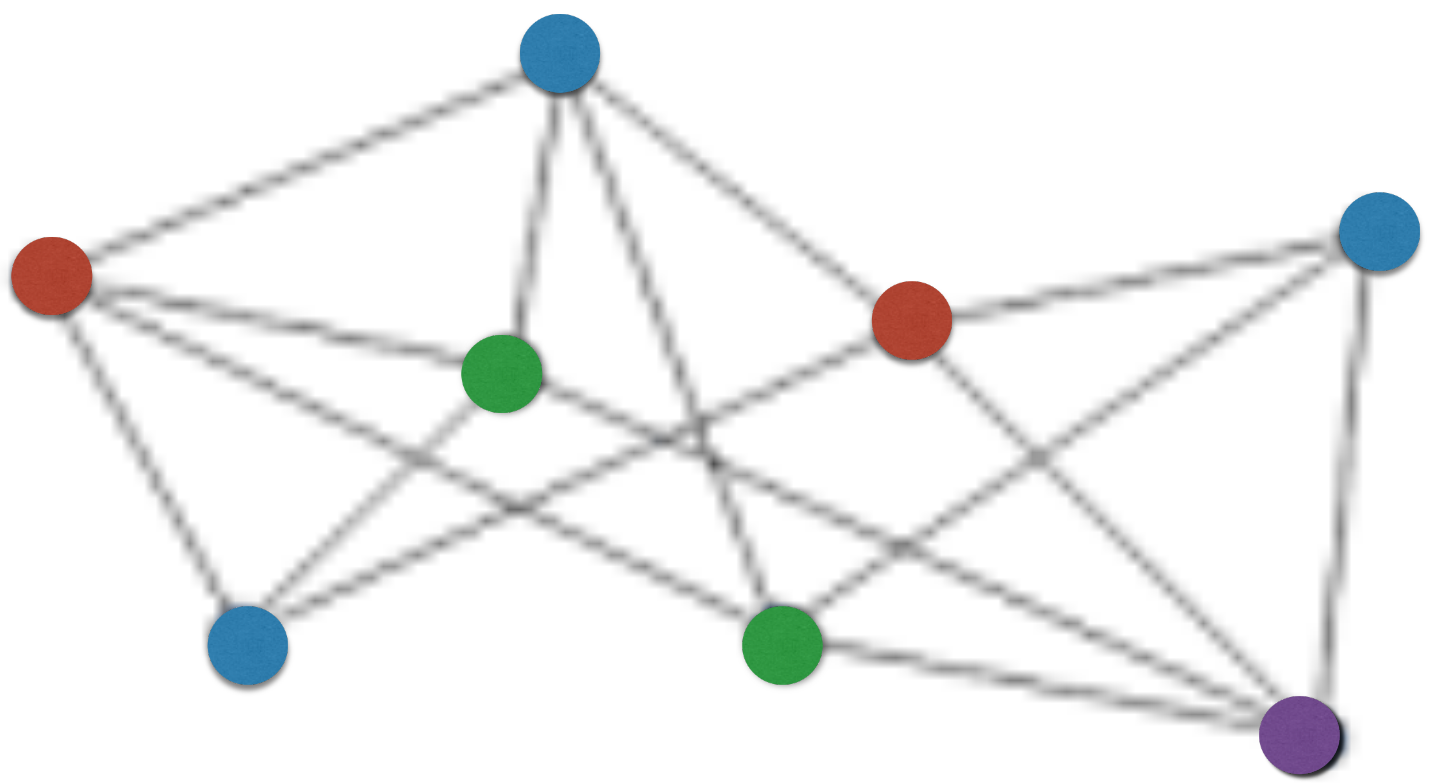
MAX Clique

Min Vertex Cover

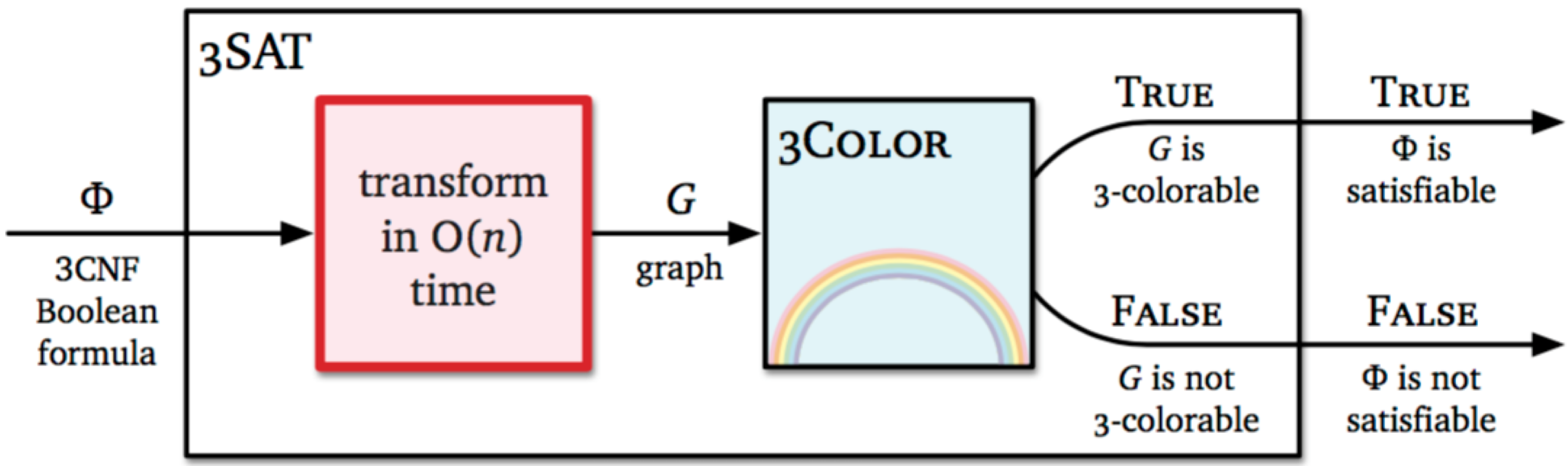
3 Coloring

3 Coloring

- Input: a graph G(V,E)
- Output: True iff G has a proper 3 coloring



what problem to start with?


3SAT
Φ
3CNF Boolean formula
transform in O(n) time
G
graph
3Color
True
G is 3-colorable
True
Φ is satisfiable
False
G is not 3-colorable
False
Φ is not satisfiable

3COL

- Given an arbitrary 3CNF formula F
  - Build a graph G as follows

    Best described in pieces

1) piece that corresponds to variables

2) piece that corresponds to clauses

3) piece that enforces logical consistency

"gadgets"

3COL

- Given an arbitrary 3CNF formula F
- Build a graph G as follows

Best described in pieces

1) Truth Gadget

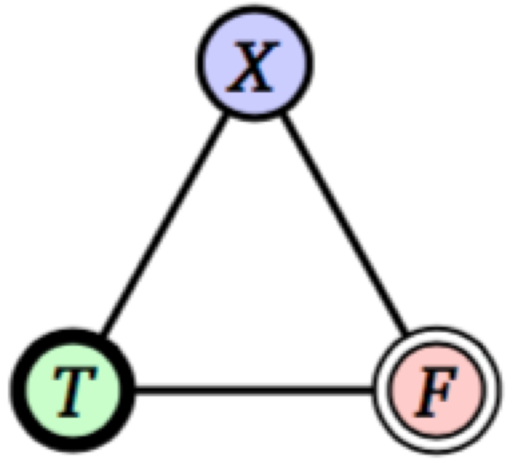
3COL

- Given an arbitrary 3CNF formula F

- Build a graph G as follows

Best described in pieces

1) Truth Gadget



3COL

- Given an arbitrary 3CNF formula F
    - Build a graph G as follows

Best described in pieces

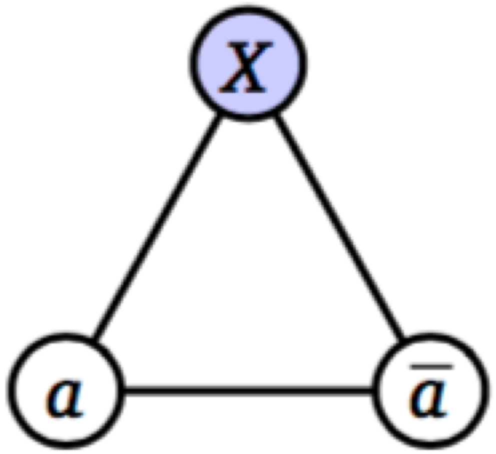
2) Variable Gadget

3COL

- Given an arbitrary 3CNF formula F
  - Build a graph G as follows

Best described in pieces

2) Variable Gadget



one vertex in the graph for every variable and one for its negation. One vertex labeled X

3COL

- Given an arbitrary 3CNF formula F
  - Build a graph G as follows

Best described in pieces

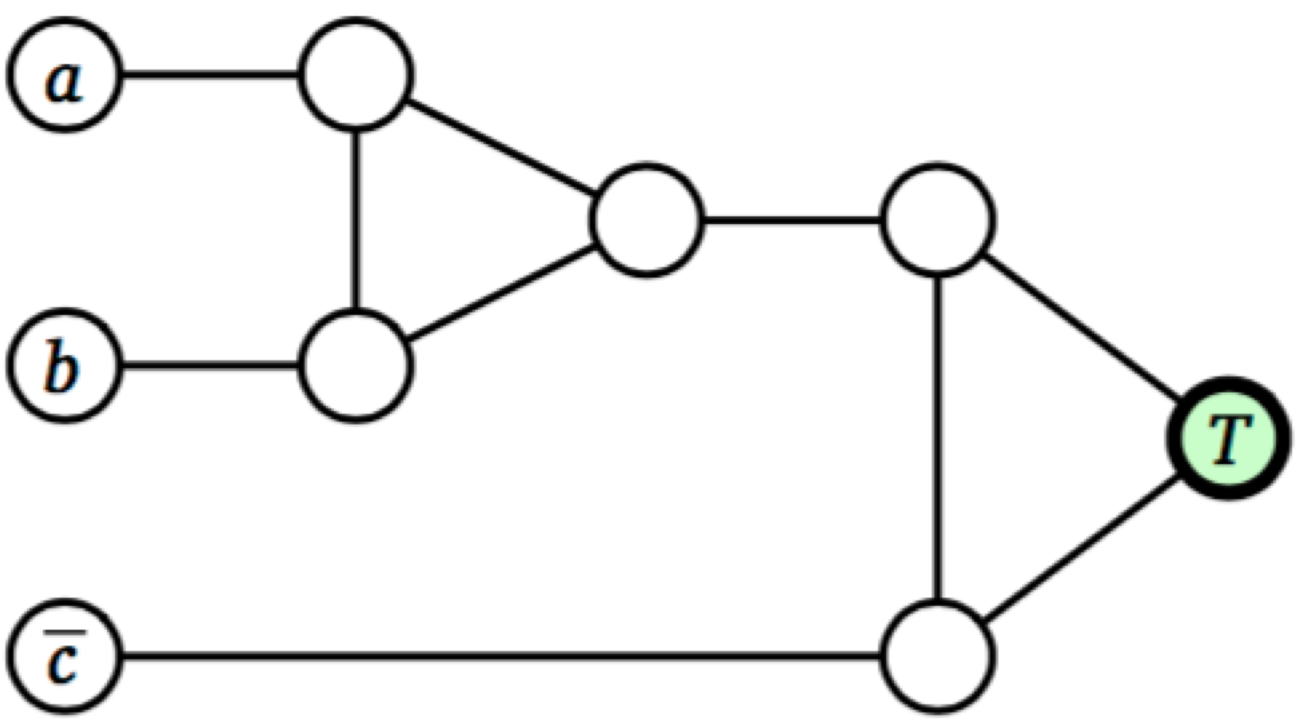
3) Clause Gadget

3COL

- Given an arbitrary 3CNF formula F
    - Build a graph G as follows

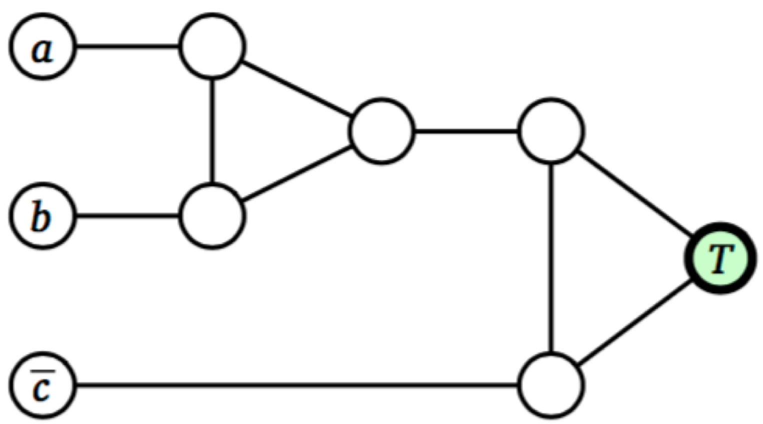
Best described in pieces

3) Clause Gadget



3COL

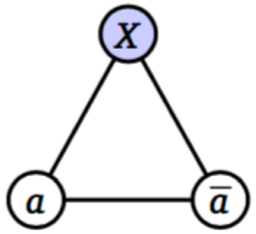
$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$



in any proper coloring
at least one of the three literals
must be colored T

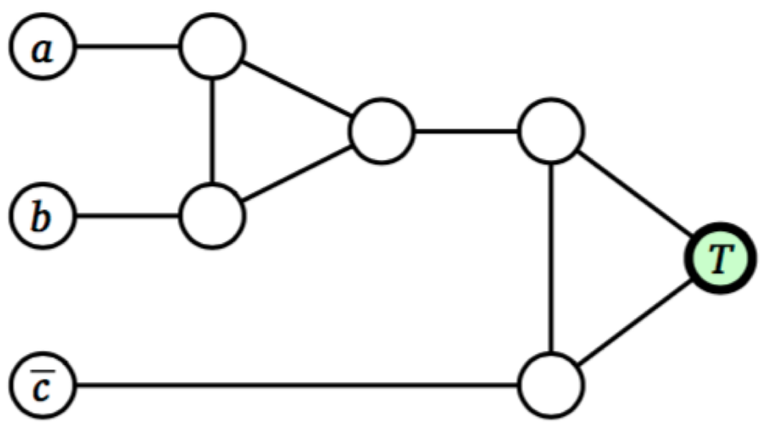
easier to prove with 2 SAT example

literal vertices,
connected to X



3COL

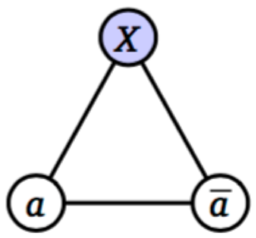
$$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$$

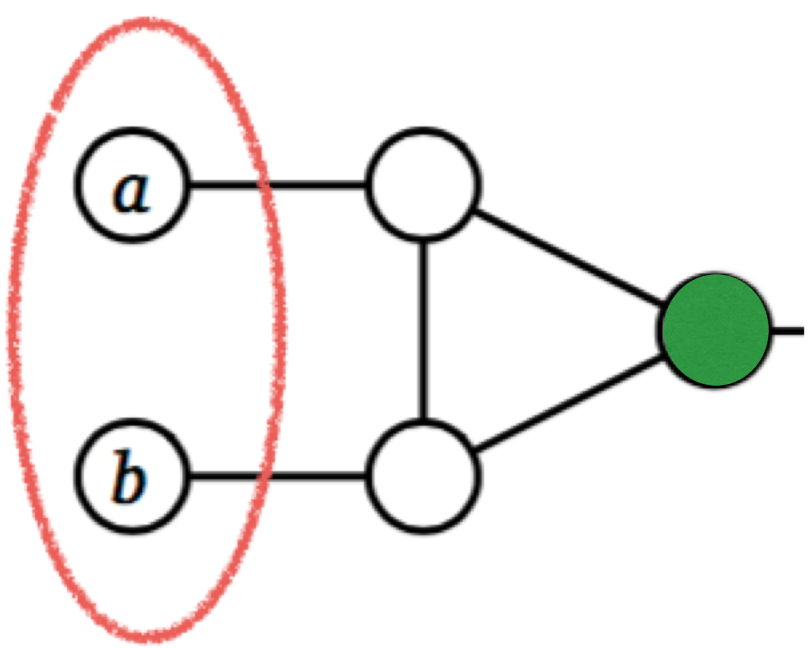


in any proper coloring
at least one of the three literals
must be colored T

easier to prove with 2 SAT example

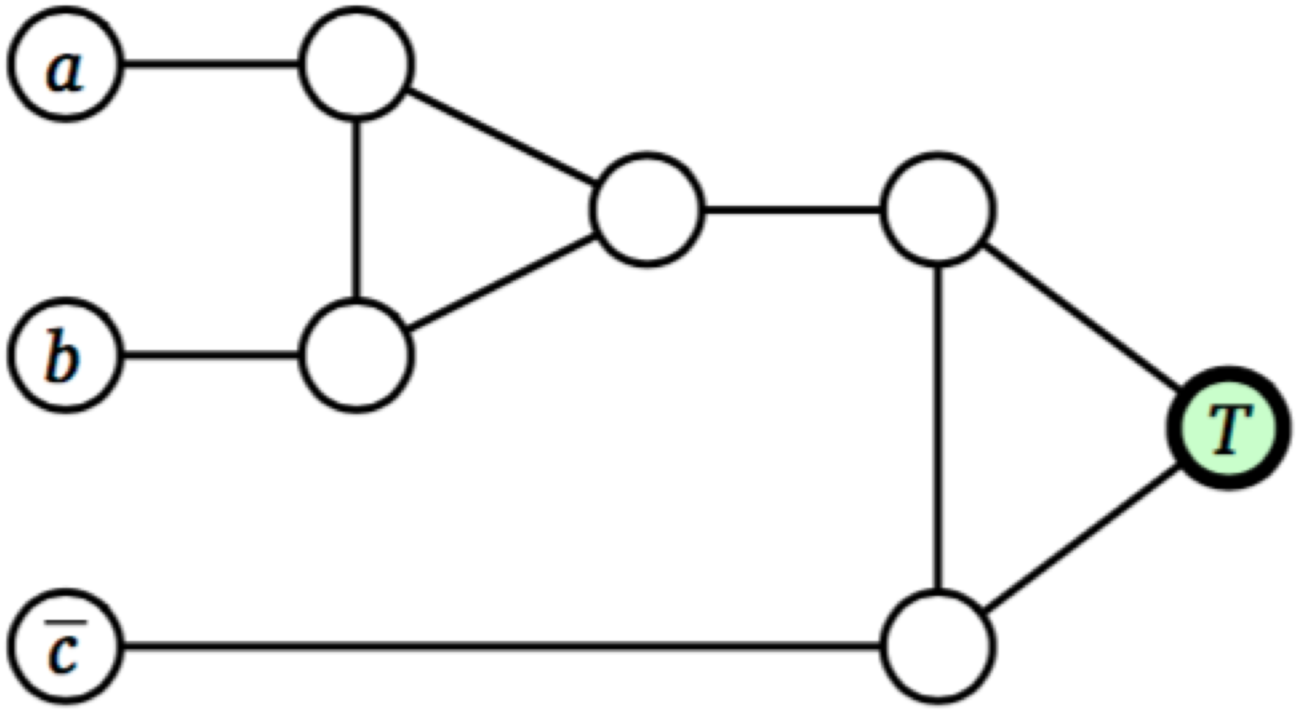
literal vertices,
connected to X





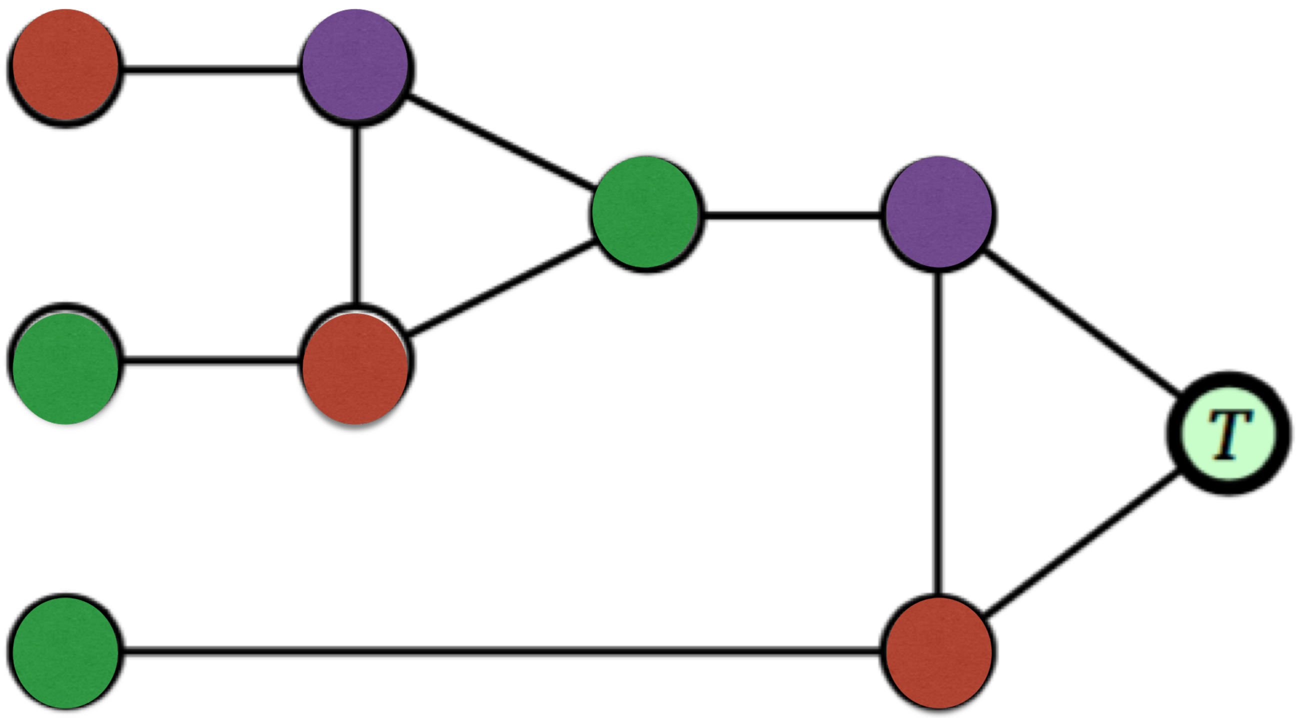
3COL

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



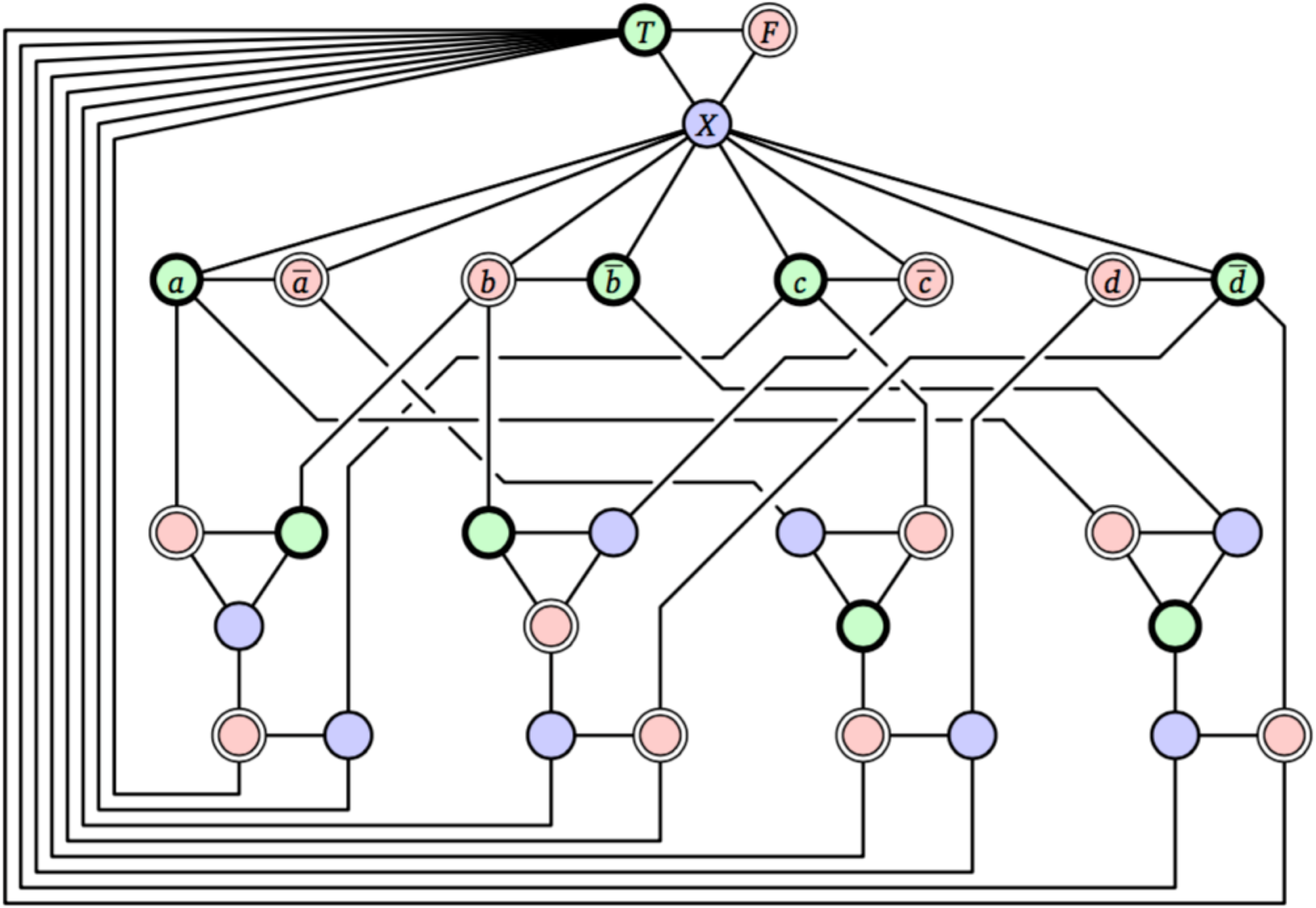
3COL

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



There are 8 possible colorings for the 3 literals on the left.

- For 7 of them one gets colored T and I can properly color the gadget
- For the 8th, all of them are colored False and I can't properly color the gadget



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

Proof

Suppose F is satisfiable

So G is 3-Colorable

Suppose G is 3-Colorable

So F is satisfiable

Proof

Suppose F is satisfiable

- Fix any satisfying assignment
- Color True literals same color as T
- Color False literals same color as F
- By case analysis:

extend the coloring to the clause gadget

So G is 3-Colorable

Suppose G is 3-Colorable

So F is satisfiable

Proof

Suppose F is satisfiable

- Fix any satisfying assignment
- Color True literals same color as T
- Color False literals same color as F
- By case analysis:
  extend the coloring to the clause gadget

So G is 3-Colorable

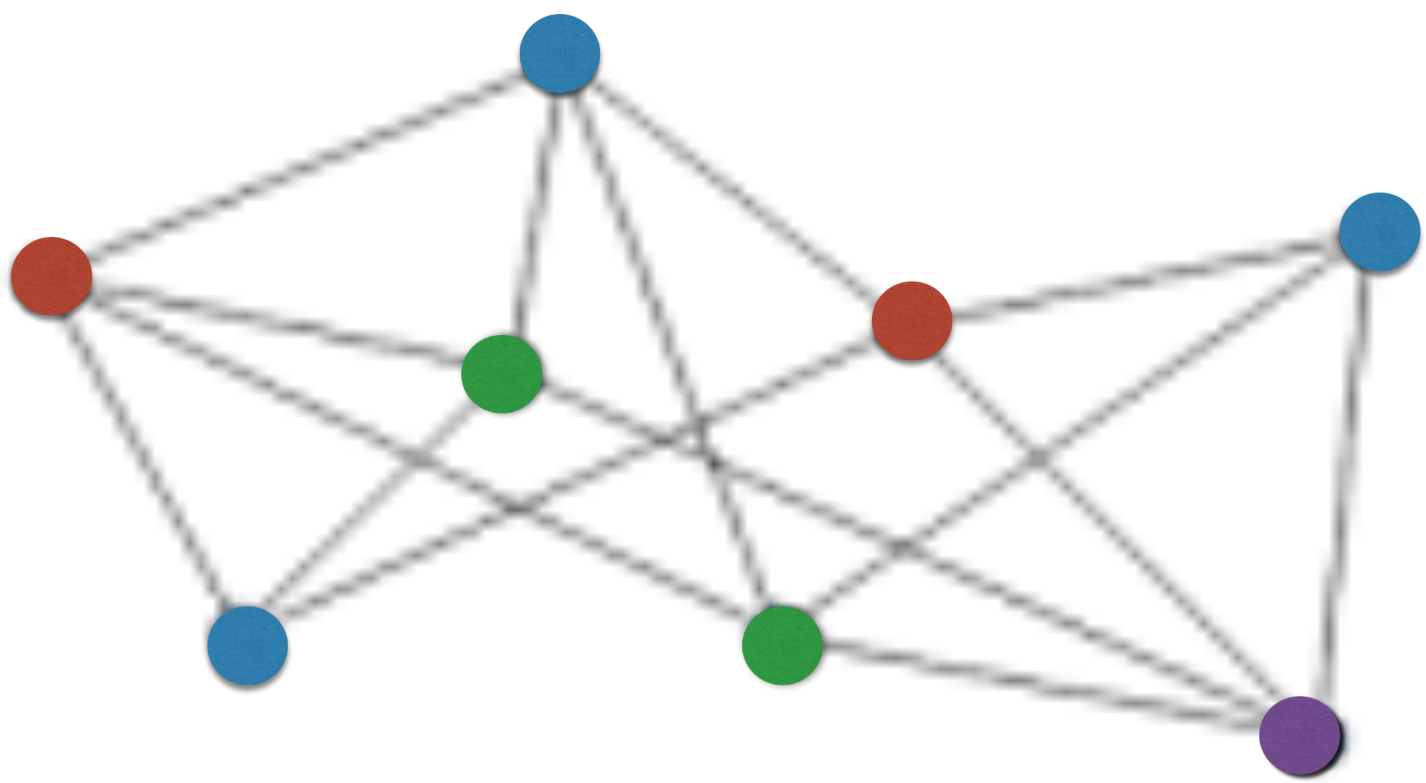
Suppose G is 3-Colorable

- Fix a proper 3 Coloring
- Each literal vertex is colored T or F
- This gives me an assignment of boolean values to variables
- By case analysis: At least one literal in each clause gadget is colored T

So F is satisfiable

4 Coloring?

- Input: a graph G(V,E)
- Output: True iff G has a proper 4 coloring



Hamiltonian Cycle

- Input: a directed graph G(V,E)
- Output: Is there a cycle in G that visits each vertex exactly once?
- Really asking if there is a way to order the vertices so that every adjacent pair is connected by an edge.
- Reduction from HC if a problem asks for ordering of vertices.
- Anti-topological sort

NP hardness

- Library of NP-hard problems

CircuitSAT

SAT

3SAT

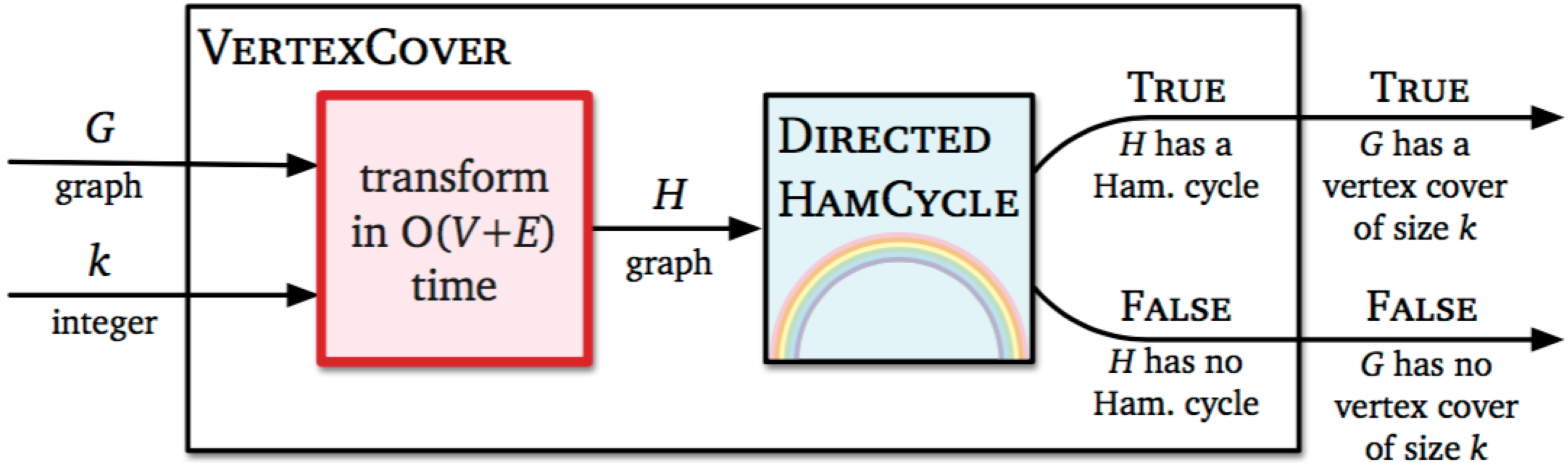
MAX IS

MAX Clique

Min Vertex Cover

3 Coloring

Hamiltonian Cycle



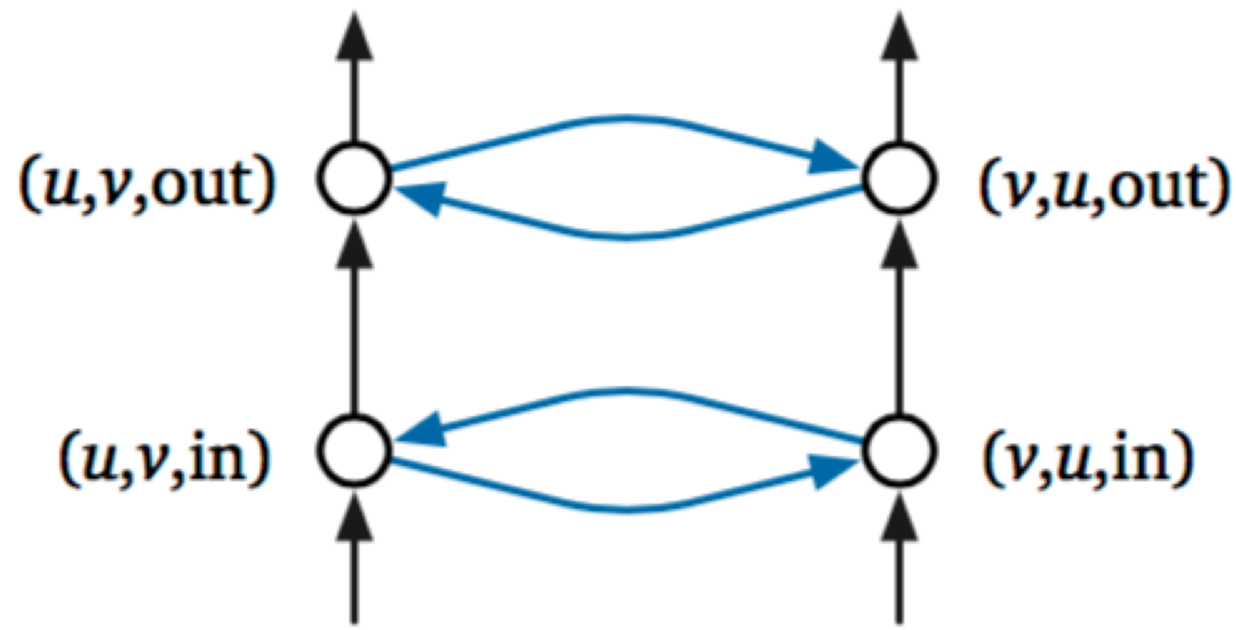
Hamiltonian Cycle

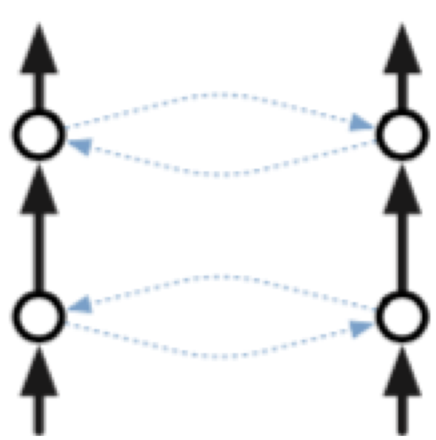
- Given an arbitrary graph G and parameter k
  - Build a graph H as follows

Best described in gadgets

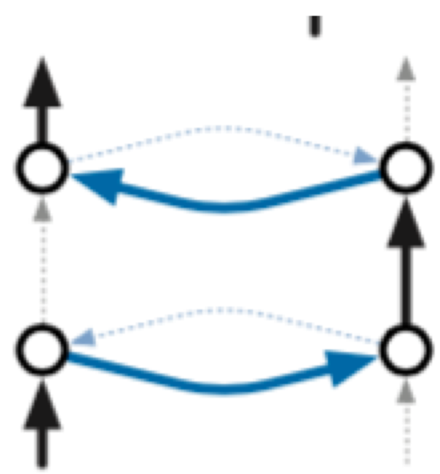
Hamiltonian Cycle

1) edge gadget

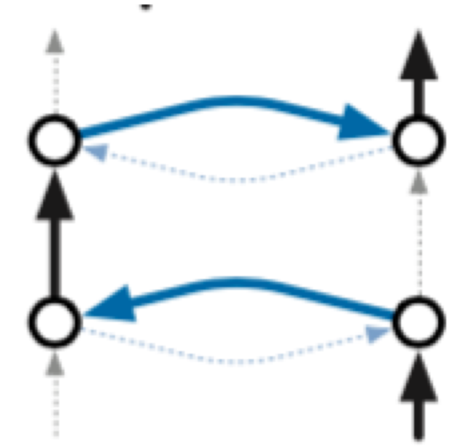




both u,v in VC



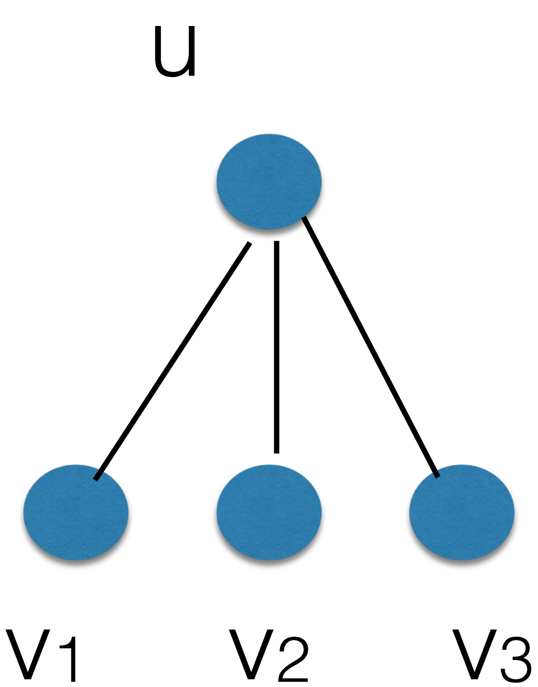
only u in VC

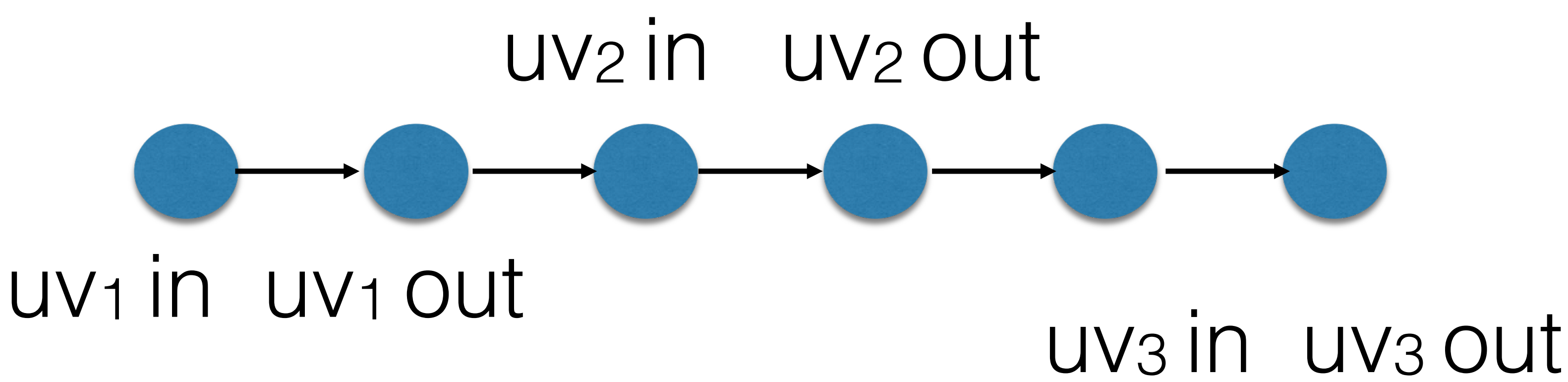


only v in VC

Hamiltonian Cycle

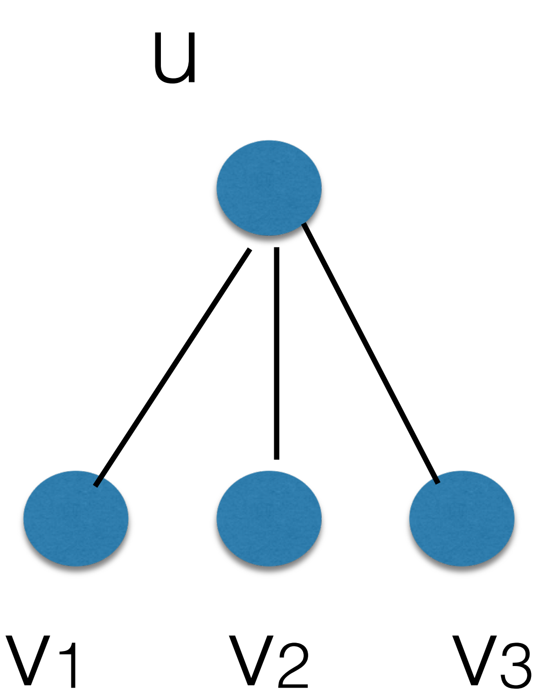
2) vertex gadget

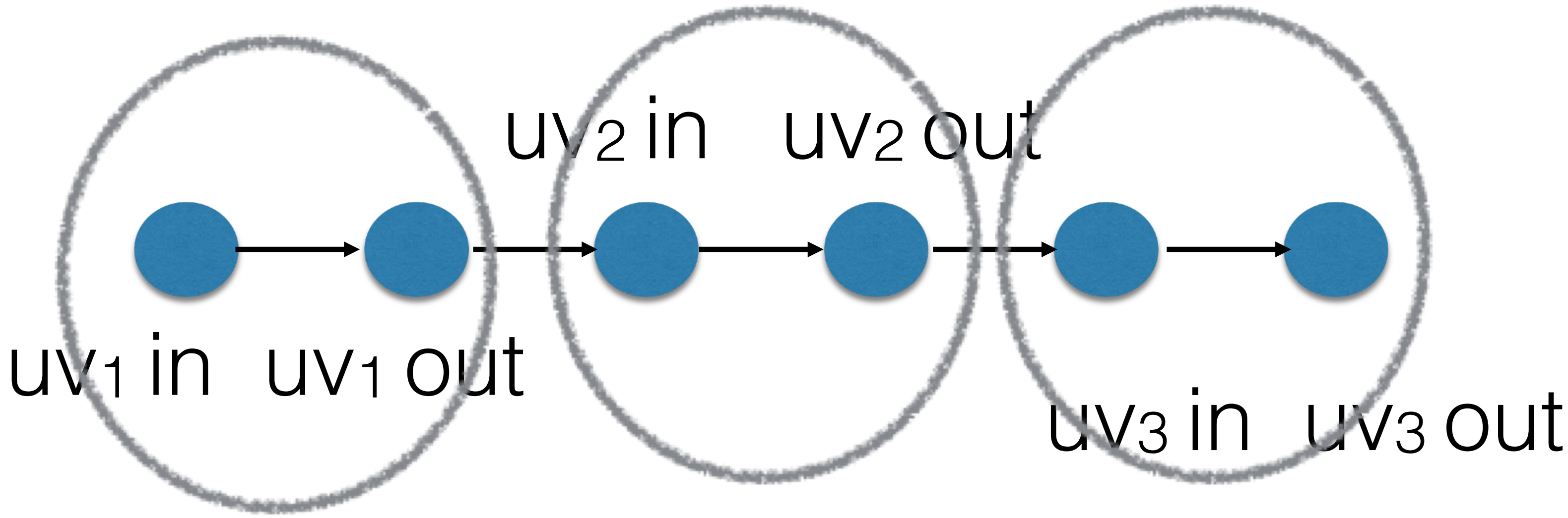




Hamiltonian Cycle

2) vertex gadget

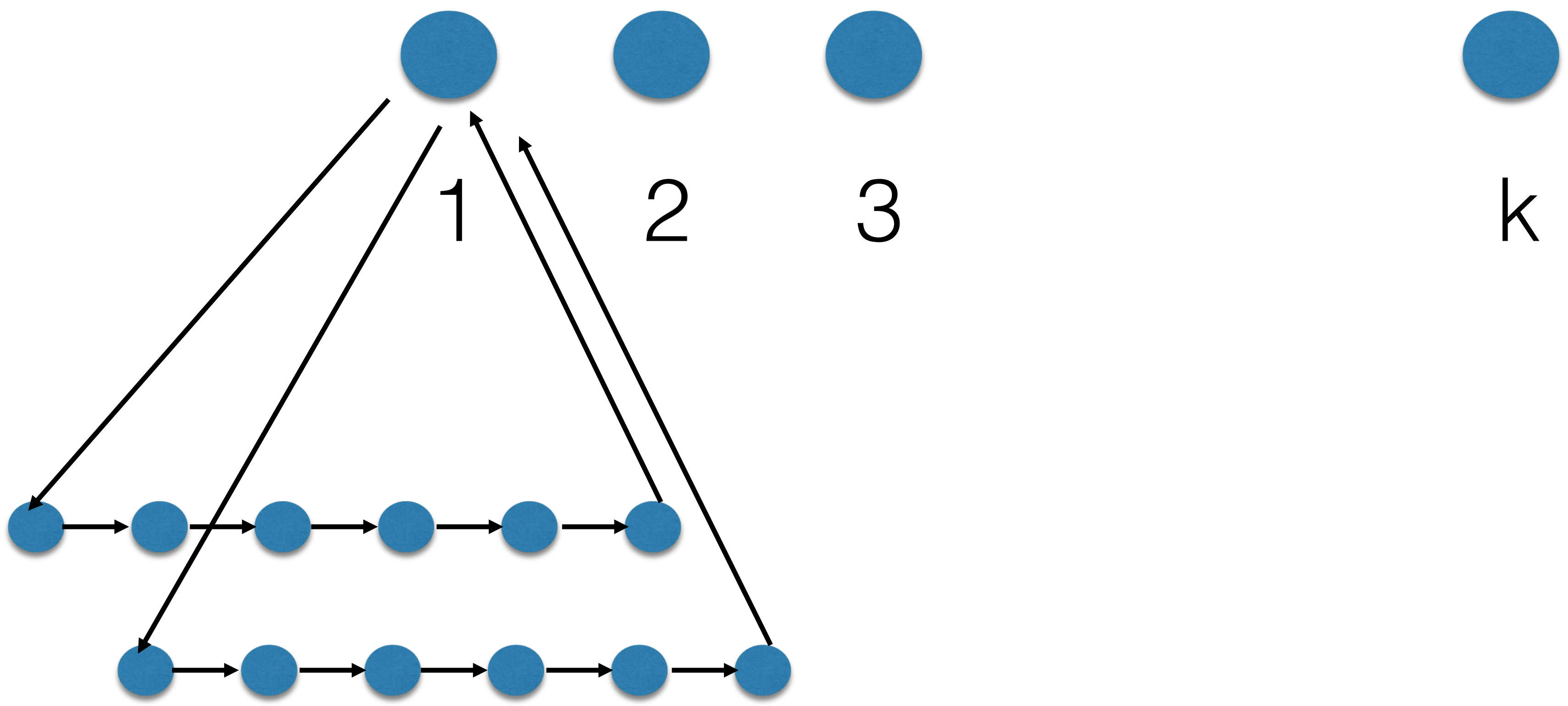


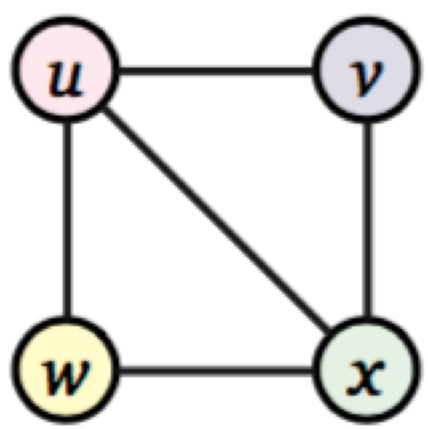


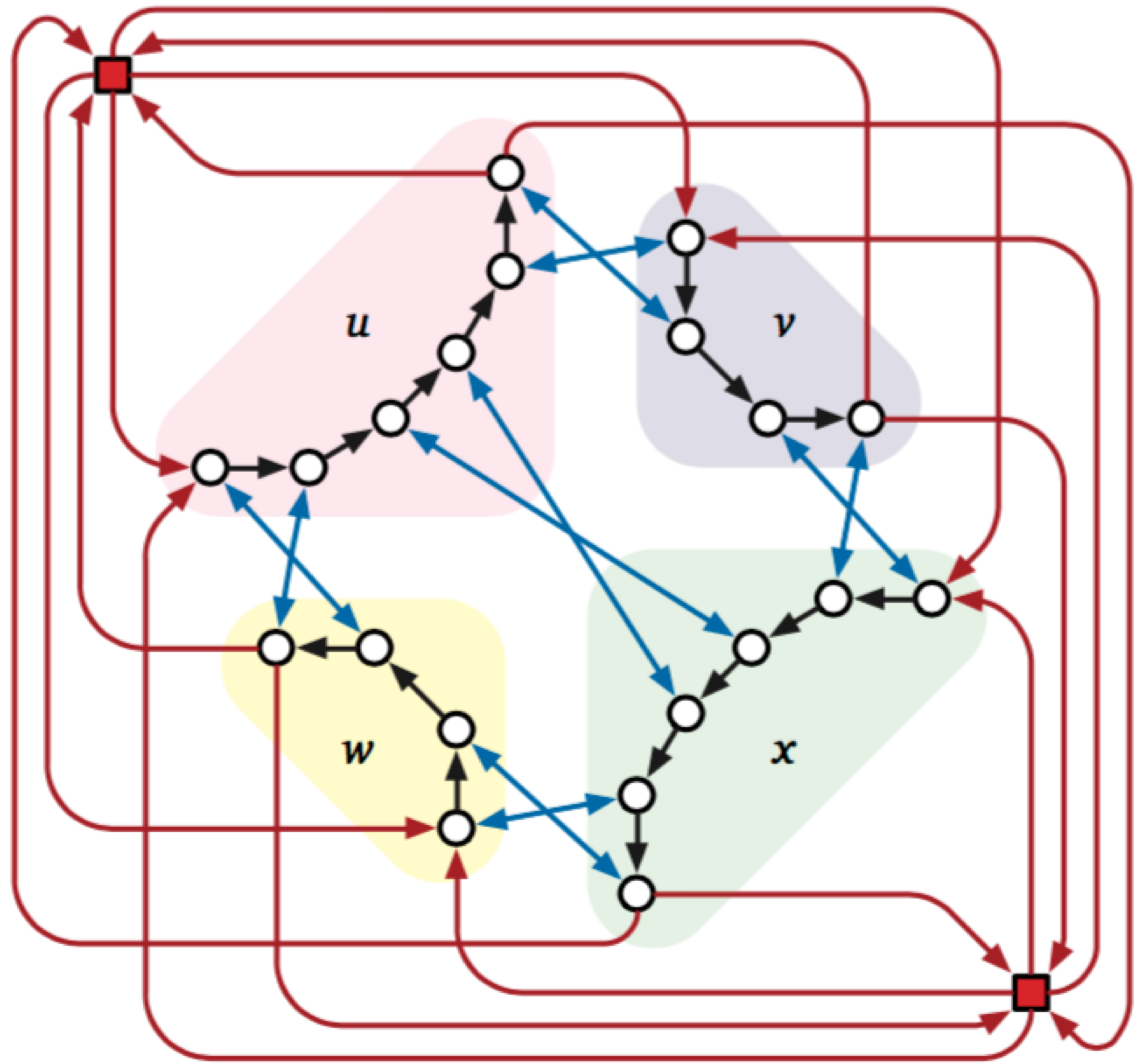
connected with edge gadget too

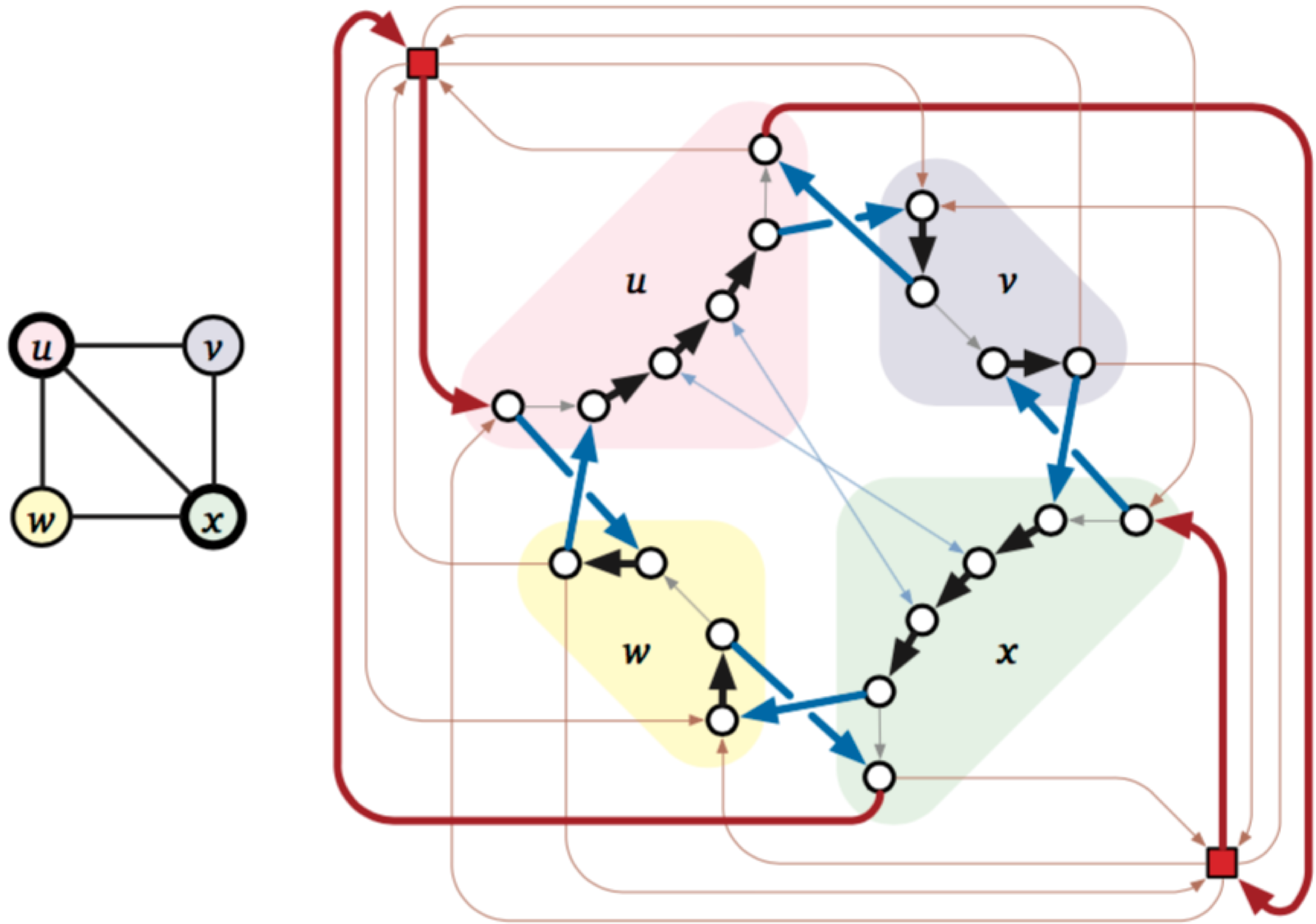
Hamiltonian Cycle

3) cover gadget




u
v
w
x


u
v
w
x



u
v
w
x
u
v
w
x

Approximation Algorithm/Ratio

Minimization problem Π: $\mathcal{A}$ is an approximation algorithm with *(relative)* approximation ratio α iff

- $\mathcal{A}$ is polynomial time algorithm
- for all instance I of Π, $\mathcal{A}$ produces a feasible solution $\mathcal{A}(I)$ such that

  val $(\mathcal{A}(I)) \leq \alpha$ val (OPT(I))

(Note: $\alpha \geq 1$)

Remark: α can depend in size of I, hence technically it is $\alpha(|I|)$.

Example: $\alpha(|I|) = \log n$

Maximization problems

Maximization problem Π: $\mathcal{A}$ is an approximation algorithm with *(relative)* approximation ratio α iff

- $\mathcal{A}$ is polynomial time algorithm
- for all instance I of Π, $\mathcal{A}$ produces a feasible solution $\mathcal{A}(\mathrm{I})$ such that

  val $(\mathcal{A}(\mathrm{I})) \geq \alpha$ val (OPT(I))

  (Note: $\alpha \leq 1$)

Very often people use $1/\alpha$ (≥ 1) as approximation ratio

Proving hardness of approximation

Proving hardness of approximation is essentially the following:

Let Π be a minimization problem

Suppose we want to prove that Π is $\alpha(|I|)$ hard to approximation where $|I|$ is the instance size

We need to show that if there is a polynomial time algorithm for Π with an approximation ratio $\alpha(|I|)$ then we can use it to solve an NP-Hard *decision problem* (any NP-Hard problem would do)

This implies that unless P=NP no $\alpha(|I|)$ approximation ratio for Π

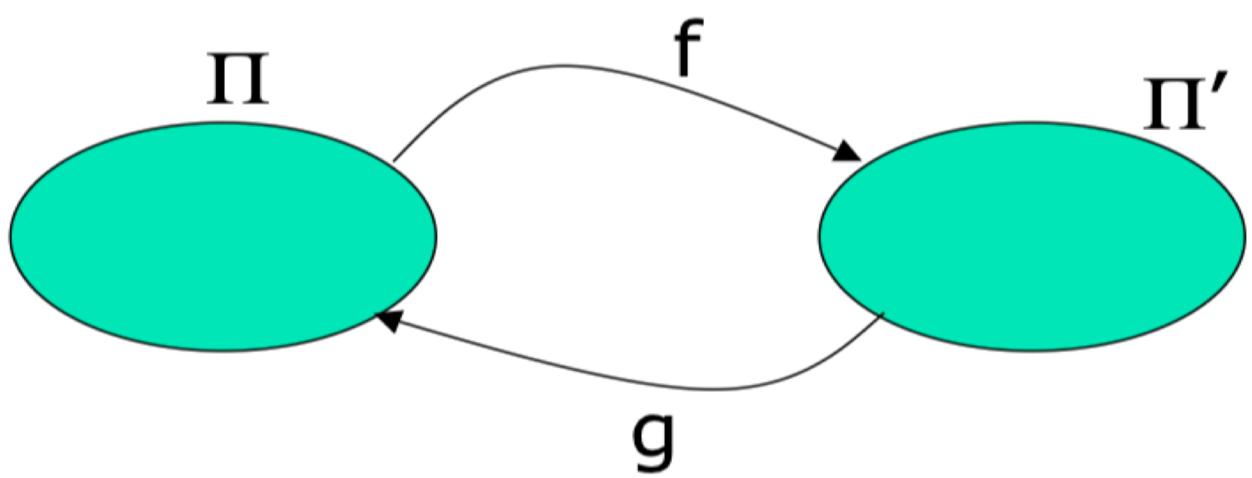
Reductions

Once we prove a particular problem Π is hard to approximate to within an α factor, we wish to use this to prove that another problem Π' is hard to approximate to within a β factor

To make it easy to compose reductions we need to define a proper notion of reduction. This is somewhat more involved than reductions to prove NP-Completeness for decision problems since we have function problems with solutions, quality of solutions etc.

We define two types of reductions

Approximation Preserving Reductions



Given an instance I of Π, $I' = f(I)$ is an instance of Π'

Given a solution s to I', $g(I, I', s)$ is a solution to I

Both f and g are poly-time computable functions

Approx preserving reductions

Other properties of f, g

We assume that both Π, Π' are minimization problems, the definitions change for min-max, max-max, etc

1. $OPT(I') \leq OPT(I)$
2. If $s \in \mathcal{S}(I')$ then $t = g(I,I',s)$ is a solution to I and $Val(t, I) \leq Val(s, I')$

(recall that $\mathcal{S}(I)$ is the set of solutions to instance I and that $Val(t, I)$ is the objective function value for soln t)

If f, g satisfy above properties then (f,g) is an approximation preserving reduction from Π to Π'

Approx preserving reductions

Using this notion of reduction allows us to claim a couple of simple but useful features

Lemma: If (f,g) is an approximation preserving reduction from Π to Π' and (f',g') is an approximation preserving reduction from Π' to Π'' then (f'',g'') is an approximation preserving reduction from Π to Π'' where $f'' = f' \circ f$ and $g'' = g \circ g'$

Approx preserving reductions

Lemma: If (f,g) is an approximation preserving reduction for Π to Π' then an α approximation to Π' where α is a constant implies an α approximation to Π

The converse of the above lemma is:

If (f,g) is an approximation reduction from Π to Π' and if Π does is NP-hard to approximate to a factor of β then Π' is NP-hard to approximate to within a factor of β

Both lemmas are straight forward exercises from the definitions

An example

We give a reduction from the Set Cover problem to the Node-Weighted Steiner tree problem

Set cover:

Given universe $\mathcal{U}$ of n elements and

sets $S_1, S_2, \ldots, S_m$ where each S_i is a subset of $\mathcal{U}$

Solution: $A \subseteq \{1,2, \ldots, m\}$ s.t $\cup_{i \in A} S_i = \mathcal{U}$

Objective function: $Val(A) = |A|$

Goal: minimization

An example

Node-weighted Steiner tree problem:

Given graph $G=(V,E)$ and node weighs $w: V \to \mathcal{R}^+$

$T \subseteq V$, terminals

Solution: a (connected) subgraph $H=(V_H, E_H)$ of G s.t

$T \subseteq V_H$

Objective function: $w(V_H)$

Goal: minimization

Reduction

- It is known that Set Cover is hard to approximate within a factor of $c \log n$ unless P=NP for some constant c
- Thus we would like to conclude that node-weighted Steiner tree is also $c \log n$ hard to approximate
- Unfortunately we cannot do this in a straight forward way using the current machinery we set up
- What we can conclude is the following:
- Since Set Cover is hard to approximate to within any constant α, node-weighted Steiner tree problem is also hard to approximate to within any constant α