Design and Analysis of Algorithm

NP Completeness

Dr. Dayal Kumar Behera

School of Computer Engineering
KIIT Deemed to be University, Bhubaneswar, India

Credits: Prof(Dr.) Biswajit Sahoo, KIIT

Course Contents

Sr #	Major and Detailed Coverage Area	Hrs
1	**NP Completeness**	4
	▪ Defination of P, NP, NP Complete, NP Hard ▪ 3-CNF Satisfiability Problem ▪ Clique Decision Problem ▪ Hamiltonian Cycle ▪ TSP	

Contents of Discussion

- Intractabe Problem
- Nondeterministic Algorithm
- P and NP Defination
- Optimization & Decision Problem
- Verification of Problem
- Reducibility
- NP Complete & NP Hard Defination
- ~~Cook's Theorem~~
- Examples of NP Complete
  - Clique Decision Problem
  - Hamiltonian Cycle
  - TSP

Tractability

Tractable **problems:**	*Intractable* **problems:**
Polynomial time	Super Polynomial time
$O(n^2)$, $O(n^3)$, $O(1)$,$O(n\lg n)$	$O(2^n)$, $O(n\,2^n)$, $O(n^n)$, $O(n!)$
Ex:- $O(n^3)$; for n=100 Number of steps = 10,00,000	Ex:- $O(2^n)$; for n=100 Number of steps ≈90,....,00

Prove

Tractability

Tractable **problems:**	*Intractable* **problems:**
Polynomial time	Super Polynomial time
$O(n^2)$, $O(n^3)$, $O(1)$, $O(n\lg n)$	$O(2^n)$, $O(n\ 2^n)$, $O(n^n)$, $O(n!)$
$O(n^{100})$ High order Polynomial	$O(n\ 2^n)$ (for n=10, small input)

Disprove

Tractability

Tractable **problems:**	*Intractable* **problems:**
Polynomial time algorithms are *Tractable* Normally	Super Polynomial time algorithms are *Intractable* in General
Not applicable for: Higher order Polynomial	Not applicable for: Small inputs

Polynomial Time Nondeterministic Algorithm

```
int Search(a, n, key)
{                                  Running Time
j=choice(1 : n)                        //O(1) : Nondeterministic

if(key==a[j])                          //1
        return(j);                     //1
else
        return(-1);                    //1
}                              Total Running time=O(1)
```

Assume time required for choice(1 : n) is O(1).

P and NP

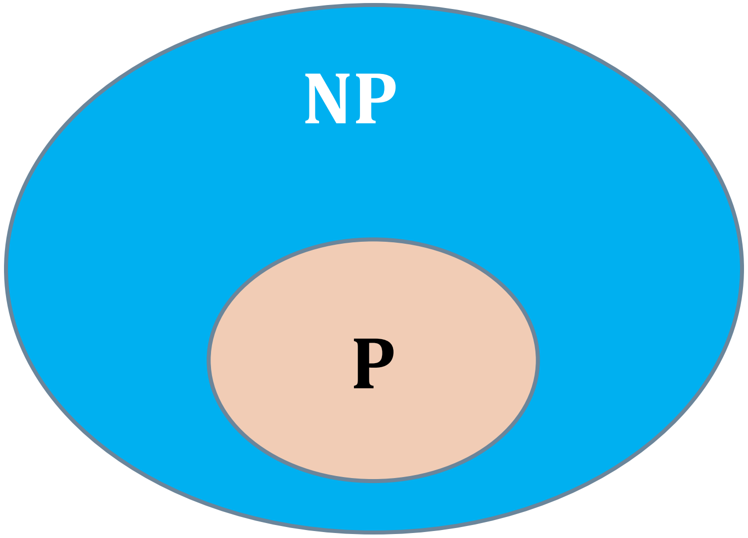
- **P** is set of problems that can be solved in polynomial time in a deterministic machine

- **NP** (*nondeterministic polynomial time*) is the set of problems that can be solved in polynomial time by a **nondeterministic** machine

A non-deterministic computer is a computer that magically “guesses” a solution, then has to verify that it is correct.

Is P = NP ?

P and NP



Today nondeterministic, Tomorrow may be deterministic

Optimization and Decision Problems

Optimization Problem: Maximize profit or Minimize Loss

Decision Problem: Answers are in Boolean (Yes/No)

Opt^n: Find MCST of the graph, G.

Dec^n: Is there any MCST exist in G with cost less than k?

Opt^n: Find TSP (optimal) of the graph, G.

Dec^n: Is there any TSP exist in G with cost less than k?

Opt^n: Find maximum profit in a 0/1 Knapsack.

Dec^n: Does 0/1 Knapsack have a profit more than k?

Optimization and Decision Problems

In fact, from the point of view of polynomial-time solvability, there is not a significant difference between the optimization (maximize or minimize) version of the problem and the decision version (decide, yes or no).

Given a method to solve the optimization version, we automatically solve the decision version as well.

Optimization and Decision Problems

Solution to decision problems takes a fraction of time more than solution to optimization problems.(cond^n check).

NP completeness is proved directly w.r.t. decision problems. However, same computational complexity will also be applicable to the optimization problems.

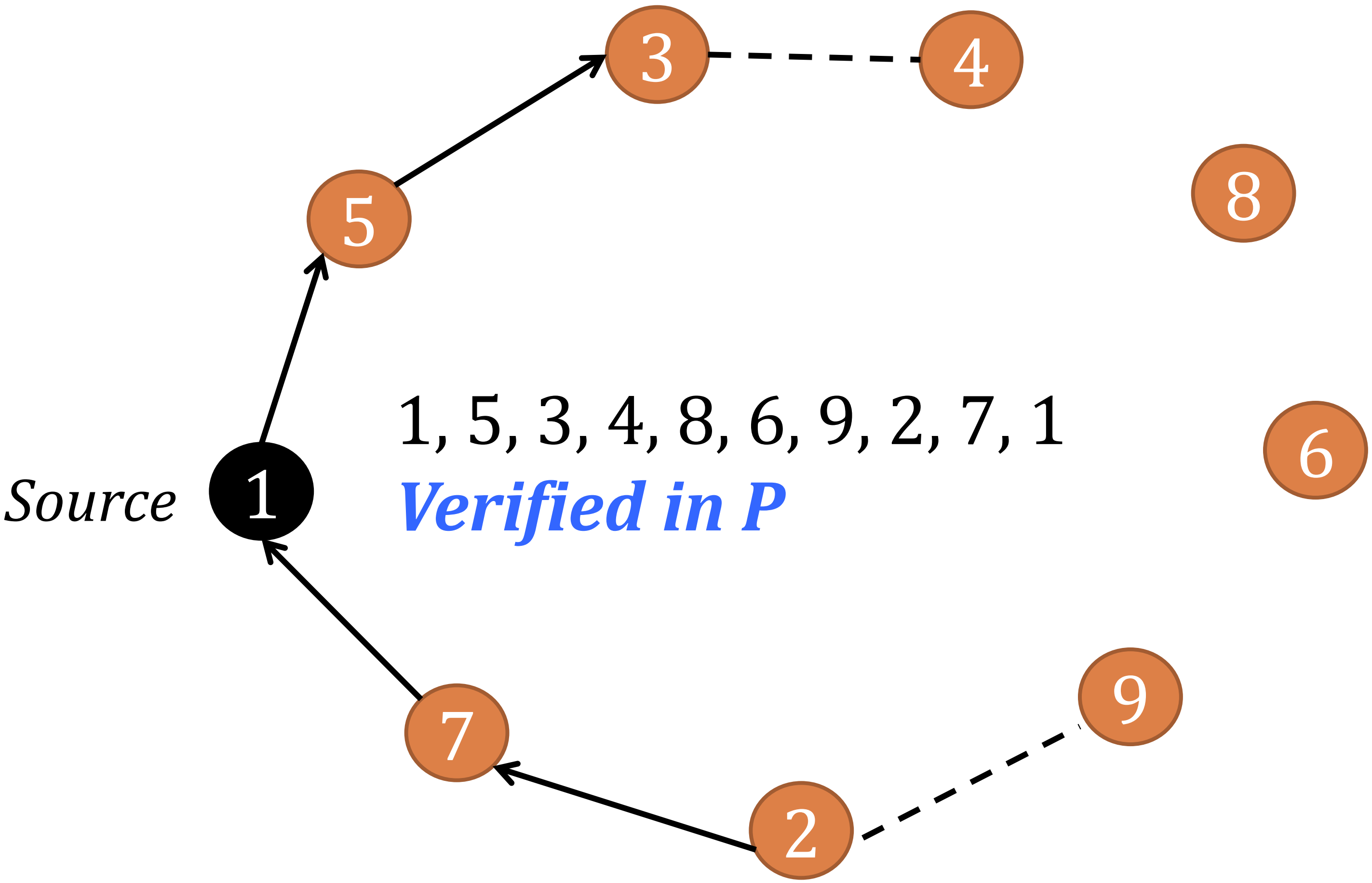
Verification of Decision Problems

Ex:- TSP, Formula Satisfiability, 0/1 Knapsack…

Given a solution (guess) to the problem, we neeed to verify it in the problem

NP problems are ***Verifiable*** in polynomial time in deterministic machine

Verification of Decision TSP in P



Verified in P

Reducibility

The crux of NP-Completeness is ***Reducibility***

Informally, a problem L can be reduced to another problem Q if *any* instance of L can be "**easily rephrased**" as an instance of Q, the solution to which provides a solution to the instance of L

Intuitively: If L reduces to Q, L is "***no harder to solve***" than Q

Reducibility Examples

Given an equation $ax^2 + bx + c = 0$, find roots of the equation.

Question: $5x + 6 = 0$;

P

$$0.x^2 + 5x + 6 = 0$$

Question: Find the value of $\sqrt{(45^2 + 46^2)}$

Apply **Pithagoras Theorem:**

P

Draw a right angle triangle with p=45, b=46, Measure h

Reducibility Examples

X: Given n integers, is the largest integer > 0 ?

Y: Given n Boolean variables, is there at least one TRUE?

X	-1	-49	-4	5	1	0	-6	8	-20
Y	F	F	F	T	T	F	F	T	F

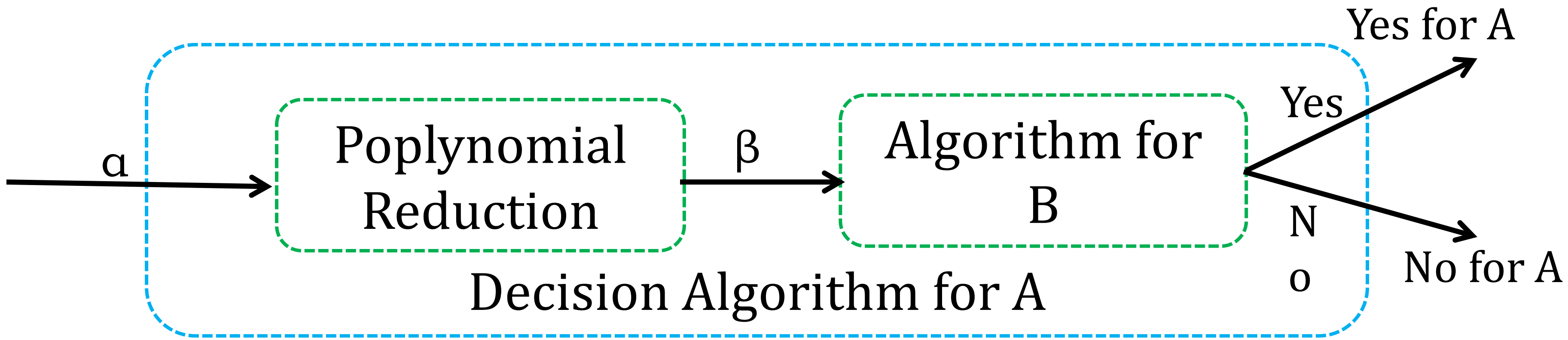
Transform X to Y by $y_i = T$ if $x_i > 0$, $y_i = F$ if $x_i <= 0$

Time required to test X = Time required to reduce X to Y + Time required to test Y

Now, X is *Polynomial-time Reducible* to Y,

So, we denote this $X \leq_p Y$

Reducibility relation



A and B are two decision Problems

If decision algorithm for B is polynomial so does A

A is no harder than B

If A hard (e.g. NPC) so does B

Transitive property of Reducibility

X, Y and Z are three problems;

X is *Polynomial-time Reducible* to Y, $X \leq_p Y$ and

Y is *Polynomial-time Reducible* to Z, $Y \leq_p Z$

Now, $X \leq_p Z$; X is *Polynomial-time Reducible* to Z

Computational Relationship

NP-Complete problems are computationaly related:

- If any *one* NP-Complete problem can be solved in polynomial time…
- …then *every* NP-Complete problem can be solved in polynomial time…
- …and in fact *every* problem in **NP** can be solved in polynomial time (which would show **P = NP**)

P & NP Defination

P = problems that can be solved in polynomial time (quick solution)

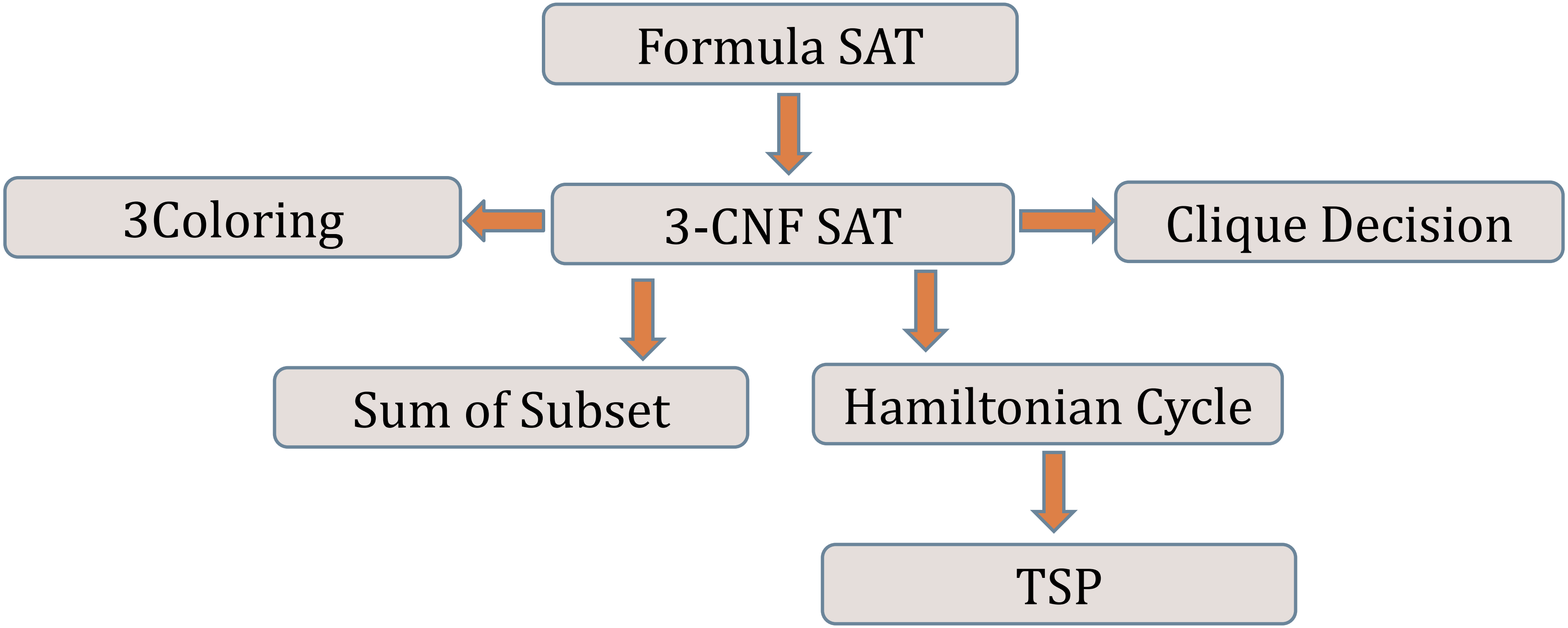
NP = problems for which a solution can be verified in polynomial time (quick verification)

Unknown whether **P = NP** ? (most suspect not)

NP Complete & NP Hard Defination

NP-Complete	NP-Hard
Problem L is NP Complete, if ➢ $L \in$ **NP** and ➢ $R \leq_p L \; \forall \; R \in$ **NP**	Problem L is NP Hard if $R \leq_p L \; \forall \; R \in$ **NP**
Problem L is NP Complete, if ➢ $L \in$ **NP** and ➢ $R \leq_p L$ for any $R \in$ **NPC**	Problem L is NP Hard if $R \leq_p L$ for any $R \in$ **NPC**

NP-Complete Problems



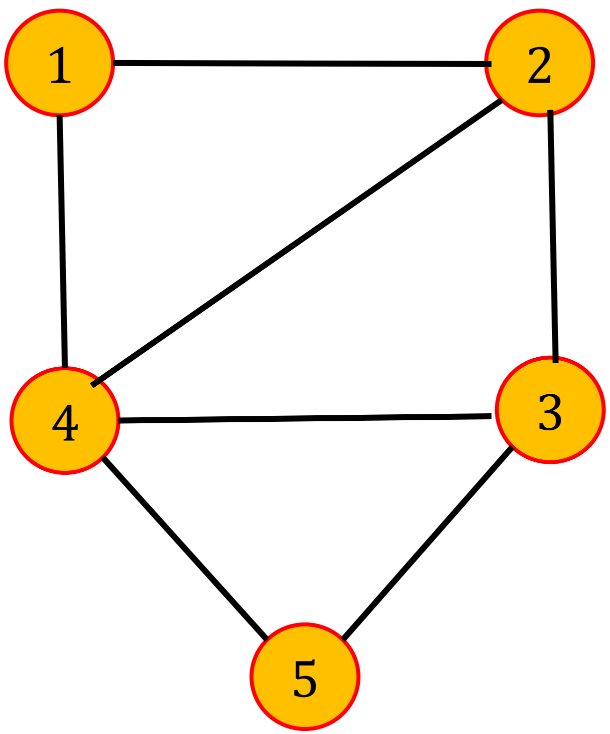
Maximum Clique Problem

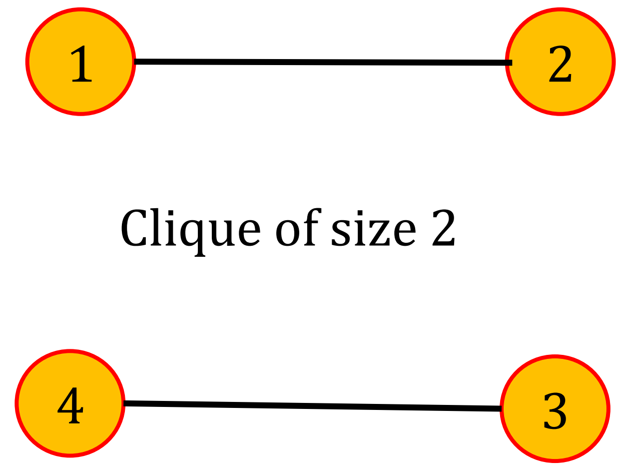
Max clique problem: A complete subgraph of a graph is a clique.

Number of edges in a complete graph, G=(V,E) is
$|E| = |V| x |V-1| / 2$

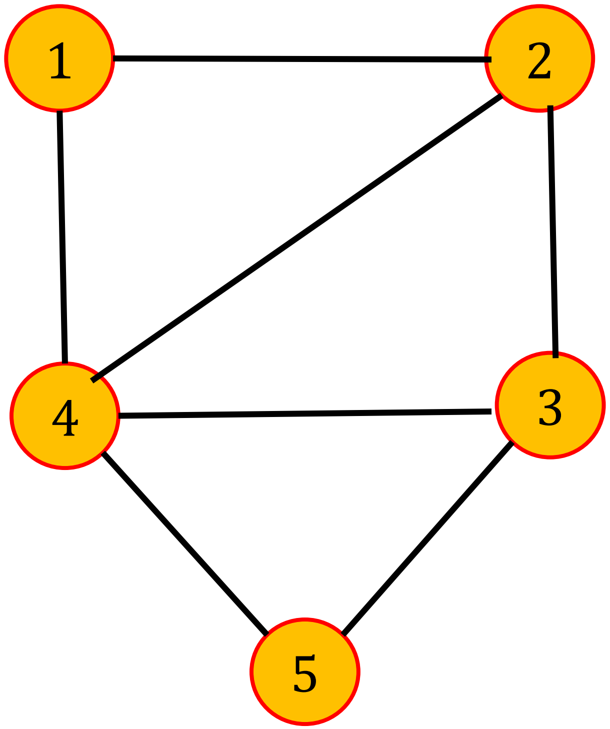
The maximal clique problem is to determine the size of a largest clique in G.

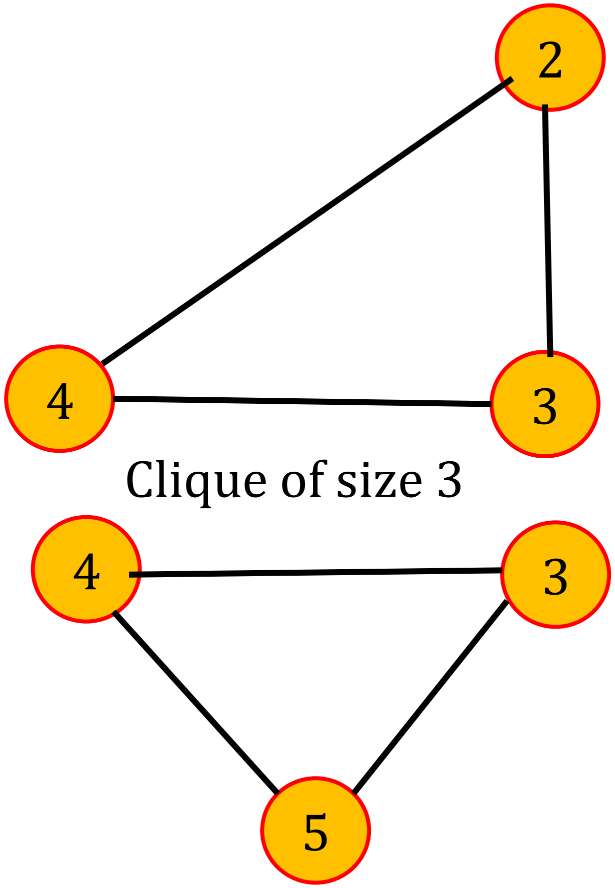
Maximum Clique Problem



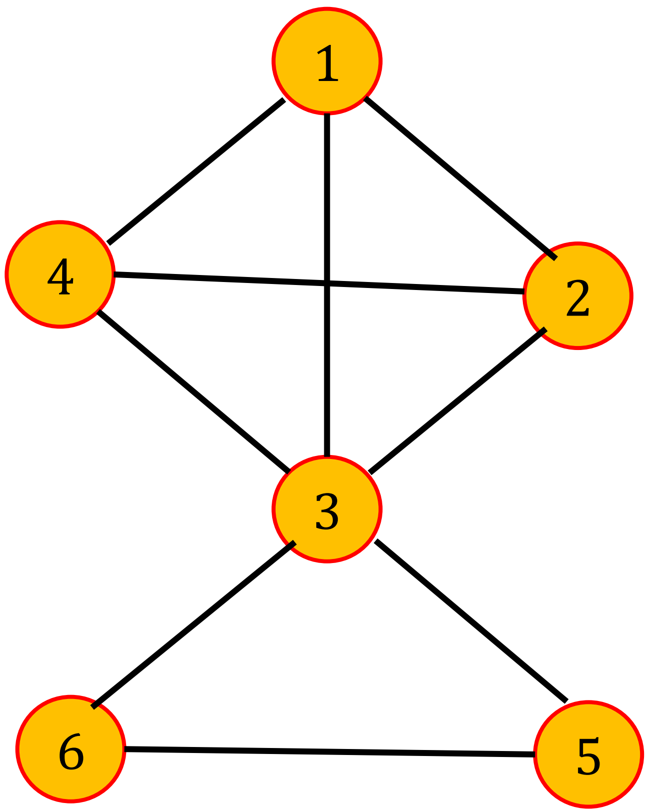


Maximum Clique Problem

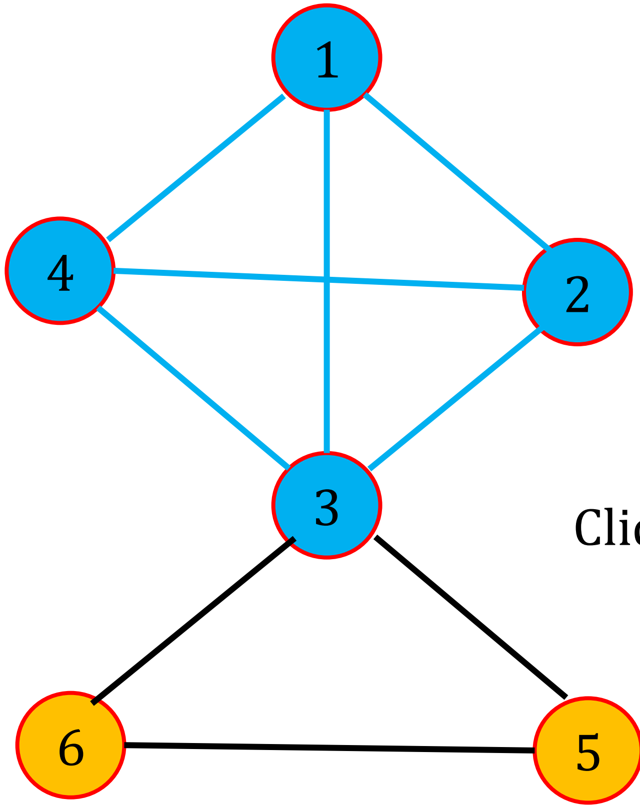




Maximum Clique Problem



Maximum Clique Problem



Clique of size 4

Clique Decision Problem (CDP)

Is there a clique of size 3 in the graph?

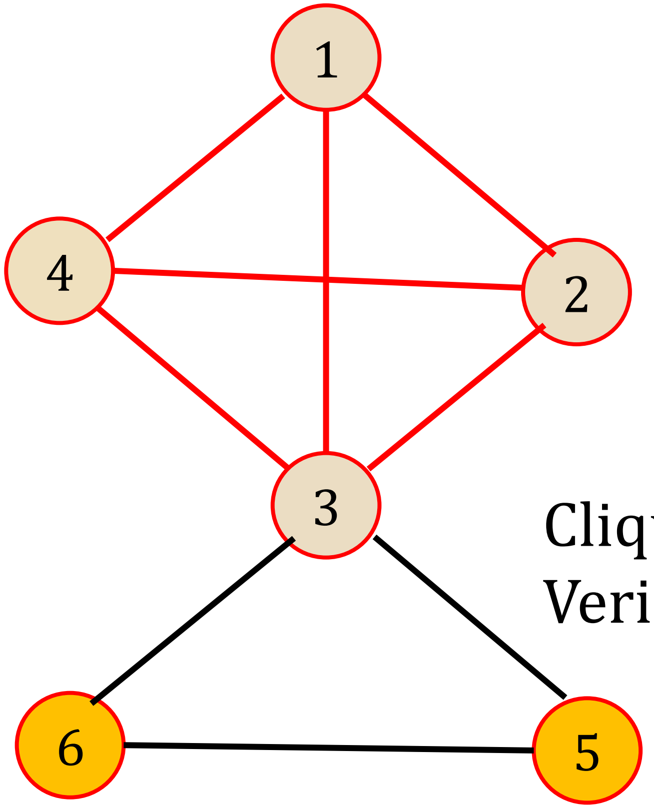
Clique Decision Problem (CDC) is in NP-Complete?

i. CDC $\in$ **NP** and
ii. 3-CNF SAT $\leq_p$ CDC

if the formula is satisfiable then the graph has a Clique of size 3.

Verification of CDP

CDC $\in$ **NP**



Clique of size 3 (1, 2, 4)
Verified in polynomial time

Clique Decision Problem (CDP)

3-CNF SAT $\leq_p$ CDP

$$\varphi = \underbrace{(\overline{x1} \lor x2 \lor \overline{x3})}_{C1} \land \underbrace{(x1 \lor \overline{x2} \lor x3)}_{C2} \land \underbrace{(x1 \lor x2 \lor \overline{x3})}_{C3} \;\xrightarrow{\leq_p}\; G\varphi$$

The formula is satisfiable iff the graph Gφ has a Clique of size 3.

$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$

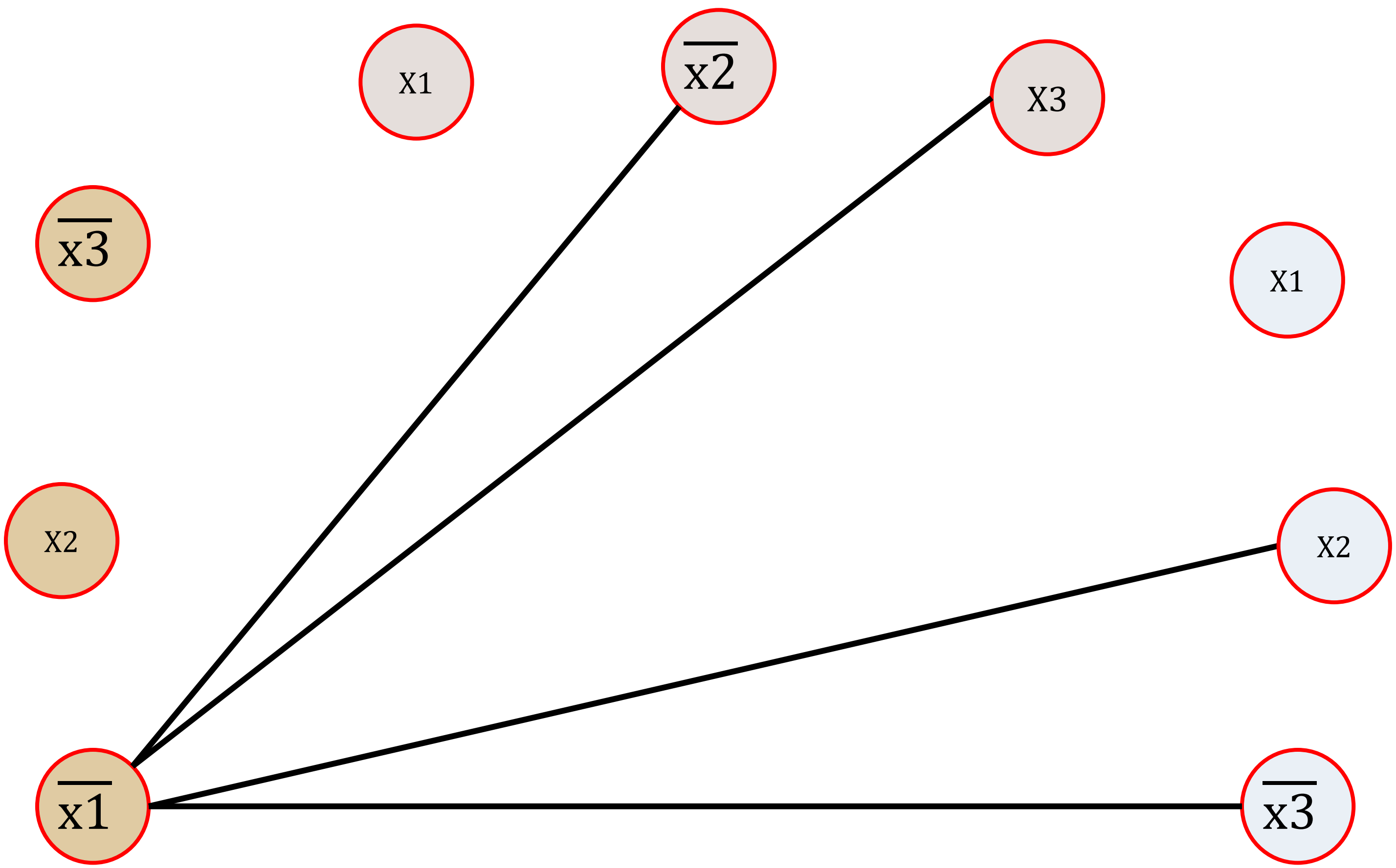
C1C2 C3

$V = \{ \langle a, i \rangle \mid a \in Ci \}$

$E = (\langle a, i \rangle \langle b, j \rangle) \mid b \neq \overline{a}, i \neq j$

$$\varphi = (\overline{x1} \lor x2 \lor \overline{x3}) \land (x1 \lor \overline{x2} \lor x3) \land (x1 \lor x2 \lor \overline{x3})$$

C1C2 C3

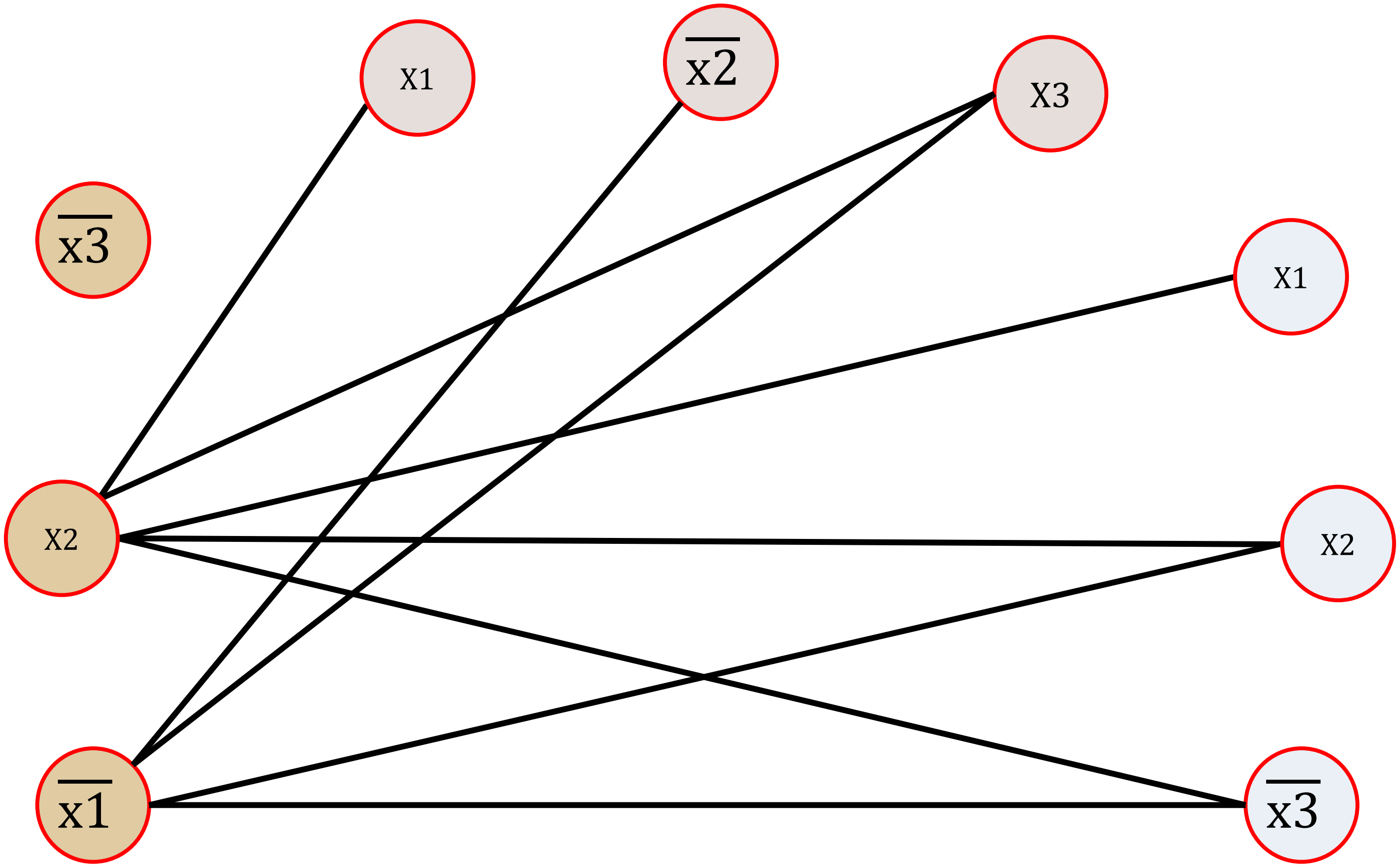


$$V = \{ \langle a, i \rangle \mid a \in C_i \}$$

$$E = (\langle a, i \rangle \langle b, j \rangle) \mid b \neq \overline{a},\ i \neq j$$

$$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

C1C2 C3

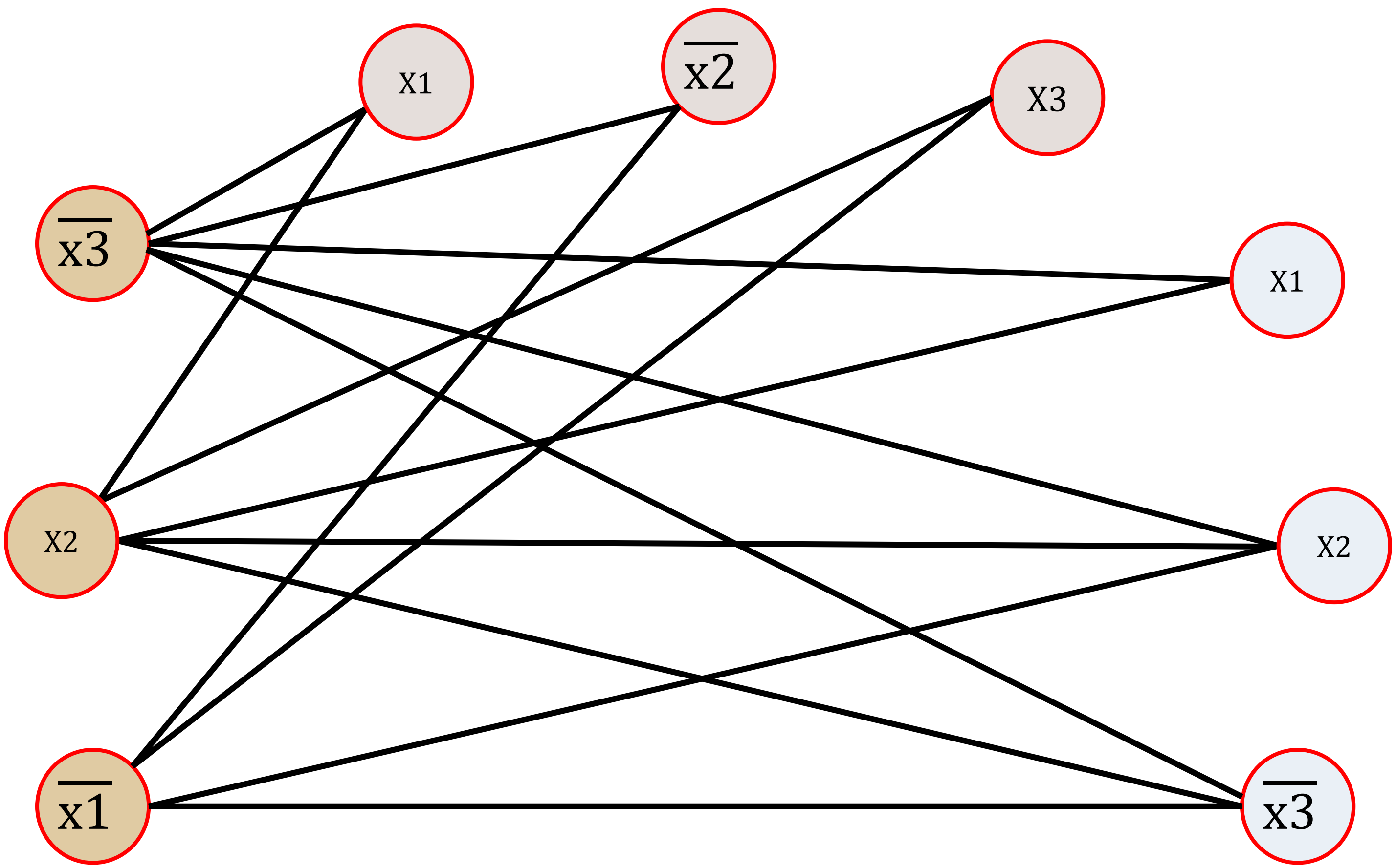


$V = \{ \langle a, i \rangle \mid a \in C_i \}$

$E = (\langle a, i \rangle \langle b, j \rangle) \mid b \neq \overline{a}, i \neq j$

$$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

C1C2 C3

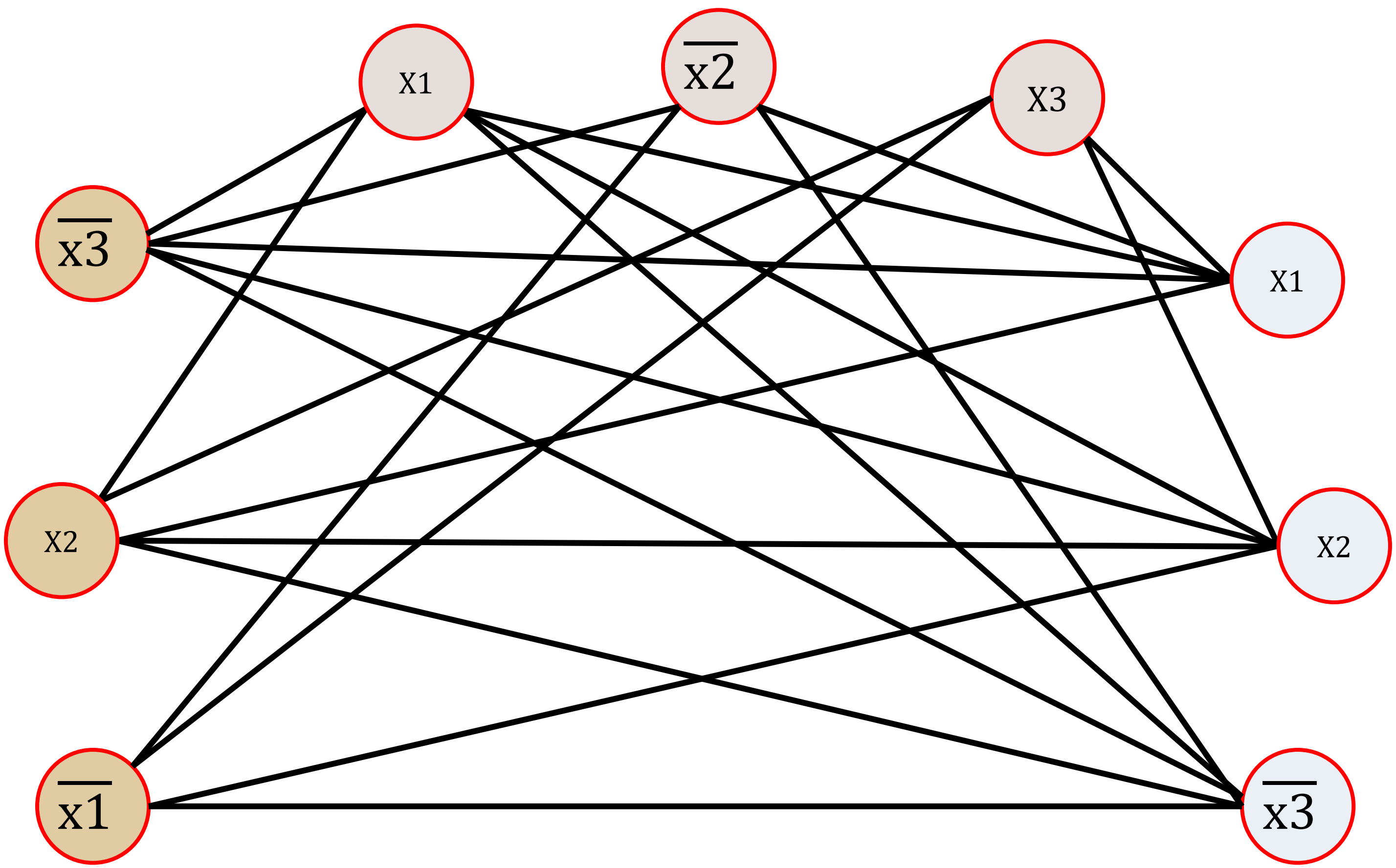


V = { <a, i> | a ∈ Ci }

E= (<a, i><b, j>) | b ≠ $\overline{a}$, i ≠ j

$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$

C1C2 C3



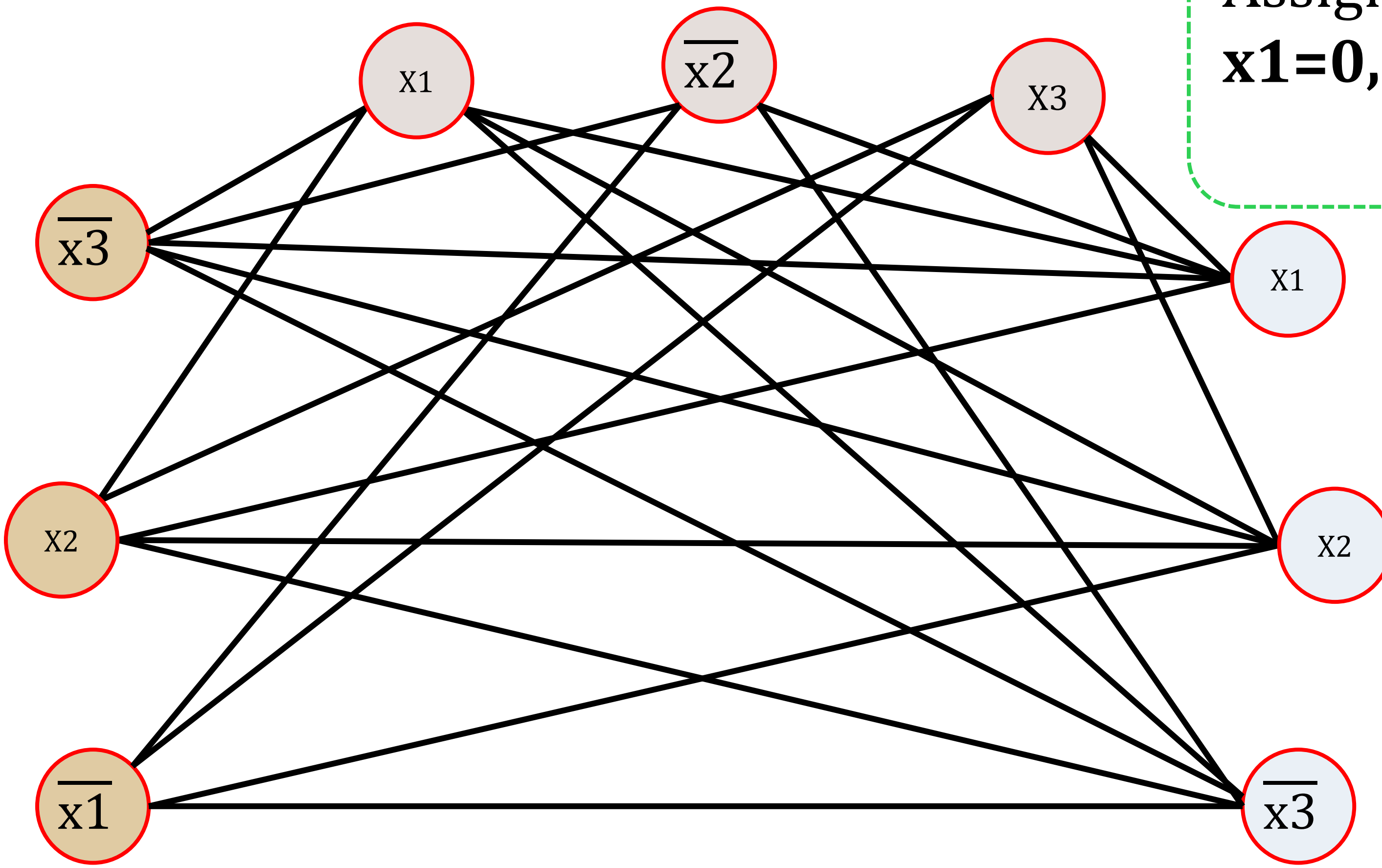
$V = \{ \langle a, i \rangle \mid a \in Ci \}$

$E = (\langle a, i \rangle \langle b, j \rangle) \mid b \neq \overline{a}, i \neq j$

$$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

C1C2 C3

Assignment:
x1=0, x2=1, x3=1



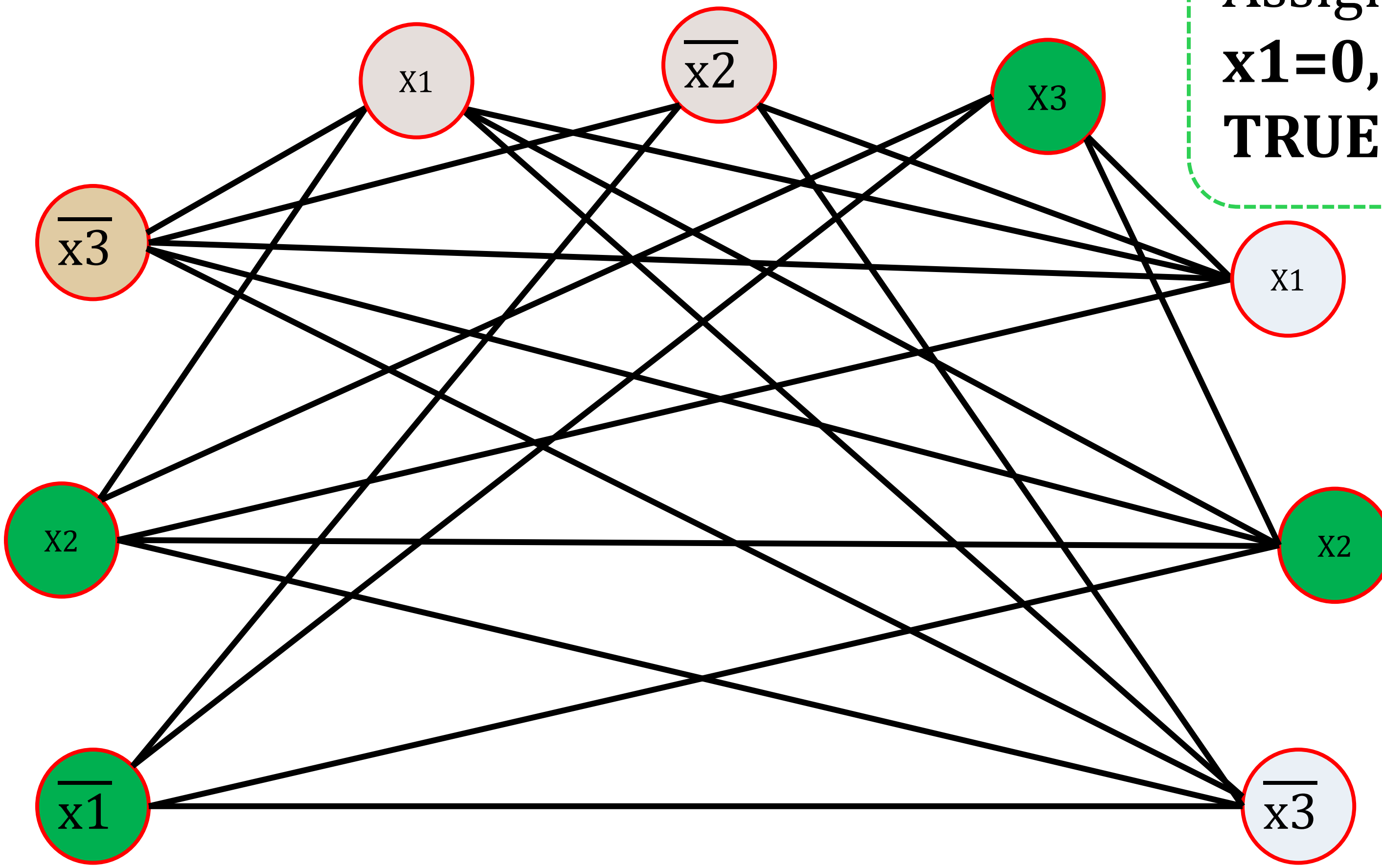
$$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

C1C2 C3

Assignment:

x1=0, x2=1, x3=1

TRUE



$$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

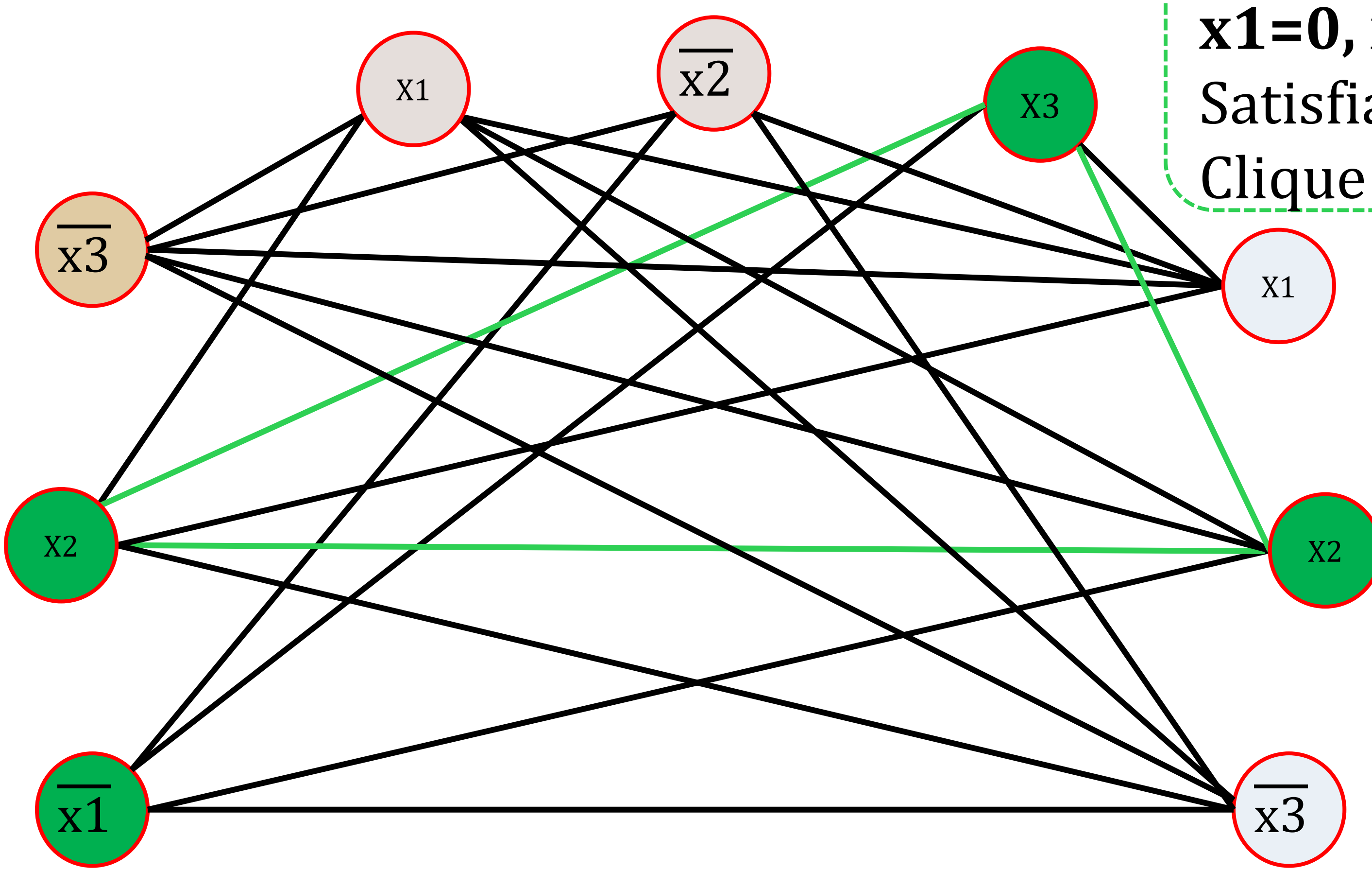
C1C2 C3

Assignment:

x1=0, x2=1, x3=1

Satisfiable

Clique of size 3



$$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

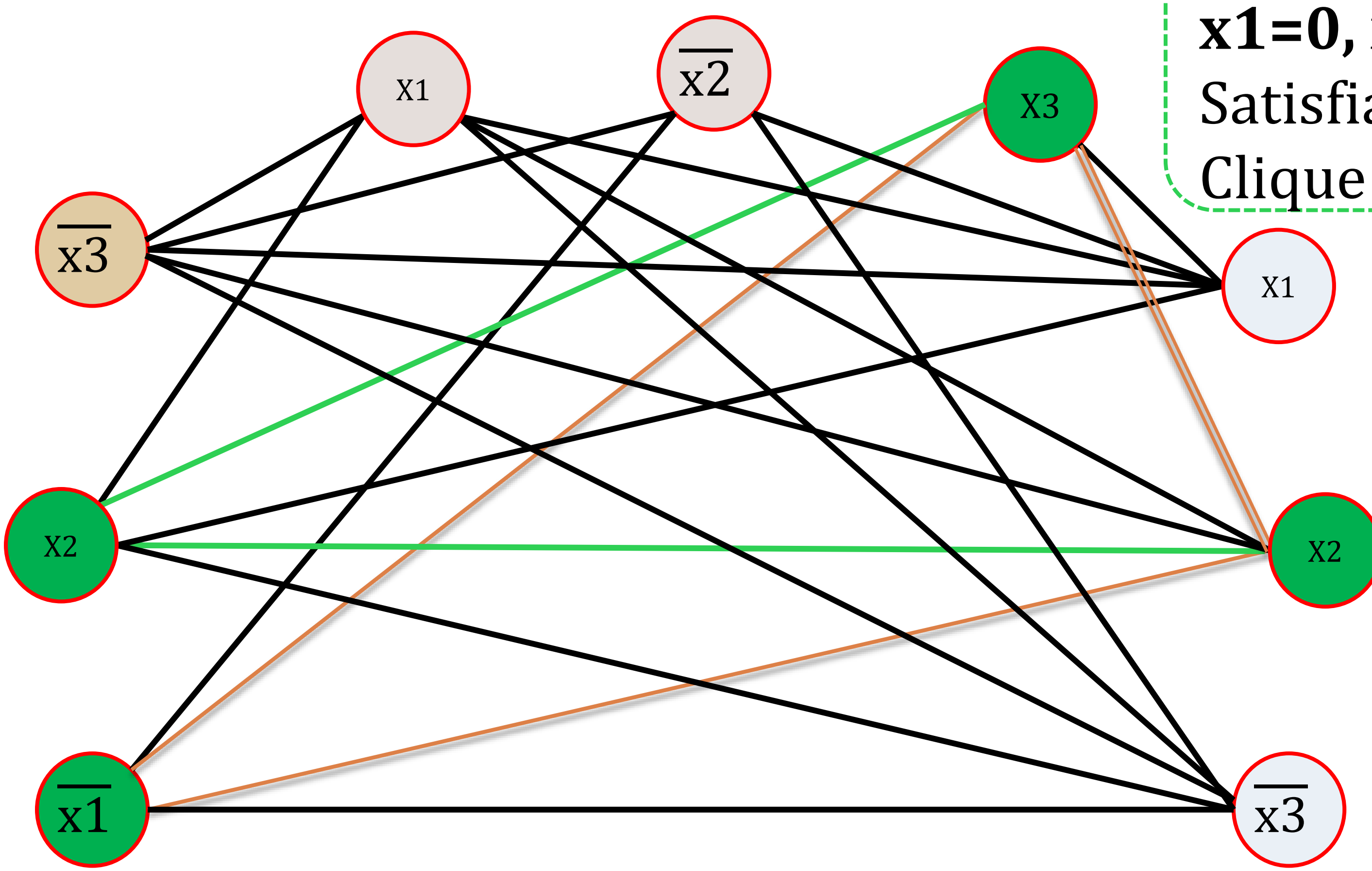
C1C2 C3

Assignment:

x1=0, x2=1, x3=1

Satisfiable

Clique of size 3



$$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

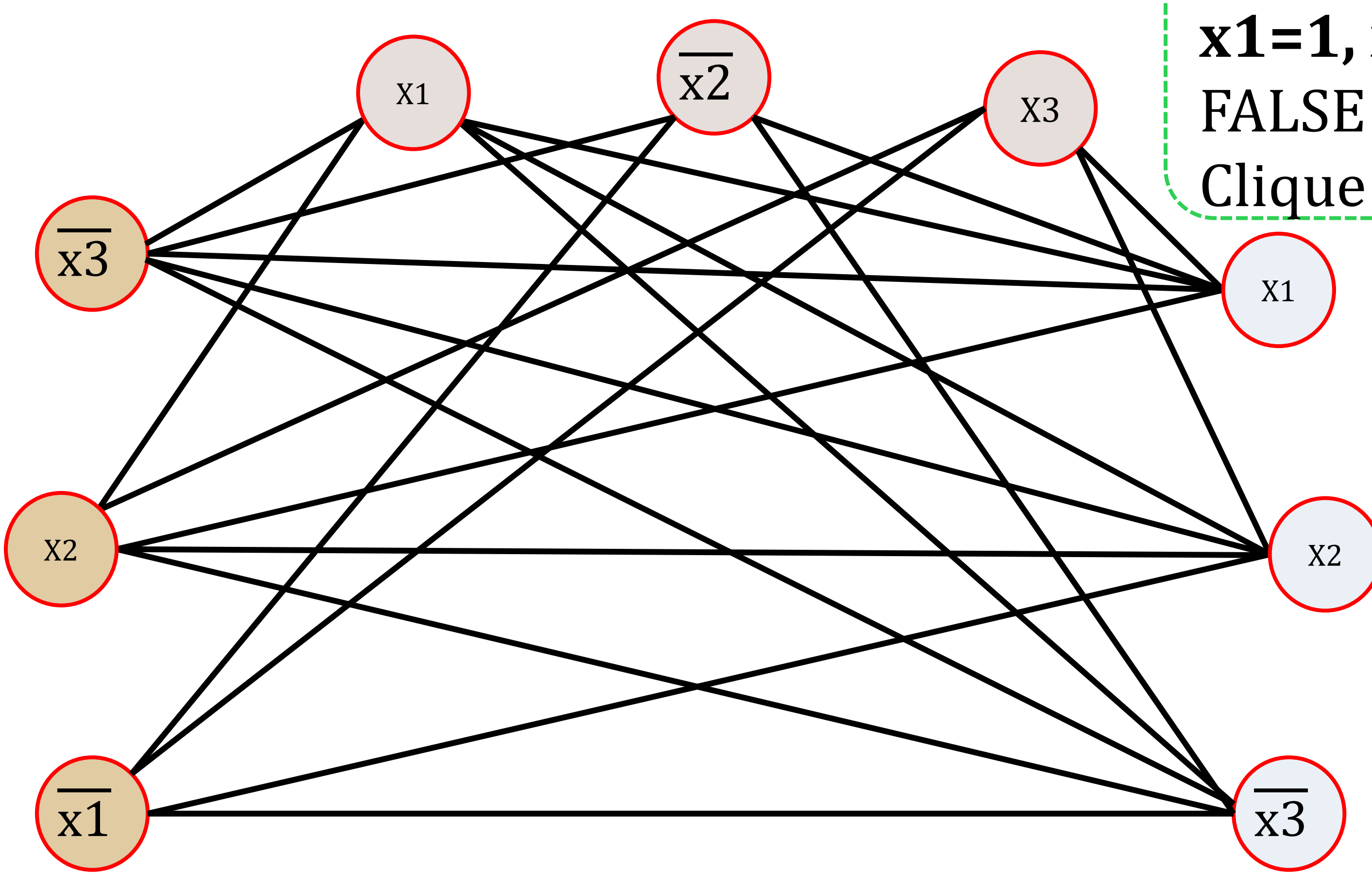
C1C2 C3

Assignment:

x1=1, x2=0, x3=1

FALSE

Clique of size 3 ?



$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$

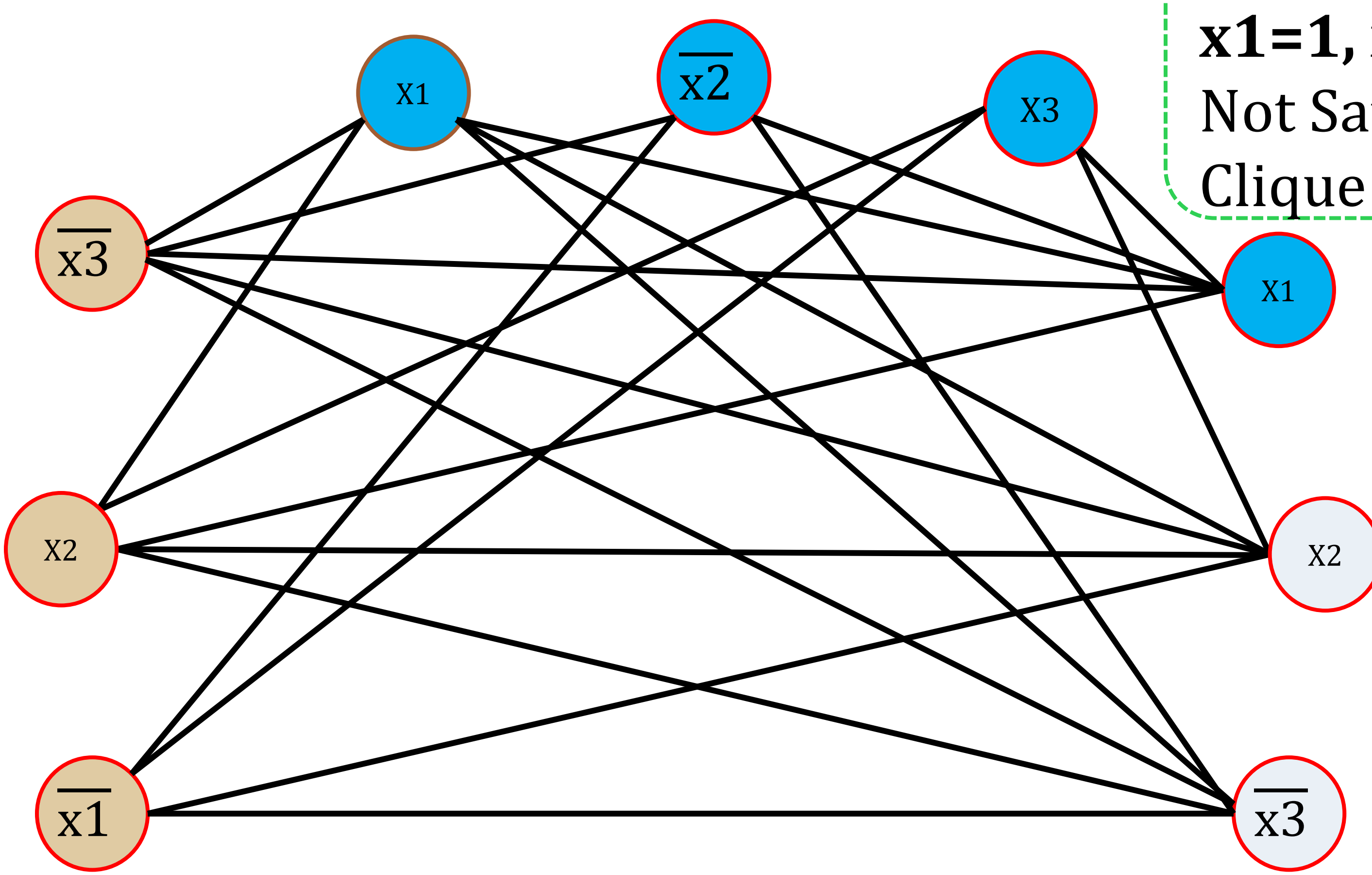
C1C2 C3

Assignment:

x1=1, x2=0, x3=1

Not Satisfiable

Clique of size 3 ?



$$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

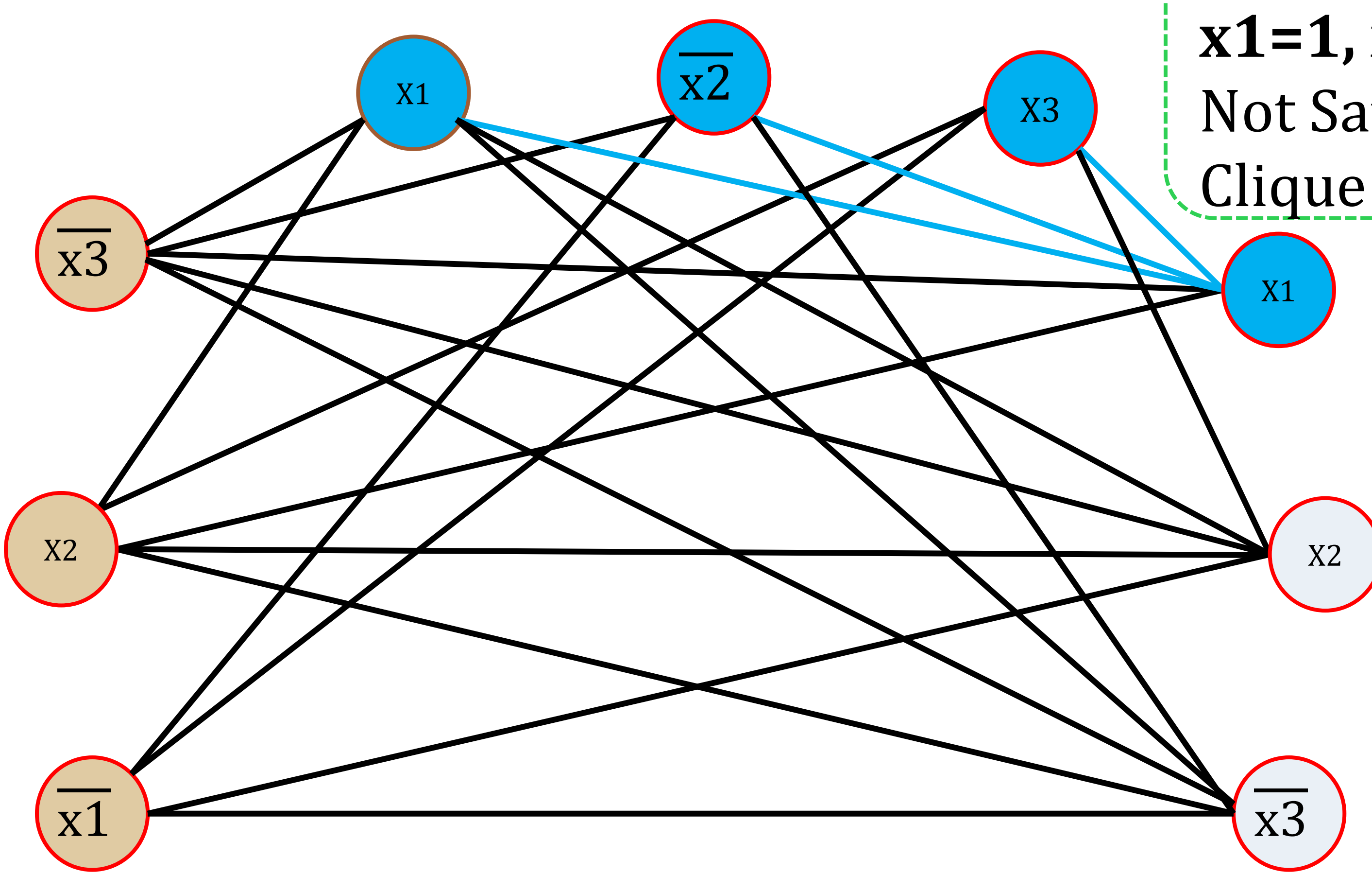
C1C2 C3

Assignment:

x1=1, x2=0, x3=1

Not Satisfiable

Clique of size 3 ?



$\varphi = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$

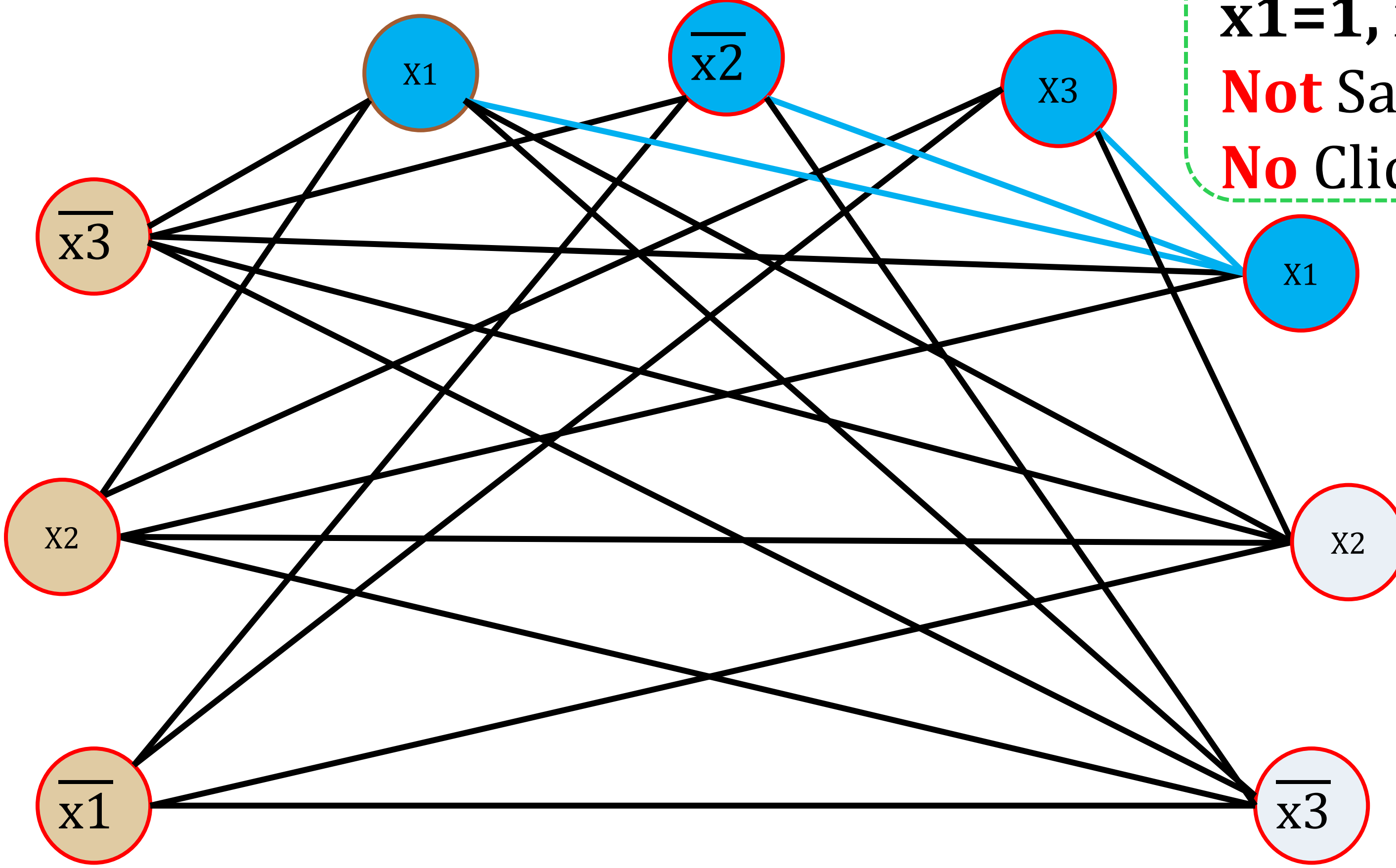
C1C2 C3

Assignment:

x1=1, x2=0, x3=1

Not Satisfiable

No Clique of size 3



Hamiltonian Cycle

Given a directed graph G = (V, E), we say that a cycle C in G is a Hamiltonian cycle if it visits each vertex exactly once. Find C in G.

Hamiltonian Cycle Problem: Given a G, does it contain a Hamiltonian cycle?

Hamiltonian Cycle

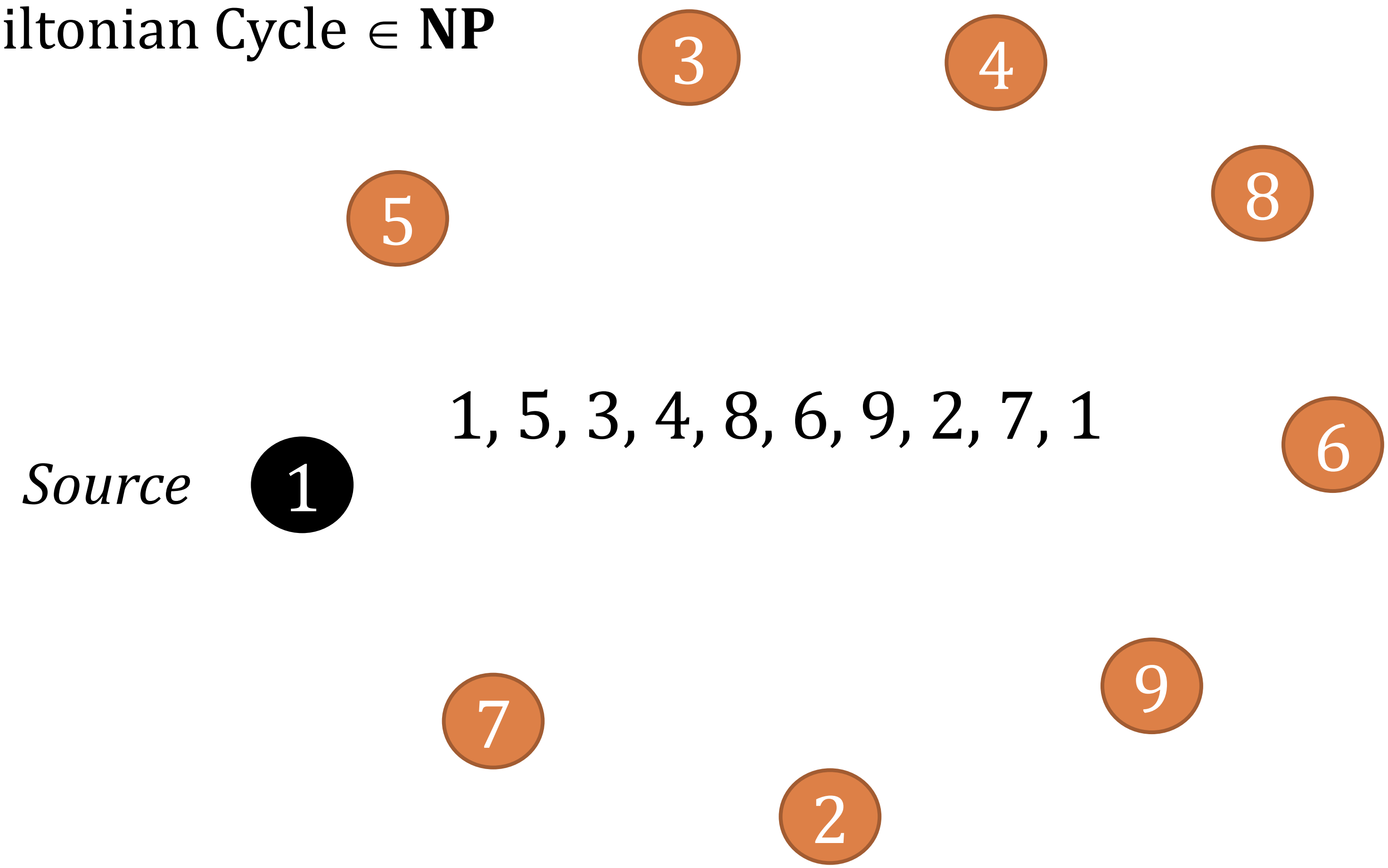
Hamiltonian Cycle is in NP-Complete ?

- Hamiltonian Cycle $\in$ **NP** and
- 3-CNF SAT $\leq_p$ Hamiltonian Cycle

if the formula is satisfiable then the graph has a Hamiltonian Cycle

Verification of Hamiltonian Cycle

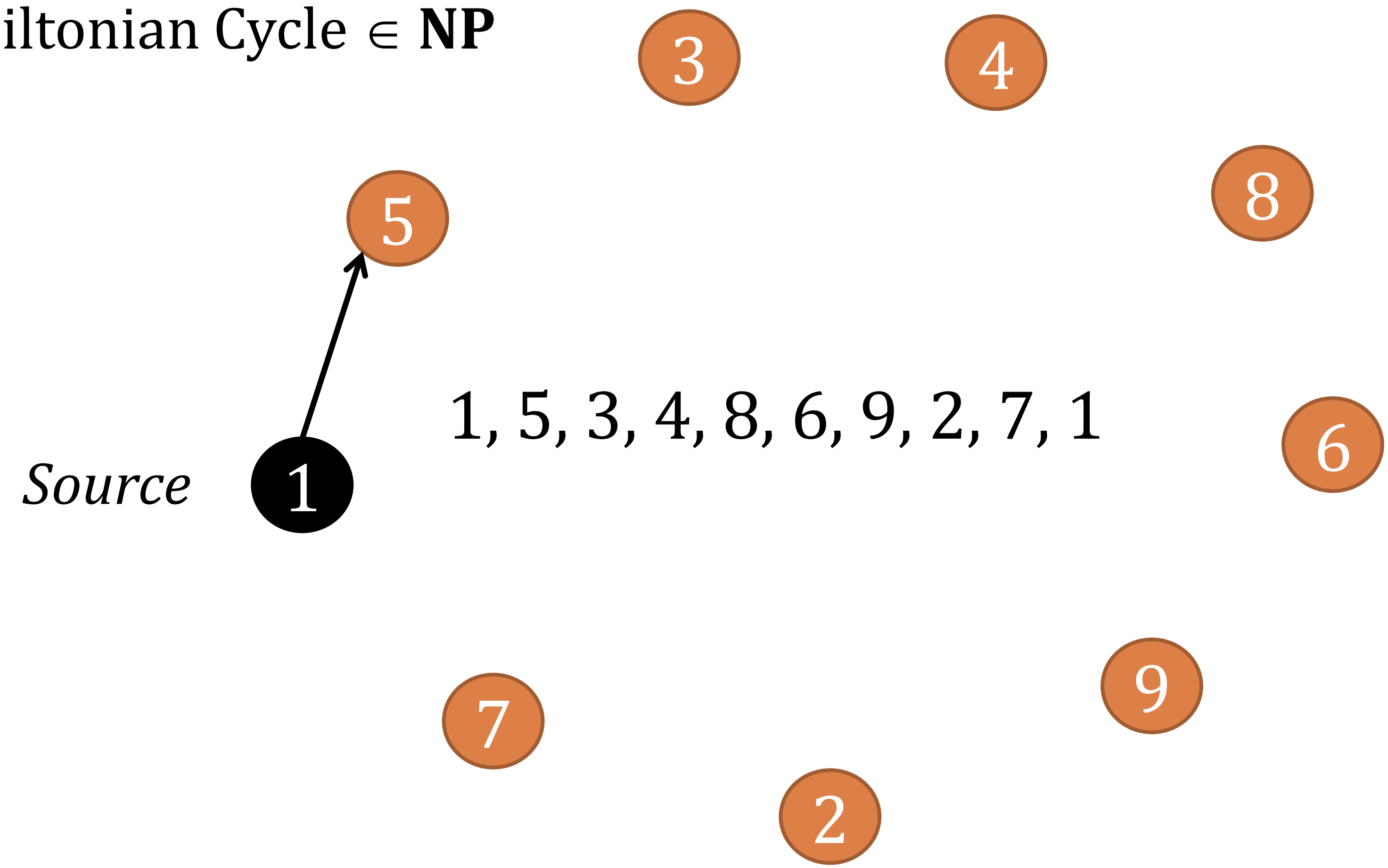
Hamiltonian Cycle $\in$ **NP**



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

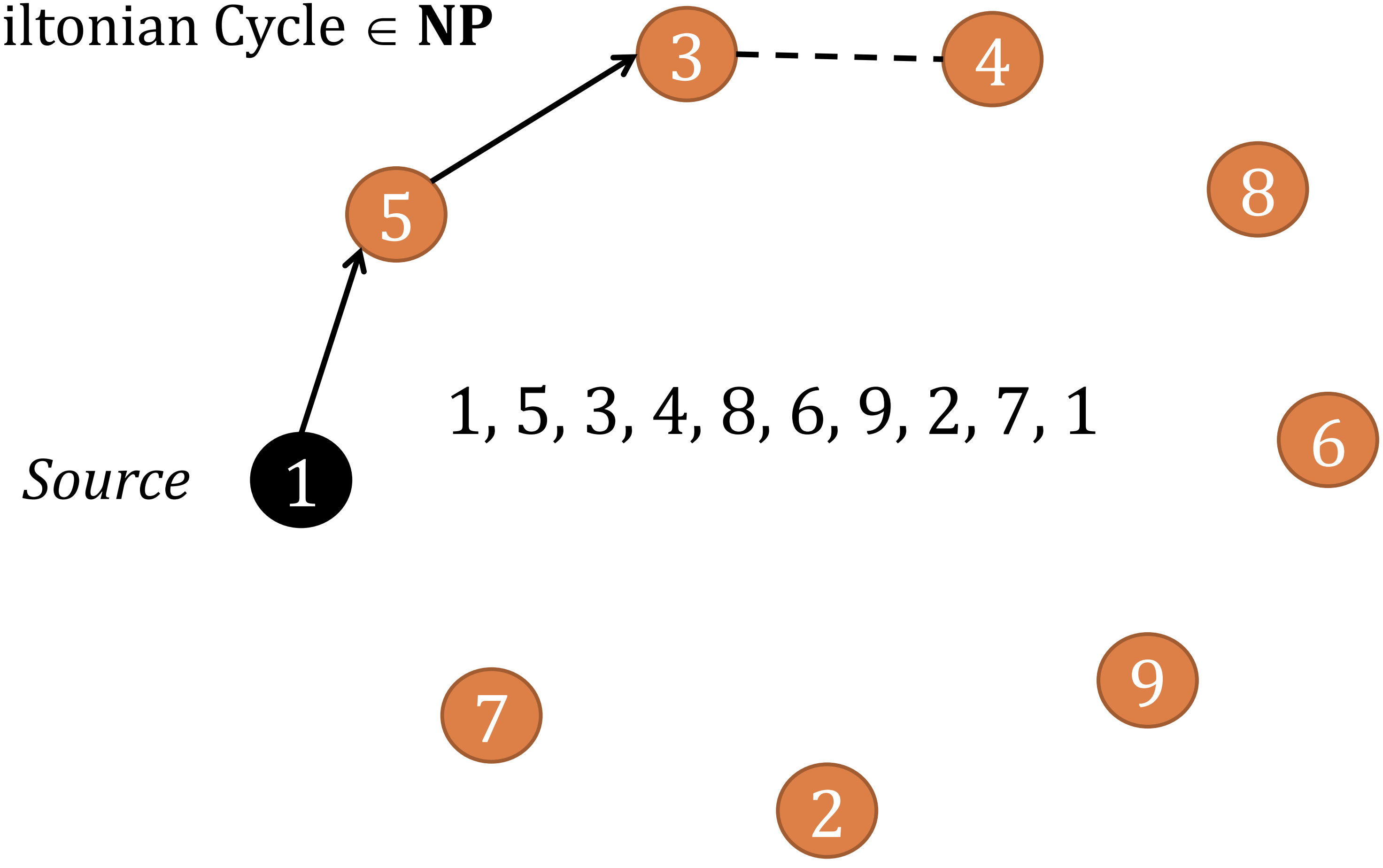
Verification of Hamiltonian Cycle

Hamiltonian Cycle $\in$ **NP**



Verification of Hamiltonian Cycle

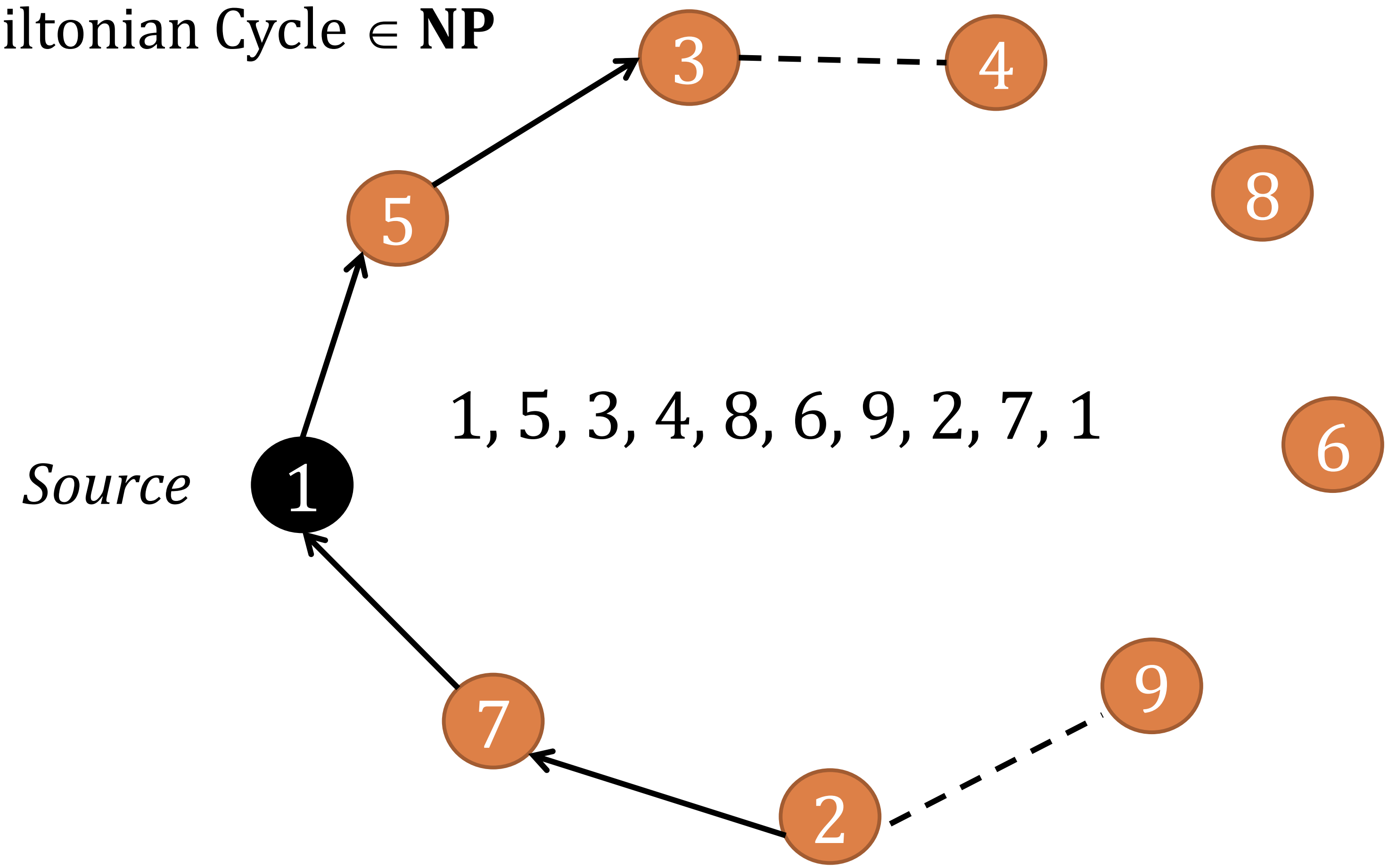
Hamiltonian Cycle ∈ **NP**



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

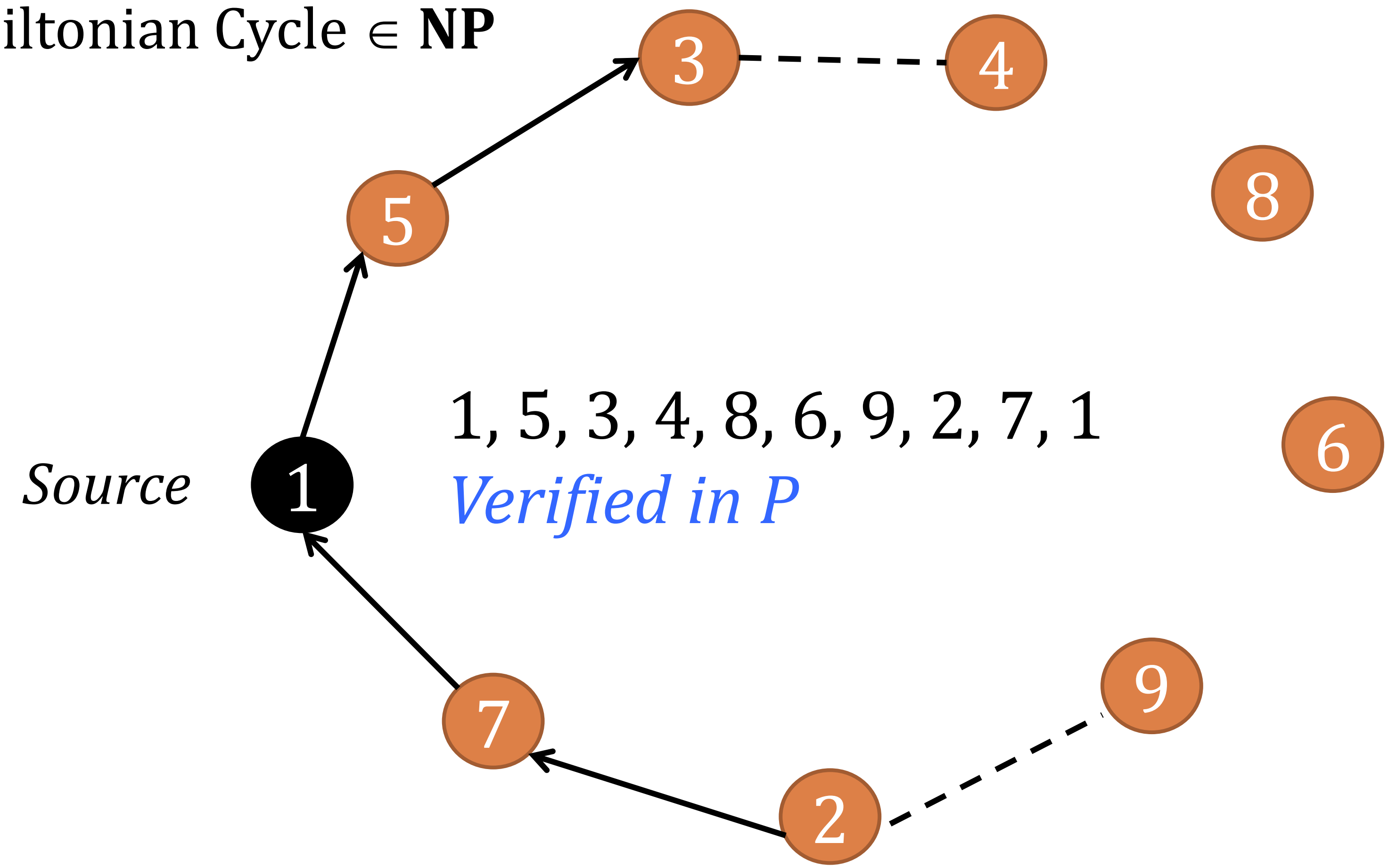
Verification of Hamiltonian Cycle

Hamiltonian Cycle $\in$ **NP**



Verification of Hamiltonian Cycle

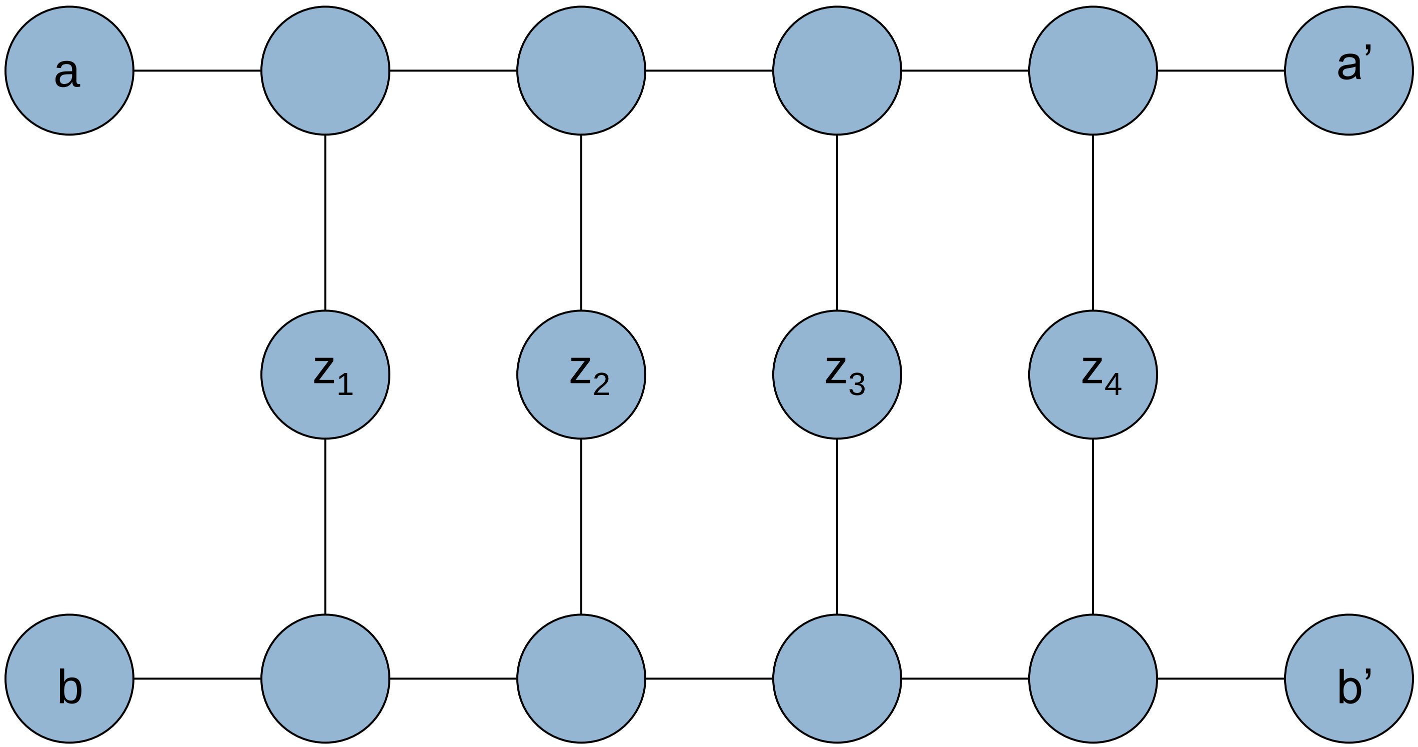
Hamiltonian Cycle ∈ **NP**



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

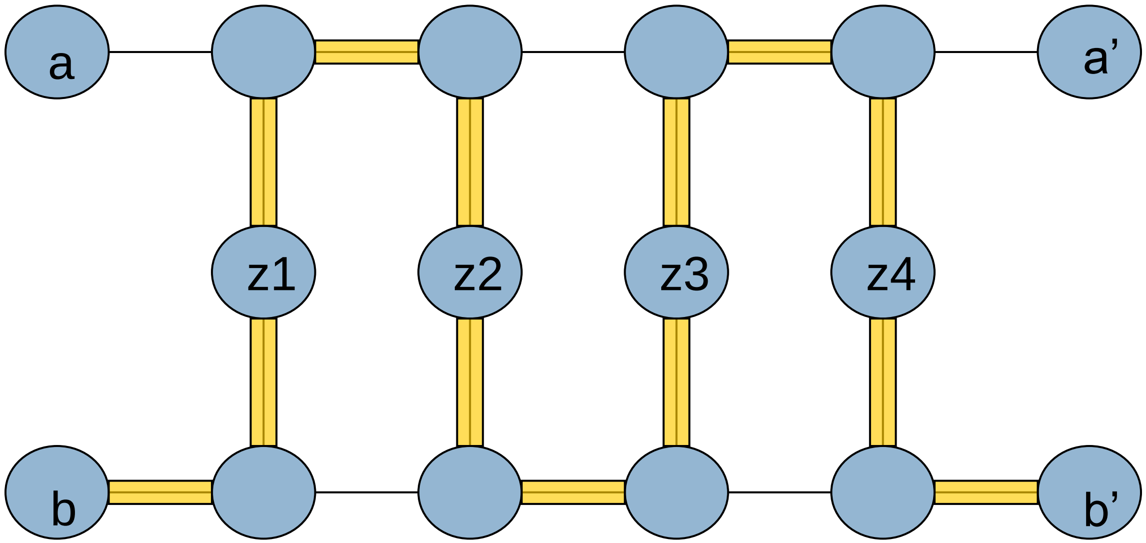
Verified in P

A Widget

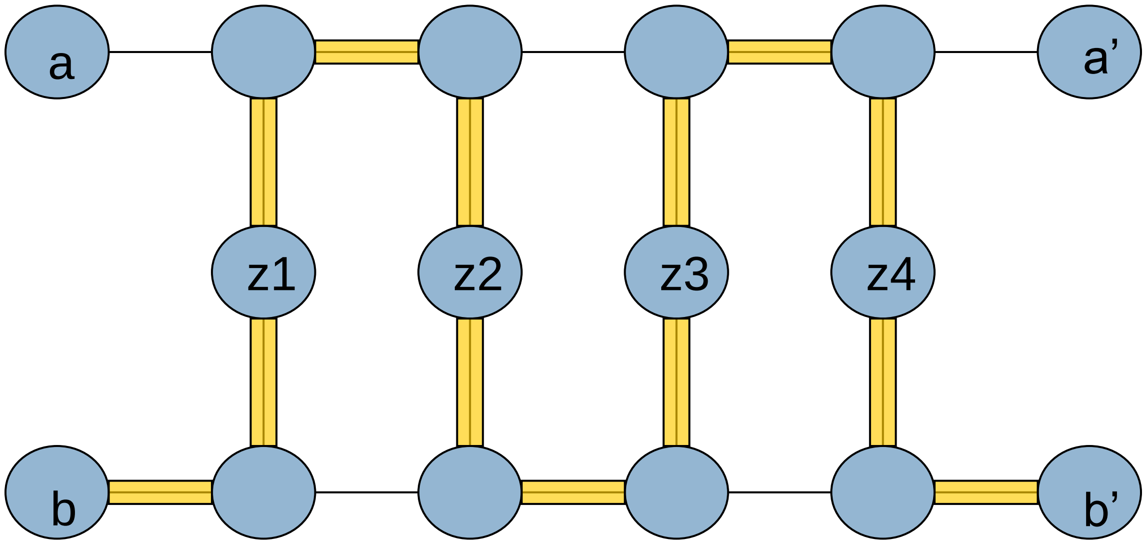


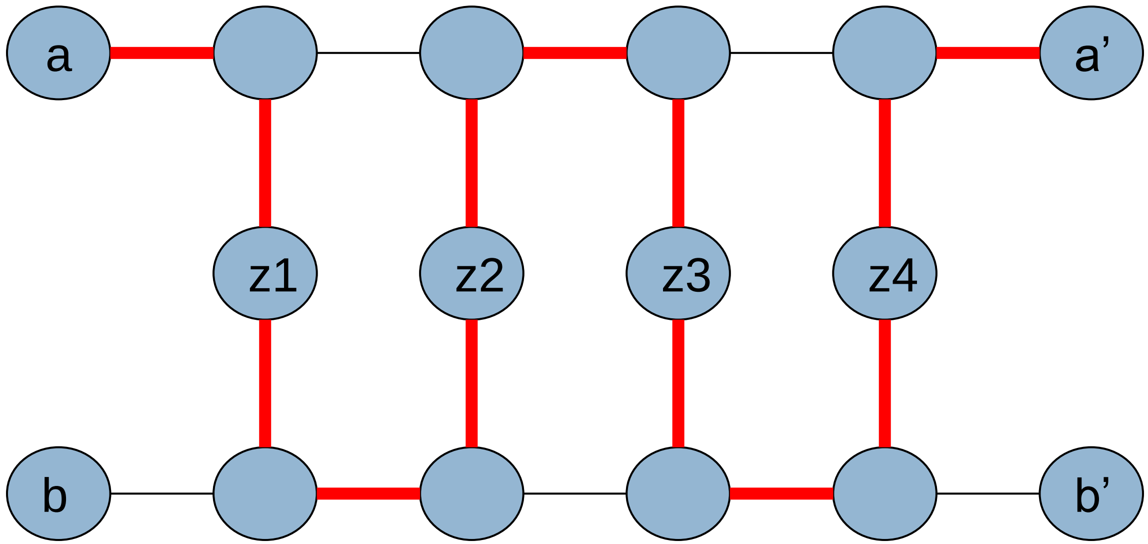
A widget is part of a graph.

Contd…



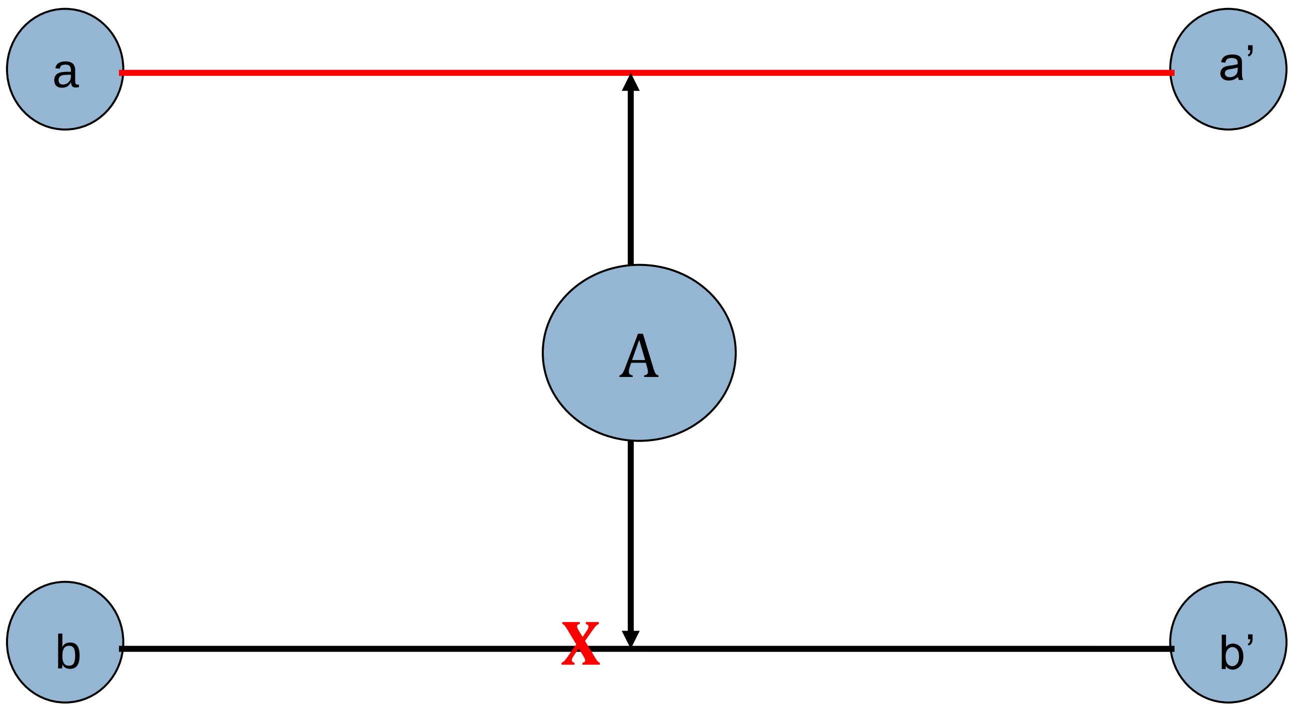
Contd...





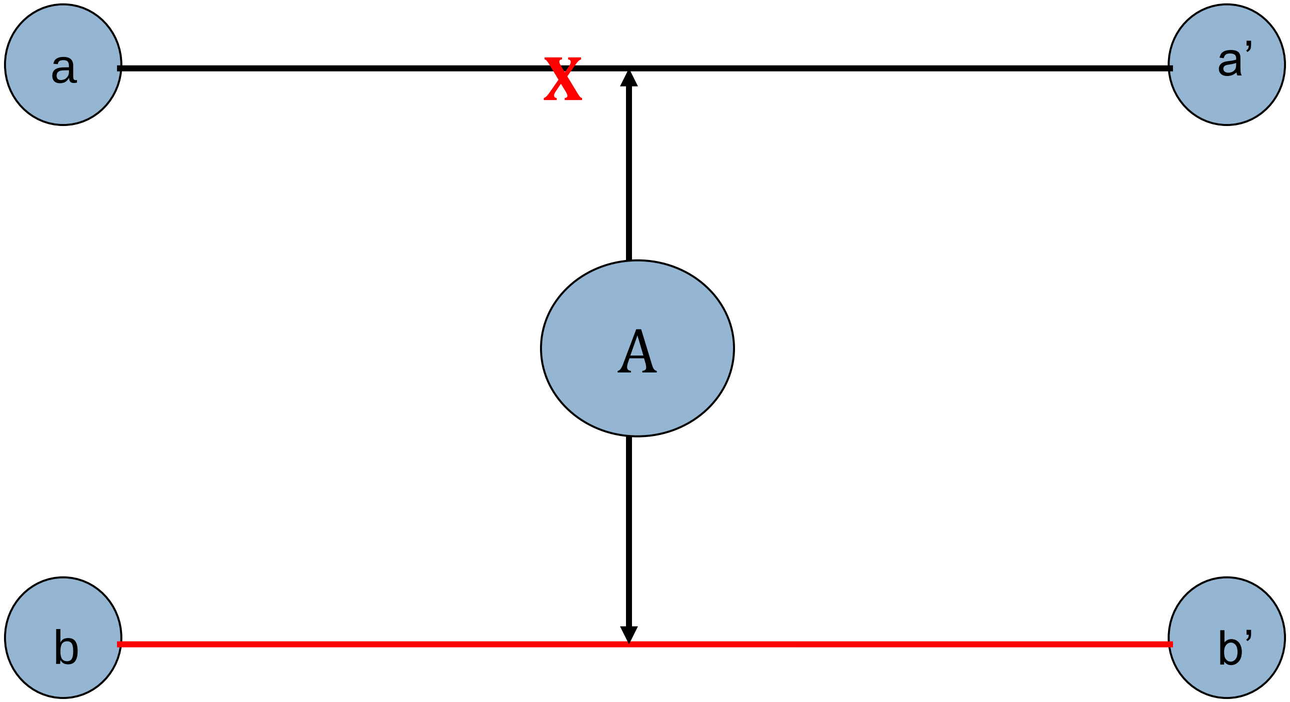
Contd…

A Widget



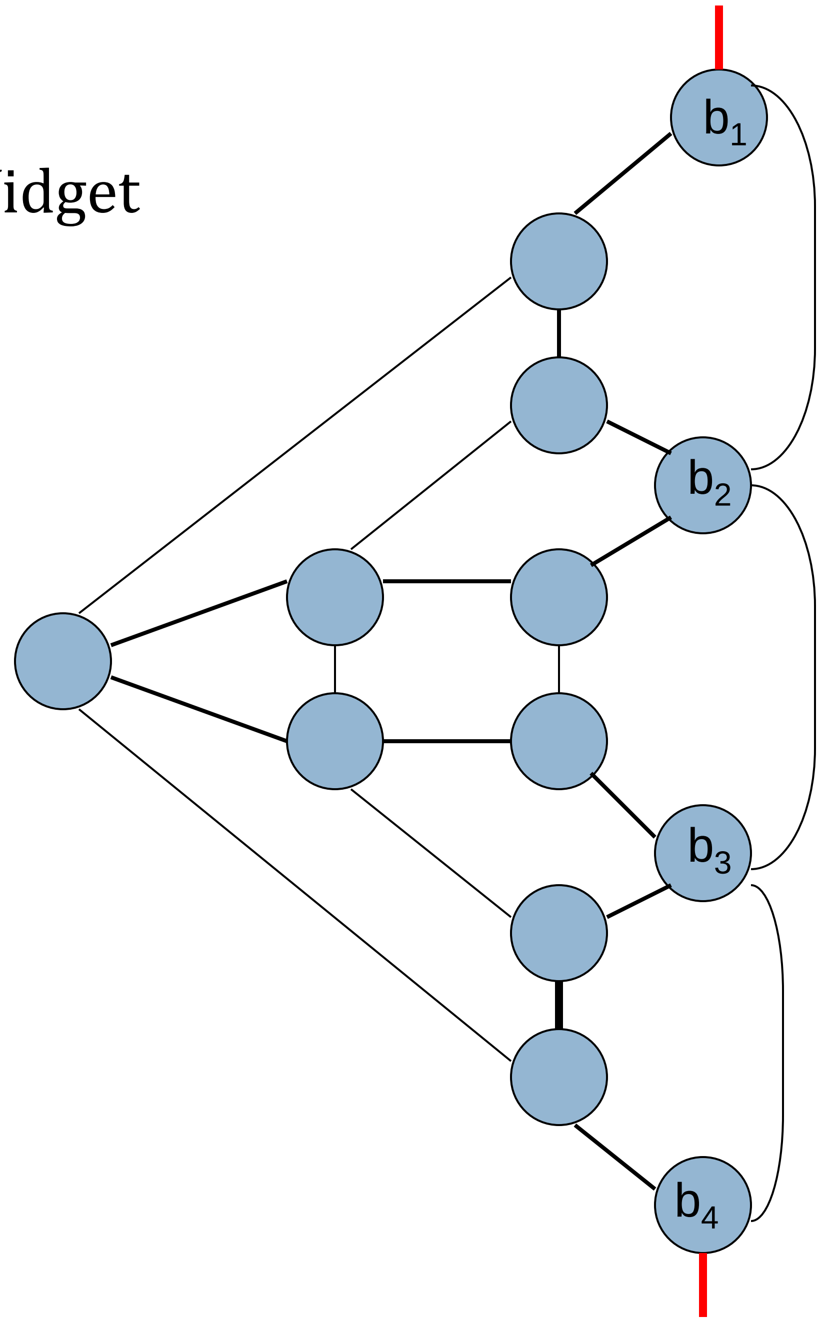
Contd...

A Widget



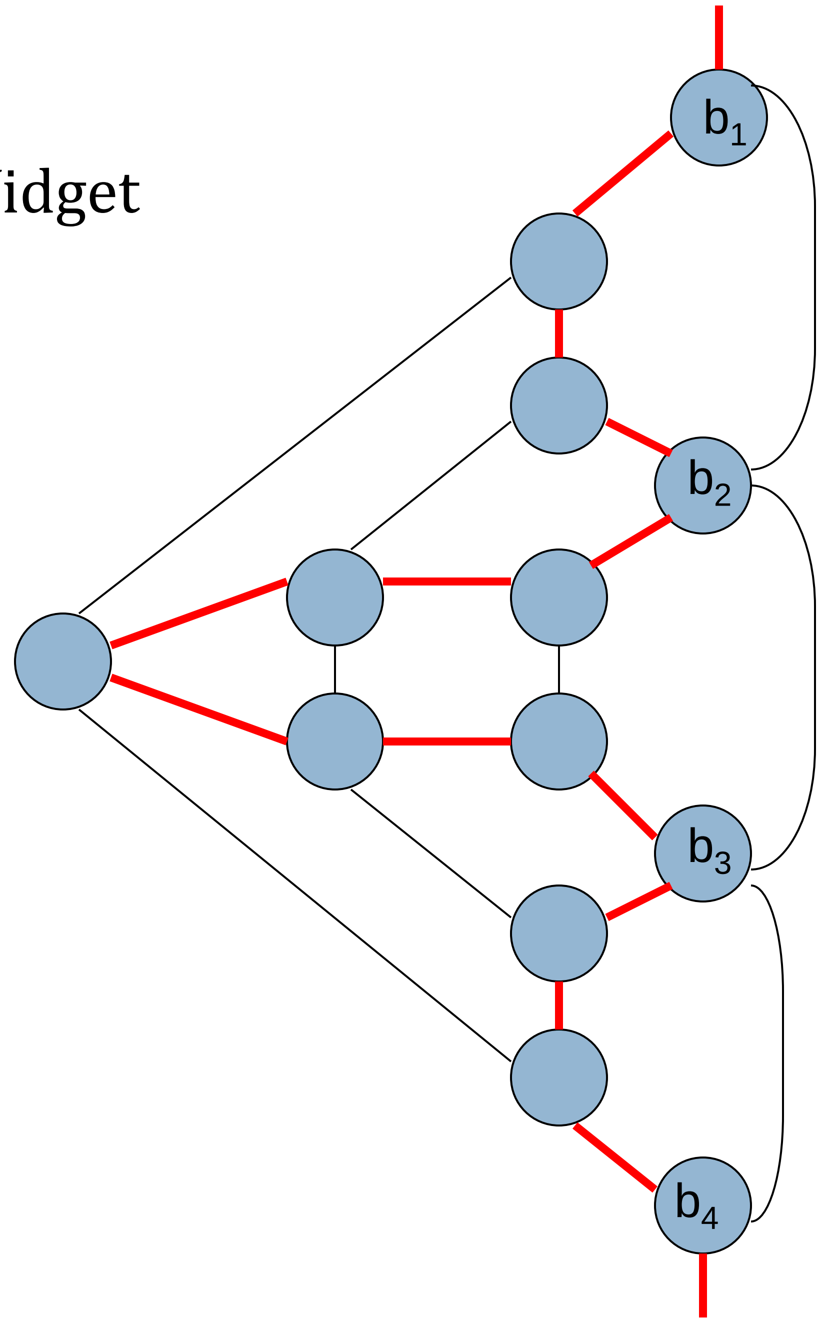
Contd...

B Widget



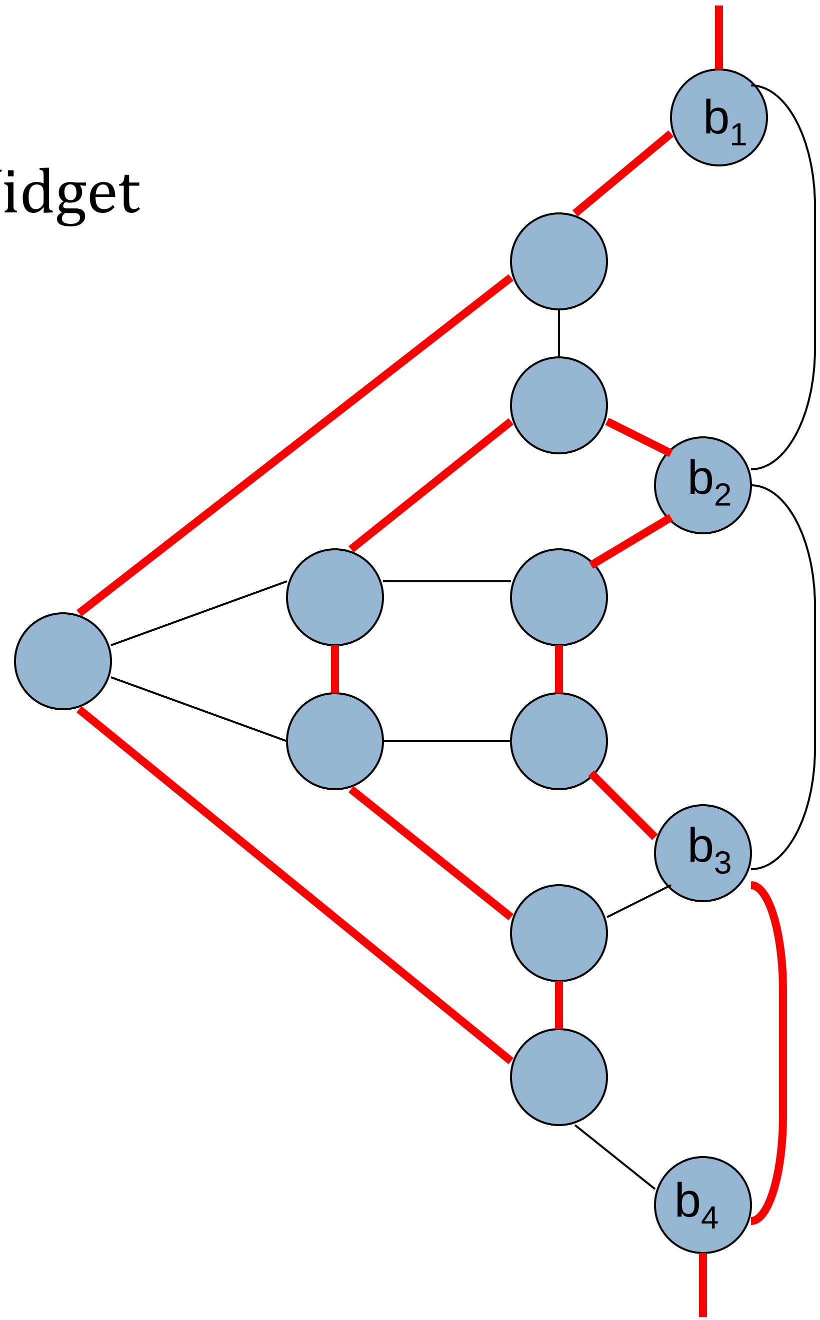
Contd...

B Widget



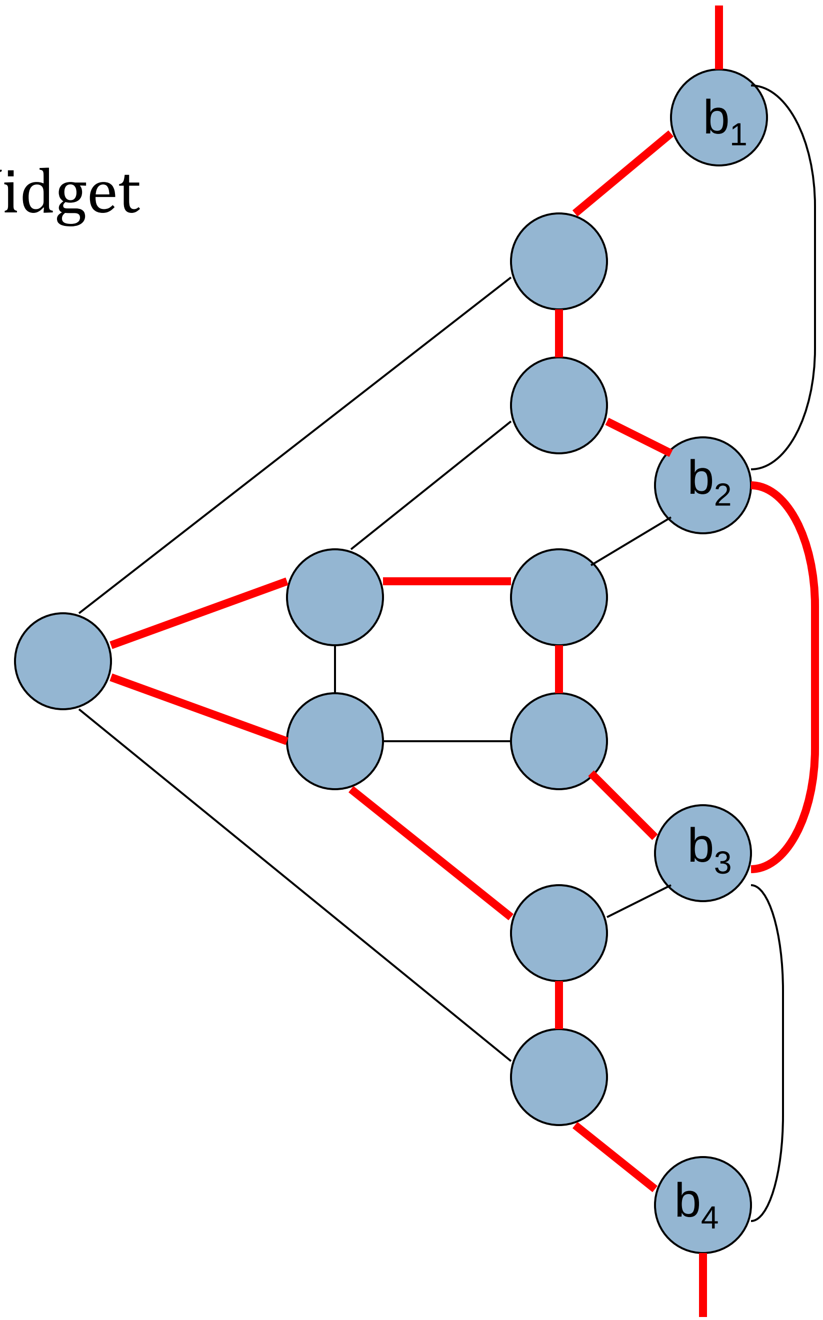
Contd...

B Widget



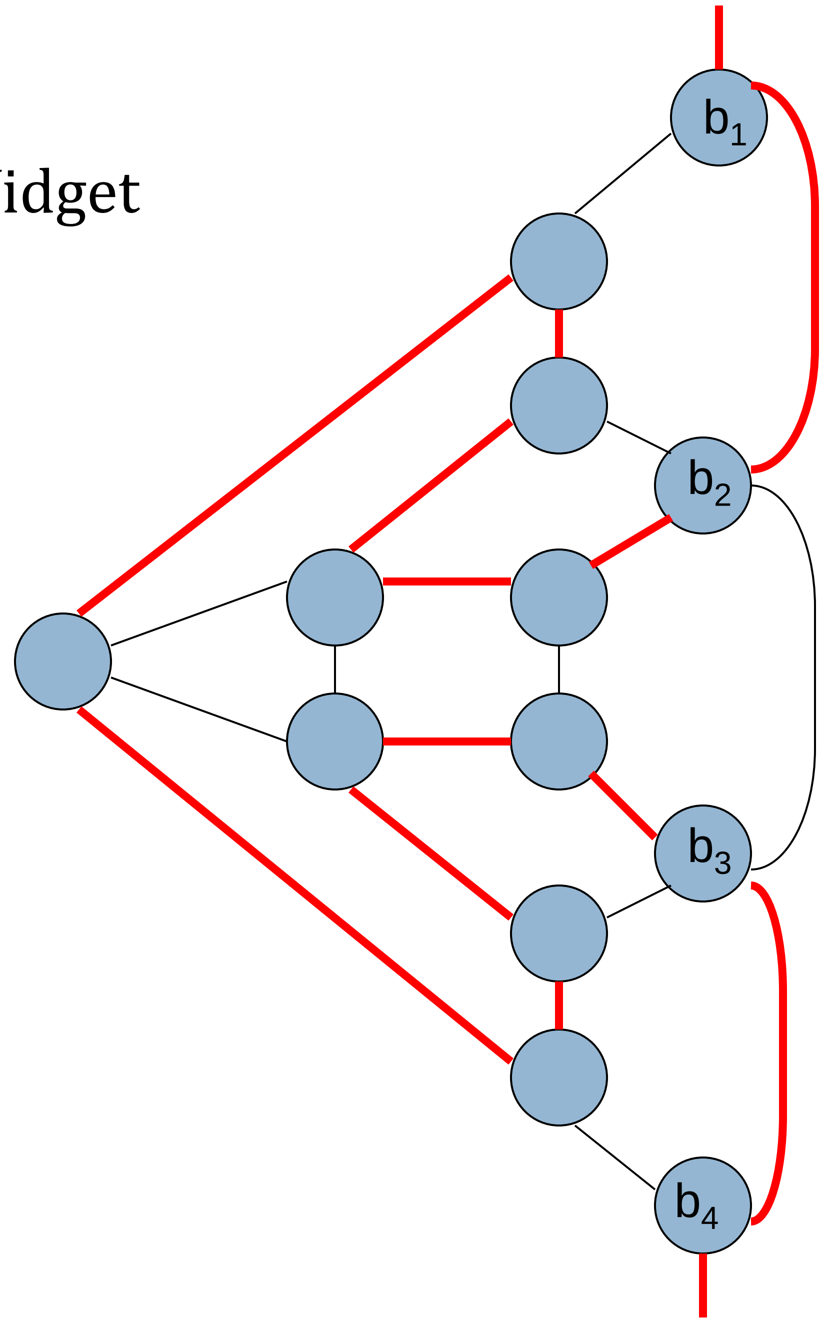
Contd...

B Widget



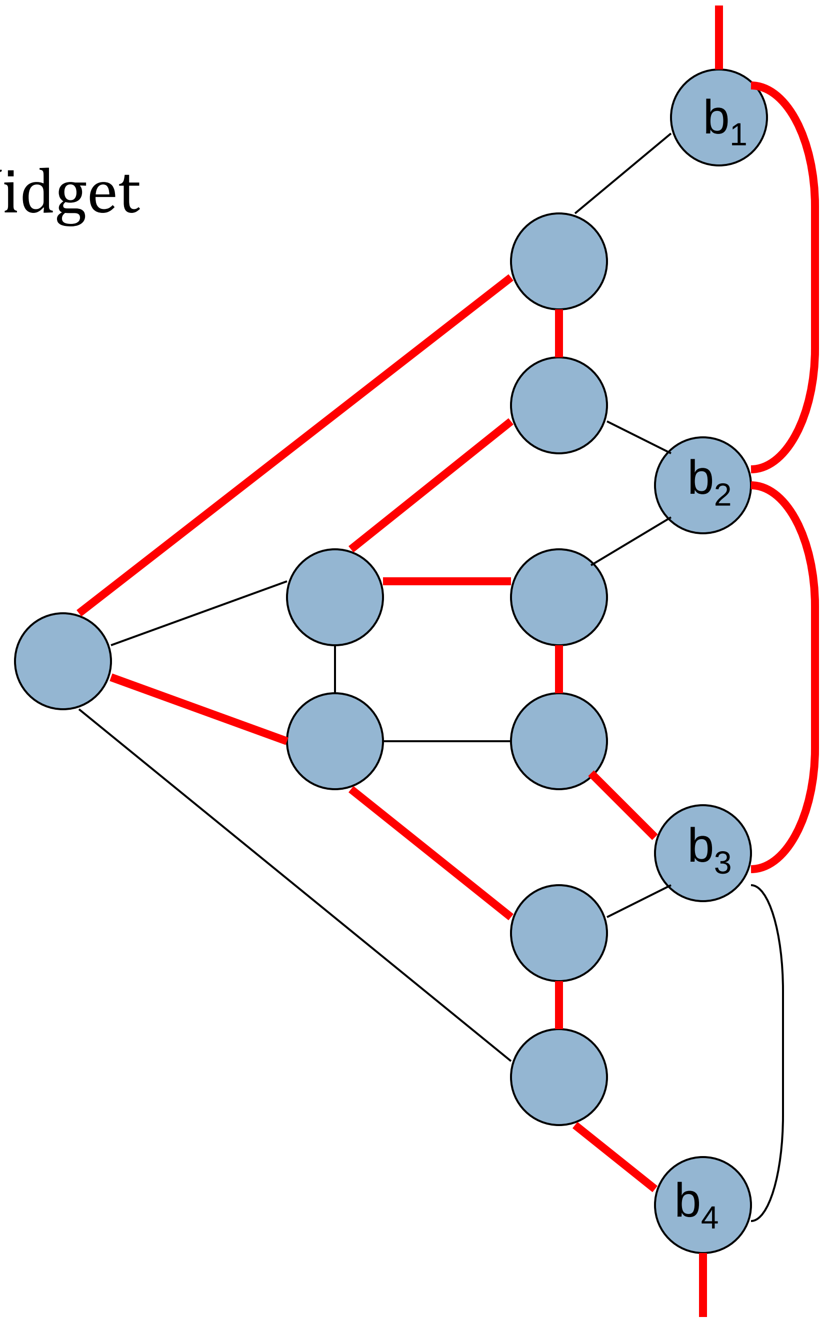
Contd...

B Widget



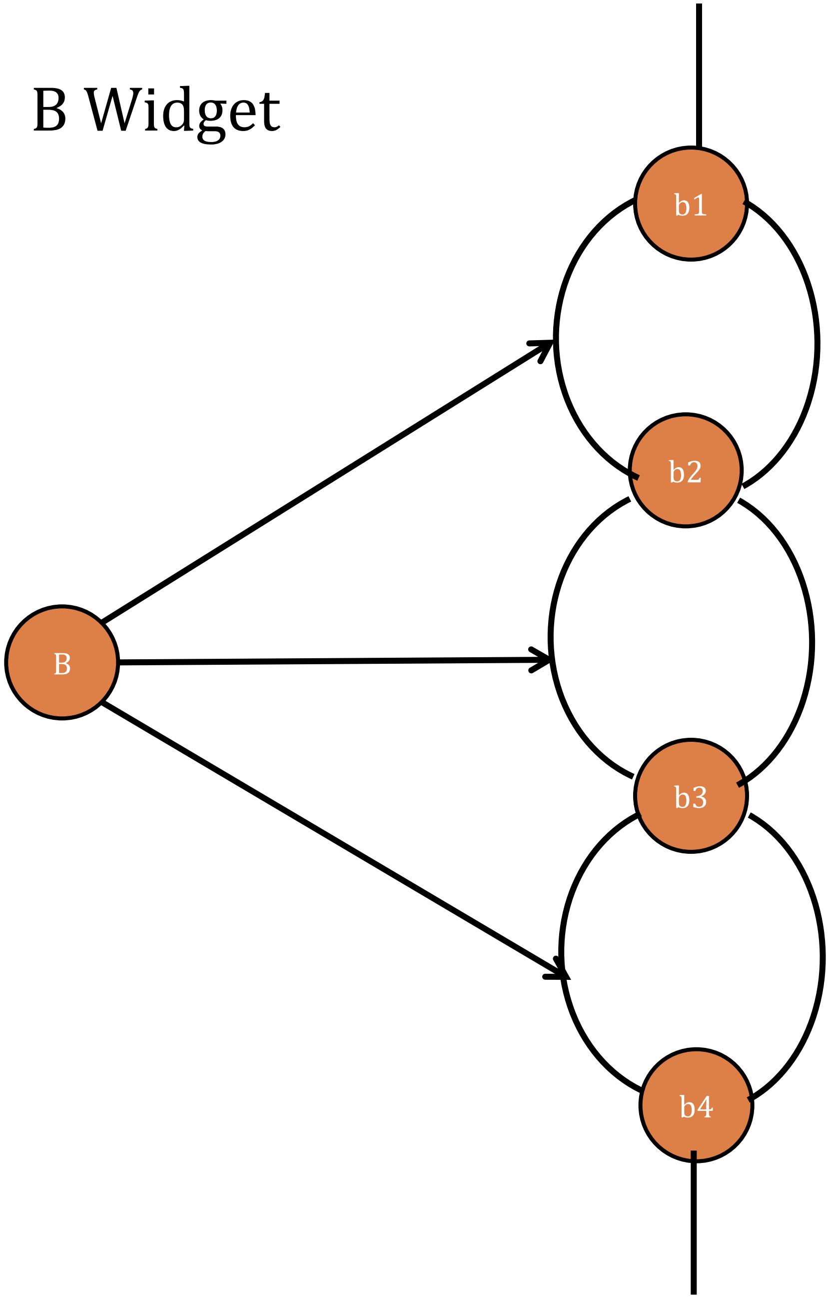
Contd...

B Widget

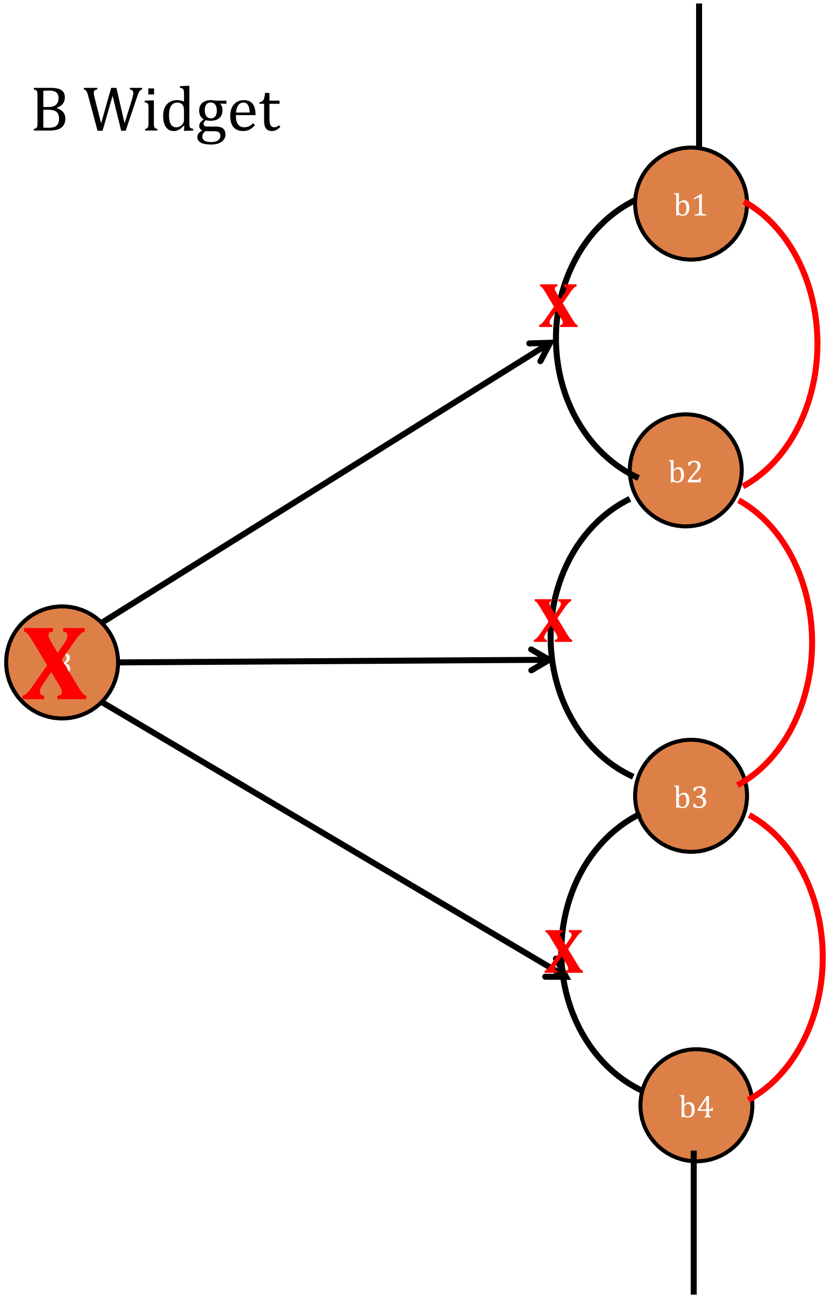


Contd...

B Widget

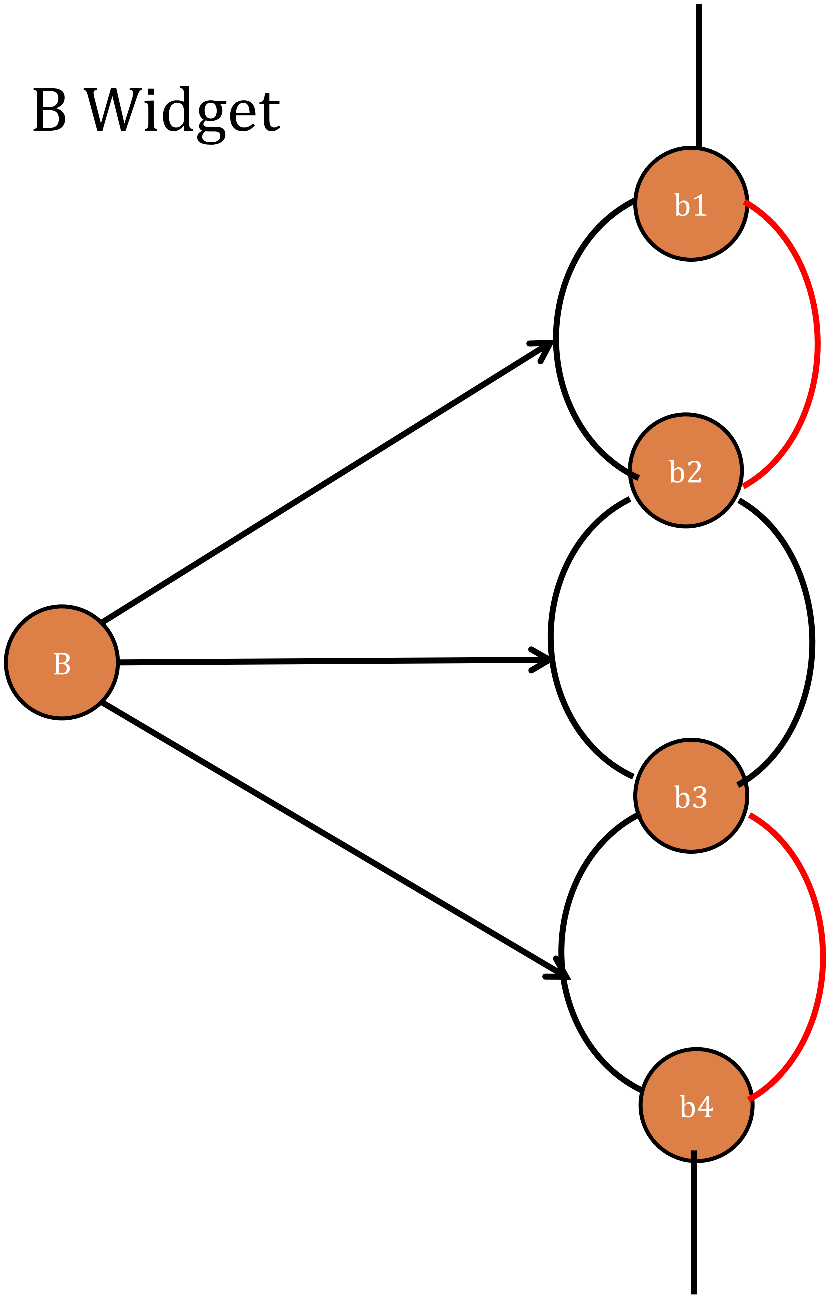


B Widget



All edges can not be traversed opposite to widget B to visit remaining vetreces of the Widget

B Widget



A subset of the edges can be traversed opposite to widget B to visit remaining vetreces of the Widget B

Reducing to Hamiltonian Cycle

3-CNF SAT $\leq_p$ Hamiltonian Cycle

Reducing to Hamiltonian Cycle

3-CNF SAT $\leq_p$ Hamiltonian Cycle

$$F = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

C1 C2 C3

$\leq_p$

GRAPH

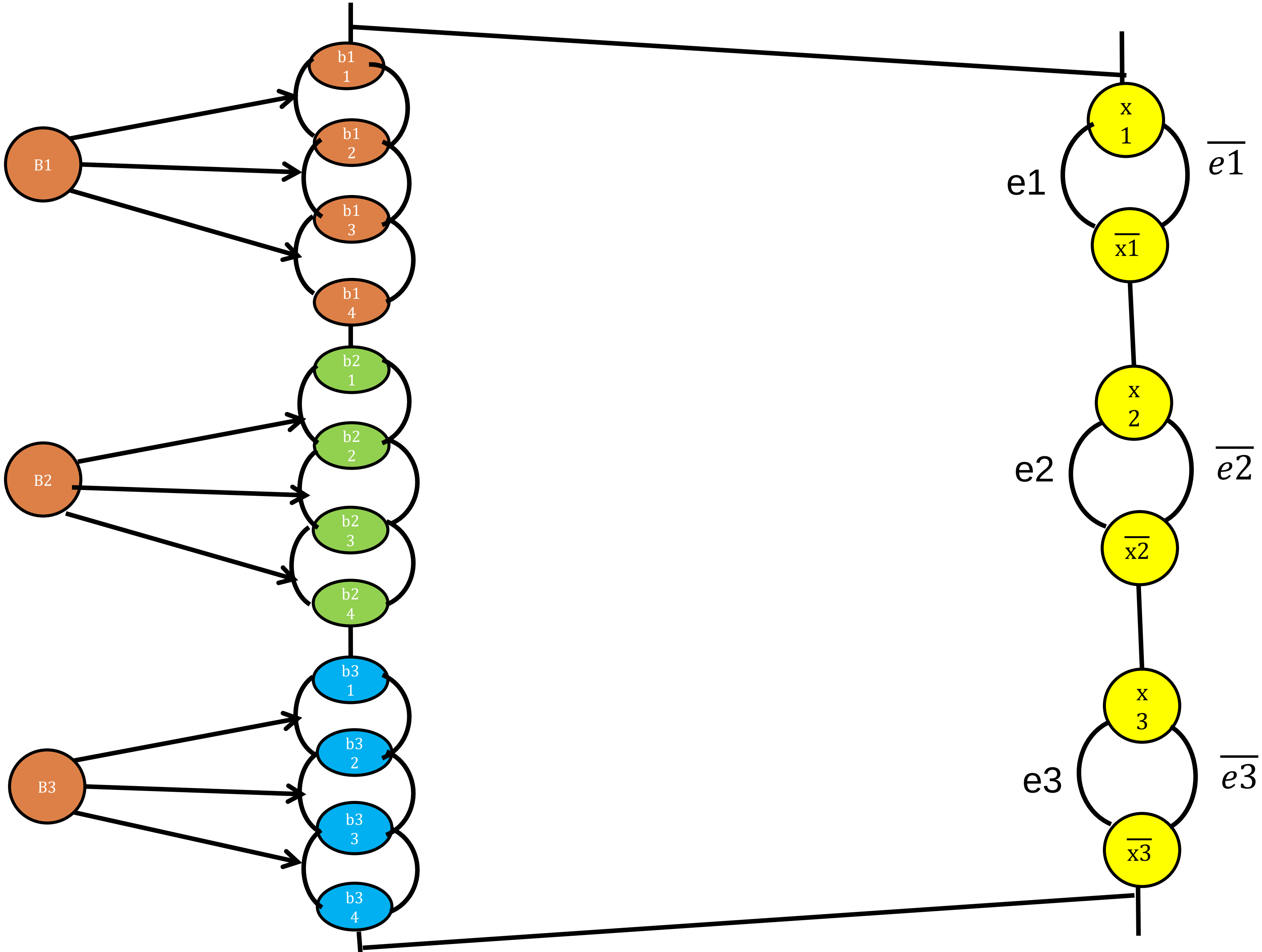
Reduring to Hamiltonian Cycle

3-CNF SAT $\leq_p$ Hamiltonian Cycle

$$F = \underbrace{(\overline{x1} \vee x2 \vee \overline{x3})}_{C1} \wedge \underbrace{(x1 \vee \overline{x2} \vee x3)}_{C2} \wedge \underbrace{(x1 \vee x2 \vee \overline{x3})}_{C3} \quad \leq_p \quad \boxed{\text{GRAPH}}$$

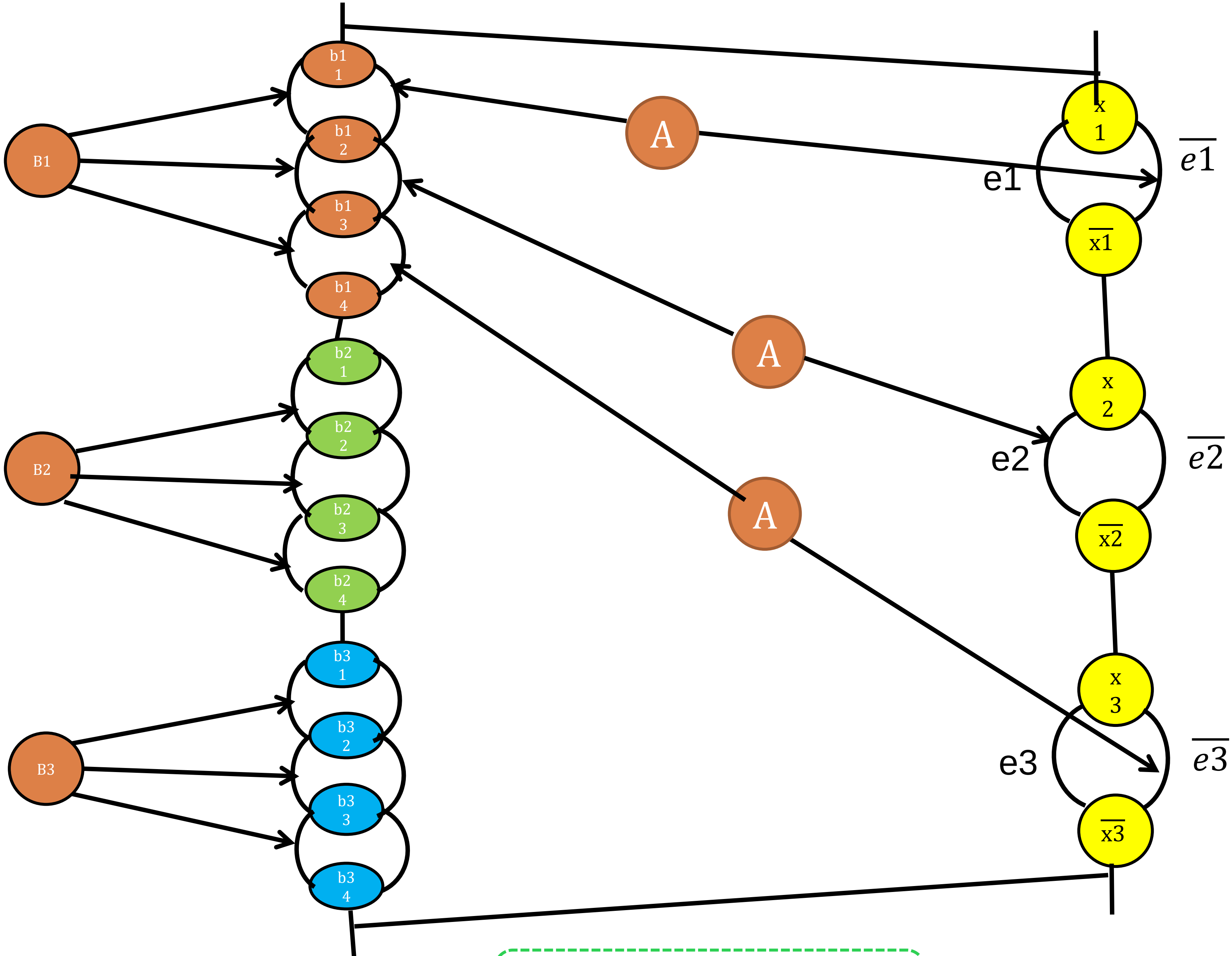
if the formula is satisfiable then the Graph has a Hamiltonian Cycle

Contd...



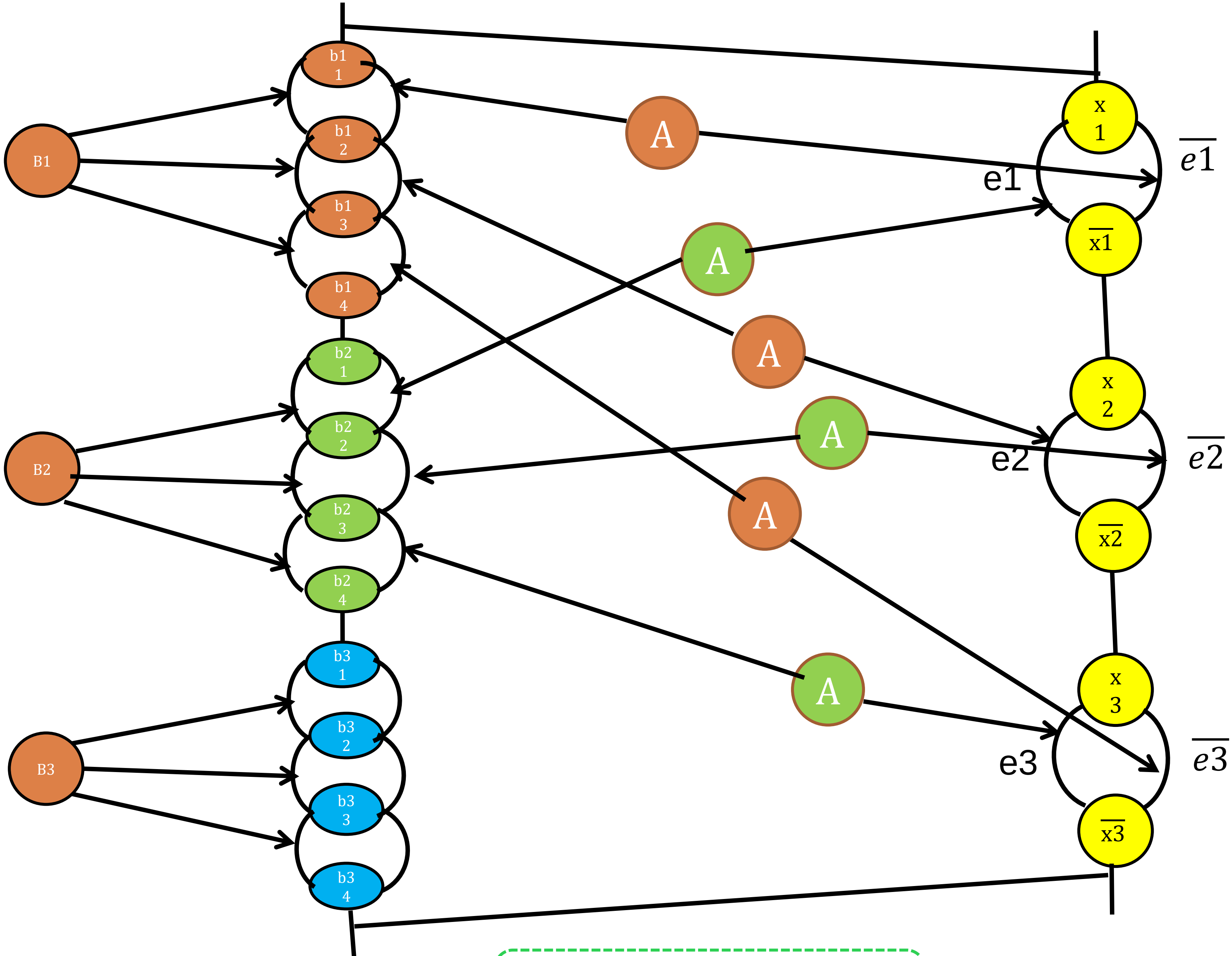
$$F = \underbrace{(\overline{x1} \vee x2 \vee \overline{x3})}_{C1} \wedge \underbrace{(x1 \vee \overline{x2} \vee x3)}_{C2} \wedge \underbrace{(x1 \vee x2 \vee \overline{x3})}_{C3}$$

Contd...



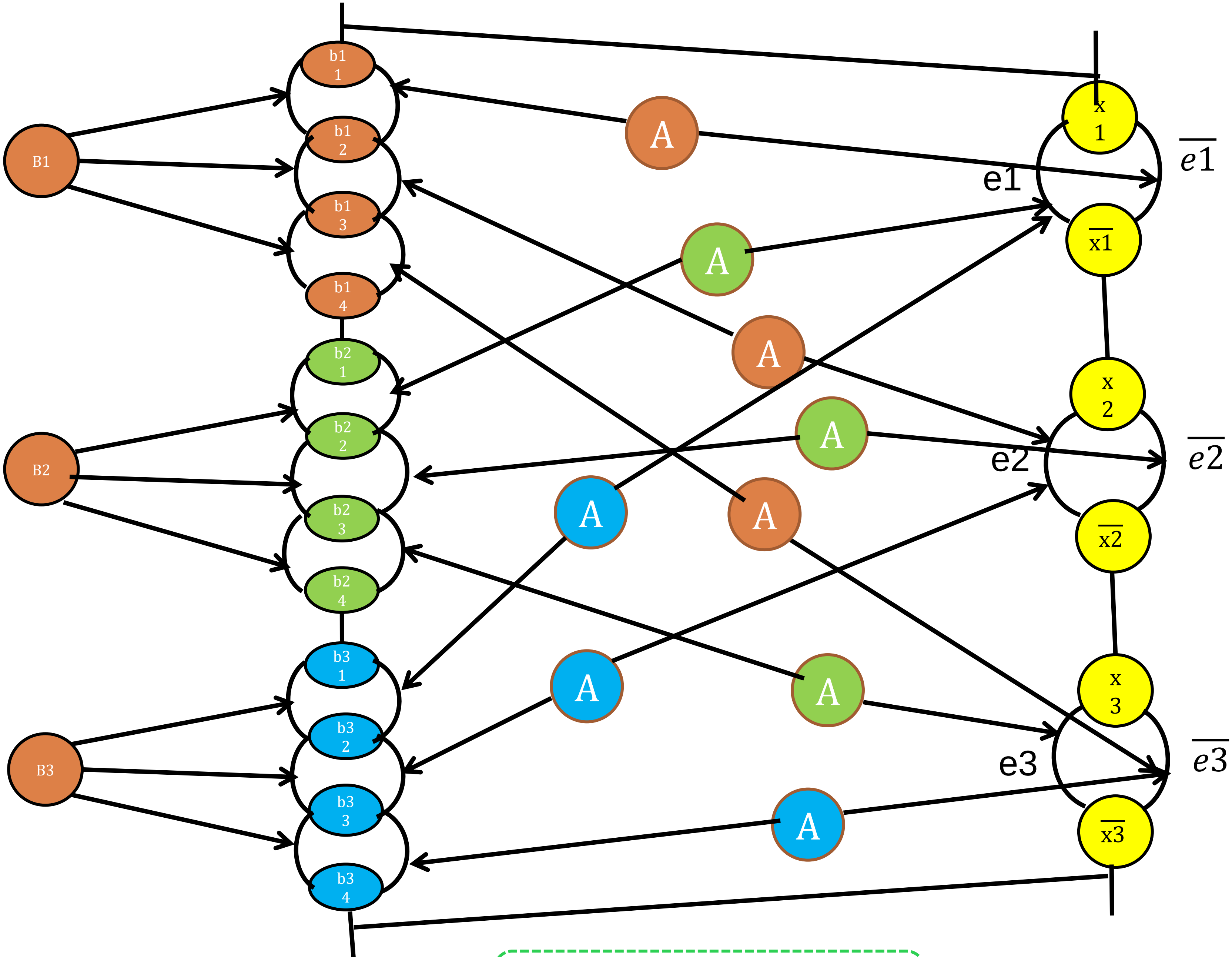
$$C1 = (\overline{x1} \vee x2 \vee \overline{x3})$$

Contd...



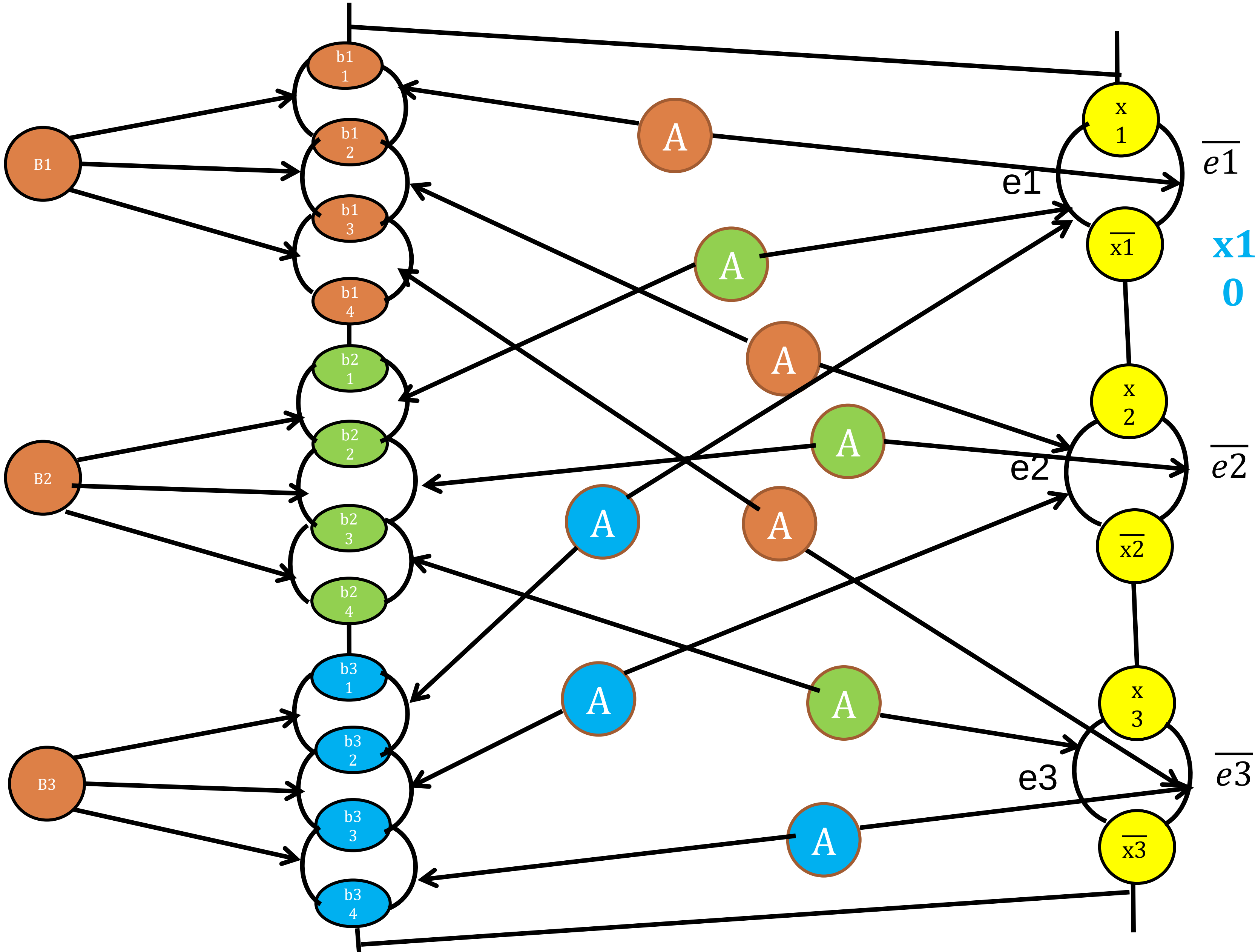
C2= (x1 V $\overline{x2}$ V x3)

Contd...



C3= (x1 V x2 V $\overline{x3}$

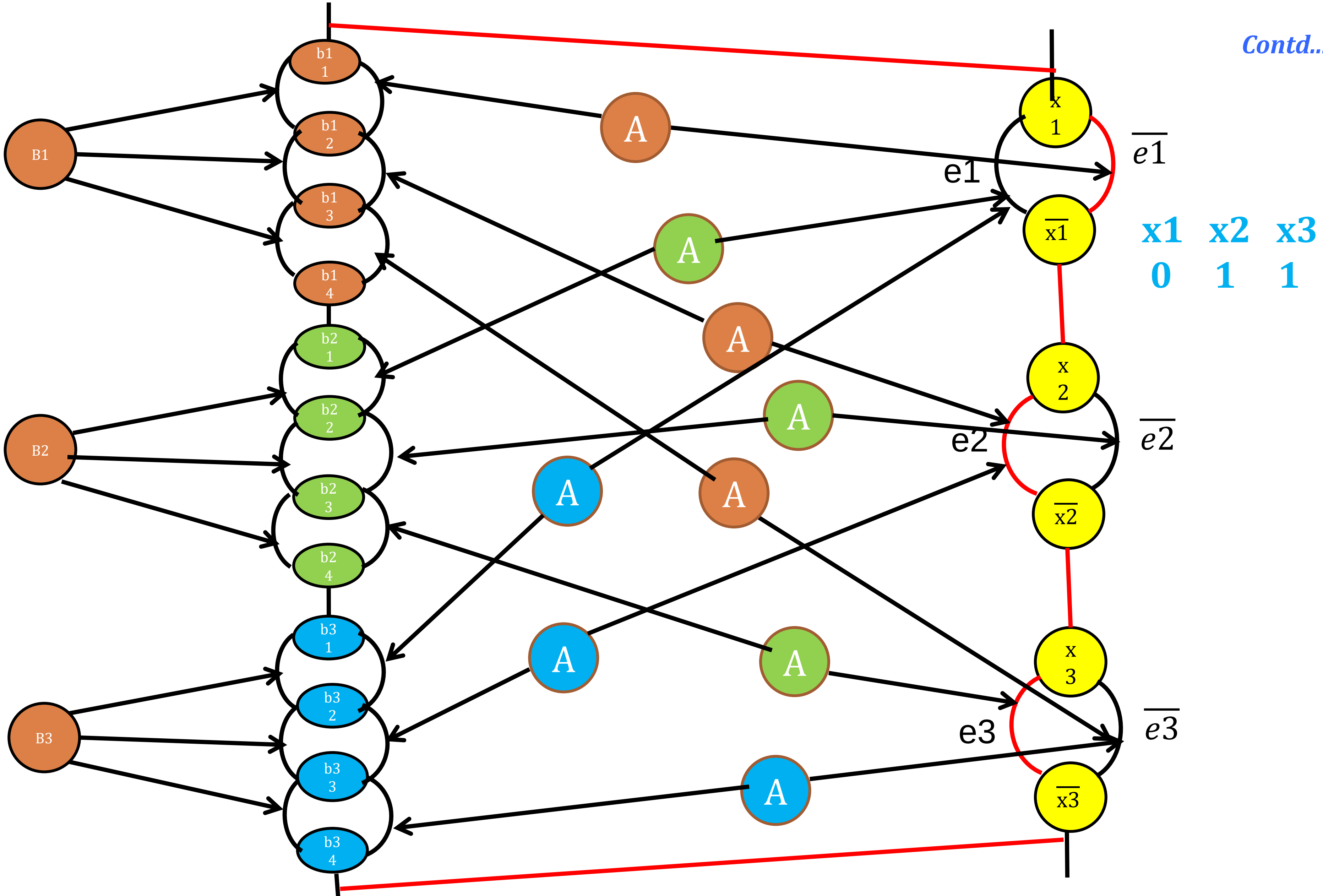
Contd...



x1	x2	x3
0	1	1

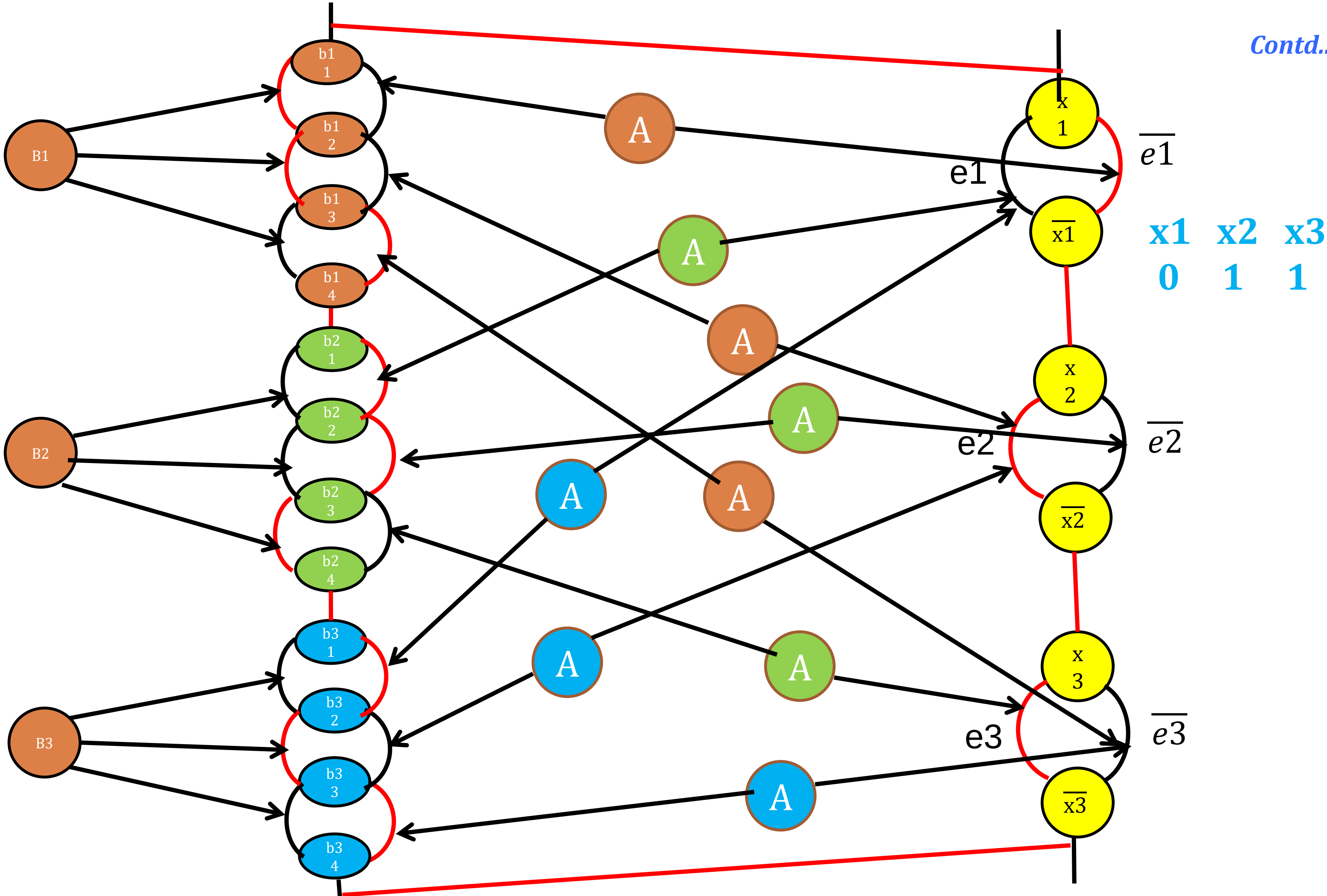
$$F = \underbrace{(\overline{x1} \vee x2 \vee \overline{x3})}_{C1} \wedge \underbrace{(x1 \vee \overline{x2} \vee x3)}_{C2} \wedge \underbrace{(x1 \vee x2 \vee \overline{x3})}_{C3}$$

Contd...



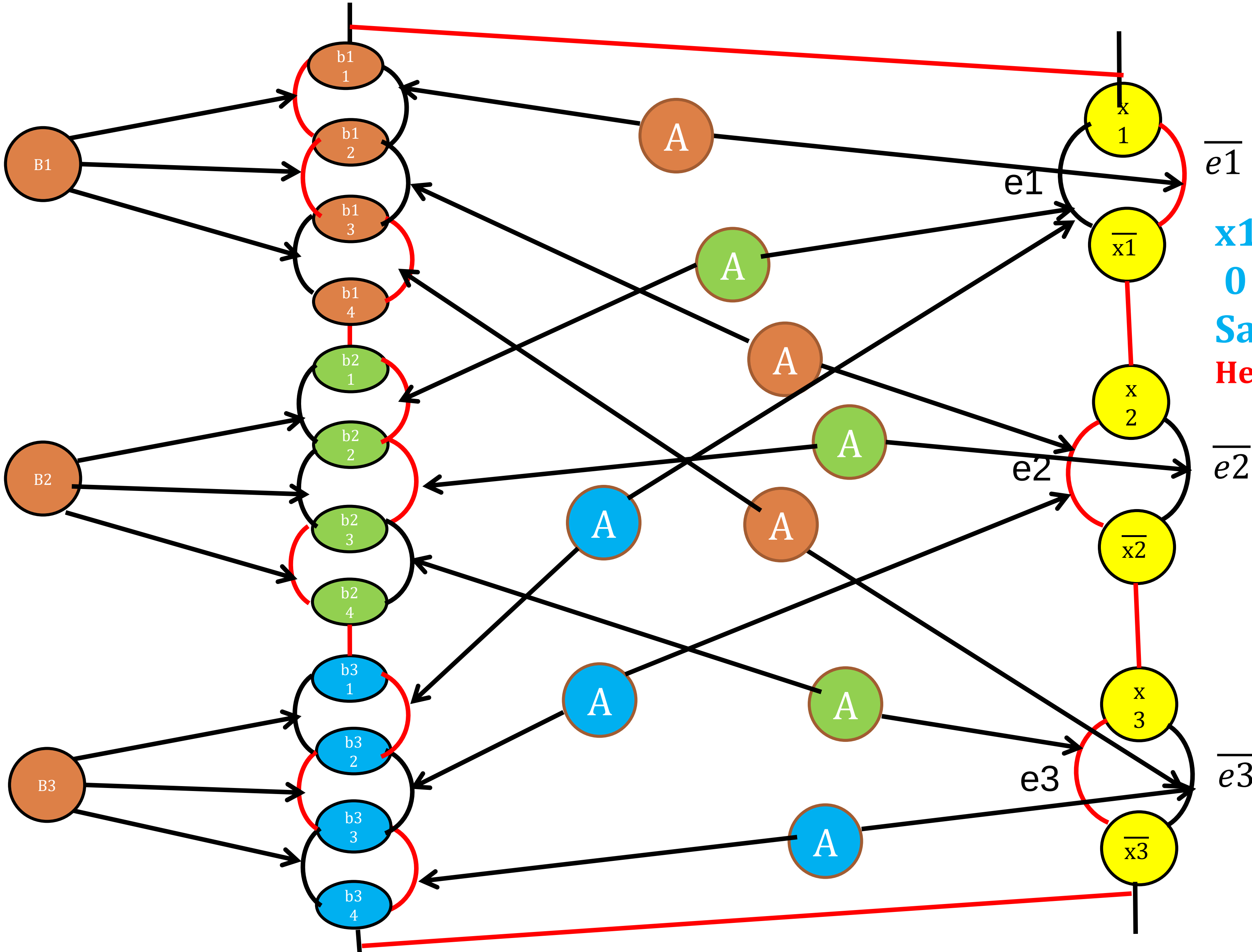
$$F = (\underbrace{\overline{x1} \vee x2 \vee \overline{x3}}_{C1}) \wedge (\underbrace{x1 \vee \overline{x2} \vee x3}_{C2}) \wedge (\underbrace{x1 \vee x2 \vee \overline{x3}}_{C3})$$

Contd...



$$F = \underbrace{(\overline{x1} \vee x2 \vee \overline{x3})}_{C1} \wedge \underbrace{(x1 \vee \overline{x2} \vee x3)}_{C2} \wedge \underbrace{(x1 \vee x2 \vee \overline{x3})}_{C3}$$

Contd...



x1 x2 x3

0 1 1

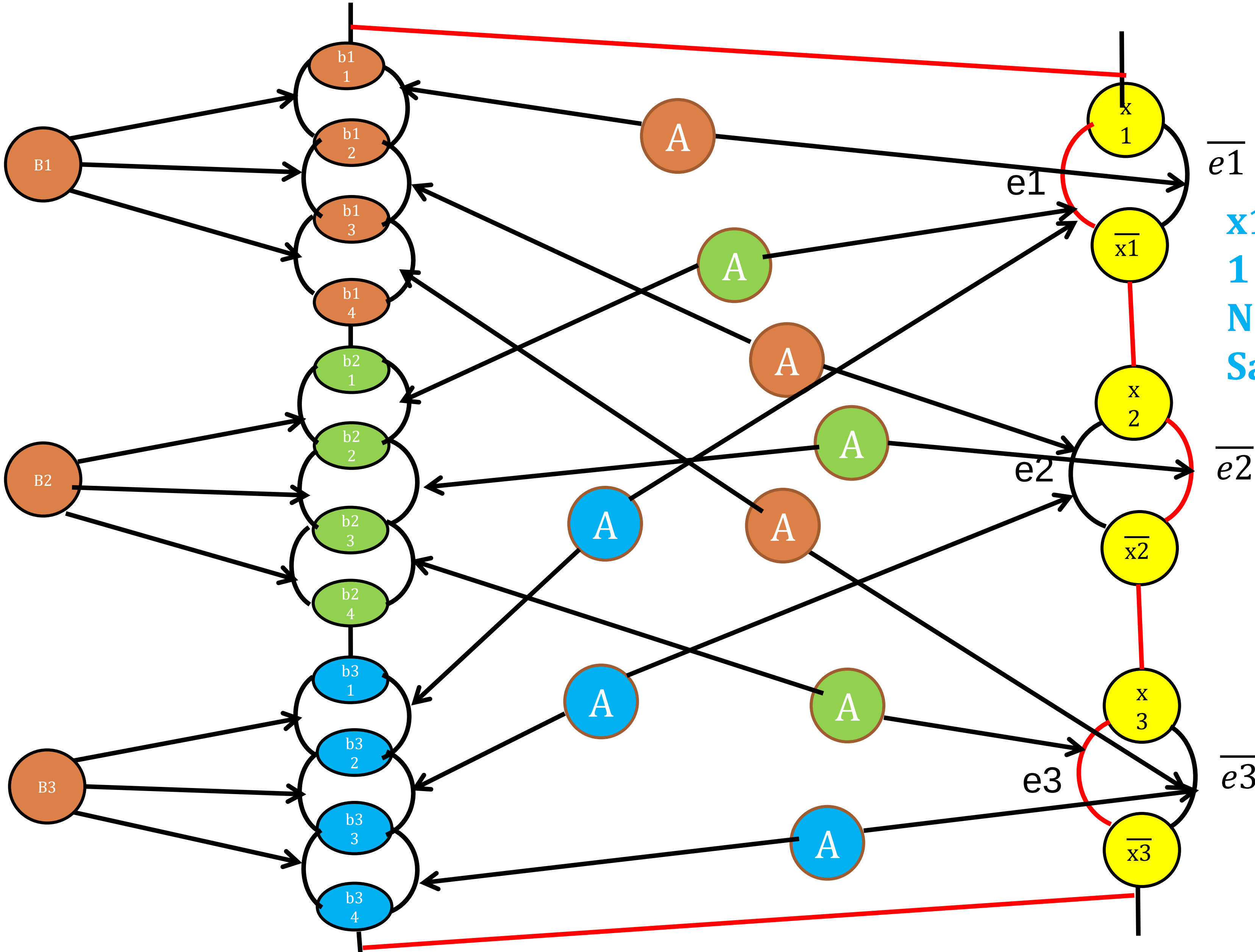
Satisfiable

Hence, Cycle

F = ($\overline{x1}$ V x2 V $\overline{x3}$) Λ (x1 V $\overline{x2}$ V x3) Λ (x1 V x2 V $\overline{x3}$)

C1 C2 C3

Contd...



x1 x2 x3

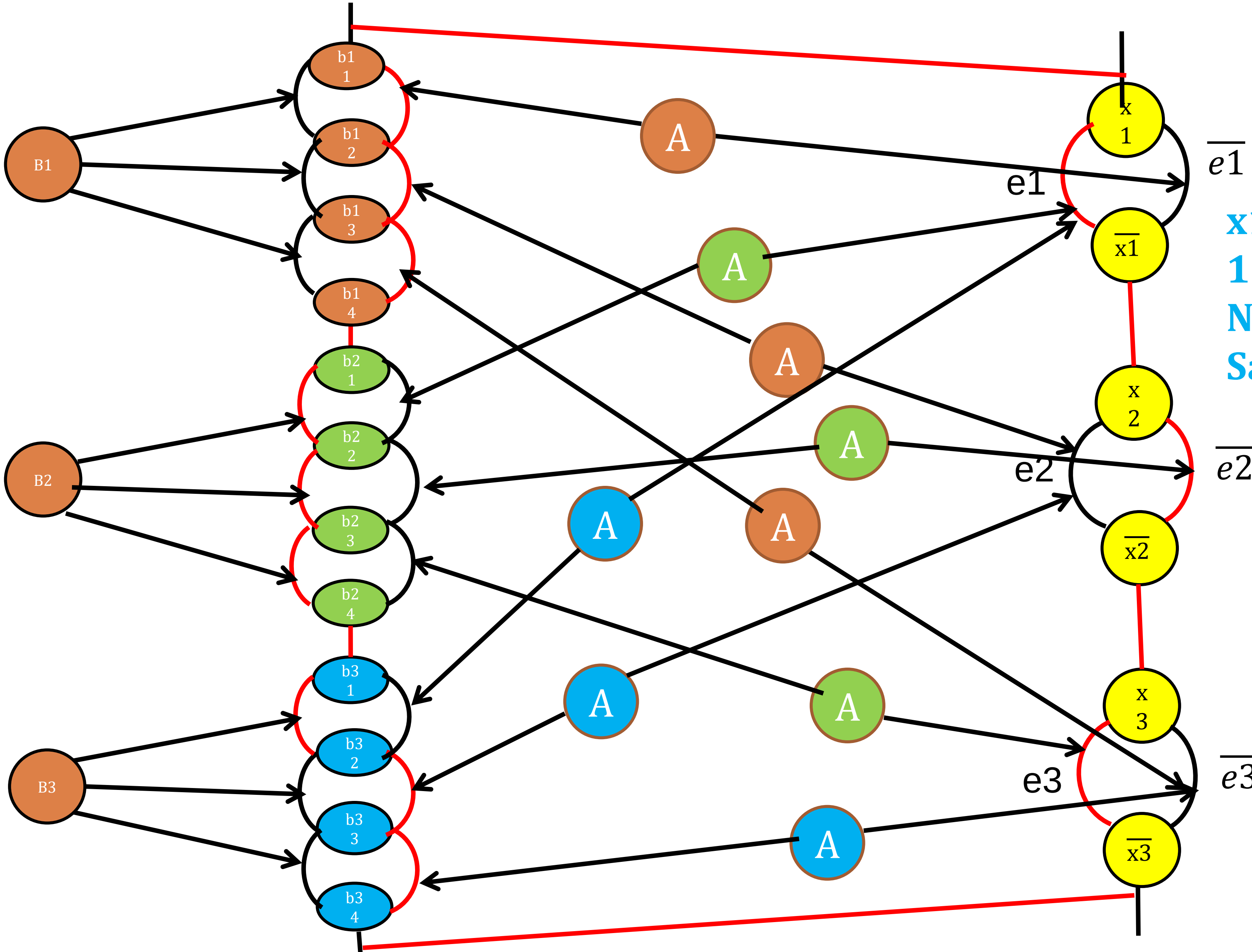
1 0 1

Not Satisfiable

$$F = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

C1 C2 C3

Contd...



x1 x2 x3

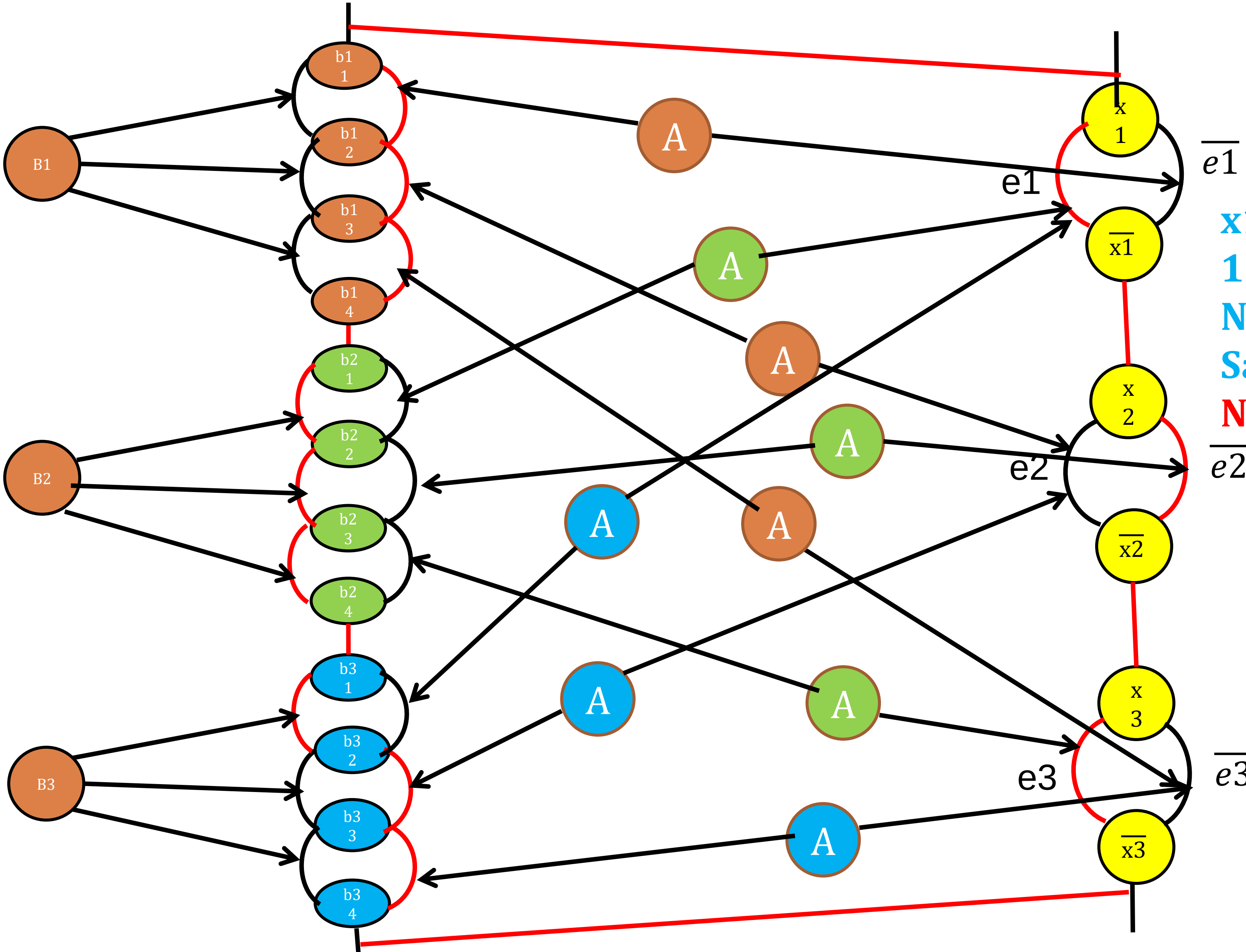
1 0 1

Not Satisfiable

$$F = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

C1 C2 C3

Contd...



x1 x2 x3

1 0 1

Not Satisfiable

No Cycle

$$F = (\overline{x1} \vee x2 \vee \overline{x3}) \wedge (x1 \vee \overline{x2} \vee x3) \wedge (x1 \vee x2 \vee \overline{x3})$$

C1 C2 C3

Travelling Salesperson Problem (TSP)

Consider a salesman who must visit n cities labeled v_1, v_2,…, v_n.

The salesman starts in city v_1, his home, and wants to find a tour—an order in which to visit all the other cities and return home. His goal is to find a tour that causes him to travel as little total distance as possible.

Decision Travelling Salesman Problem: Given a set of distances on n cities, and a bound D, is there a tour of length at most D?

Travelling Salesperson Problem (TSP)

TSP is in NP-Complete ?

i. TSP $\in$ **NP**

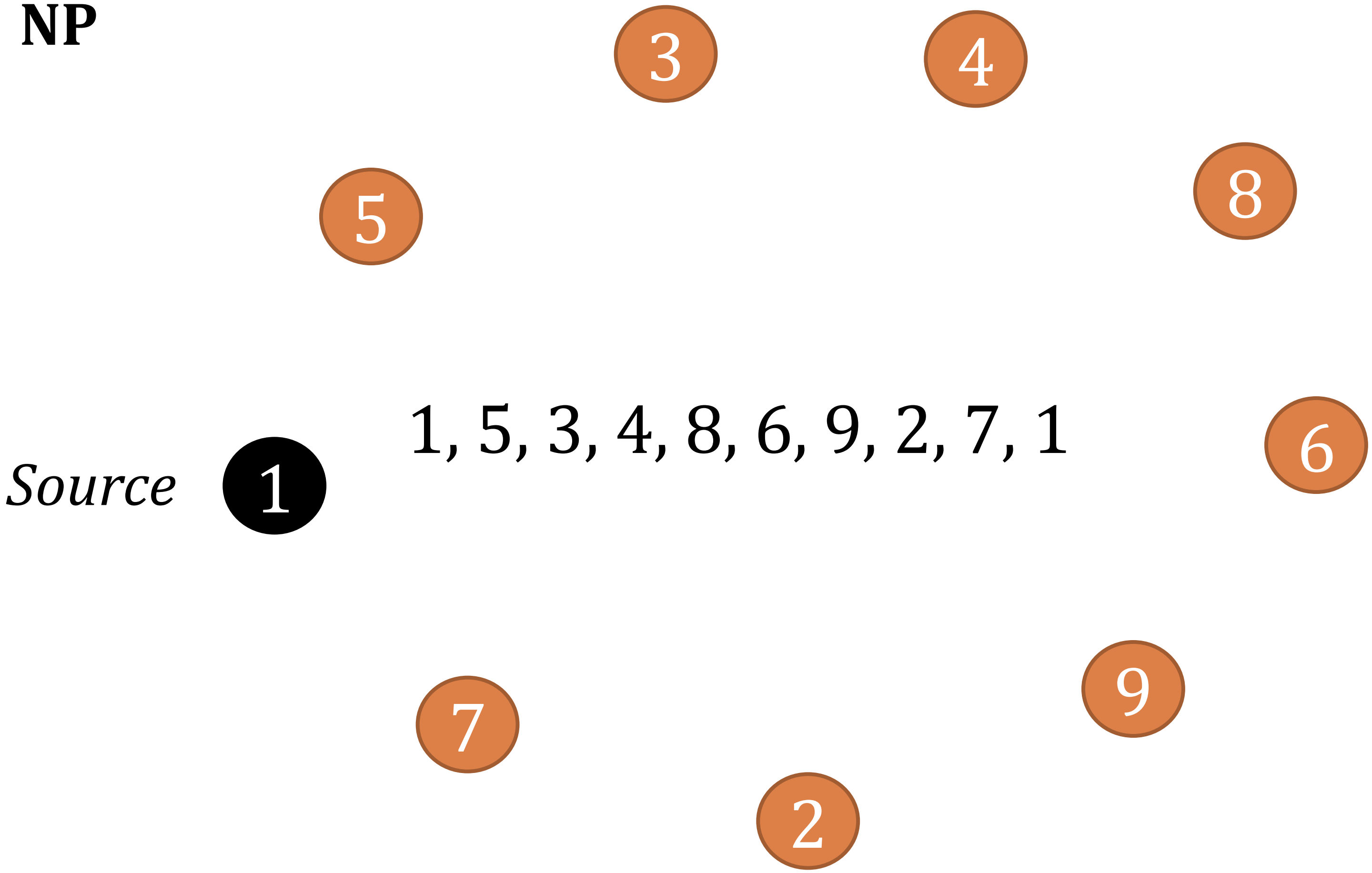
ii. Hamiltonian Cycle $\leq_p$ TSP

if the graph (G) has a Hamiltonian Cycle then the graph (G') must have a TSP

TSP: Does the graph have a TSP whose cost is k?

Verification of decision TSP

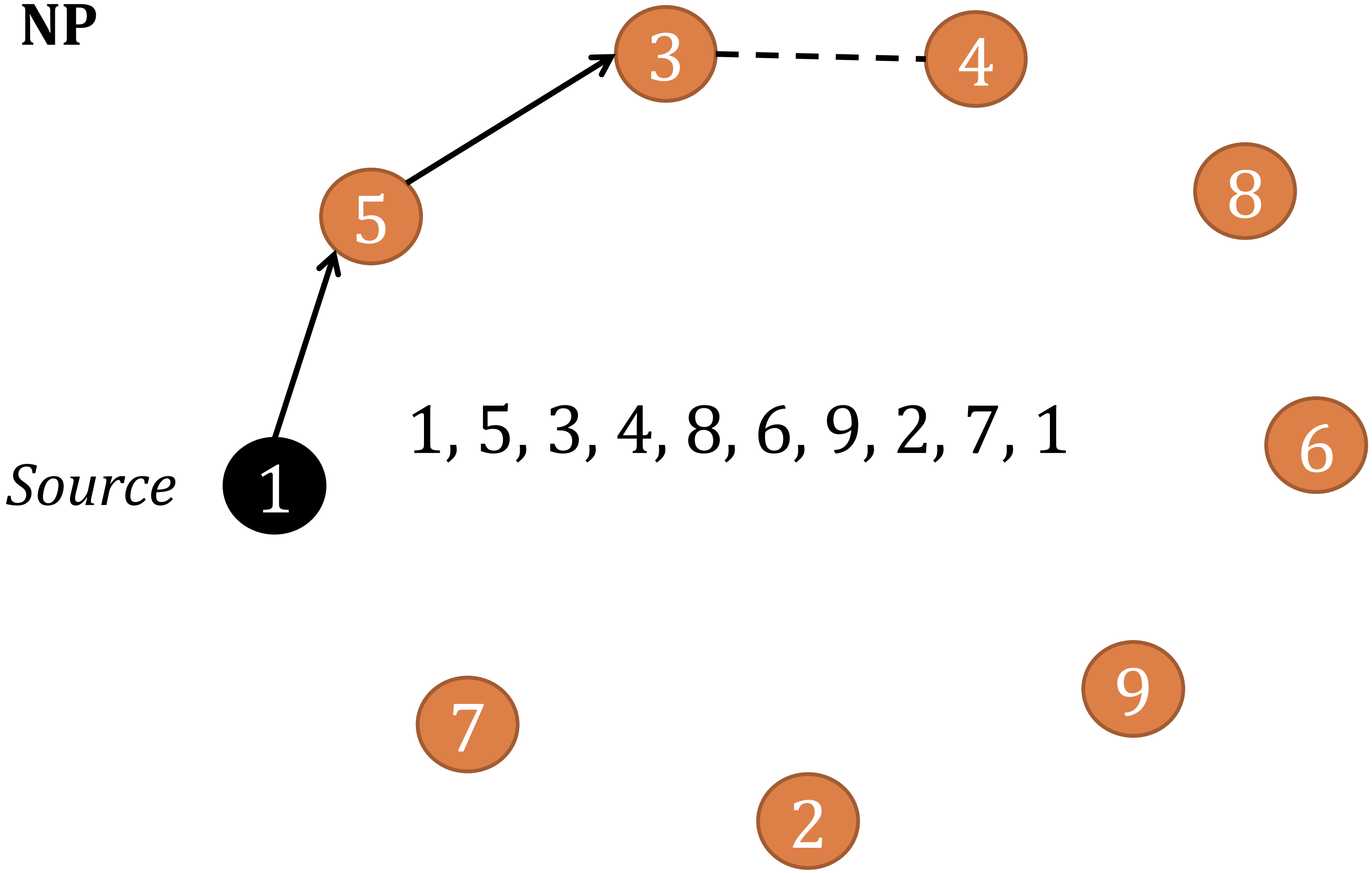
TSP $\in$ **NP**



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

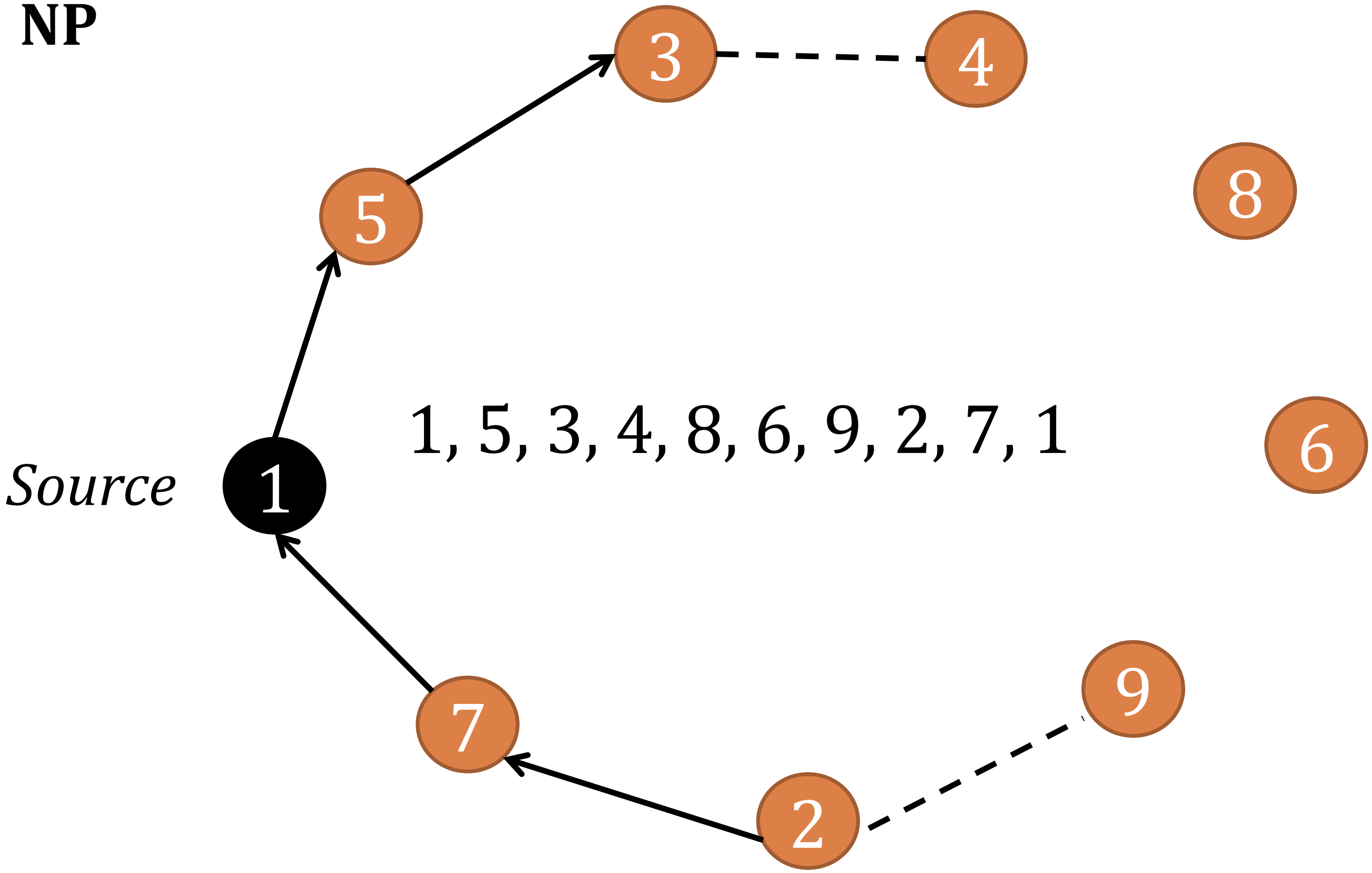
Verification of decision TSP

TSP $\in$ **NP**



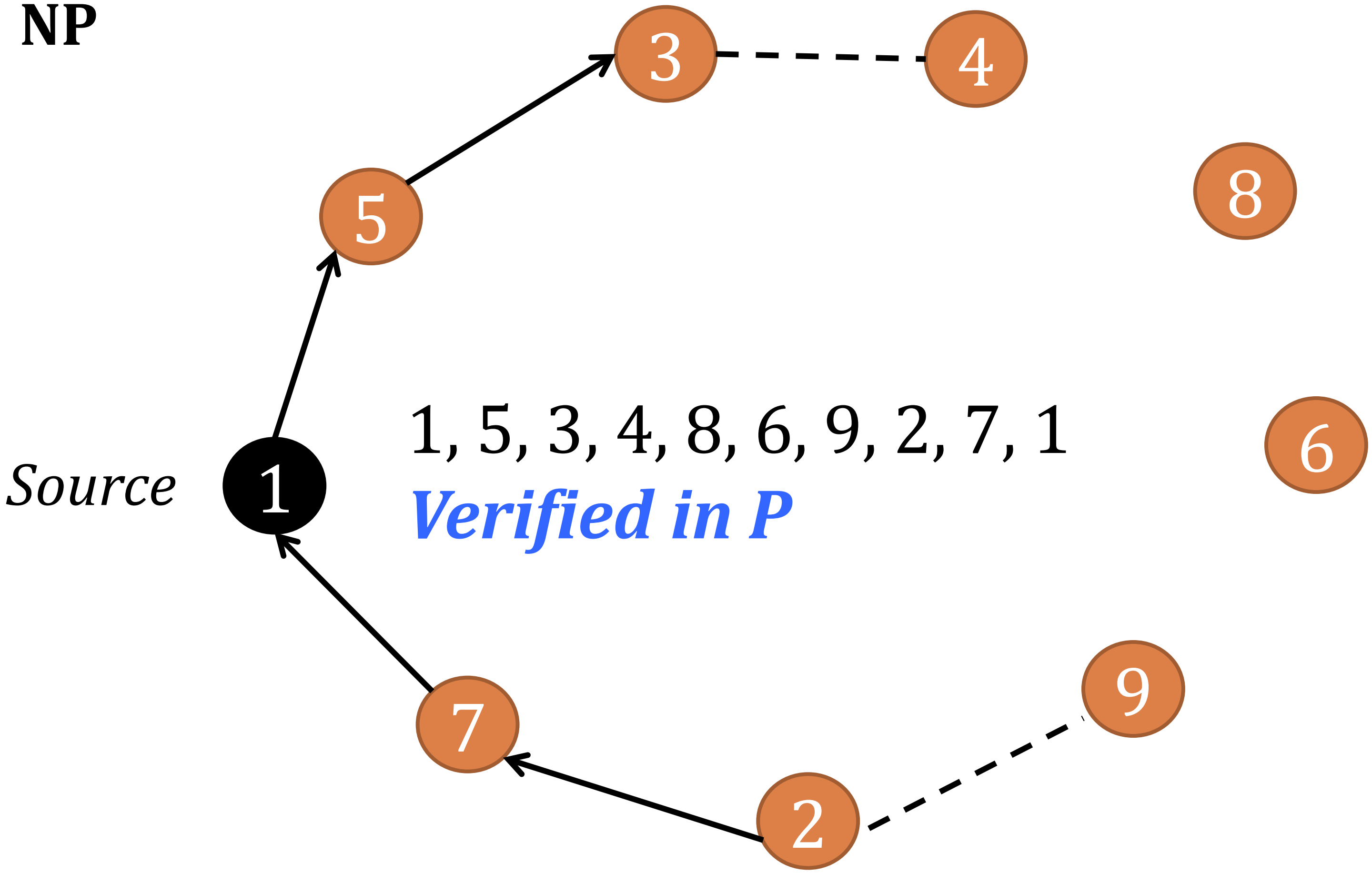
Verification of decision TSP

TSP $\in$ **NP**



Verification of decision TSP

TSP $\in$ **NP**



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

Verified in P

Reducing to Travelling Salesperson Problem (TSP)

Hamiltonian Cycle $\leq_p$ TSP

G G'

Reducing to Travelling Salesperson Problem (TSP)

Hamiltonian Cycle $\leq_p$ TSP

G G'

Cost matrix of G is reduced to cost matrix of G'

cost(i, j) = 1 if an edge is there between i to j else 0

Reducing to Travelling Salesperson Problem (TSP)

Hamiltonian Cycle $\leq_p$ TSP

G G'

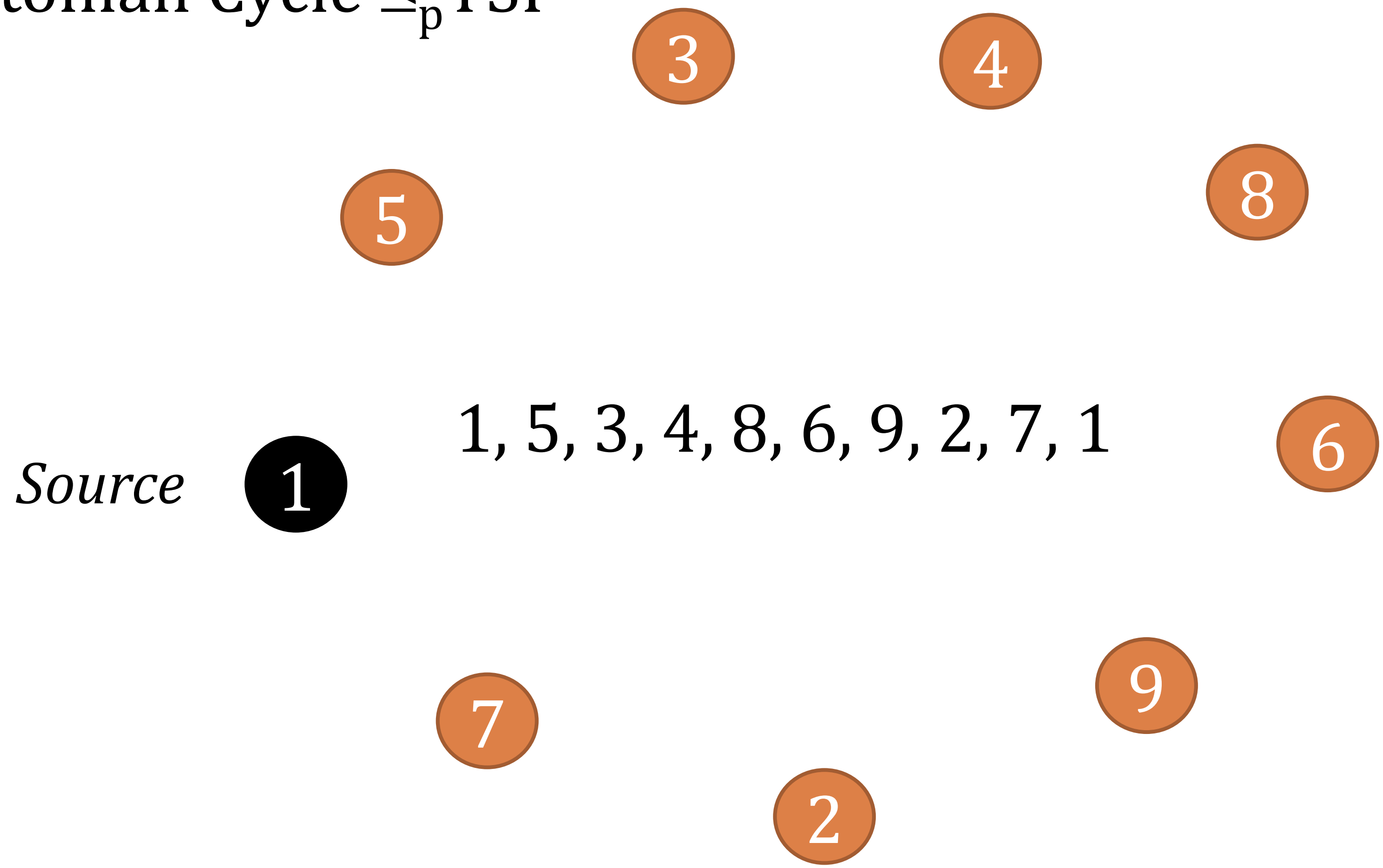
Cost matrix of G is reduced to cost matrix of G'

cost(i, j) = 1 if an edge is there between i to j else 0

The reduction can be done in P i. e. $O(n^2)$, considering n number of vertices

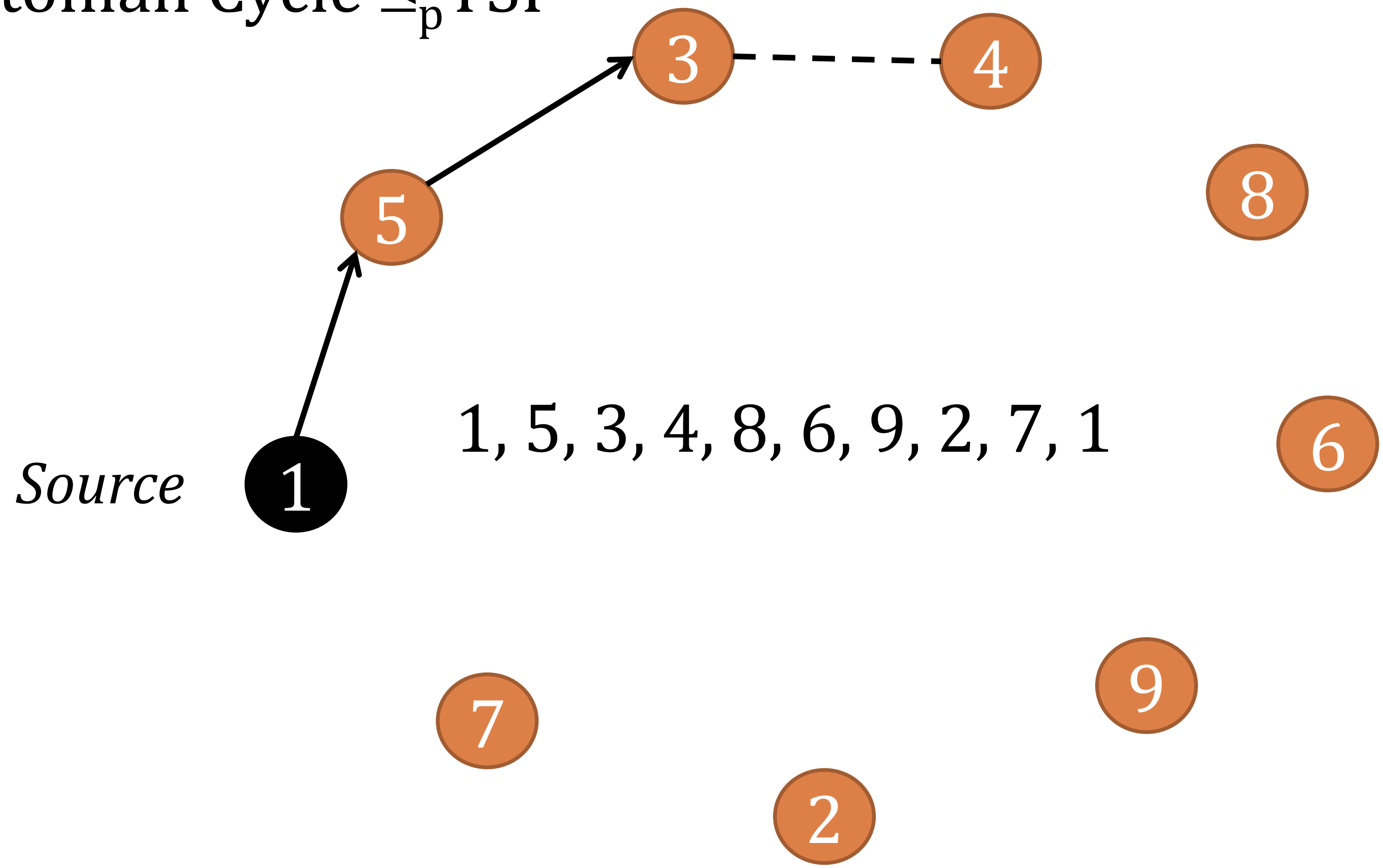
TSP of cost k?
Hamiltonian Cycle?

Hamiltonian Cycle $\leq_p$ TSP



TSP of cost k?
Hamiltonian Cycle?

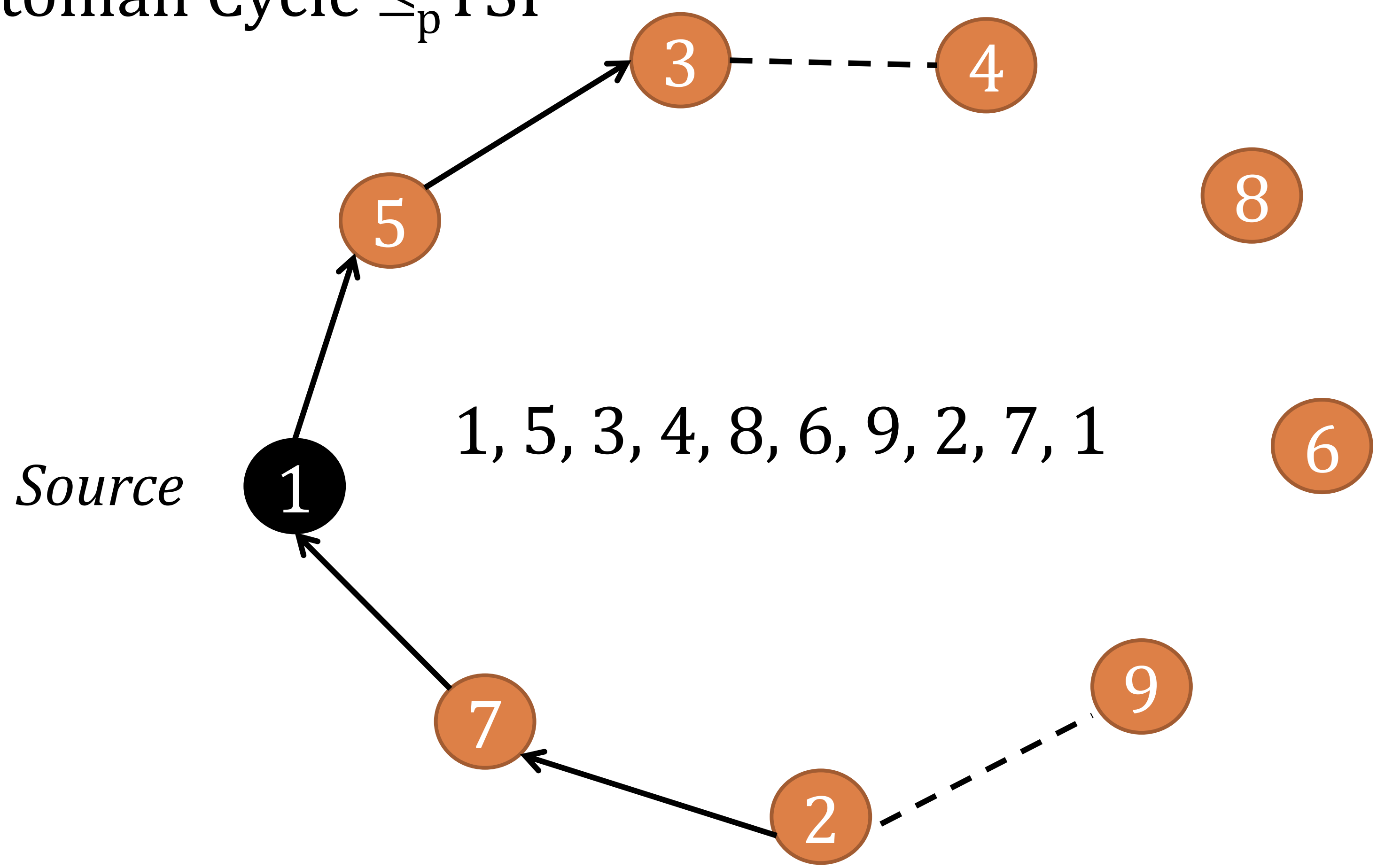
Hamiltonian Cycle $\leq_p$ TSP



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

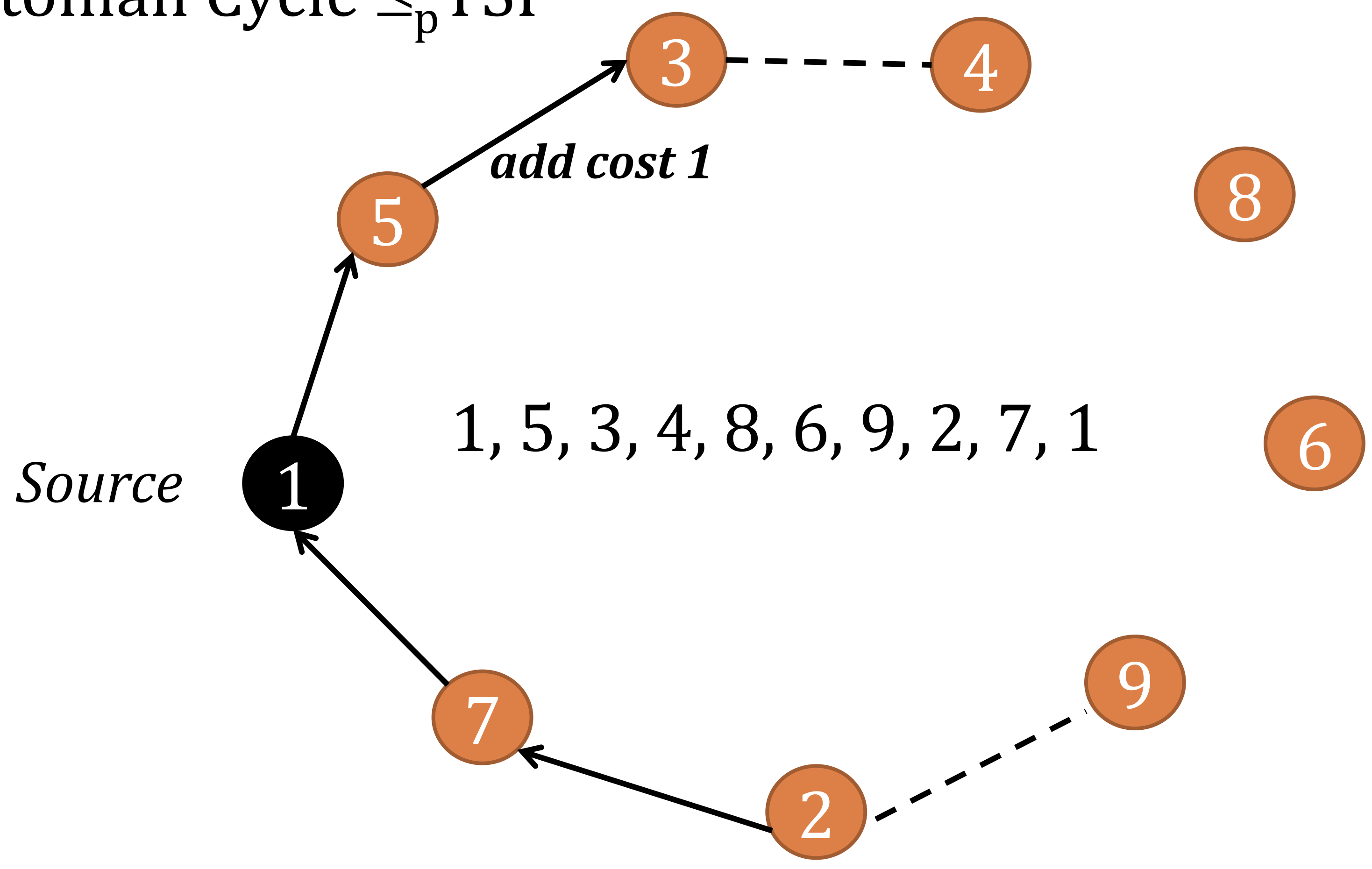
TSP of cost k?
Hamiltonian Cycle?

Hamiltonian Cycle $\leq_p$ TSP



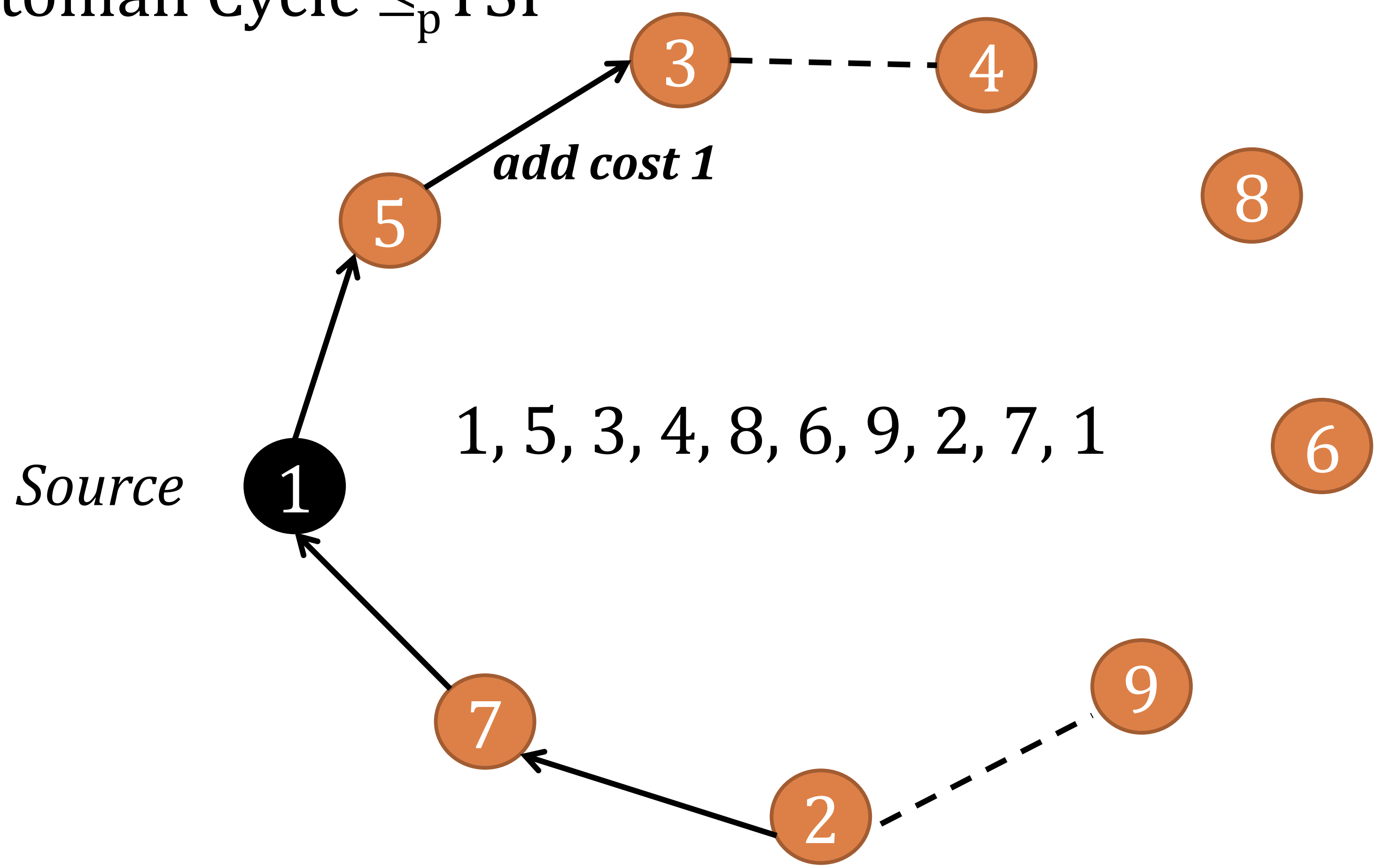
TSP of cost k?
Hamiltonian Cycle?

Hamiltonian Cycle $\leq_p$ TSP



TSP of cost k?
Hamiltonian Cycle?

Hamiltonian Cycle $\leq_p$ TSP

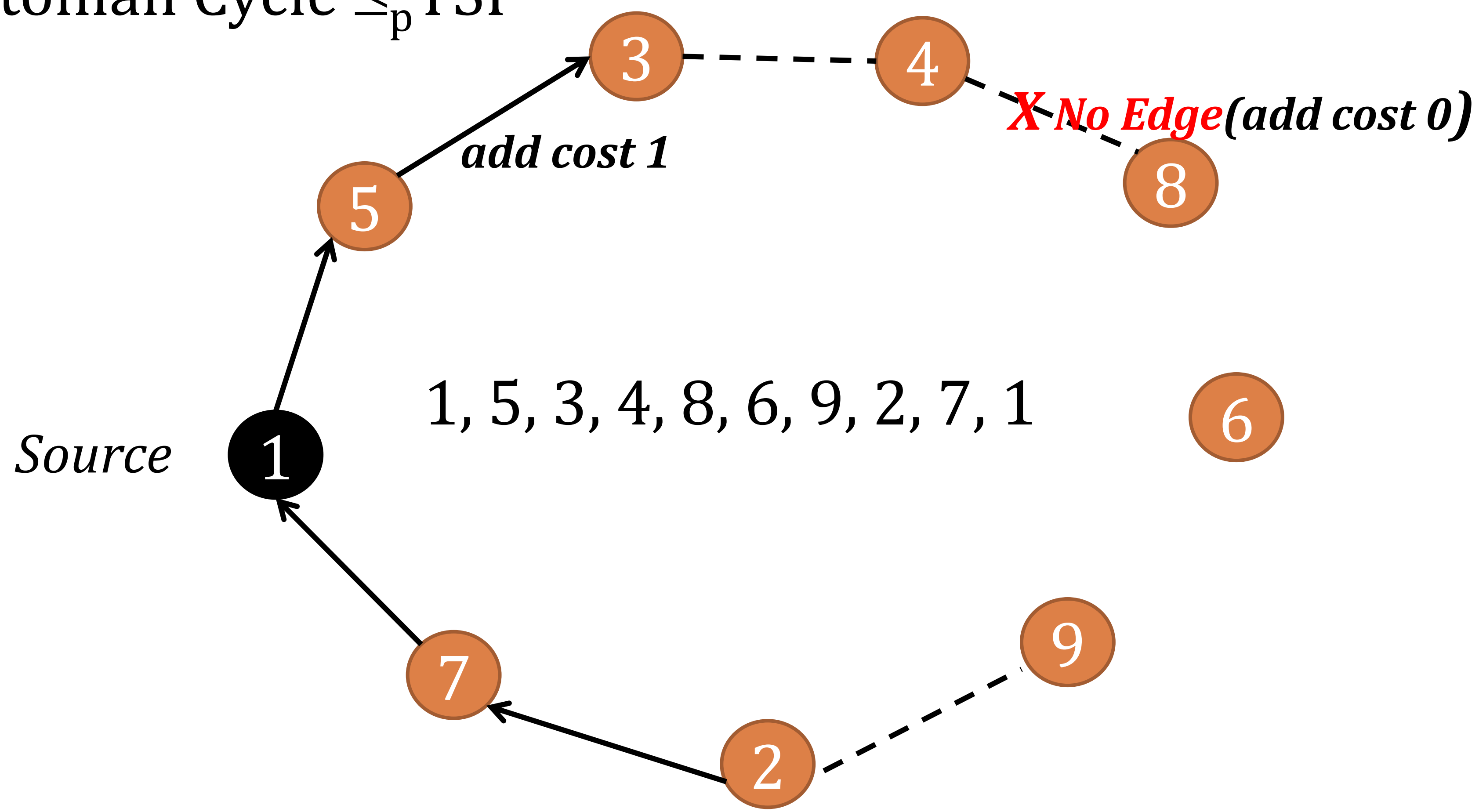


All edges are present, Hamiltonian cycle exists

TSP exists with cost k

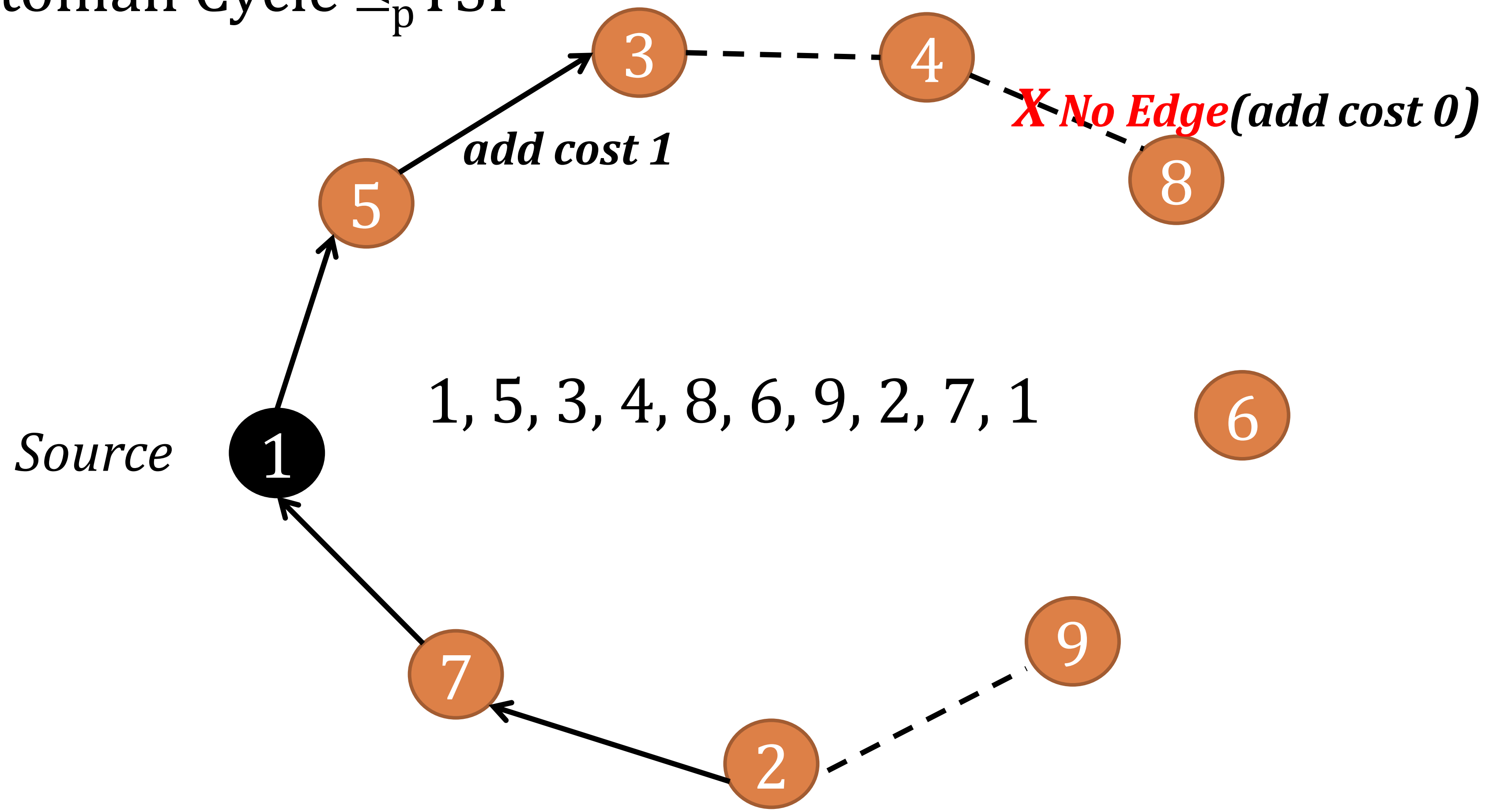
TSP of cost k?
Hamiltonian Cycle?

Hamiltonian Cycle $\leq_p$ TSP



TSP of cost k?
Hamiltonian Cycle?

Hamiltonian Cycle $\leq_p$ TSP



No Hamiltonian cycle exists

No TSP exist with cost k

THANK YOU!