Algorithms, Lecture 3 on NP : Nondeterministic Polynomial Time

Last week:

Defined Polynomial Time Reductions:

Problem X is poly time reducible to Y

$X \leq_P Y$

if can solve X using poly computation and a poly number of calls to an algorithm solving Y.

“Up to poly factors, X is at least as easy as Y”
“Up to poly factors, Y is harder than X”

Last class:

Defined NP:

decision (yes/no) problems

can check certificates for yes answers in ptime

Have poly time *proof checkers*:

A poly time algorithm A so that

if X(s) = yes,

there exists a w so that A(s,w) = yes, and

if X(s) = no,

for all w, A(s,w) = no

Last class:

Defined NP-Complete:

1. $X \in NP$, and
2. X is NP-hard

X is NP-hard if:

1. for all $Y \in NP$, $Y \leq_P X$, or
2. CircuitSat $\leq_P X$, or
3. $Y \leq_P X$, for some NP-Hard Y, such as SAT, Independent Set, Vertex Cover

Last class:

Proved CircuitSAT is NP-Hard.

Proved CircuitSAT $\leq_P$ SAT.

In fact, proved CircuitSAT $\leq_P$ 3-SAT

Where 3-SAT is SAT,
but each clause has at most 3 terms.

Last class:

Proved CircuitSAT is NP-Hard.

Proved CircuitSAT $\leq_P$ SAT.

In fact, proved CircuitSAT $\leq_P$ 3-SAT

Where 3-SAT is SAT,
but each clause has at most 3 terms.

Also possible to force each variable
to appear at most 3 times

Also possible to force each variable to appear at most 3 times

Say a variable x appears k times.

Create k new variables, $x_1, \ldots, x_k$,
one for each occurrence.

Add clauses

$$x_1 \vee \overline{x_2}\ ,\ x_2 \vee \overline{x_3}\ ,\ \ldots,\ x_{k-1} \vee \overline{x_k}\ ,\ x_k \vee \overline{x_1}$$

Only satisfied if all are equal.

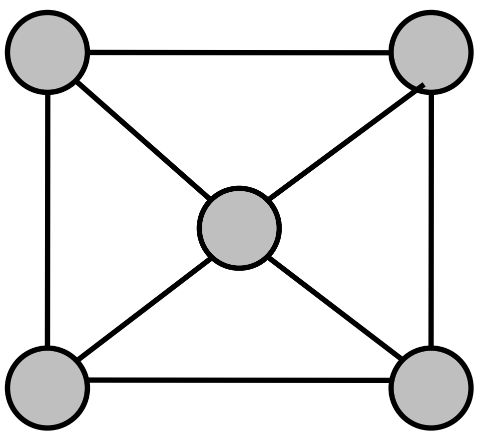
Today

Will prove more problems are NP-complete:

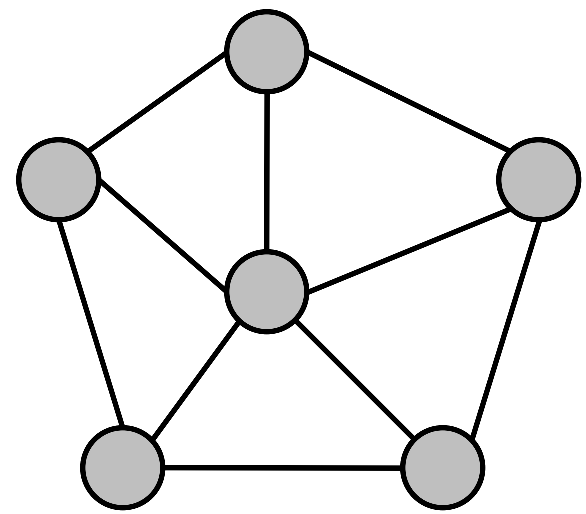
3-coloring
Hamiltonian Cycle
Travelling Salesman Problem

k-Coloring

Given a graph $G = (V,E)$, does there exist
$f : V \to \{1,2,\ldots, k\}$ (colors)
So that for all $(u,v) \in E$ $f(u) \neq f(v)$?



3-colorable



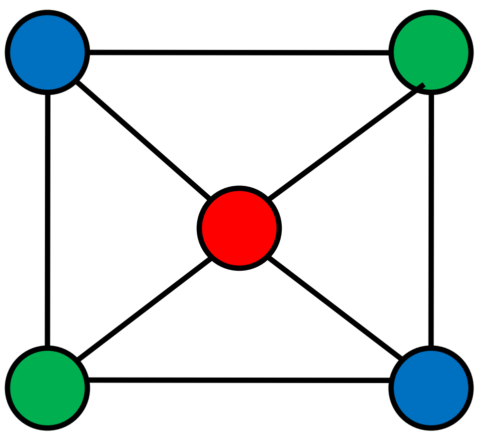
Not 3-colorable

k-Coloring

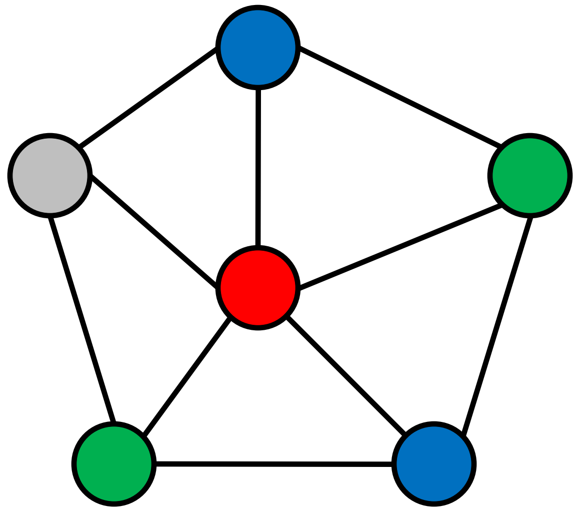
Given a graph $G = (V,E)$, does there exist

$f : V \rightarrow \{1,2,\ldots, k\}$ (colors)

So that for all $(u,v) \in E$ $f(u) \neq f(v)$?



3-colorable



Not 3-colorable

k-Coloring is NP-Complete

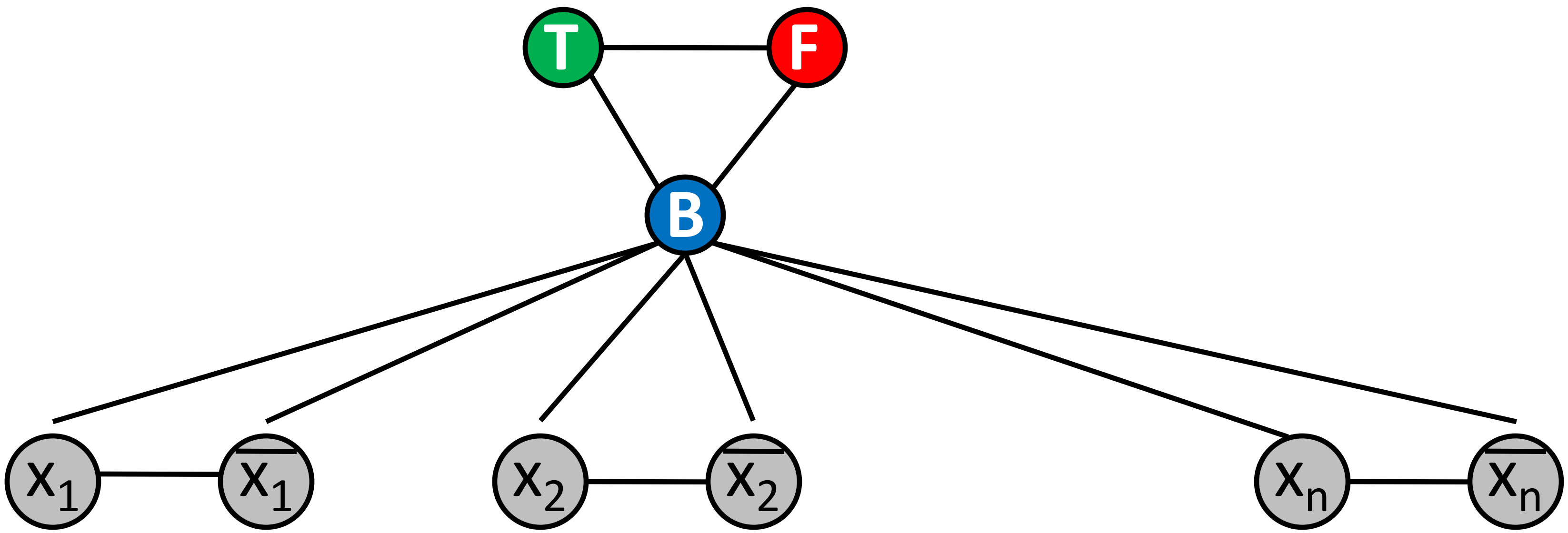
Clearly in NP, because can check a proposed coloring
To prove NP-hard, will show 3-SAT $\leq_P$ 3-Coloring

Given a collection of clauses $C_1, \ldots, C_k$, each with at most 3 terms, on variables $x_1, \ldots, x_n$

produce graph $G = (V,E)$ that is
3-colorable iff the clauses are satisfiable

3-Coloring is NP-Complete – variable gadgets

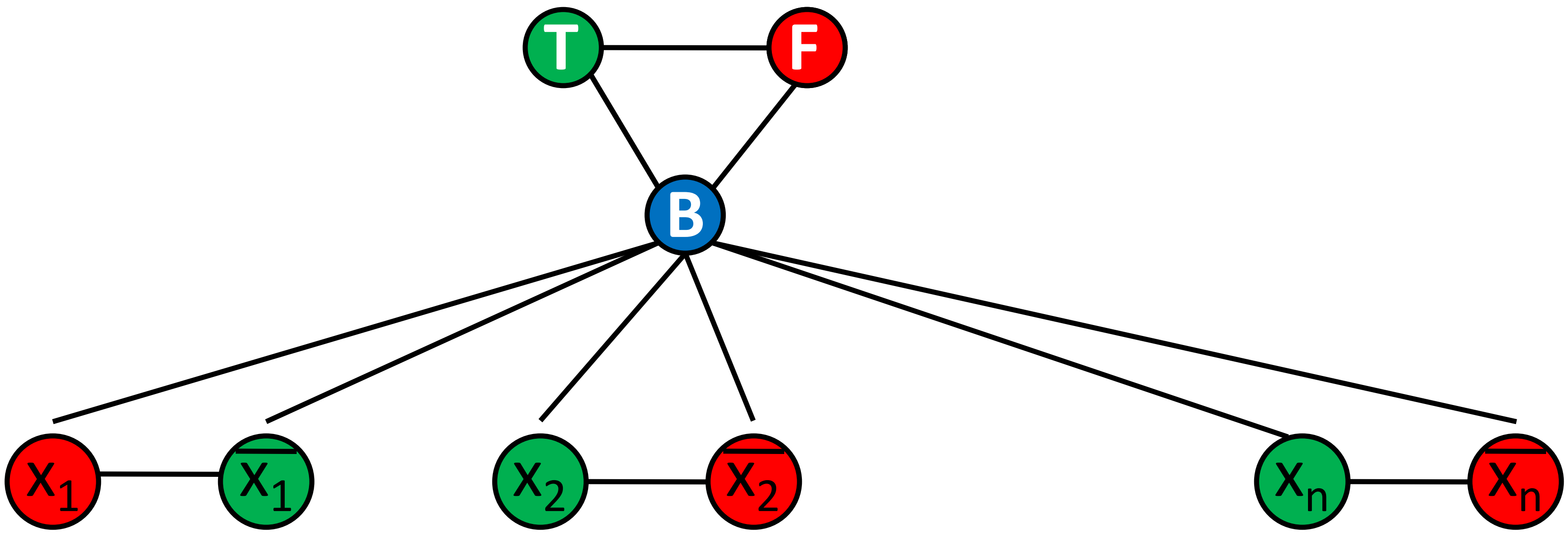
Create 3 special nodes: T, F, B (base), and one node for each term: x_i and $\overline{x_i}$



In every 3-coloring, one of x_i and $\overline{x_i}$ is colored T and one is colored F

3-Coloring is NP-Complete – variable gadgets

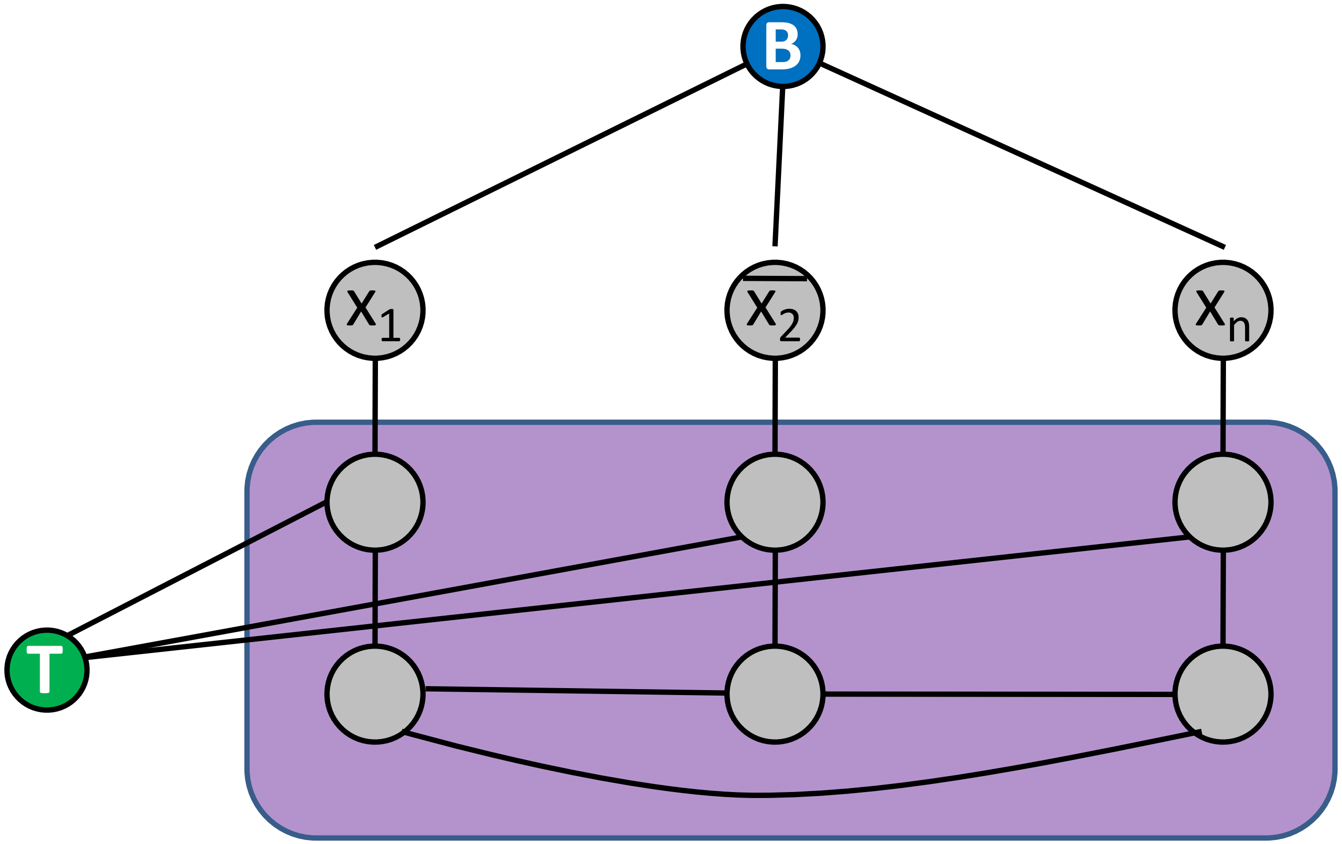
Create 3 special nodes: T, F, B (base), and one node for each term: x_i and $\overline{x_i}$



In every 3-coloring, one of x_i and $\overline{x_i}$ is colored T and one is colored F

3-Coloring is NP-Complete – clause gadgets

Consider clause $x_1 \vee \overline{x_2} \vee x_n$

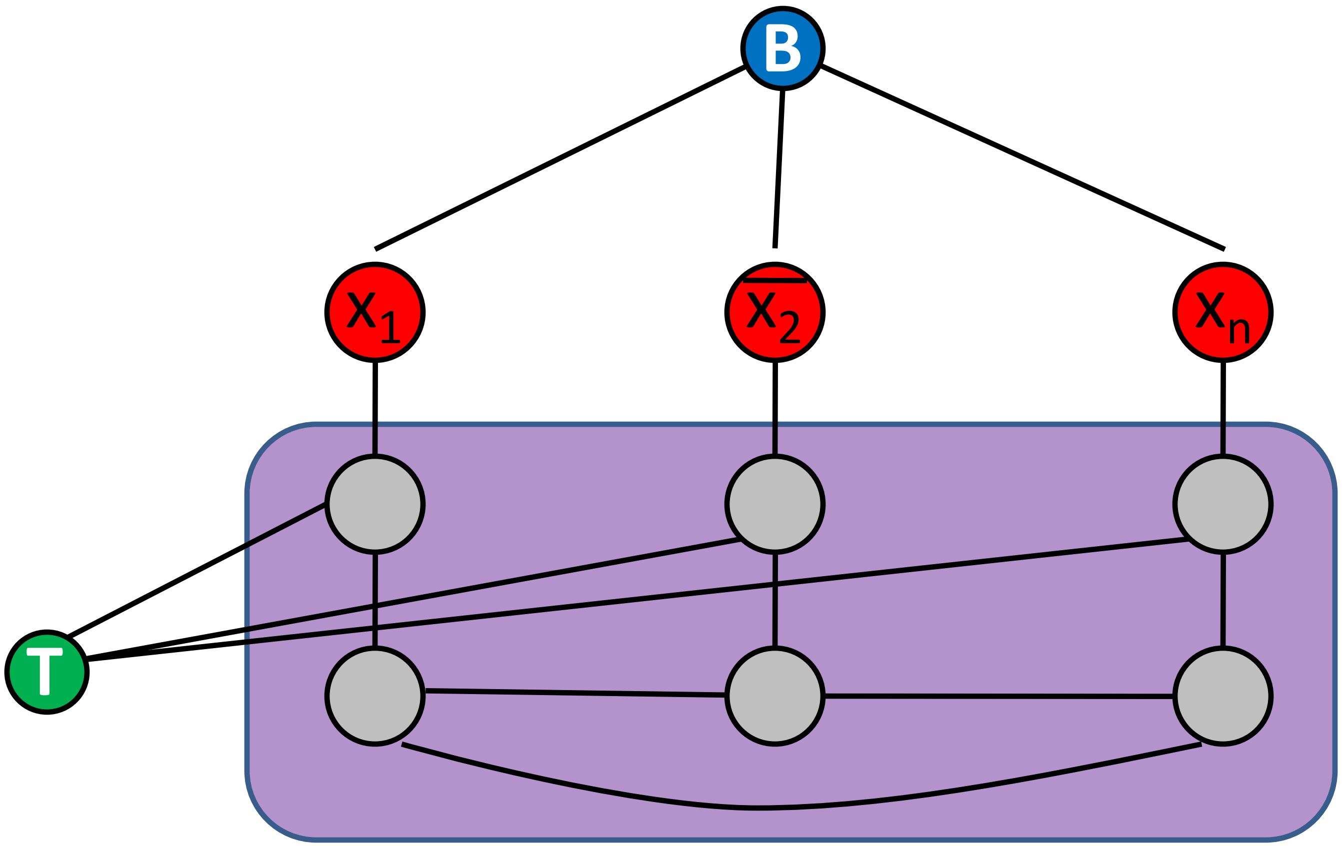


Claim: 3-colorable iff terms colored to satisfy clause

3-Coloring is NP-Complete – clause gadgets

Claim: 3-colorable iff terms colored to satisfy clause

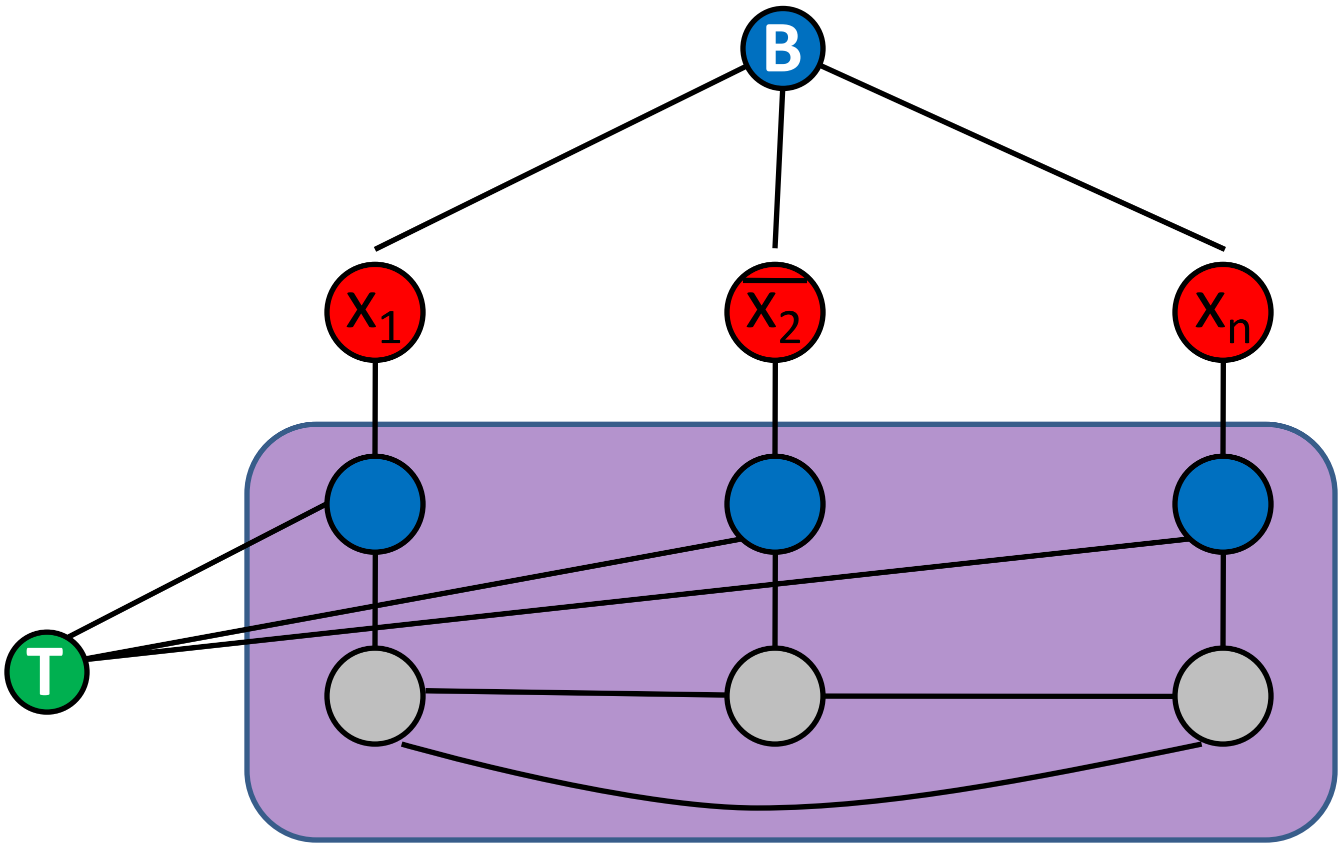
1. If terms all colored F, then cannot 3-color



3-Coloring is NP-Complete – clause gadgets

Claim: 3-colorable iff terms colored to satisfy clause

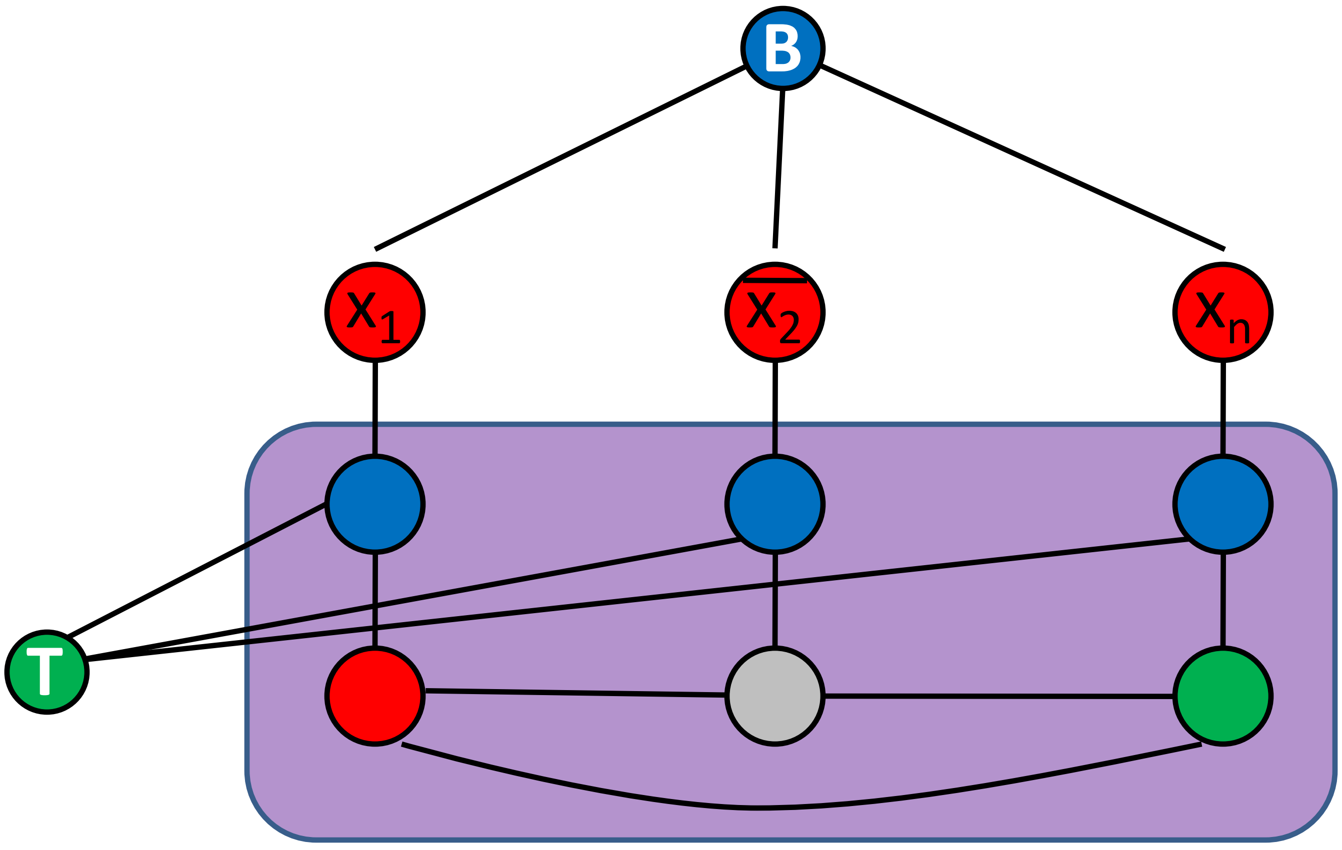
1. If terms all colored F, then cannot 3-color



3-Coloring is NP-Complete – clause gadgets

Claim: 3-colorable iff terms colored to satisfy clause

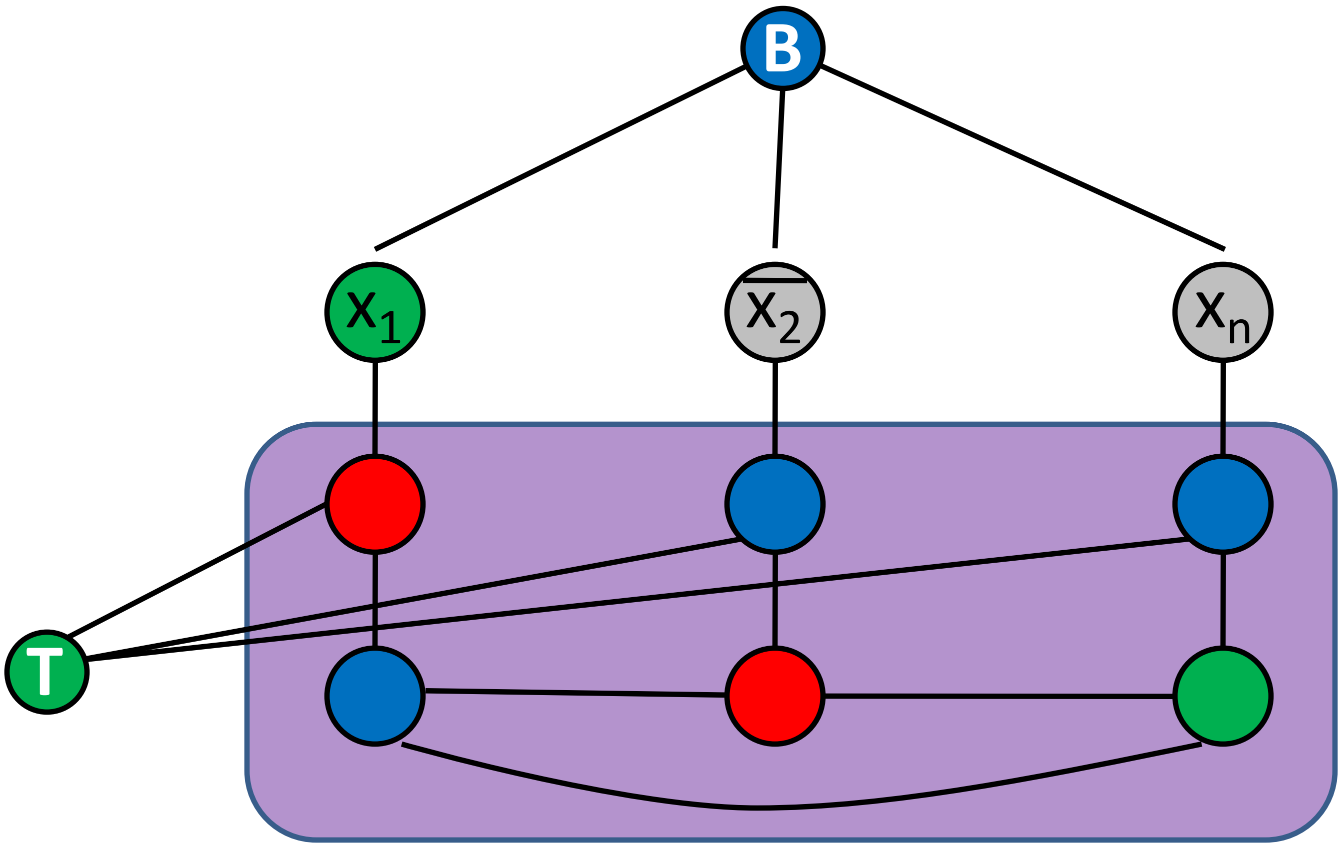
1. If terms all colored F, then cannot 3-color



3-Coloring is NP-Complete – clause gadgets

Claim: 3-colorable iff terms colored to satisfy clause

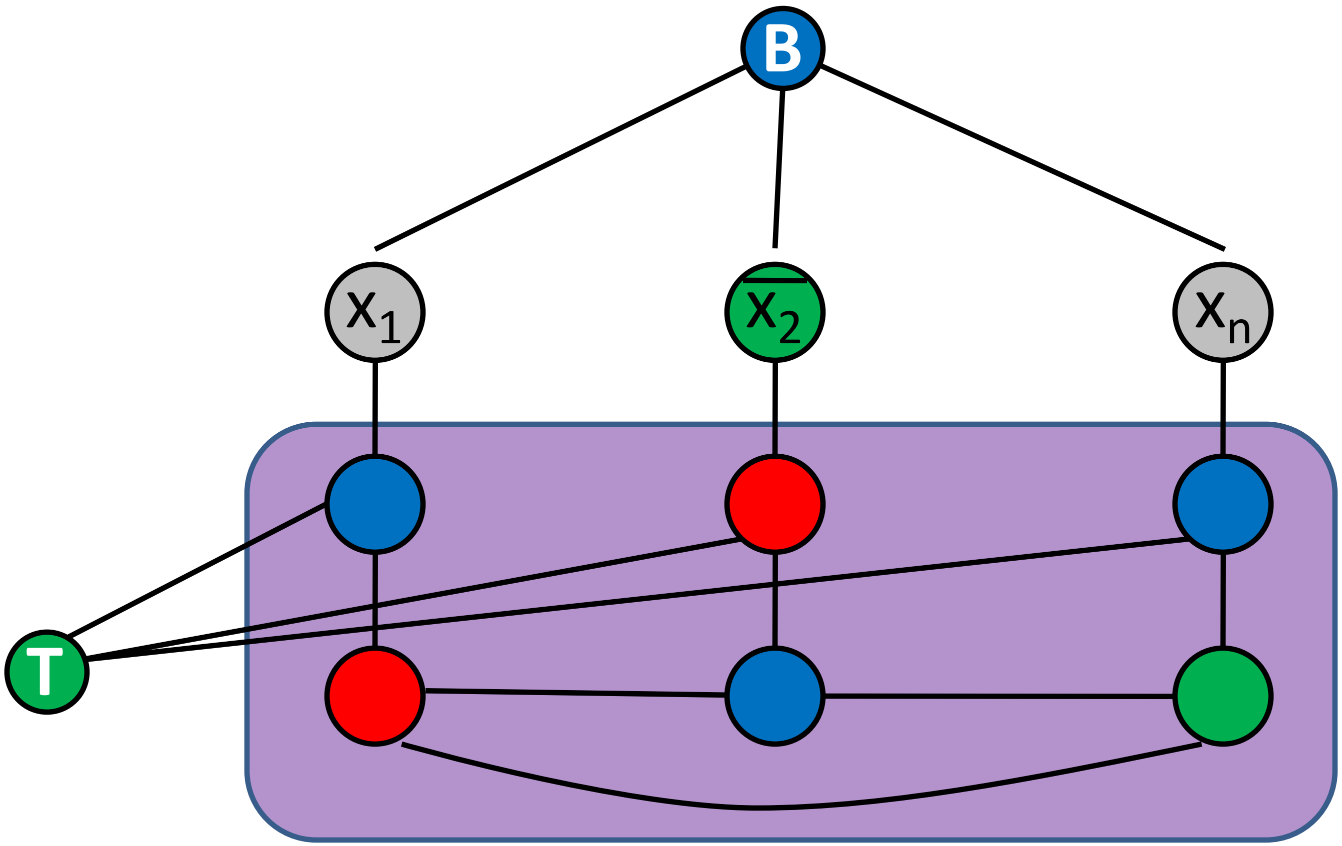
2. If some term true, can 3-color



3-Coloring is NP-Complete – clause gadgets

Claim: 3-colorable iff terms colored to satisfy clause

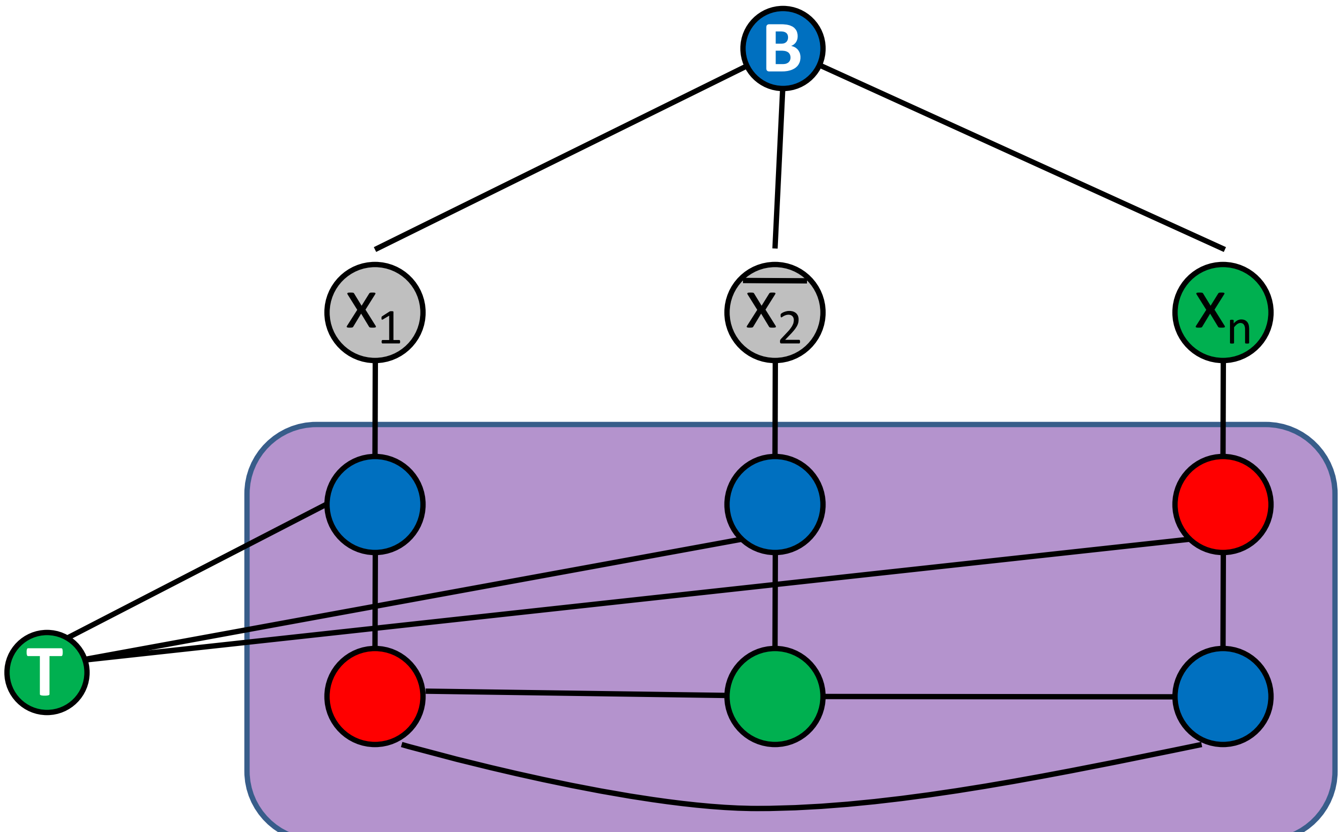
2. If some term true, can 3-color



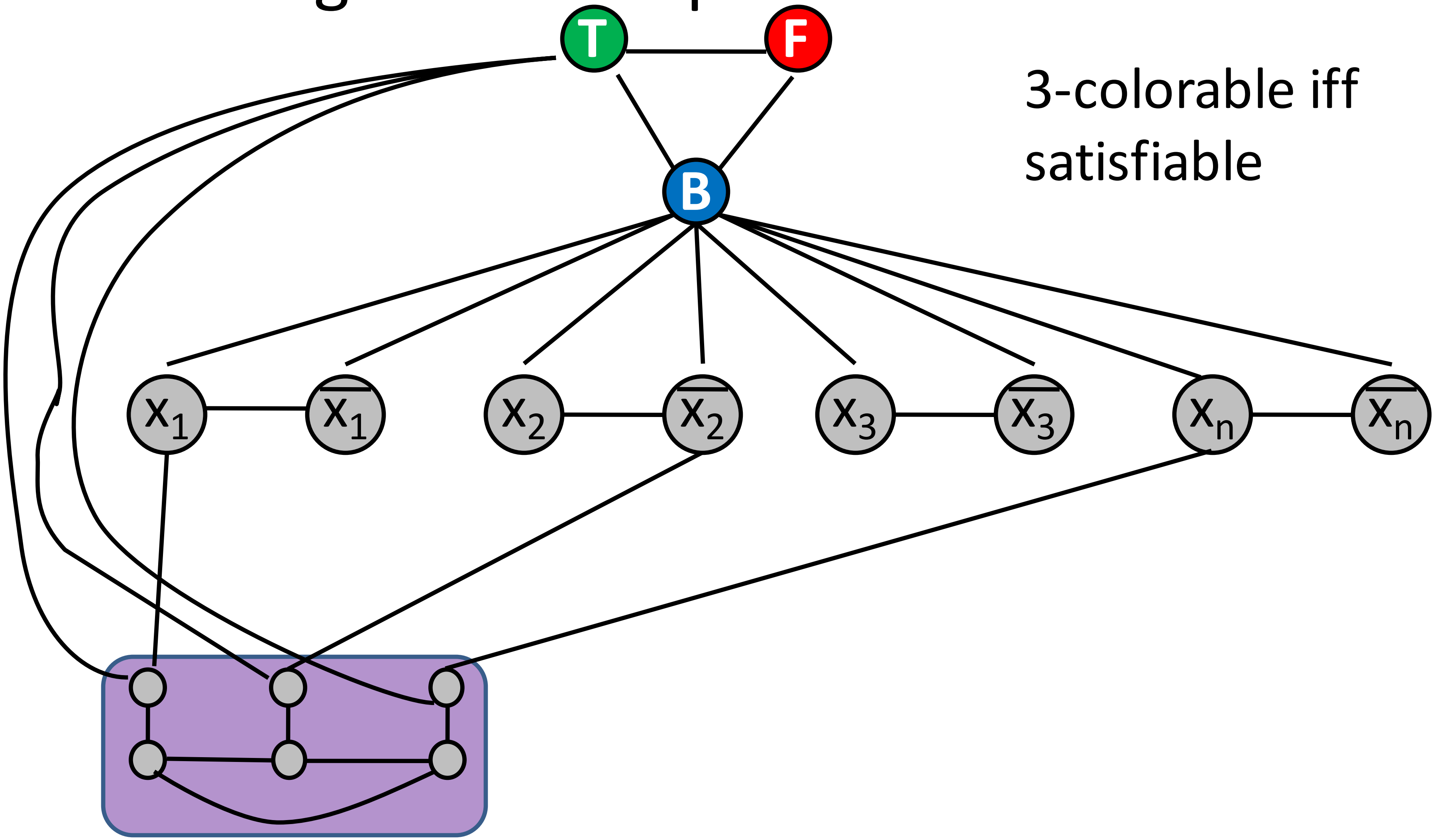
3-Coloring is NP-Complete – clause gadgets

Claim: 3-colorable iff terms colored to satisfy clause

2. If some term true, can 3-color



3-Coloring is NP-Complete

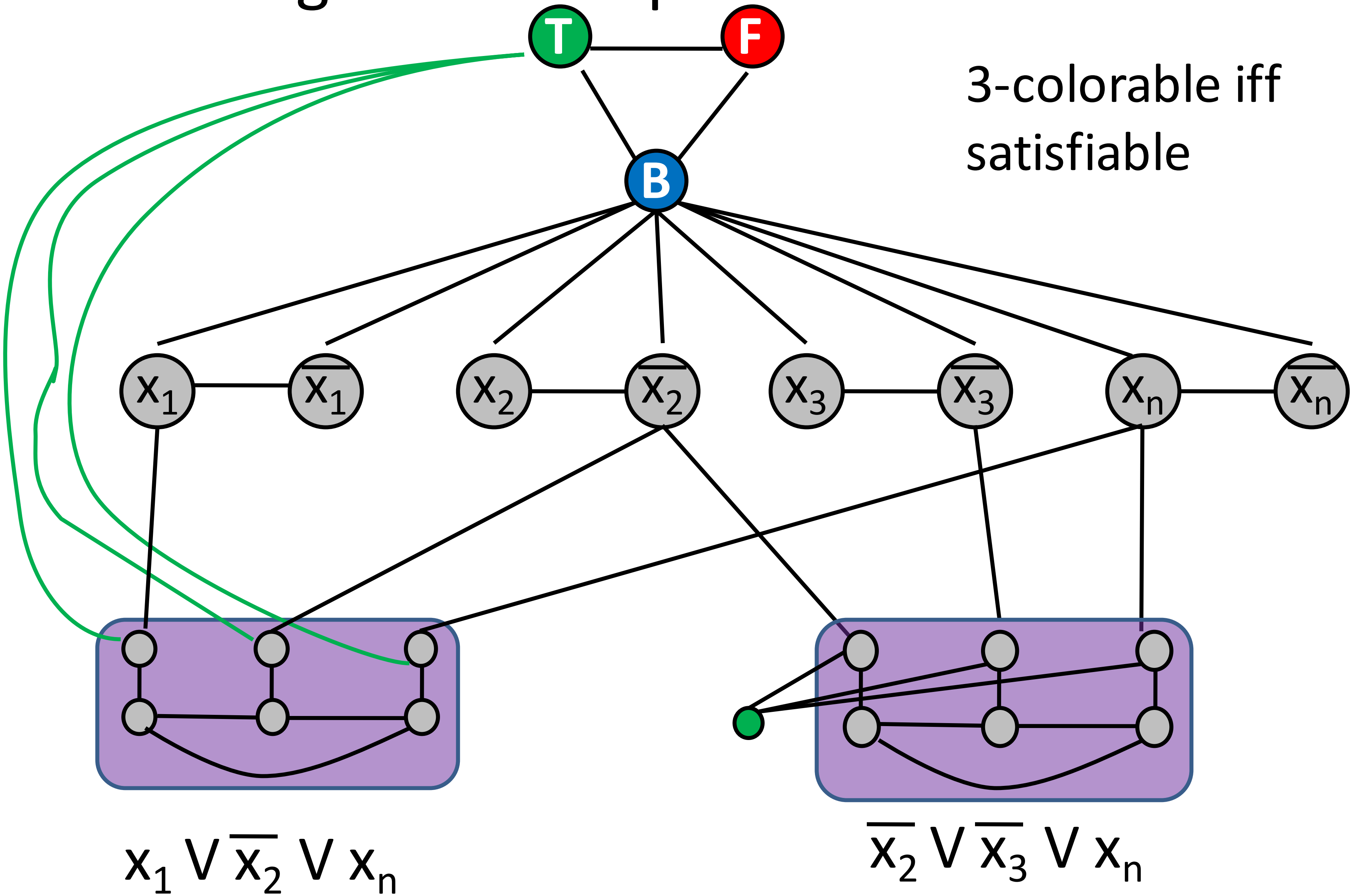


3-colorable iff satisfiable

$x_1 \vee \overline{x_2} \vee x_n$

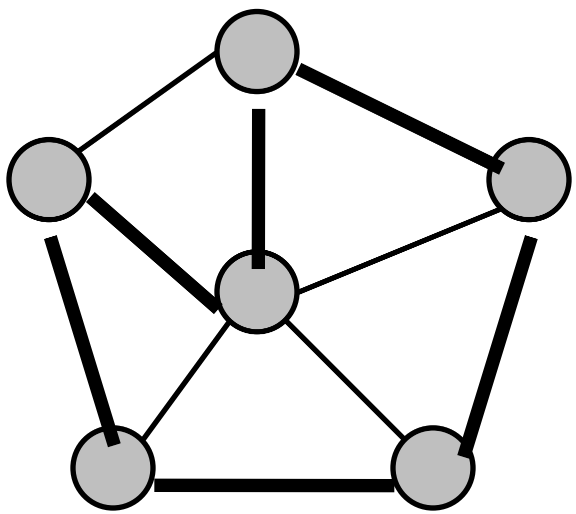
3-Coloring is NP-Complete

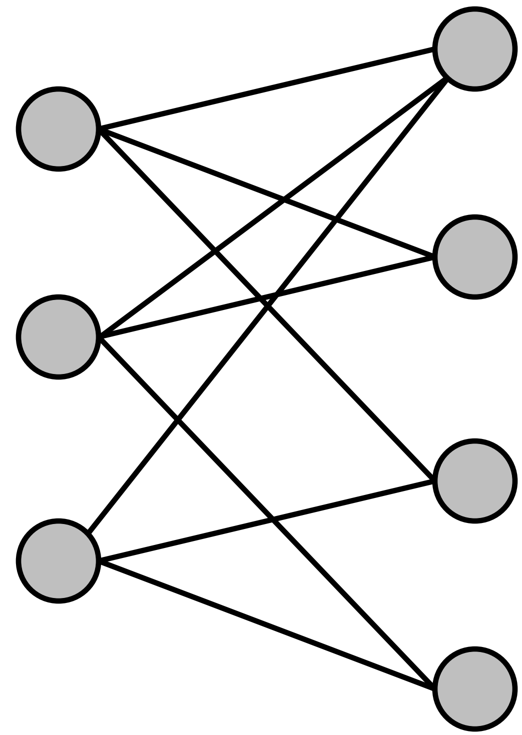
3-colorable iff satisfiable



Hamiltonian Cycle:

A cycle in a graph that hits each vertex once.





Directed Hamiltonian Cycle:

same, but in a directed graph

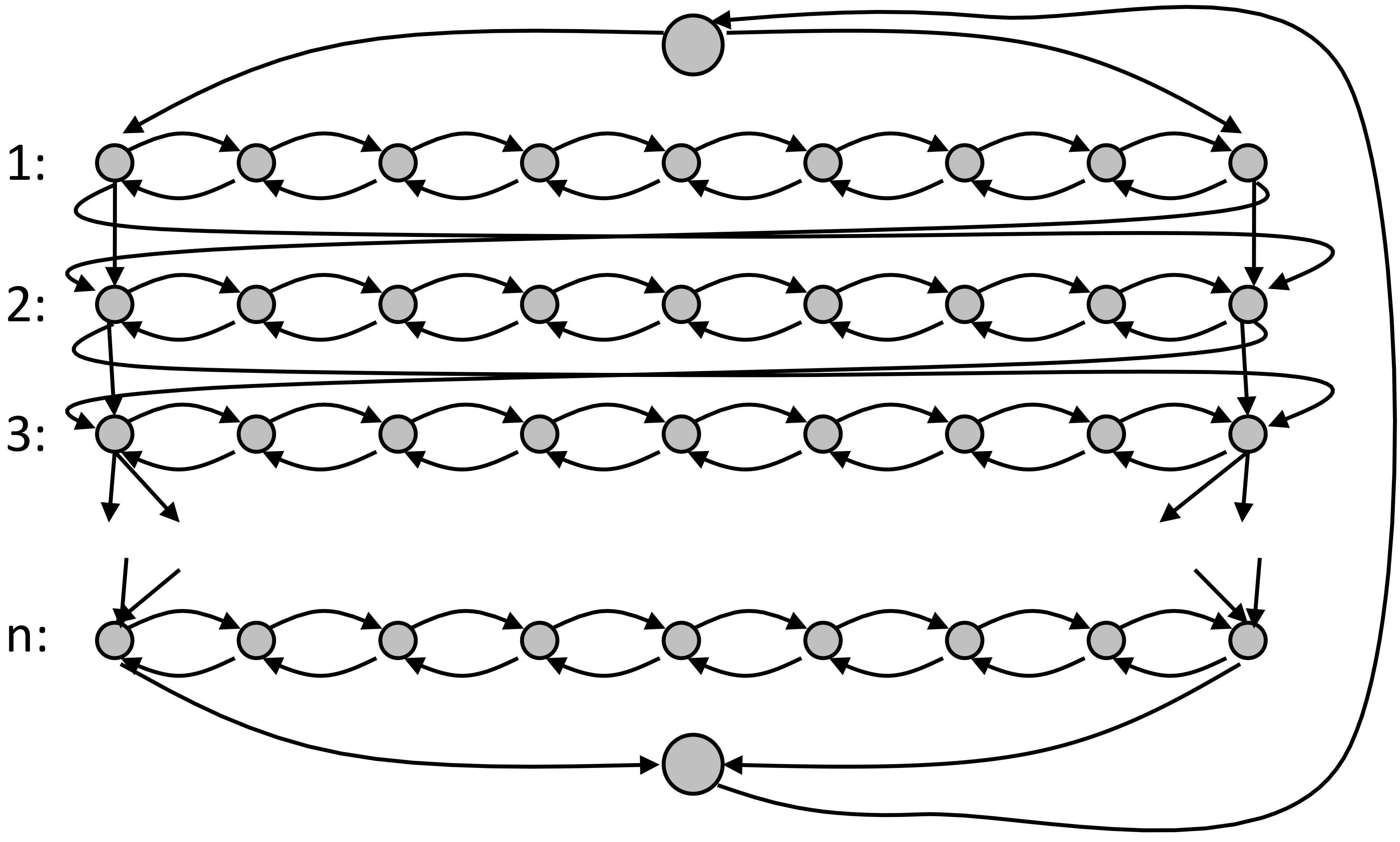
Directed Ham Cycle is NP-Complete

Clearly in NP, because can check if a cycle is
Hamiltonian

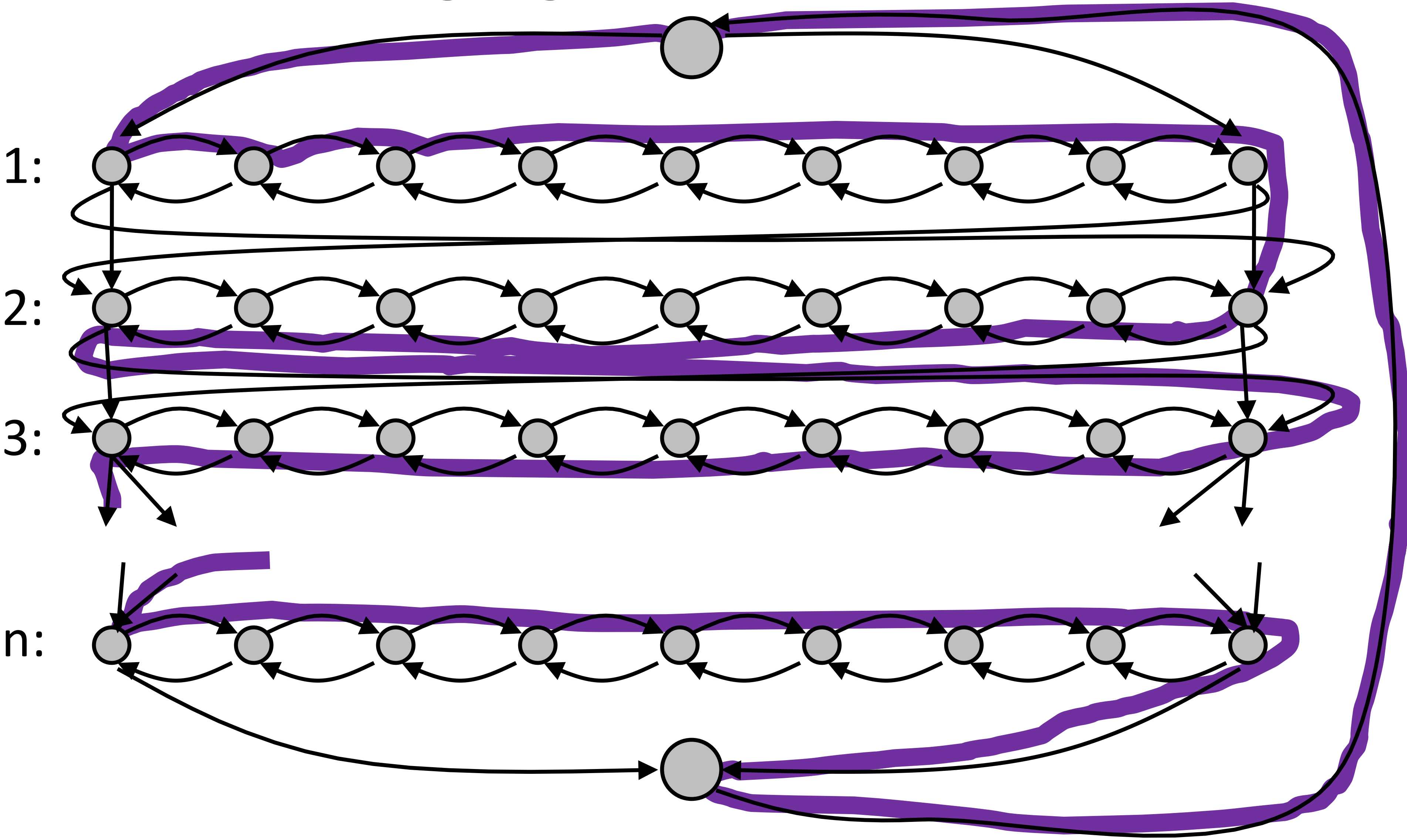
To prove NP-hard, will show
$3\text{-SAT} \leq_P$ Directed Ham Cycle

Produce directed graph $G = (V,E)$ that
has Ham Cycle iff the clauses are satisfiable

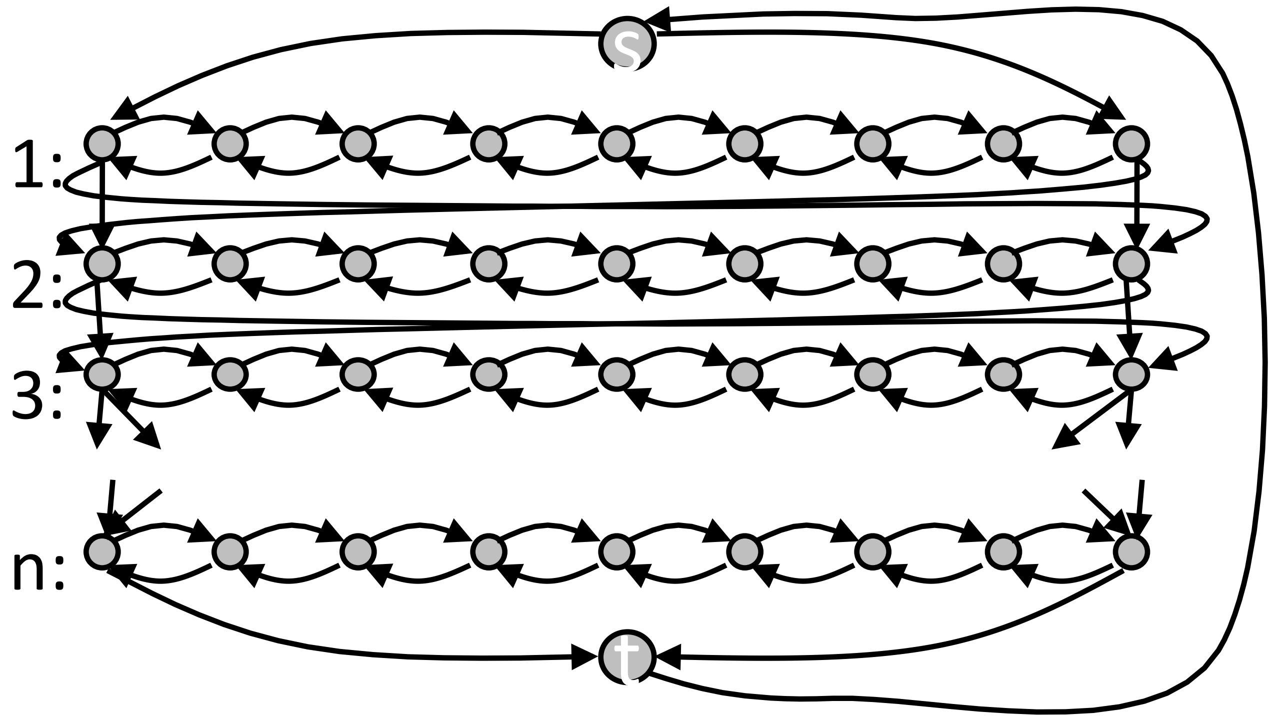
Start: create graph with 2^n Ham Cycles, then create gadgets to restrict them



Start: create graph with 2^n Ham Cycles, then create gadgets to restrict them

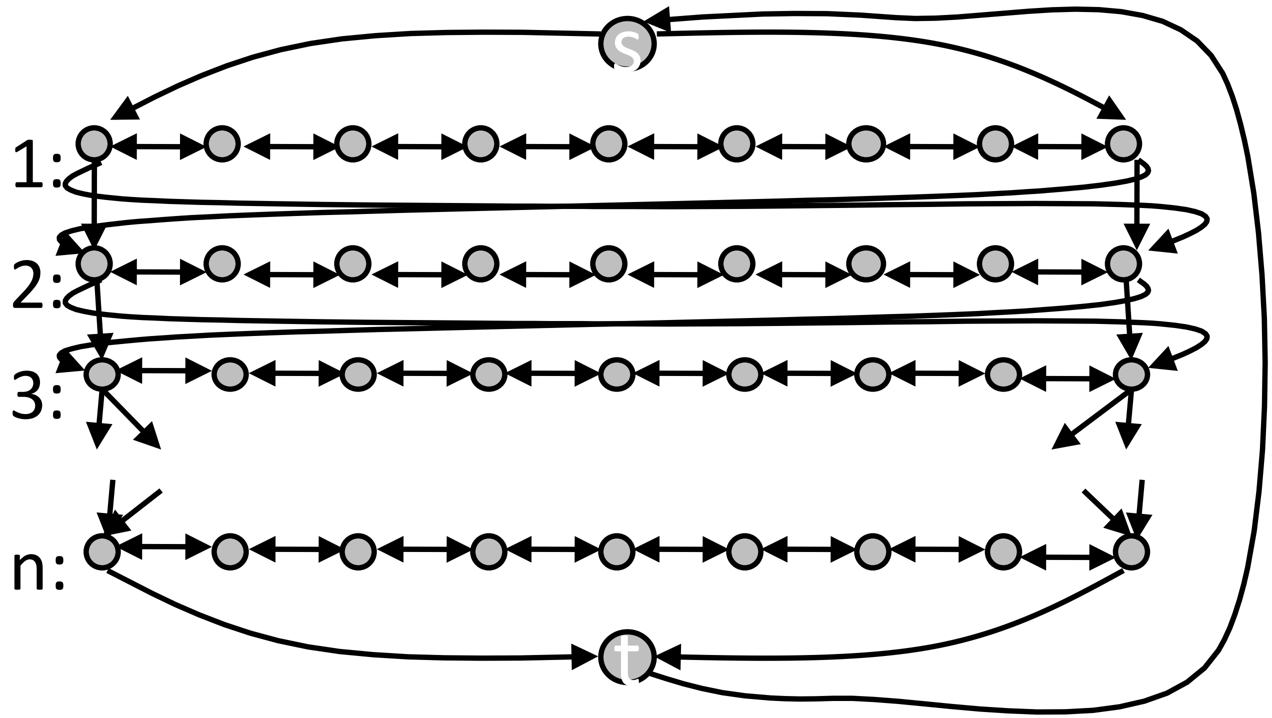


Start: create graph with 2^n Ham Cycles, then create gadgets to restrict them



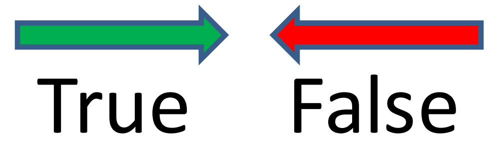
Must go top-to-bottom, and can traverse each row left-to-right (True) or right-to-left (False)

Start: create graph with 2^n Ham Cycles, then create gadgets to restrict them



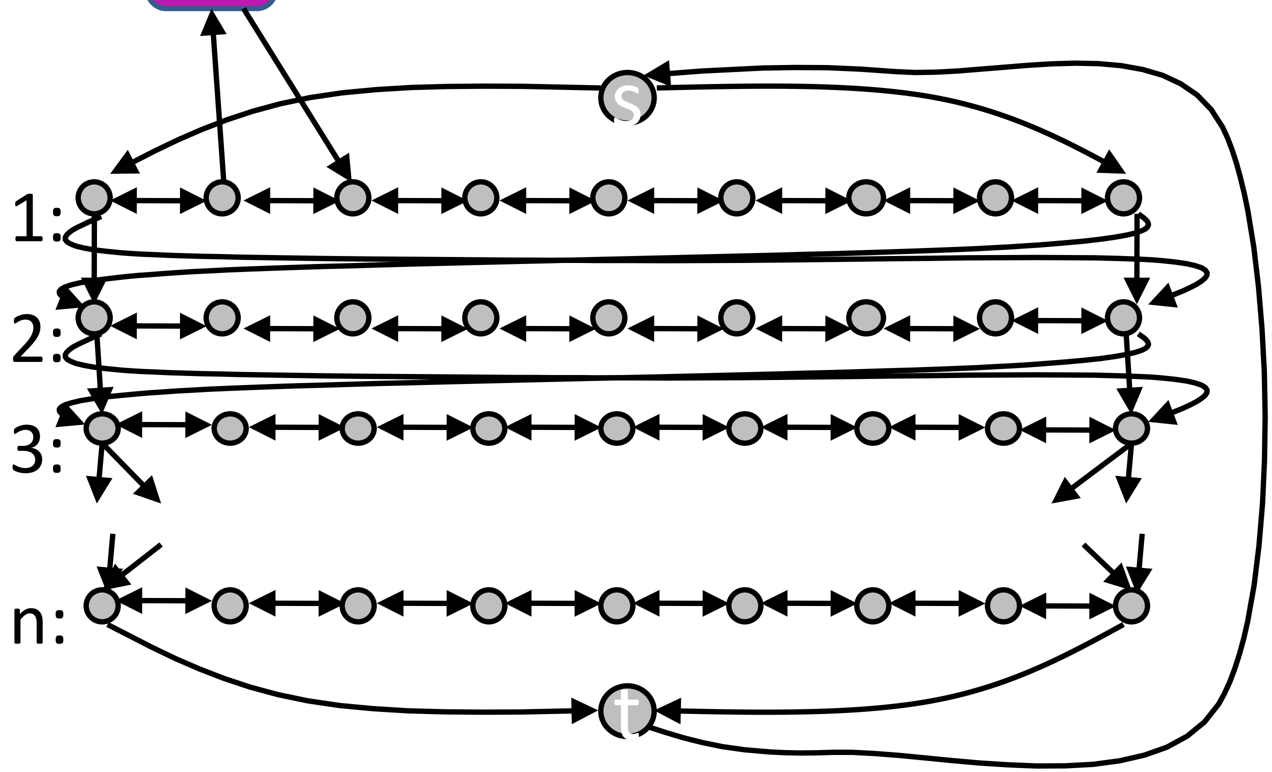
Must go top-to-bottom, and can traverse each row left-to-right (True) or right-to-left (False)

Clause gadgets



clause

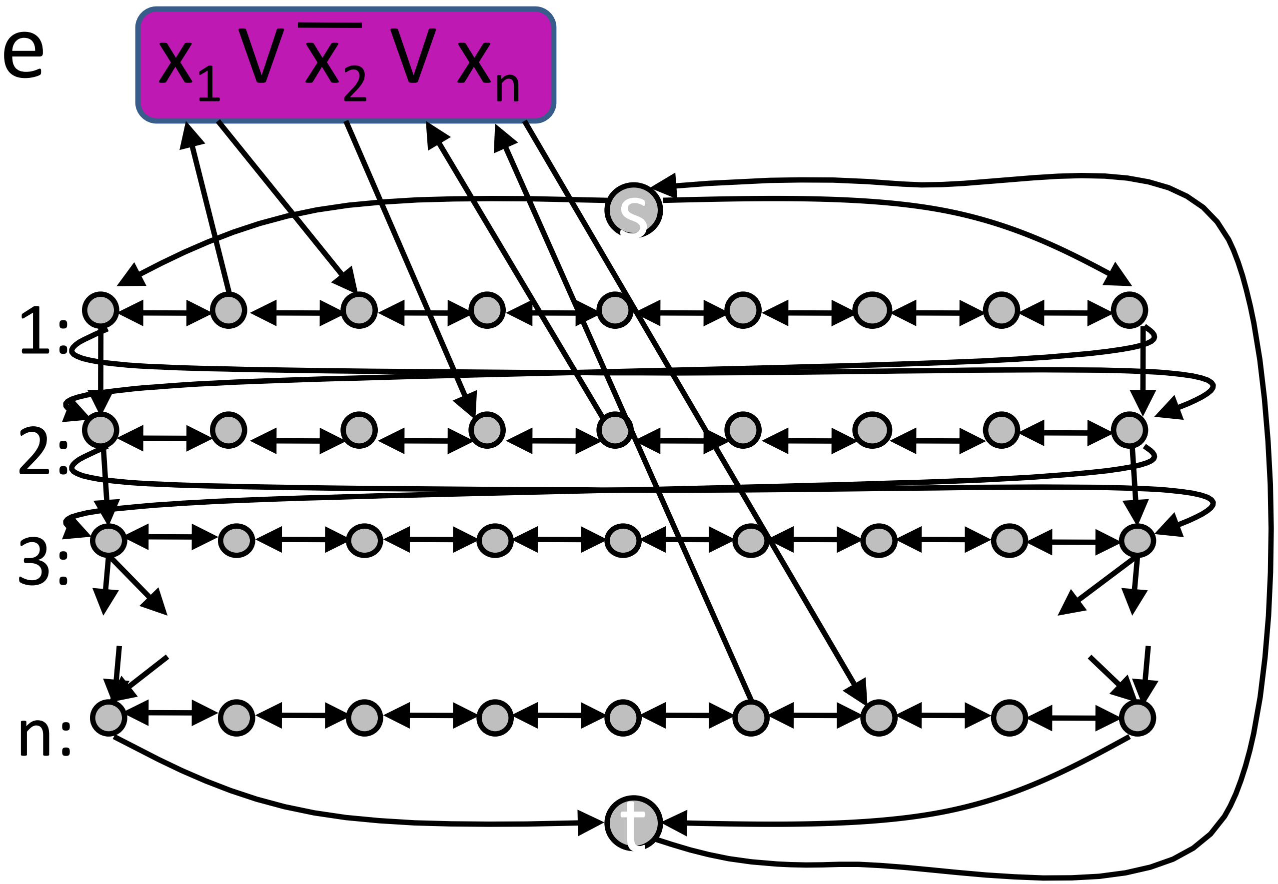
forces traverse first row



Clause gadgets

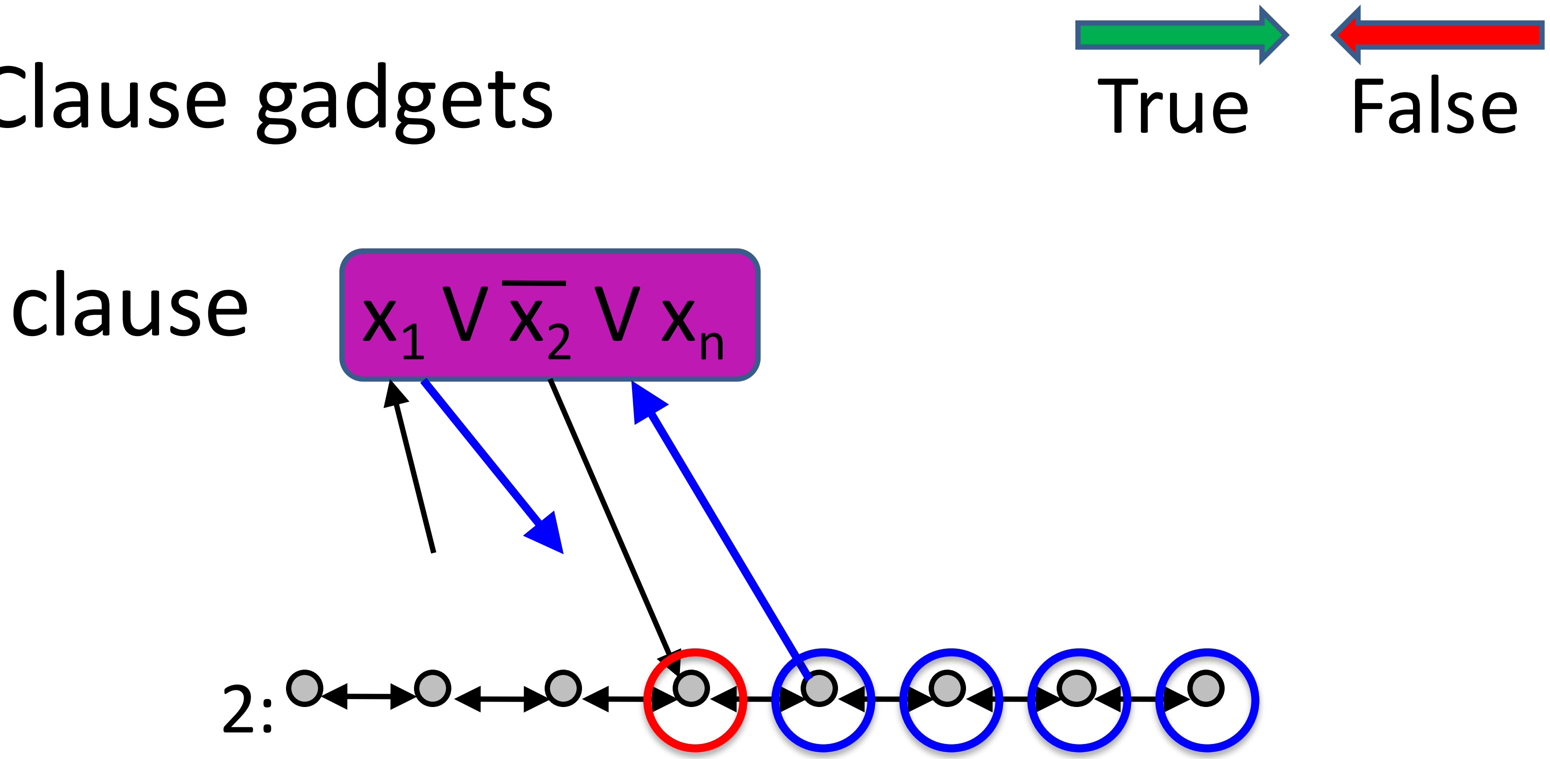
True False

clause



Forces traverse 1, or 2, or n

Clause gadgets

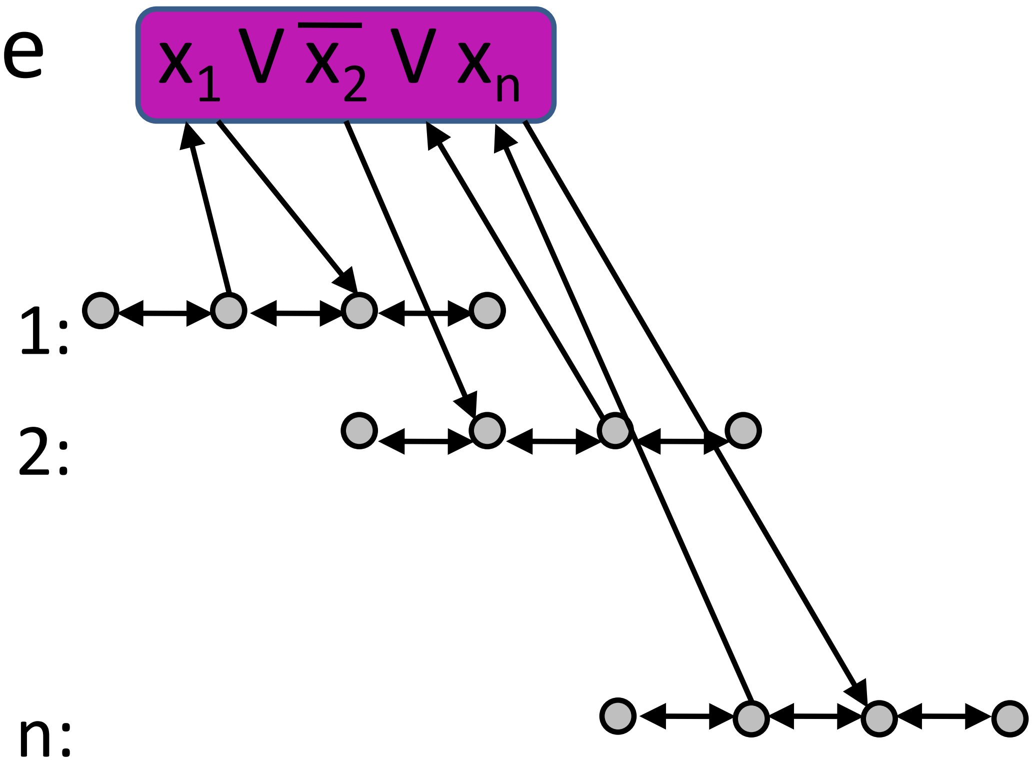


To see must come back to same row, note that if do not is no hamiltonian path through unused down-link

Clause gadgets

→ True ← False

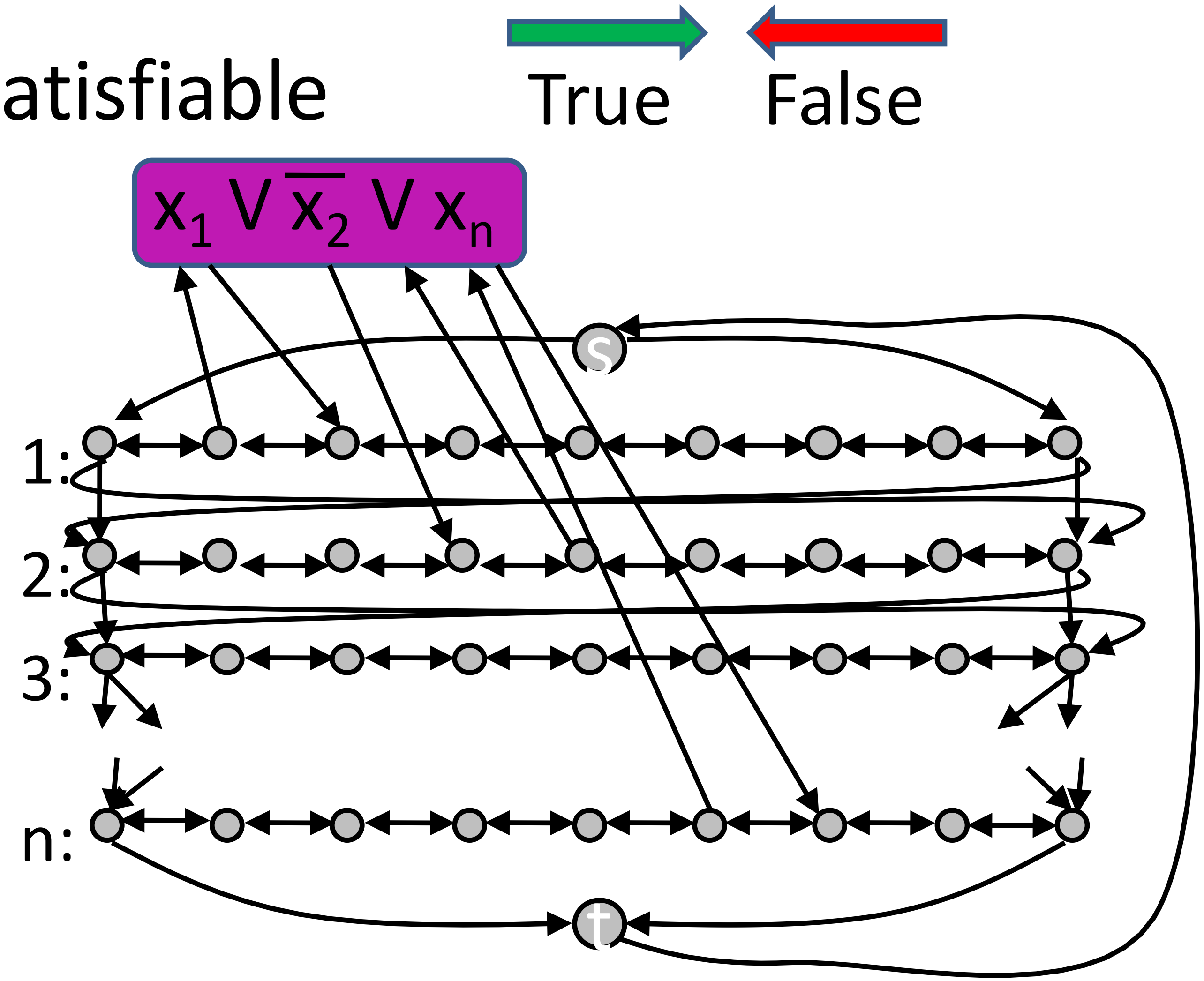
clause



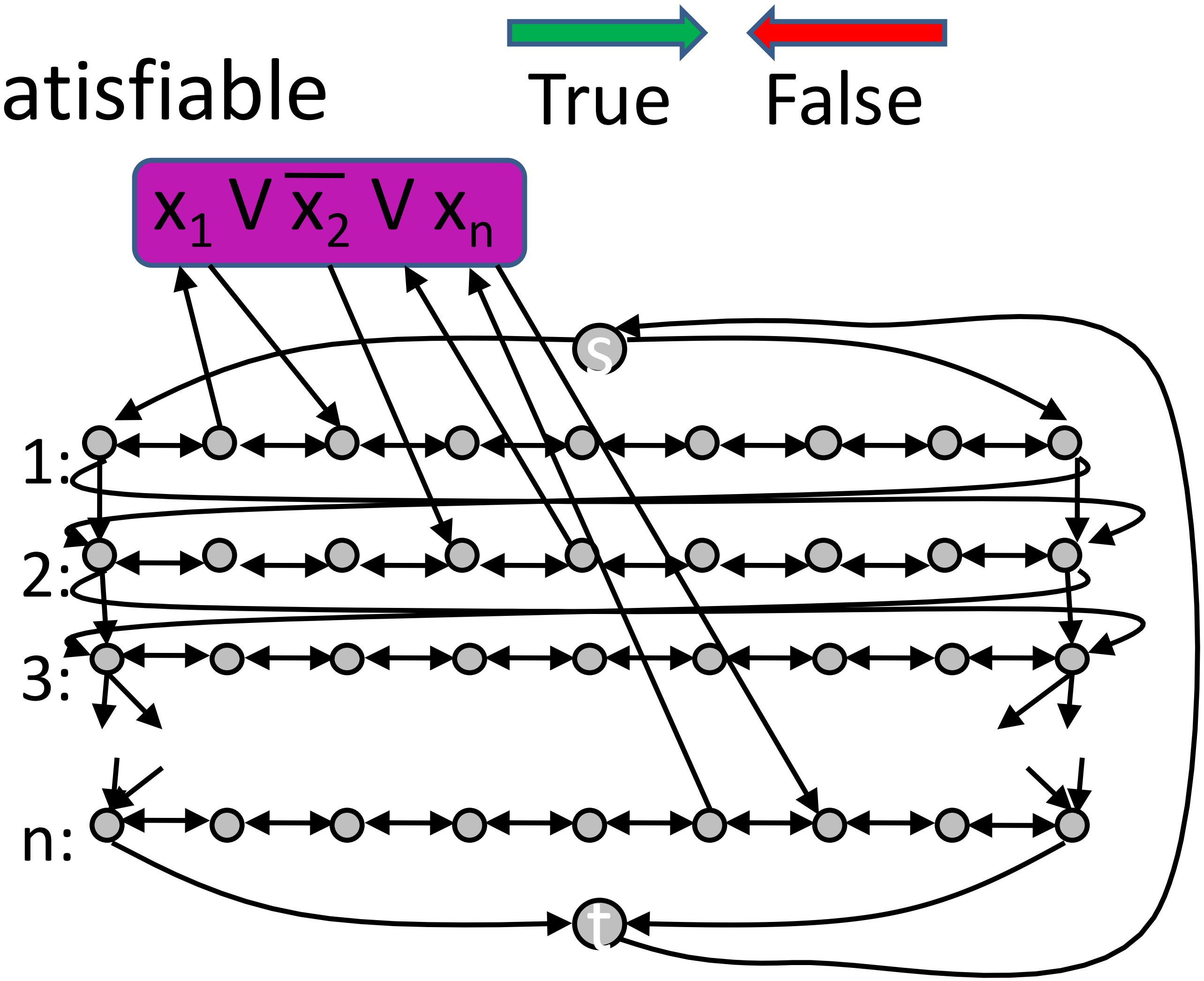
Forces traverse 1 →, or 2 ←, or n →

Ham cycle iff satisfiable

Pf. If satisfiable, traverse in order indicated by vars, picking up each clause once using some true term.



Ham cycle iff satisfiable



Pf. If Ham Cycle, must go top to bottom

assign vars by direction

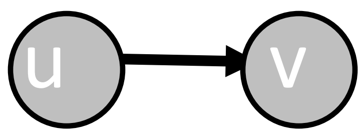
if visit each clause node, then is made true by term on row from which make the visit.

Directed Ham Cycle $\leq_P$ Ham Cycle

1. In directed problem,
   answer same if reverse all arrows.

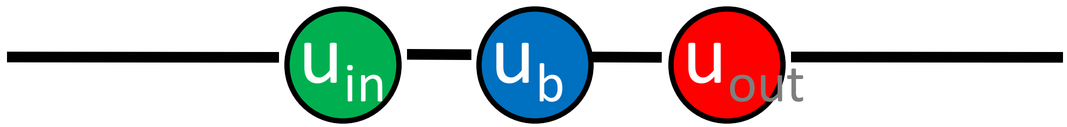
2. To transform to undirected,
   replace each vertex v with three vertices:

v_{in}, v_{base}, v_{out}



Replace directed (u,v) edge with (u_{out} , v_{in})

Directed Ham Cycle $\leq_P$ Ham Cycle

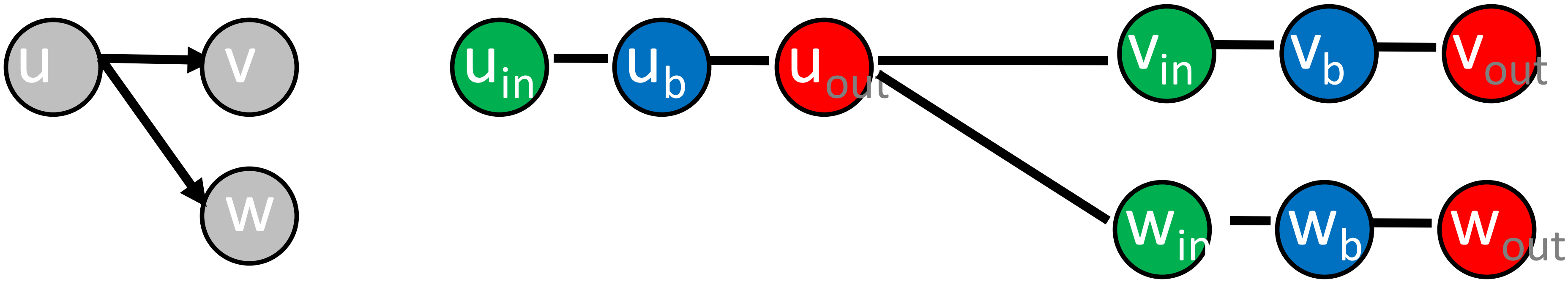


Claim: If these are only edges to u_b, then in every Hamiltonian cycle u_b must be adjacent to u_{in}

Proof: if it is not, then once enter u_b can not get out

Directed Ham Cycle $\leq_P$ Ham Cycle

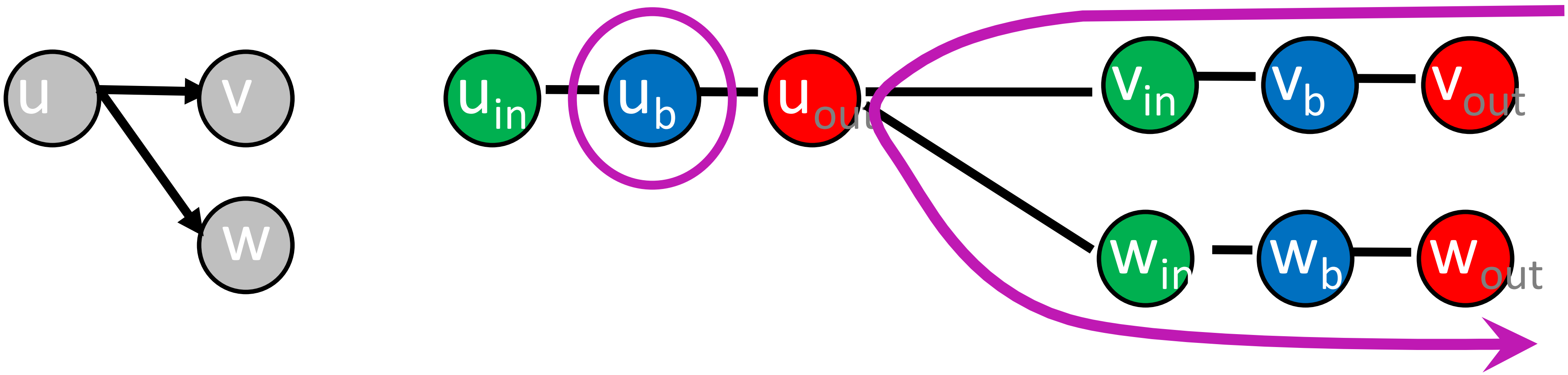
Replace directed (u,v) edge with (u_{out} , v_{in})



Directed Ham Cycle in original -> Ham Cycle

Directed Ham Cycle $\leq_P$ Ham Cycle

Replace directed (u,v) edge with (u_{out} , v_{in})

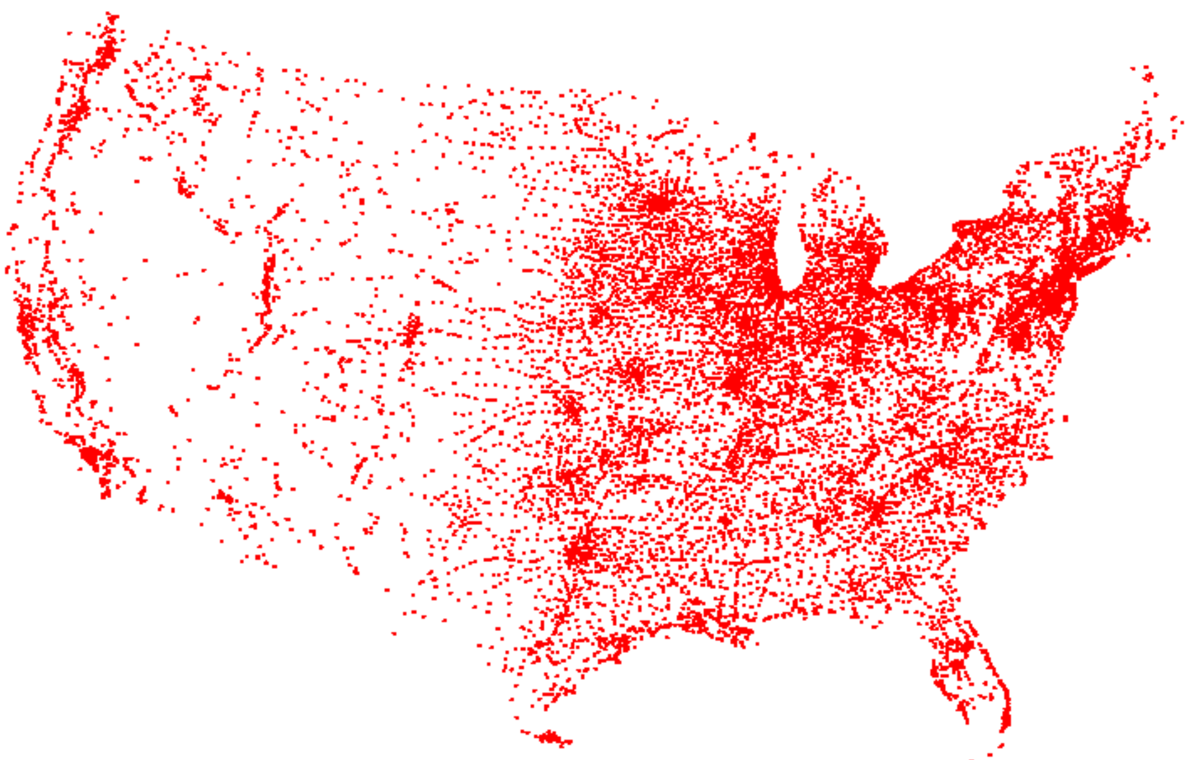


Lemma:

Every Ham Cycle in the undirected graph must go in, base, out, in, base, out, in, base, out, etc, must correspond to a Ham Cyc in directed graph

TSP (Travelling Salesperson Problem)

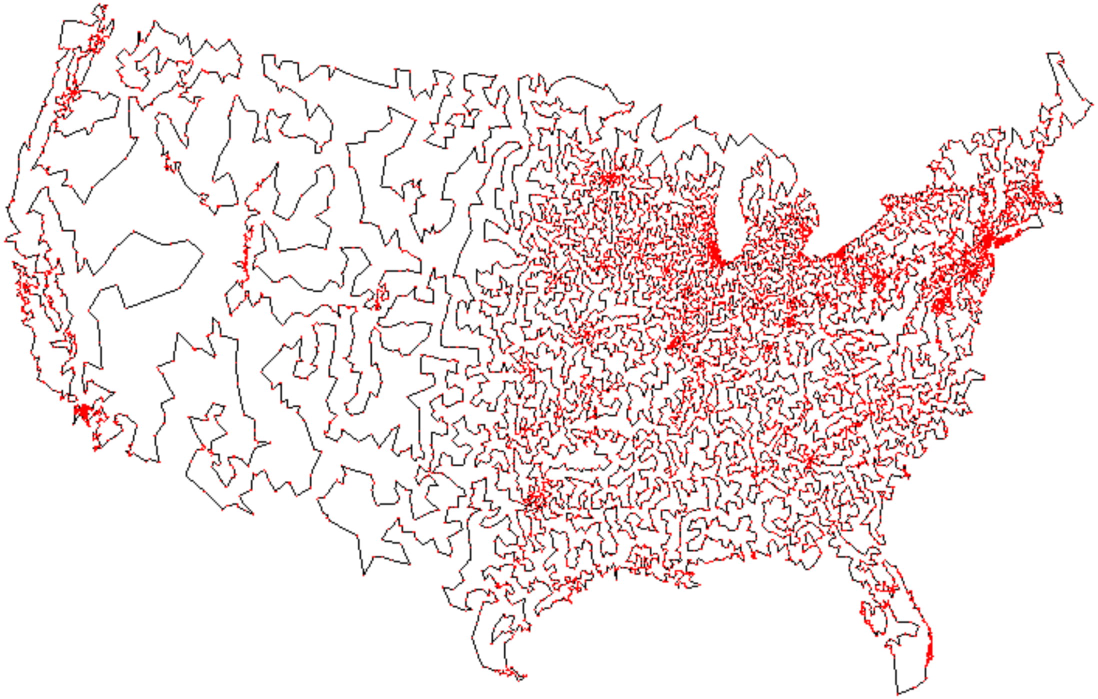
Given n locations, a distance function d(u,v) and a total distance D, does there exist a tour through all locations of total distance at most L?



http://www.tsp.gatech.edu/usa13509/usa13509_info.html

TSP (Travelling Salesperson Problem)

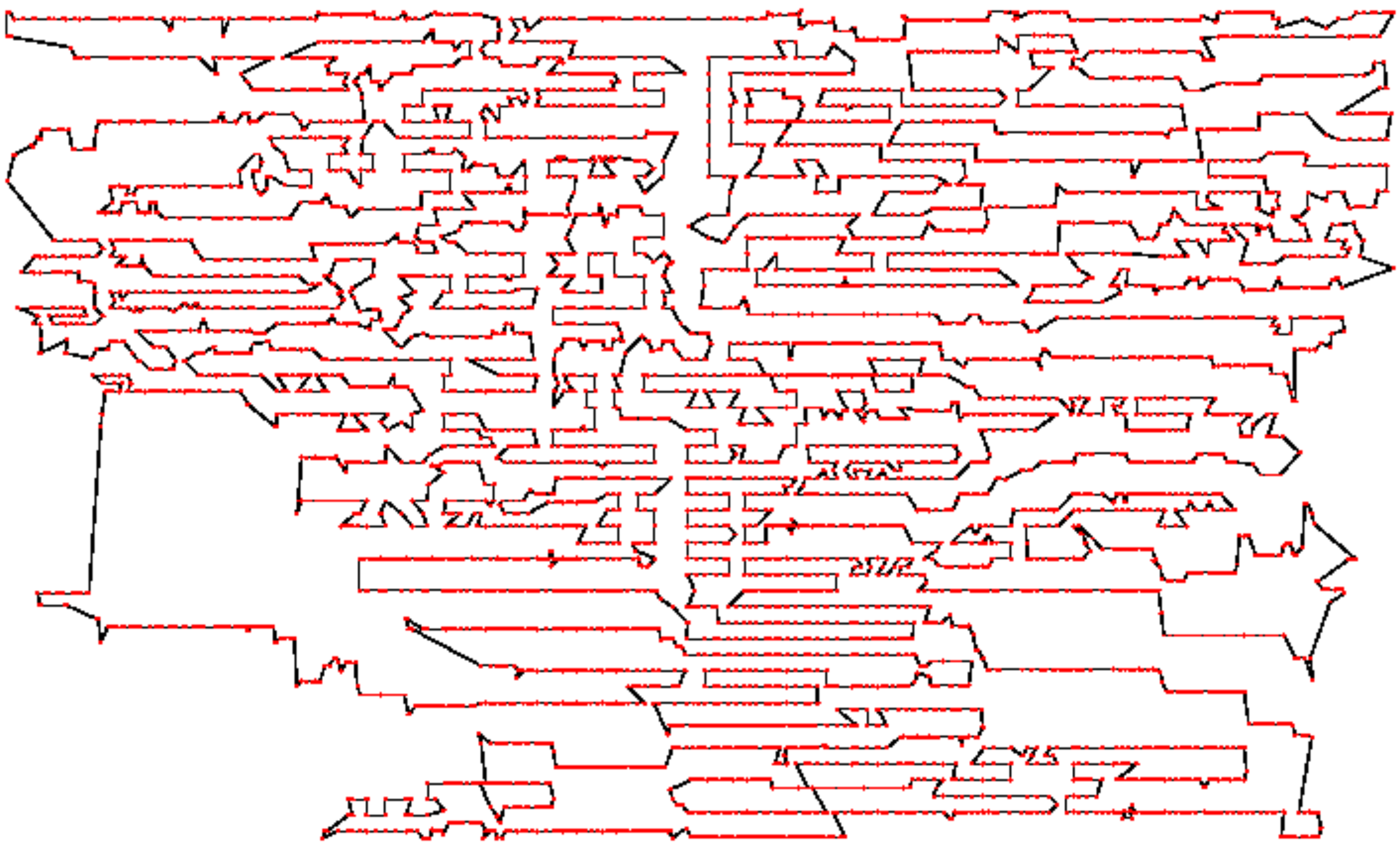
Given n locations, a distance function d(u,v) and a total distance D, does there exist a tour through all locations of total distance at most L?



http://www.tsp.gatech.edu/usa13509/usa13509_sol.html

RL5915 optimal solution

An optimal solution for RL5915 is given by the following tour, which has length 565530.



http://www.tsp.gatech.edu/rl5915/rl5915_sol.html

TSP is NP-complete

Ham Cycle $\leq_P$ TSP

Given graph $G = (V,E)$,
create one location for each vertex,

$$d(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 2 & \text{otherwise} \end{cases}$$

Target distance = $|V|$

A tour of all locations that returns to start and has total length $|V|$ must use exactly $|V|$ edges of G

TSP is NP-complete

Ham Cycle $\leq_P$ TSP

Given graph $G = (V,E)$,
create one location for each vertex,

$$d(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 2 & \text{otherwise} \end{cases}$$

This is an abstract distance function.

TSP is NP-complete

Ham Cycle $\leq_P$ TSP

Given graph $G = (V,E)$,
create one location for each vertex,

$$d(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 2 & \text{otherwise} \end{cases}$$

This is an abstract distance function.

Remains NP-hard for integer points in plane.

Issue with Planar TSP

If input is locations of points, instead of distances

The problem is not known to be in NP, because do not know if can compare distances in polynomial time.

For integers $x_1, \ldots, x_n$ integer t, do not have poly time algorithm to test if

$$\sum_i \sqrt{x_i} \leq t$$

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.

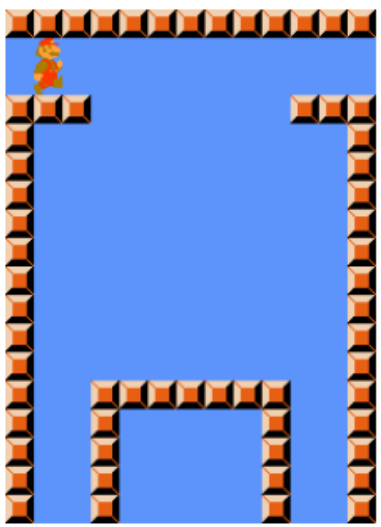


Figure 4: Variable gadget for Mario

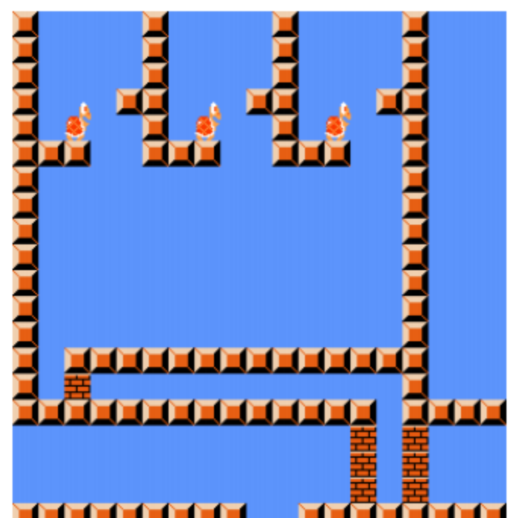


Figure 5: Clause gadget for Mario

Candy Crush is NP-hard

Toby Walsh

NICTA and University of NSW, Sydney, Australia

Abstract

We prove that playing Candy Crush to achieve a given score in a fixed number of swaps is NP-hard.

Keywords: computational complexity, NP-completeness, Candy Crush.

Bejeweled, Candy Crush and other Match-Three Games are (NP-)Hard

L. Gualà[1], S. Leucci[2], and E. Natale[3]

[1]Università degli Studi di Roma *Tor Vergata*
`guala@mat.uniroma2.it`
[2]Università degli Studi dell'Aquila
`stefano.leucci@univaq.it`
[3]*Sapienza* Università di Roma
`natale@di.uniroma1.it`

March 25, 2014