Strategy

- We have seen that 3SAT is NP-complete.
- We now reduce 3SAT to some other problems.

Hamiltonian Path

Theorem: HAMPATH is NP-complete.

Must demonstrate

- HAMPATH $\in NP$
- every $A \in NP$ poly-time reducible to HAMPATH.

Note that

- we have already seen the first
- we will show that 3SAT $\leq_P$ HAMPATH

Hamiltonian Path

For any 3cnf formula ϕ,

- we construct a graph G
- with vertices s and t
- such that a hamiltonian path exists from s to t iff ϕ is satisfiable.

Hamiltonian Path

Here is a 3cnf formula ϕ:

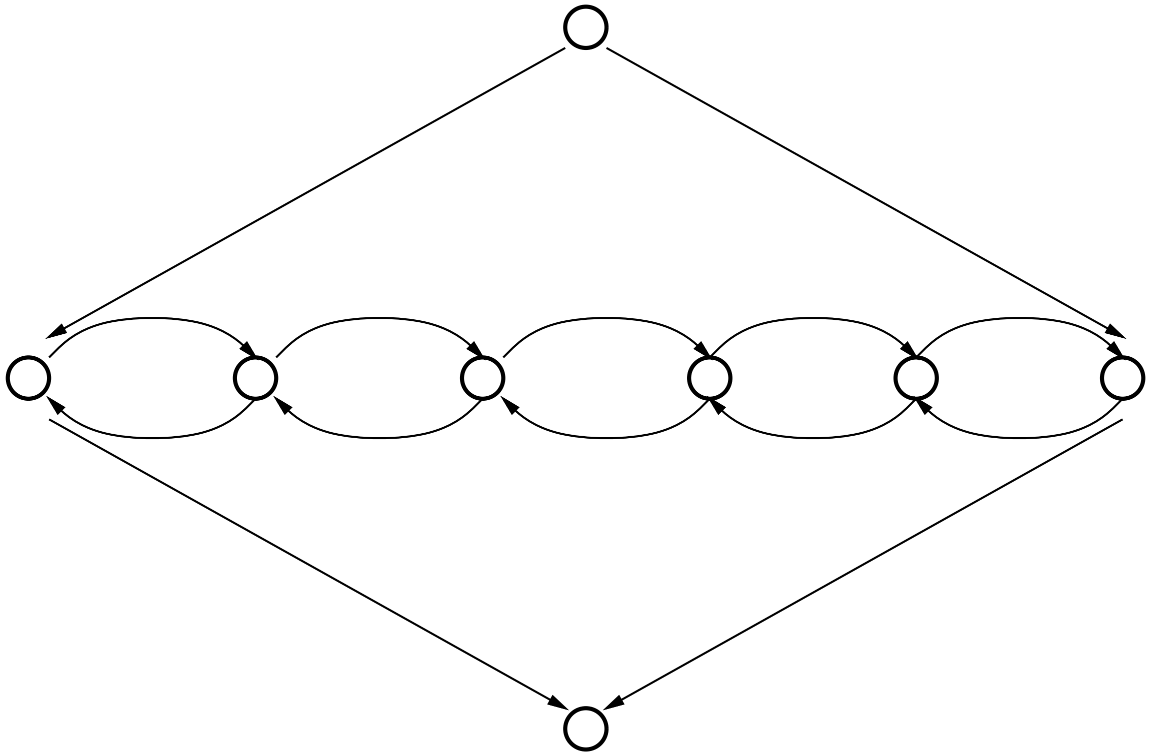
$$(a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots (a_k \vee b_k \vee c_k) \wedge$$

where

- each a_i, b_i, c_i is x_i or $\overline{x_i}$
- the ℓ variables are $x_1, \ldots, x_\ell$.

Hamiltonian Path

Each variable is represented by the following graph:

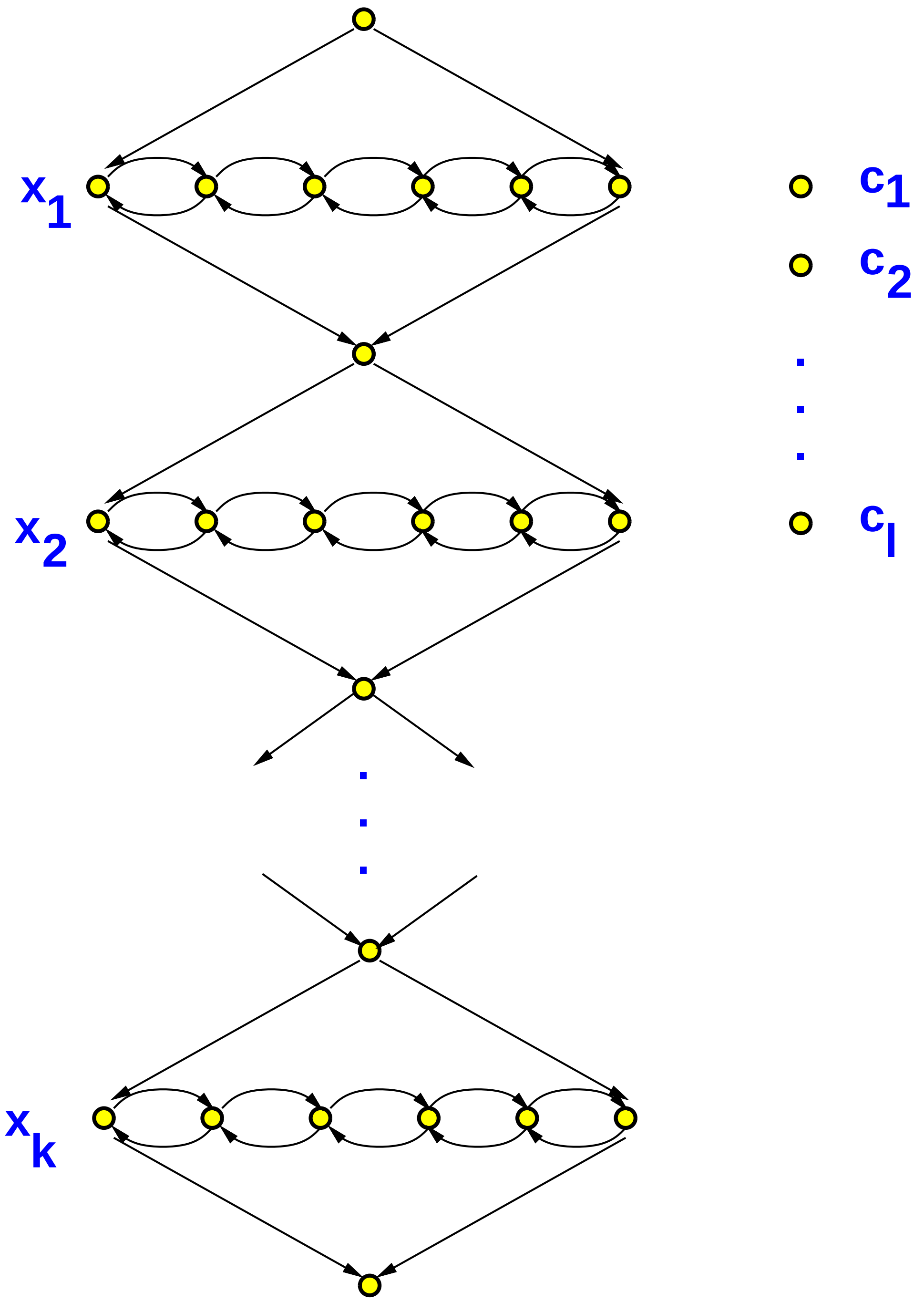


Hamiltonian Path

Each clause of ϕ is a single node.

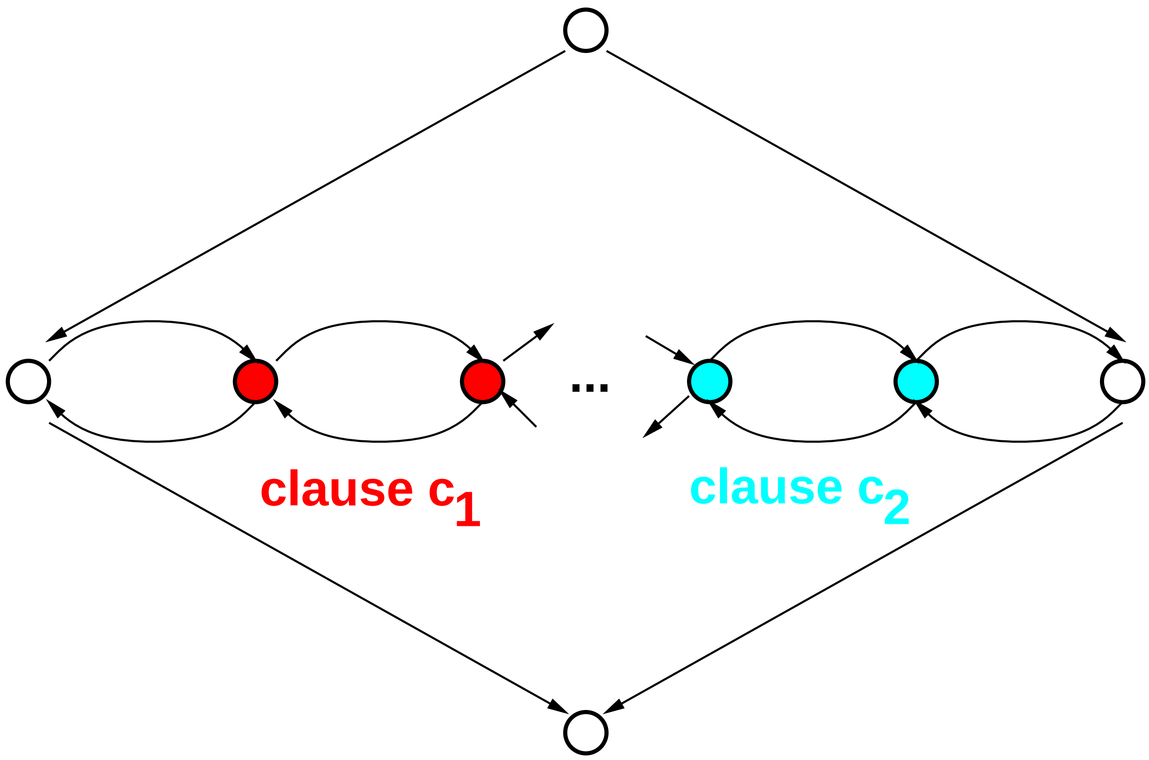
Hamiltonian Path

Global structure of graph (missing edges)



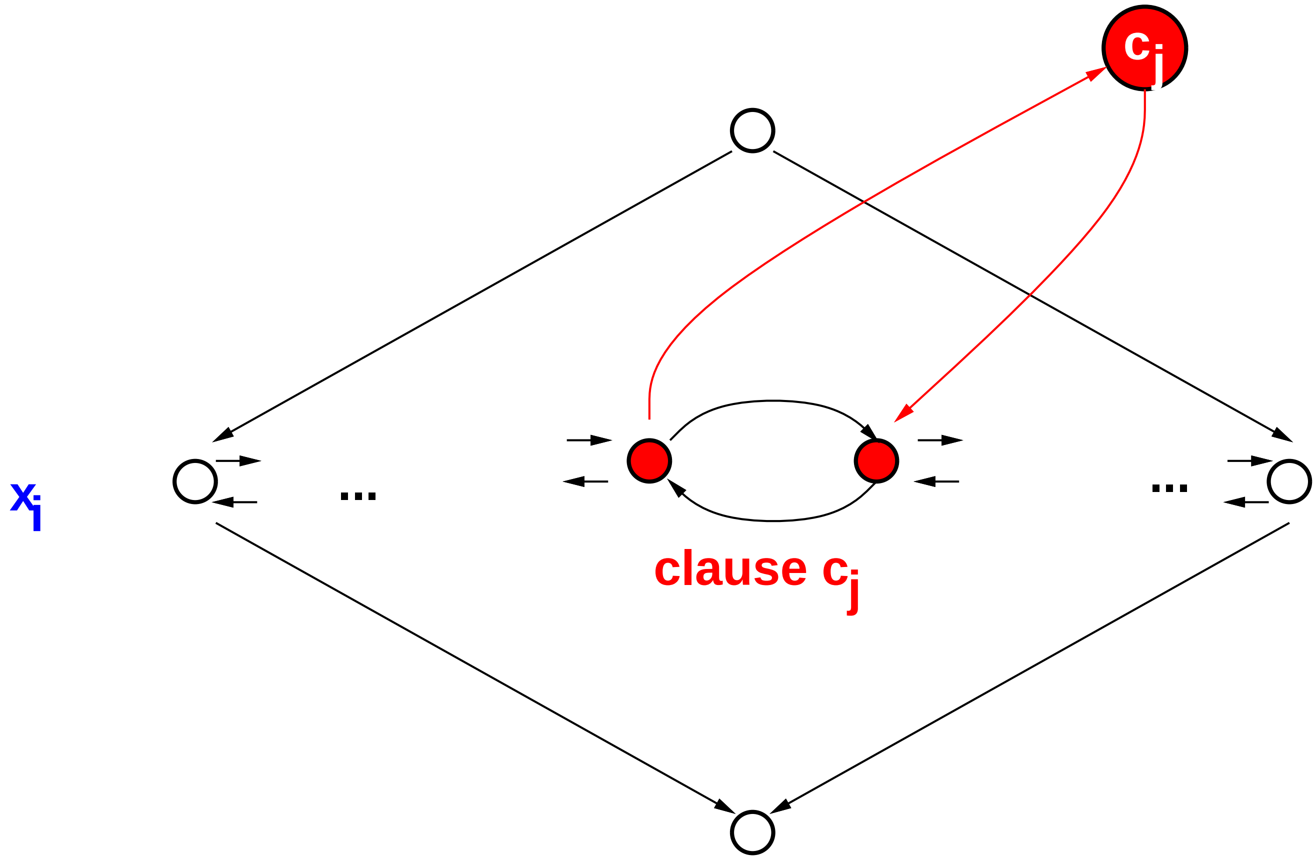
Hamiltonian Path

Center of each diamond has $2k$ nodes, one for each clause.



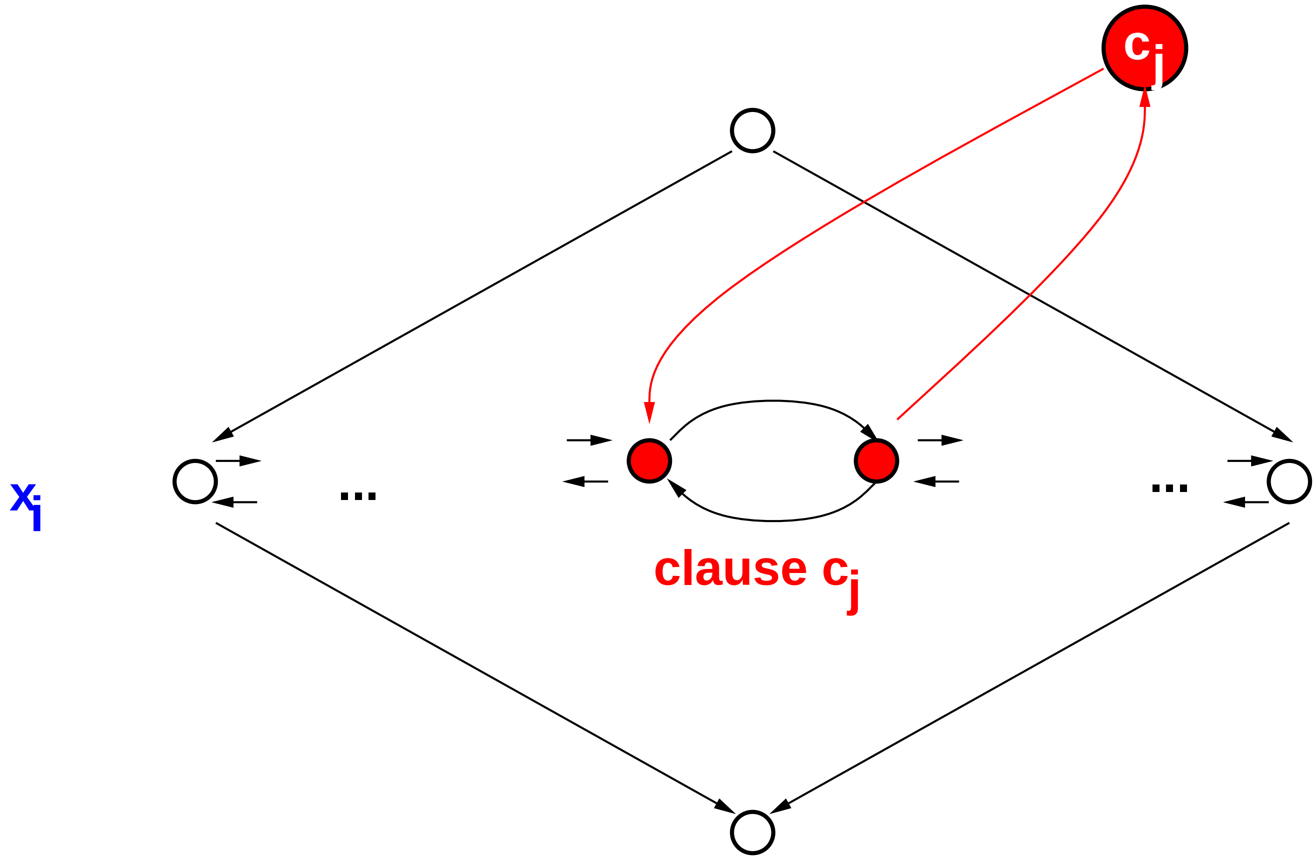
Hamiltonian Path

If variable x_i appears in clause c_j, add this "detour"



Hamiltonian Path

If $\overline{x_i}$ appears in clause c_j, add this "detour"



Hamiltonian Path

After adding edges from “diamonds” to clause vertexes, G is complete.

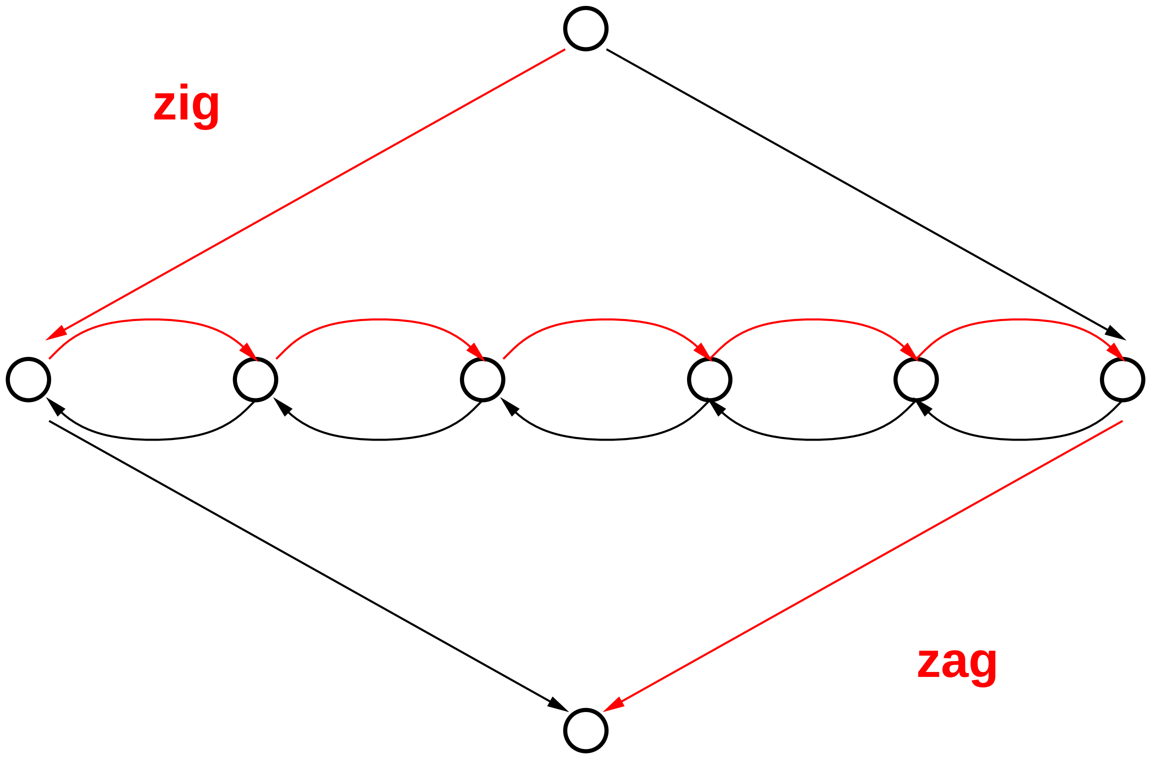
Claim: If ϕ is satisfiable, then G has a hamiltonian path.

Strategy:

- ignore clause nodes for now
- traverse diamonds

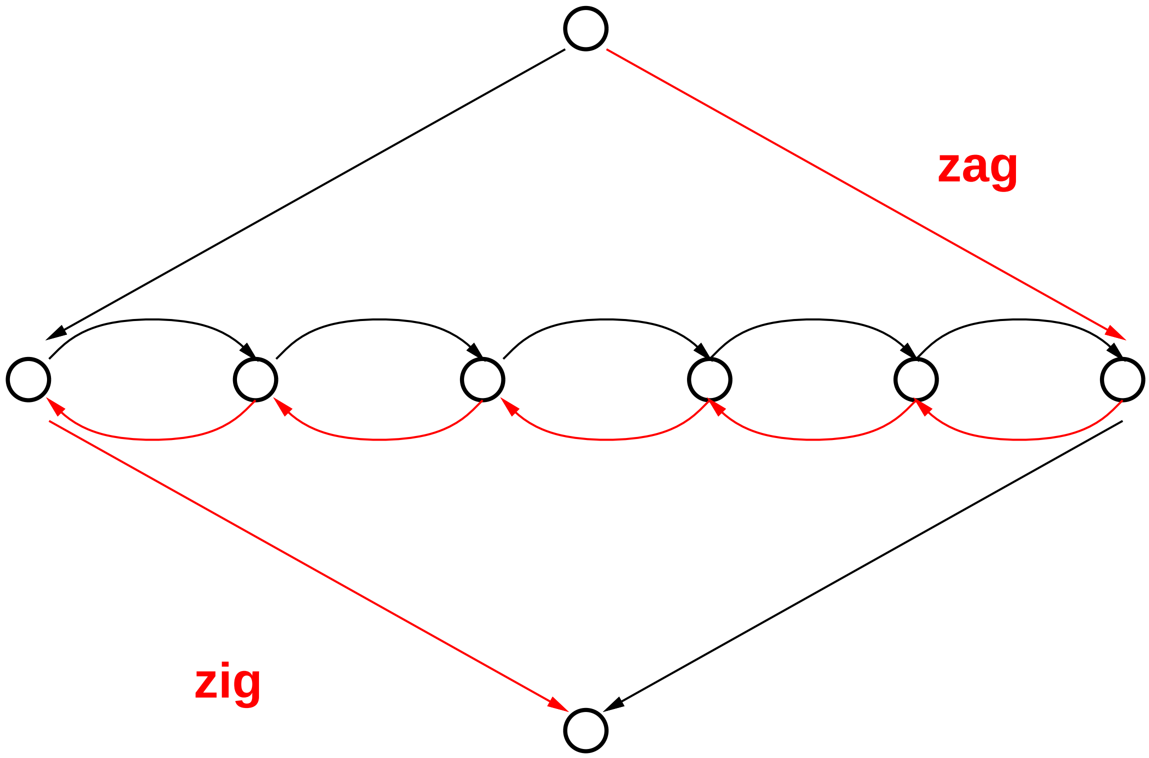
Hamiltonian Path

If x_i is **true** in the assignment, then **zig-zag**.



Hamiltonian Path

If x_i is **false** in the assignment, then **zag-zig**.

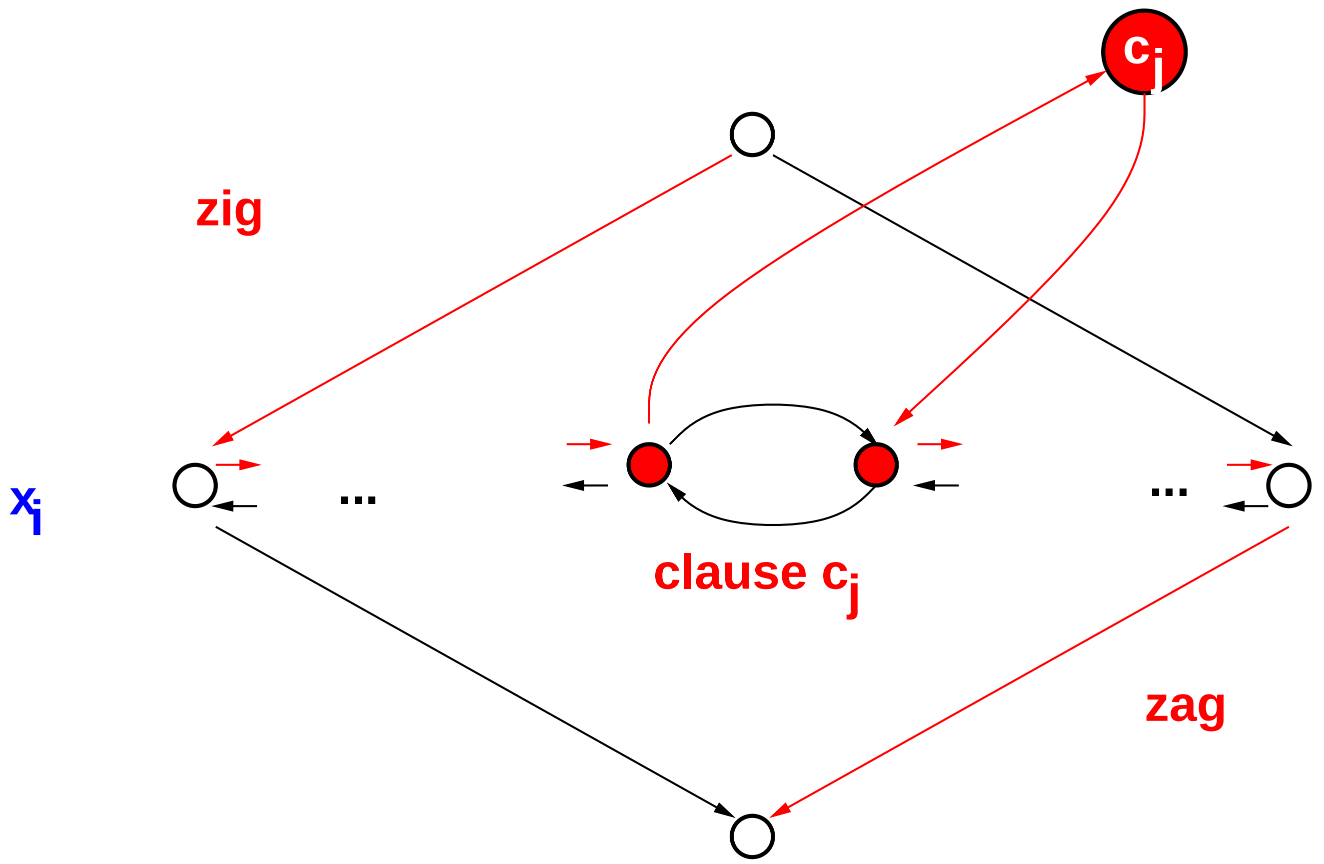


Hamiltonian Path

Add clause nodes.

- Each c_j is assigned one **true** literal.
- For each clause, pick one.

If we select x_i in c_i, add "detour"

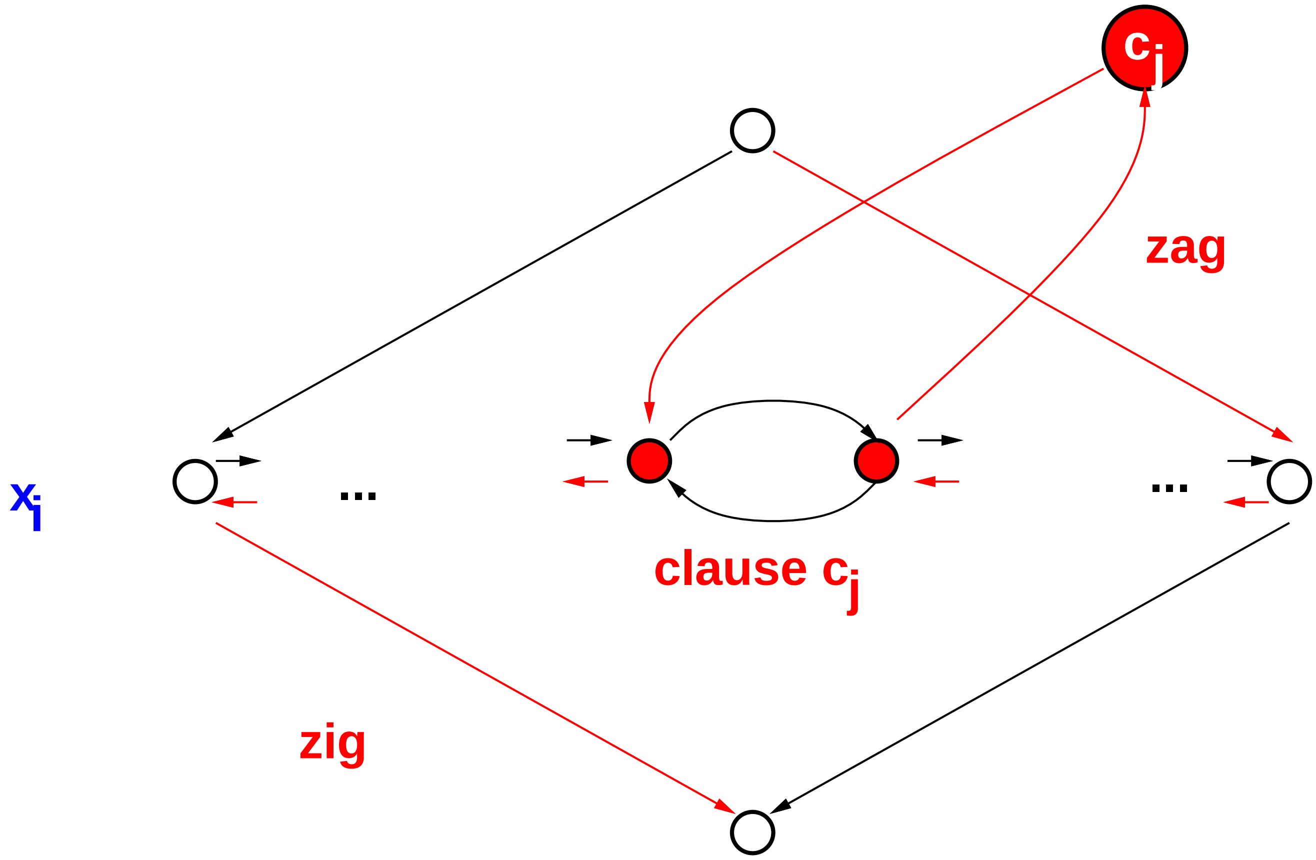


Hamiltonian Path

Add clause nodes.

- Each c_j is assigned one **true** literal.
- For each clause, pick one.

If we select $\overline{x_i}$ in c_i, add "detour"



This completes one direction of the reduction.

Hamiltonian Path

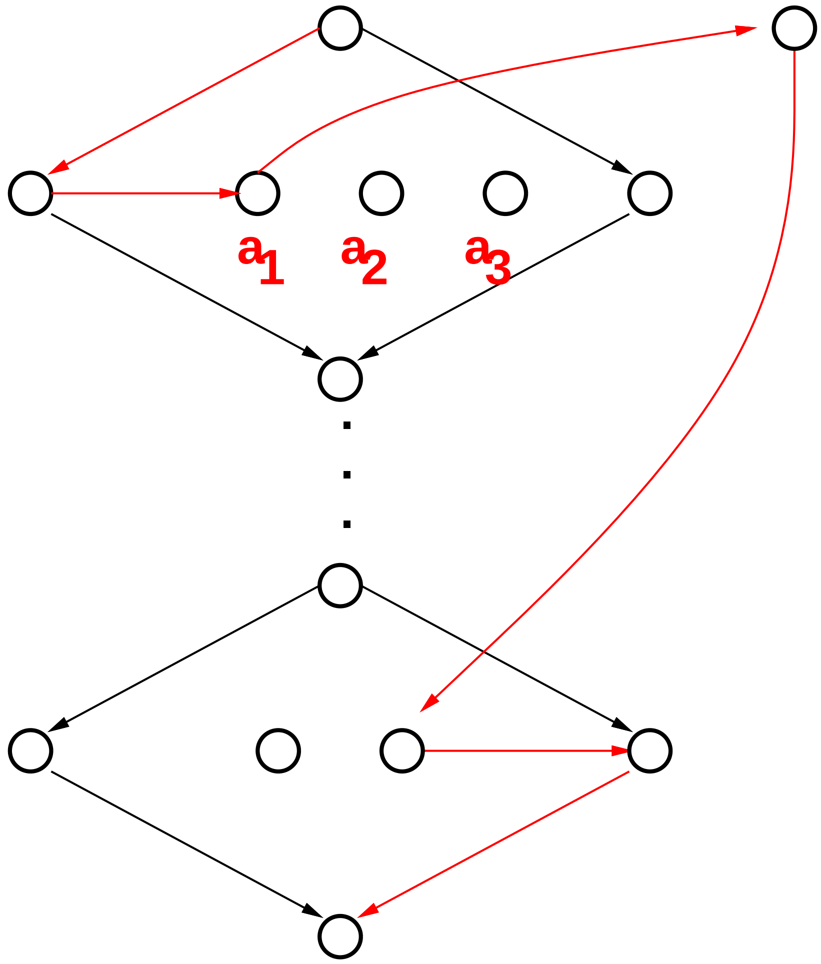
Claim: If G has a hamiltonian path from s to t, then ϕ has a satisfying assignment.

Definition: A **normal** hamiltonian path is one that traverses the diamonds in order.

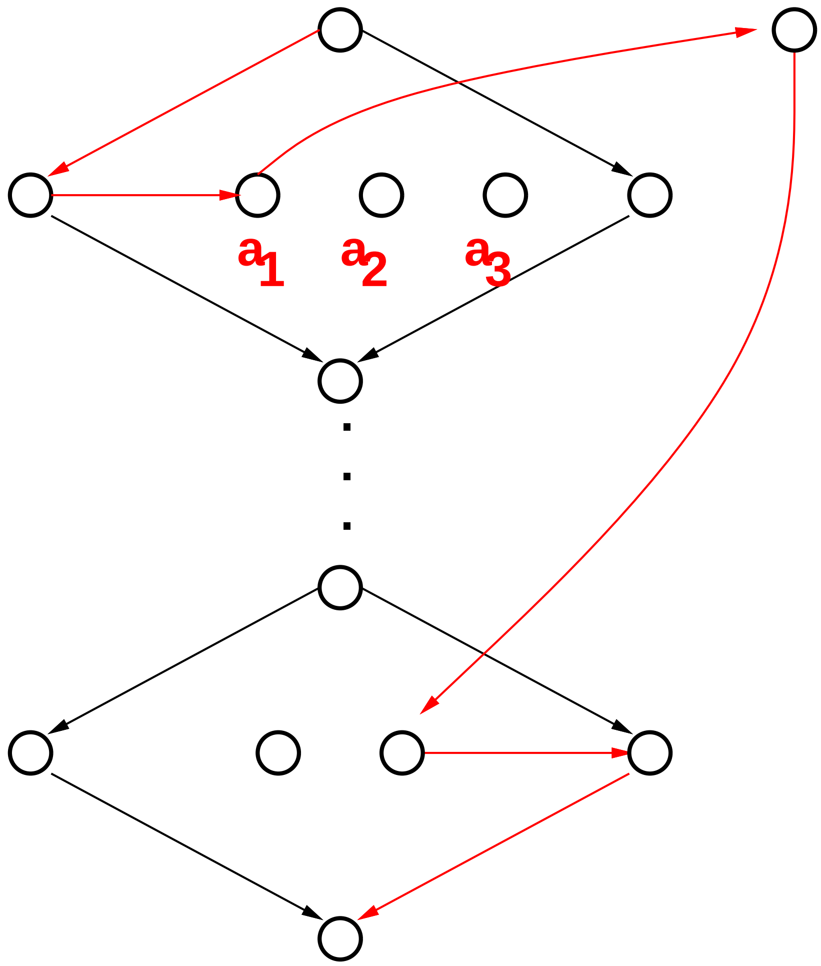
- if x_i diamond zig-zags, assign **true**.
- if x_i diamond zags-zig, assign **false**.
- each clause vertex appears once
- source of detour determines which literal is assigned **true**.

Hamiltonian Path

Claim: Every hamiltonian path in G is normal.



Hamiltonian Path



- only arrows to a_2 from a_1, a_3, c
- paths from a_1 or c go elsewhere
- path from a_2 would leave no exit

Any hamiltonian path is normal, Q.E.D.

Subset Sum

Theorem: SUBSET-SUM is NP-complete.

Must demonstrate

- SUBSET-SUM $\in NP$
- every $A \in NP$ poly-time reducible to SUBSET-SUM.

Note that

- we have already seen the first
- we will show that 3SAT $\leq_P$ SUBSET-SUM

Subset Sum

For every SUBSET-SUM problem,

- We construct some very large numbers,
- in decimal notation.
- Variables are pairs of numbers, and
- clauses are positions in their decimal representations.

Subset Sum

Let ϕ be a boolean formula with

- variables $x_1, \ldots, x_\ell$
- clauses $c_1, \ldots, c_k$.

We construct a table whose rows are numbers in decimal notation.

Subset Sum

Rows of table are labeled with

- variable encodings: $y_1, z_1, y_2, z_2, \ldots, y_\ell, z_\ell$,
- clause encodings: $g_1, h_1, g_2, h_2, \ldots, g_k, h_k$,
- goal t

Subset Sum

Each x_i encoded as y_i, z_i.

Decimal representation has two parts.

- left-hand part: digit i is 1, rest 0
- right-hand part: one digit for each clause
  - j^{th} digit of y_i is 1 if c_i contains x_i
  - j^{th} digit of z_i is 1 if c_i contains $\overline{x_i}$

Subset Sum

$$(x_1 \vee \overline{x_2} \vee \cdots) \wedge \cdots (\overline{x_1} \vee \overline{x_2} \vee \cdots)$$

	1	2	...	ℓ	c_1	...	c_k
y_1	1	0	...	0	1	...	0
z_1	1	0	...	0	0	...	1
y_2	0	1	...	0	0	...	0
z_2	0	1	...	0	1	...	1

Subset Sum

S also contains g_j and h_j for each clause c_j

- they are equal
- digit $\ell + j$ is 1, rest 0.

	1	...	ℓ	c_1	c_2	...	c_k
$\vdots$							
g_1	0	...	0	1	0	...	0
h_1	0	...	0	1	0	...	0
g_2	0	...	0	0	1	...	0
h_2	0	...	0	0	1	...	0

Subset Sum

Bottom line of table is goal t

- first ℓ digits are 1
- last k digits are 3

	1	...	ℓ	c_1	c_2	...	c_k
$\vdots$							
t	1	...	1	3	3	...	3

Subset Sum

	1	2	...	ℓ	c_1	c_2	...	c_k
y_1	1	0	...	0	1	0	...	0
z_1	1	0	...	0	0	0	...	1
y_2	0	1	...	0	0	0	...	0
z_2	0	1	...	0	1	0	...	1
$\vdots$								
g_1	0	0	...	0	1	0	...	0
h_1	0	0	...	0	1	0	...	0
g_2	0	0	...	0	0	1	...	0
h_2	0	0	...	0	0	1	...	0
$\vdots$								
t	1	1	...	1	3	3	...	3

Subset Sum

Claim: If ϕ is satisfiable, then some subset of S sums to t.

- select y_i if x_i is *true*.
- select z_i if x_i is *false*.

Adding them up

- yields a 1 in the first ℓ digits
- matching t so far

Each of the last k digits

- lies between 1 and 3
- some literal is true
- no more than 3 can be true
- so pick enough h_i, g_i to bring digit up to 3

This completes one direction.

Subset Sum

Claim: If some subset of S sums to t, then ϕ is satisfiable.

- all digits in S are 0 or 1.
- each column contains ≤ 5 ones.
- therefore, subset causes no carries between columns
- subset must have each y_i or z_i, but not both.

Subset Sum

Here is the satisfying assignment.

- if subset includes y_i, then x_i is *true*
- if subset includes z_i, then x_i is *false*

This assignment must satisfy ϕ because

- each of final k columns sums to 3.
- at most 2 come from g_i, h_i
- at least one comes from true literal

Final point: transformation takes $O(n^2)$ stages.

Not-All-Equal Satisfiability

A 3cnf formula is *satisfiable* if there is a truth assignment

- in each clause,
- not all literals are *false*.

Definition: A 3cnf formula is *nae-satisfiable* if there is a truth assignment

- in each clause,
- not all literals are *equal*.

$$\text{NAESAT} = \{\langle\phi\rangle | \phi \text{ is nae-satisfiable 3cnf formula}\}$$

Not-All-Equal Satisfiability

Theorem: NAESAT is NP-complete.

By reduction from CIRCUIT-SAT.

- given a circuit C
- we build a 3cnf formula ϕ
- so that C is satisfiable iff $\phi \in$ NAESAT

Not-All-Equal Satisfiability

For each NOT gate in C with input w_i and output w_j, add clauses

$$(w_i \vee w_j \vee z) \wedge (\overline{w_i} \vee \overline{w_j} \vee z)$$

Note introduction of padding variable z!

3SAT

For each OR gate in C with inputs w_i, w_j and output w_k, add clauses

$$(\overline{w_i} \vee w_k \vee z) \wedge (\overline{w_j} \vee w_k \vee z) \wedge (w_i \vee w_j \vee \overline{w_k})$$

Equivalent to earlier construction, but now "padded" with z.

3SAT

For each AND gate in C with inputs w_i, w_j and output w_k, add clauses

$$(w_i \vee \overline{w_k} \vee z) \wedge (w_j \vee \overline{w_k} \vee z) \wedge (\overline{w_i} \vee \overline{w_j} \vee w_k)$$

Equivalent to earlier construction, but now “padded” with z.

Not-All-Equal Satisfiability

Claim: If there exists an nae-satisfying assignment for ϕ, then C is satisfiable.

For any nae-satisfying assignment to ϕ,

- the *complementary* assignment is also nae-satisfying.
- in one of them the extra variable z is *false*
- ignoring z, we have the usual reduction

Not-All-Equal Satisfiability

Claim: If there exists a satisfying assignment for C, then ϕ is nae-satisfiable.

Truth assignment for C ensures that

- not all literals are *false*.
- assign *false* to z (OK because it's padding)

Any clause with a z nae-satisfiable.

Remaining clauses from AND gates:

$$(w_i \vee \overline{w_k} \vee z) \wedge (w_j \vee \overline{w_k} \vee z) \wedge (\overline{w_i} \vee \overline{w_j} \vee w_k)$$

Cannot satisfy all three terms.

OR is similar case analysis.