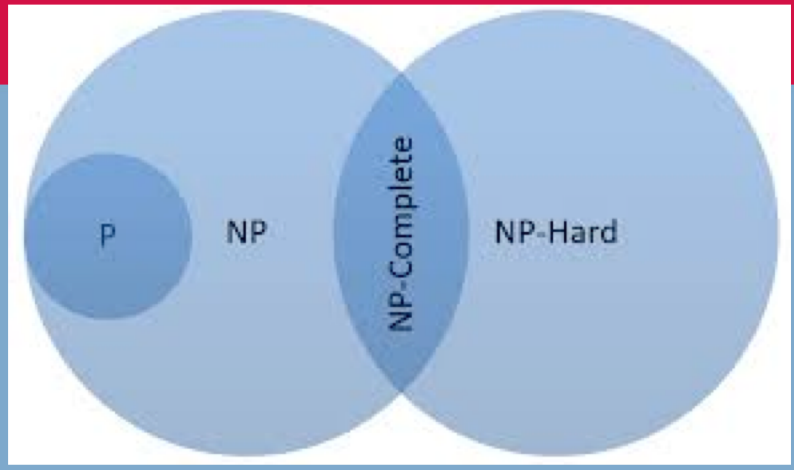
CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

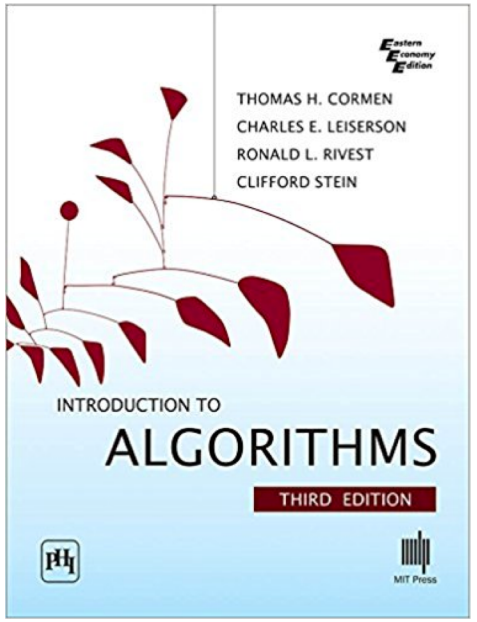
Metaheuristic Optimization

NP Complexity

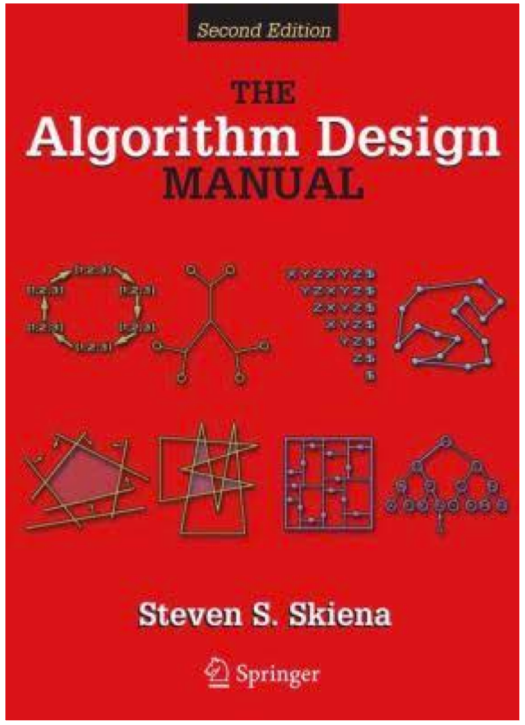


Dr. Diarmuid Grimes

Books

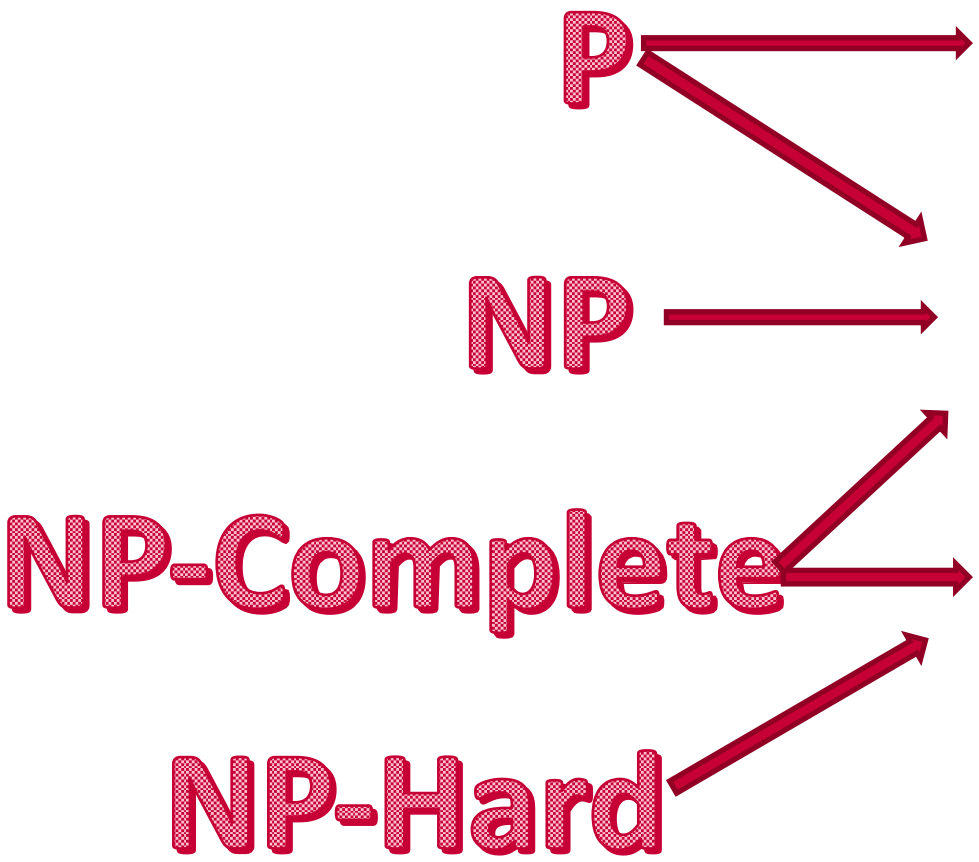


Introduction to Algorithms Complexity
Chapter 34 – NP-completeness



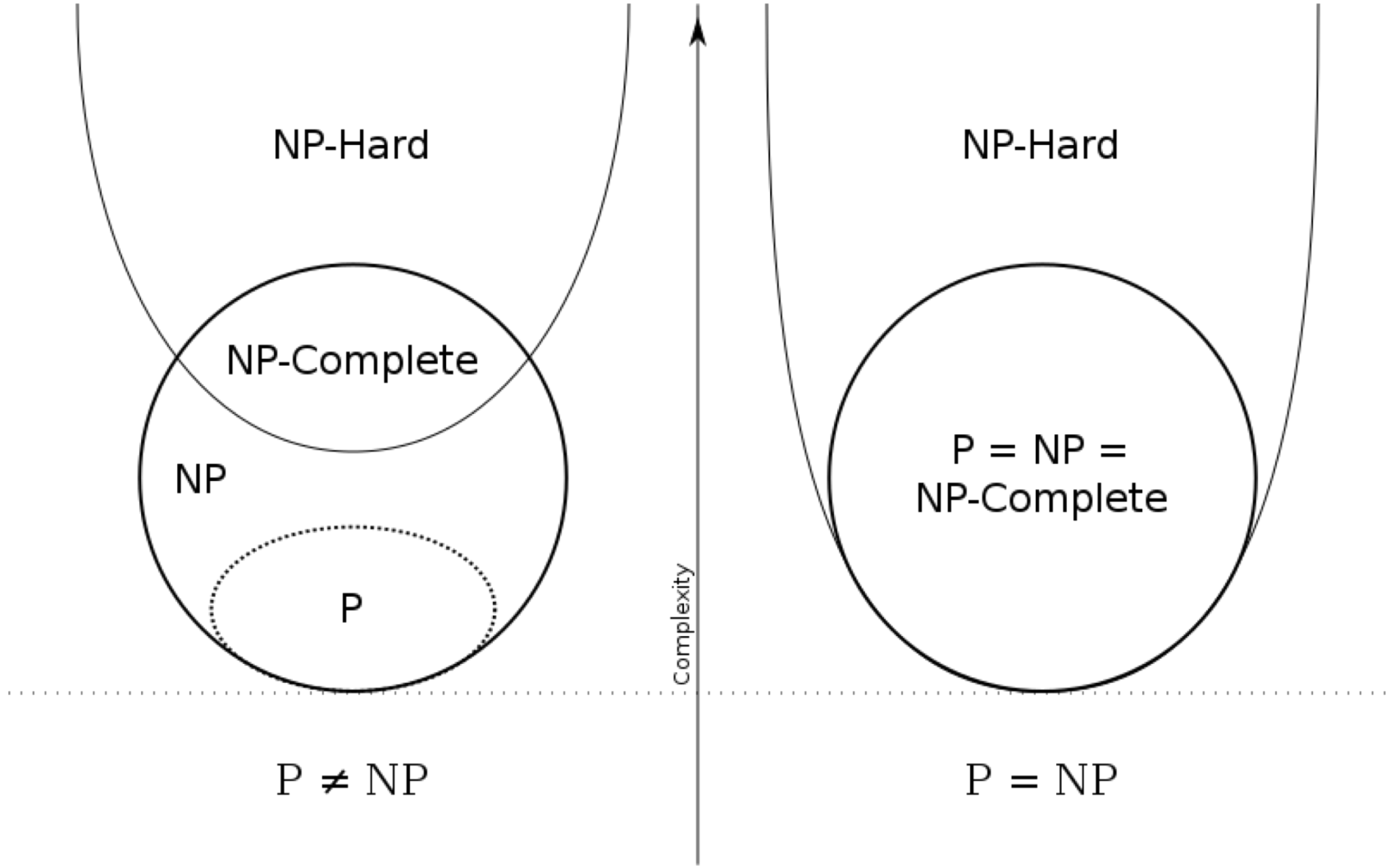
The Algorithm Design Manual
Steven S. Skiena
Chapter 9
Intractable Problems and Approximation Algorithms

Recap



1. Decision problem can be solved in polynomial time
2. Candidate solution for problem can be verified in polynomial time
3. All problems in NP can be *reduced* in polynomial time to this problem

Recap


NP-Hard
NP-Complete
NP
P
Complexity
NP-Hard
P = NP =
NP-Complete
P ≠ NP
P = NP

Implications

- Suppose that $L_1 \leq_p L_2$
- If there is a polynomial time algorithm for L_2, then there is a polynomial time algorithm for L_1.
- If there is no polynomial time algorithm for L_1, then there is no polynomial time algorithm for L_2.
- Note the asymmetry!

NP-Completeness

Definition of NP-Complete

L is NP-complete if and only if

(1) L is in NP and

(2) for all L' in NP, $L' \leq_p L$.

In other words, L is at least as hard as every language in NP.

Implication of NP-Completeness

Theorem: Suppose L is NP-complete.

(a) If there is a poly time algorithm for L, then P = NP.

(b) If there is no poly time algorithm for L, then there is no poly time algorithm for any NP-complete language.

Showing NP-Completeness

- How to show that a problem (language) L is NP-complete?
- Direct approach: Show

  (1) L is in NP

  (2) every other language in NP is polynomially reducible to L.
- Better approach: once we know some NP-complete problems, we can use reduction to show other problems are also NP-complete. How?

Showing NP-Completeness with a Reduction

To show L is NP-complete:

(1) Show L is in NP.

(2.a) Choose an appropriate known NP-complete language L'.

(2.b) Show $L' \leq_p L$.

Why does this work? By transitivity: Since every language L'' in NP is polynomially reducible to L', L'' is also polynomially reducible to L.

The First NP-Complete Problem: Satisfiability - SAT

First NP-Complete Problem

How do we get started? Need to show via brute force that some problem is NP-complete.

- Logic problem "satisfiability" (or SAT).
- Given a boolean expression (collection of boolean variables connected with ANDs and ORs), is it satisfiable, i.e., is there a way to assign truth values to the variables so that the expression evaluates to TRUE?

Definition of SAT

- SAT = all (and only) strings that encode satisfiable CNF formulas.

SAT is NP-Complete

- Cook's Theorem: SAT is NP-complete.
- Proof ideas 2 parts: one relatively easy, one relatively hard!
- SAT is in NP: Given a candidate solution (a truth assignment) for a CNF formula, verify in polynomial time (by plugging in the truth values and evaluating the expression) whether it satisfies the formula (makes it true).
- More complicated: show that any problem in NP can be reduced to an instance of the SAT problem, i.e. that SAT is NP-Hard

SAT is NP-Complete

- How to show that every language in NP is polynomially reducible to SAT?
- Key idea: the common thread among all the languages in NP is that each one is solved by some nondeterministic Turing machine (a formal model of computation) in polynomial time.
- Given a description of a poly time TM, construct in poly time, a CNF formula that simulates the computation of the TM.

Proving NP-Completeness By Reduction

So now we have an initial NP-Complete problem (SAT)

To show L is NP-complete:

(1) Show L is in NP.

(2.a) Choose an appropriate known NP-complete language L'.

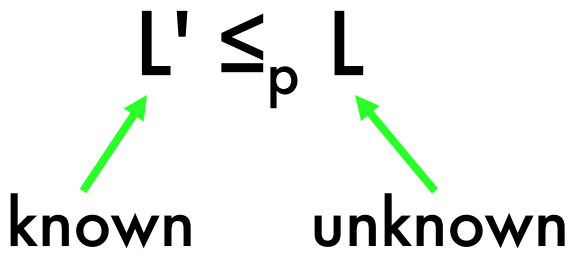
(2.b) Show $L' \leq_p L$: Describe an algorithm to compute a function f such that

- f is poly time
- f maps inputs for L' to inputs for L s.t. x is in L' if and only if f(x) is in L

Get the Direction Right!

- We want to show that L is at least as hard (time-consuming) as L'.
- So if we have an algorithm A for L, then we can solve L' with polynomial overhead
- Algorithm for L':

```
input: x
compute y = f(x)
run algorithm A for L on y
return whatever A returns
```



Definition of 3SAT

- 3SAT is a special case of SAT: each clause contains exactly 3 literals.
- Is 3SAT in NP?
  - Yes, because SAT is in NP.
- Is 3SAT NP-complete?
  - Not obvious. It has a more regular structure, which can perhaps be exploited to get an efficient algorithm
  - In fact, 2SAT does have a polynomial time algorithm

Showing 3SAT is NP-Complete by Reduction

(1) To show 3SAT is in NP, use same algorithm as for SAT to verify a candidate solution (truth assignment)

(2.a) Choose SAT as known NP-complete problem.

(2.b) Describe a reduction from SAT inputs to 3SAT inputs

- computable in poly time
- SAT input is satisfiable iff constructed 3SAT input is satisfiable

- Every instance of SAT can be represented as 3SAT such that
  - if the SAT instance is unsatisfiable then the 3SAT instance is unsatisfiable,
  - Else, every solution for the SAT instance must also be a solution for the 3SAT instance

Reduction from SAT to 3SAT

- We're given an arbitrary CNF formula $C = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ over set of variables U
  - each c_i is a clause (disjunction of literals)
- We will replace each clause c_i with a set of clauses C_i', and may use some extra variables U_i' just for this clause
- Each clause in C_i' will have exactly 3 literals
- Transformed input will be **conjunction** of all the clauses in all the C_i'
- New clauses are carefully chosen…

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

    $(z_1 \vee y_i^1 \vee y_i^2)$

    $(z_1 \vee \overline{y_i^1} \vee y_i^2)$

    $(z_1 \vee y_i^1 \vee \overline{y_i^2})$

    $(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

Any solution for the new problem must be a solution (k=3) for the original problem (k=1)

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

    $(z_1 \vee y_i^1 \vee y_i^2)$

    $(z_1 \vee \overline{y_i^1} \vee y_i^2)$

    $(z_1 \vee y_i^1 \vee \overline{y_i^2})$

    $(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

Table with all possible positions

Z_1	Y^1	Y^2	$-Y^1$	$-Y^2$	C^1	C^2	C^3	C^4	F
1	1	1							
1	1	0							
1	0	1							
1	0	0							
0	1	1							
0	1	0							
0	0	1							
0	0	0							

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

    $(z_1 \vee y_i^1 \vee y_i^2)$

    $(z_1 \vee \overline{y_i^1} \vee y_i^2)$

    $(z_1 \vee y_i^1 \vee \overline{y_i^2})$

    $(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

Table with all possible positions

Z_1	Y^1	Y^2	$-Y^1$	$-Y^2$	C^1	C^2	C^3	C^4	F
1	1	1	0						
1	1	0	0						
1	0	1	1						
1	0	0	1						
0	1	1	0						
0	1	0	0						
0	0	1	1						
0	0	0	1						

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

    $(z_1 \vee y_i^1 \vee y_i^2)$

    $(z_1 \vee \overline{y_i^1} \vee y_i^2)$

    $(z_1 \vee y_i^1 \vee \overline{y_i^2})$

    $(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

Table with all possible positions

Z_1	Y^1	Y^2	$-Y^1$	$-Y^2$	C^1	C^2	C^3	C^4	F
1	1	1	0	0					
1	1	0	0	1					
1	0	1	1	0					
1	0	0	1	1					
0	1	1	0	0					
0	1	0	0	1					
0	0	1	1	0					
0	0	0	1	1					

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

$(z_1 \vee y_i^1 \vee y_i^2)$

$(z_1 \vee \overline{y_i^1} \vee y_i^2)$

$(z_1 \vee y_i^1 \vee \overline{y_i^2})$

$(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

Table with all possible positions

Z	Y^1	Y^2	$-Y^1$	$-Y^2$	C^1	C^2	C^3	C^4	F
1	1	1	0	0	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1
0	1	1	0	0					
0	1	0	0	1					
0	0	1	1	0					
0	0	0	1	1					

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

    $(z_1 \vee y_i^1 \vee y_i^2)$

    $(z_1 \vee \overline{y_i^1} \vee y_i^2)$

    $(z_1 \vee y_i^1 \vee \overline{y_i^2})$

    $(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

Table with all possible positions

Z_1	Y^1	Y^2	$-Y^1$	$-Y^2$	C^1	C^2	C^3	C^4	F
1	1	1	0	0	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1
0	1	1	0	0	1				
0	1	0	0	1	1				
0	0	1	1	0	1				
0	0	0	1	1	0				

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

    $(z_1 \vee y_i^1 \vee y_i^2)$

    $(z_1 \vee \overline{y_i^1} \vee y_i^2)$

    $(z_1 \vee y_i^1 \vee \overline{y_i^2})$

    $(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

Table with all possible positions

Z_1	Y^1	Y^2	$-Y^1$	$-Y^2$	C^1	C^2	C^3	C^4	F
1	1	1	0	0	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1
0	1	1	0	0	1	1			
0	1	0	0	1	1	0			
0	0	1	1	0	1	1			
0	0	0	1	1	0	1			

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

    $(z_1 \vee y_i^1 \vee y_i^2)$

    $(z_1 \vee \overline{y_i^1} \vee y_i^2)$

    $(z_1 \vee y_i^1 \vee \overline{y_i^2})$

    $(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

Table with all possible positions

Z_1	Y^1	Y^2	$-Y^1$	$-Y^2$	C^1	C^2	C^3	C^4	F
1	1	1	0	0	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1		
0	1	0	0	1	1	0	1		
0	0	1	1	0	1	1	0		
0	0	0	1	1	0	1	1		

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

    $(z_1 \vee y_i^1 \vee y_i^2)$

    $(z_1 \vee \overline{y_i^1} \vee y_i^2)$

    $(z_1 \vee y_i^1 \vee \overline{y_i^2})$

    $(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

Table with all possible positions

Z_1	Y^1	Y^2	$-Y^1$	$-Y^2$	C^1	C^2	C^3	C^4	F
1	1	1	0	0	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	0	
0	1	0	0	1	1	0	1	1	
0	0	1	1	0	1	1	0	1	
0	0	0	1	1	0	1	1	1	

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

    $(z_1 \vee y_i^1 \vee y_i^2)$

    $(z_1 \vee \overline{y_i^1} \vee y_i^2)$

    $(z_1 \vee y_i^1 \vee \overline{y_i^2})$

    $(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

Table with all possible positions

Z_1	Y^1	Y^2	$-Y^1$	$-Y^2$	C^1	C^2	C^3	C^4	F
1	1	1	0	0	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	0	0
0	1	0	0	1	1	0	1	1	0
0	0	1	1	0	1	1	0	1	0
0	0	0	1	1	0	1	1	1	0

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 2: k = 2.
  - Use extra variable y_i^1.
  - Replace c_i with 2 clauses:

    $(z_1 \vee z_2 \vee y_i^1)$

    $(z_1 \vee z_2 \vee \overline{y_i^1})$

1 Extra Variable
And
2 Clauses

Proof

Truth Table with all possible positions

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 3: k = 3.
  - No extra variables are needed.
  - Keep c_i:

    $(z_1 \vee z_2 \vee z_3)$

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: $k > 3$.
  - Use extra variables $y_i^1, \ldots, y_i^{k-3}$.
  - Replace c_i with k-2 clauses:

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee y_i^3)$

$\ldots$

$\ldots$

$(\overline{y_i^{k-5}} \vee z_{k-3} \vee y_i^{k-4})$

$(\overline{y_i^{k-4}} \vee z_{k-2} \vee y_i^{k-3})$

$(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$

k-3 Extra Variable
And
K-2 Clauses

Reduction from SAT to 3SAT

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: e.g. k = 4.

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee z_4)$

For each clause (k=4)

1 Extra Variable
And
2 Clauses

Correctness of Reduction

- Show that CNF formula C is satisfiable iff the 3-CNF formula C' constructed is satisfiable.
- =>: Suppose C is satisfiable. Come up with a satisfying truth assignment for C'.
- For variables in U, use same truth assignments as for C.
- How to assign T/F to the new variables?

Correctness of Reduction

- <=: Suppose the newly constructed 3SAT formula C' is satisfiable. We must show that the original SAT formula C is also satisfiable.
- Use the same satisfying truth assignment for C as for C' (ignoring new variables).
- Show each original clause has at least one true literal in it.

Original Clause Has a True Literal

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 1: k = 1.
  - Use extra variables y_i^1 and y_i^2.
  - Replace c_i with 4 clauses:

    $(z_1 \vee y_i^1 \vee y_i^2)$

    $(z_1 \vee \overline{y_i^1} \vee y_i^2)$

    $(z_1 \vee y_i^1 \vee \overline{y_i^2})$

    $(z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$

For every assignment of y_i^1 and y_i^2, in order for all 4 clauses to have a true literal, z_1 must be true.

Truth Assignment for New Variables

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 2: k = 2.
  - Use extra variable y_i^1.
  - Replace c_i with 2 clauses:

    $(z_1 \vee z_2 \vee y_i^1)$

    $(z_1 \vee z_2 \vee \overline{y_i^1})$

Since either z_1 or z_2 is true, it does not matter how we assign y_i^1

For either assignment of y_i^1, in order for both clauses to have a true literal, z_1 or z_2 must be true.

Truth Assignment for New Variables

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 3: k = 3.
  - No extra variables are needed.
  - Keep c_i:

    $(z_1 \vee z_2 \vee z_3)$

No new variables.

Truth Assignment for New Variables

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: $k > 3$.
  - Use extra variables $y_i^1, \ldots, y_i^{k-3}$.
  - Replace c_i with k-2 clauses:

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee y_i^3)$

. . .

. . .

$(\overline{y_i^{k-5}} \vee z_{k-3} \vee y_i^{k-4})$

$(\overline{y_i^{k-4}} \vee z_{k-2} \vee y_i^{k-3})$

$(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$

If first true literal is z_1 or z_2, set all y_i's to false: then all later clauses have a true literal

Truth Assignment for New Variables

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: $k > 3$.
  - Use extra variables $y_i^1, \ldots, y_i^{k-3}$.
  - Replace c_i with k-2 clauses:

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee y_i^3)$

. . .

. . .

$(\overline{y_i^{k-5}} \vee z_{k-3} \vee y_i^{k-4})$

$(\overline{y_i^{k-4}} \vee z_{k-2} \vee y_i^{k-3})$

$(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$

If first true literal is z_{k-1} or z_k, set all y_i's to true: then all earlier clauses have a true literal

Truth Assignment for New Variables

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: $k > 3$.
  - Use extra variables $y_i^1, \ldots, y_i^{k-3}$.
  - Replace c_i with k-2 clauses:

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee y_i^3)$

. . .

. . .

$(\overline{y_i^{k-5}} \vee z_{k-3} \vee y_i^{k-4})$

$(\overline{y_i^{k-4}} \vee z_{k-2} \vee y_i^{k-3})$

$(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$

If first true literal is in between, set all earlier y_i's to true and all later y_i's to false

Original Clause Has a True Literal

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: e.g. k = 4.

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee z_4)$

Suppose in contradiction Z_1, Z_2, Z_3, Z_4 are false.
Then Y^1 must be true, but the second clause will be false ($-Y^1$)

If at least one (Z1, Z2, Z3, Z4) is True.
Then we could say Y^1 or $-Y^1$ will evaluate to true

For instance, Z1 = True (the first clause will be true, regardless the value of Y^1)
And we can say that (–Y1) = True and the second clause will be true

Original Clause Has a True Literal

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: e.g. k =5

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee z_5)$

For each clause (k=5)

2 Extra Variables
And
3 Clauses

Original Clause Has a True Literal

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: e.g. k =5

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee z_5)$

Same argument as before (k=4)

Suppose in contradiction Z_1, Z_2, and Z_3, Z_4 are false.
Then Y^1 must be true, but the second clause will be false ($-Y^1$)

If at least one (Z1, Z2, Z3, Z4, Z5) is True.
Then we could say Y^1 or $-Y^1$ will evaluate to true

Original Clause Has a True Literal

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: $k > 3$.
  - Use extra variables $y_i^1, \ldots, y_i^{k-3}$.
  - Replace c_i with k-2 clauses:

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee y_i^3)$

$\ldots$

$\ldots$

$(\overline{y_i^{k-5}} \vee z_{k-3} \vee y_i^{k-4})$

$(\overline{y_i^{k-4}} \vee z_{k-2} \vee y_i^{k-3})$

$(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$

Suppose in contradiction all z_i's are false.

k-3 Extra Variable
And
K-2 Clauses

Original Clause Has a True Literal

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: $k > 3$.
  - Use extra variables $y_i^1, \ldots, y_i^{k-3}$.
  - Replace c_i with k-2 clauses:

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee y_i^3)$

. . .

. . .

$(\overline{y_i^{k-5}} \vee z_{k-3} \vee y_i^{k-4})$

$(\overline{y_i^{k-4}} \vee z_{k-2} \vee y_i^{k-3})$

$(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$

Suppose in contradiction all z_i's are false. Then y_i^1 must be true,

Original Clause Has a True Literal

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: $k > 3$.
  - Use extra variables $y_i^1, \ldots, y_i^{k-3}$.
  - Replace c_i with k-2 clauses:

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee y_i^3)$

$\ldots$

$\ldots$

$(\overline{y_i^{k-5}} \vee z_{k-3} \vee y_i^{k-4})$

$(\overline{y_i^{k-4}} \vee z_{k-2} \vee y_i^{k-3})$

$(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$

Suppose in contradiction all z_i's are false. Then y_i^1 must be true, so y_i^2 must be true…

Original Clause Has a True Literal

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: $k > 3$.
  - Use extra variables $y_i^1, \ldots, y_i^{k-3}$.
  - Replace c_i with k-2 clauses:

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee y_i^3)$

$\ldots$

$\ldots$

$(\overline{y_i^{k-5}} \vee z_{k-3} \vee y_i^{k-4})$

$(\overline{y_i^{k-4}} \vee z_{k-2} \vee y_i^{k-3})$

$(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$

Suppose in contradiction all z_i's are false. Then y_i^1 must be true, so y_i^2 must be true…

Original Clause Has a True Literal

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: $k > 3$.
  - Use extra variables $y_i^1, \ldots, y_i^{k-3}$.
  - Replace c_i with k-2 clauses:

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee y_i^3)$

$\ldots$

$\ldots$

$(\overline{y_i^{k-5}} \vee z_{k-3} \vee y_i^{k-4})$

$(\overline{y_i^{k-4}} \vee z_{k-2} \vee y_i^{k-3})$

$(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$

Suppose in contradiction all z_i's are false. Then y_i^1 must be true, so y_i^2 must be true…

Original Clause Has a True Literal

Let $c_i = z_1 \vee z_2 \vee \ldots \vee z_k$

- Case 4: $k > 3$.
  - Use extra variables $y_i^1, \ldots, y_i^{k-3}$.
  - Replace c_i with k-2 clauses:

$(z_1 \vee z_2 \vee y_i^1)$

$(\overline{y_i^1} \vee z_3 \vee y_i^2)$

$(\overline{y_i^2} \vee z_4 \vee y_i^3)$

. . .

. . .

$(\overline{y_i^{k-5}} \vee z_{k-3} \vee y_i^{k-4})$

$(\overline{y_i^{k-4}} \vee z_{k-2} \vee y_i^{k-3})$

$(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$

Suppose in contradiction all z_i's are false. Then y_i^1 must be true, … so last clause is false.

Why is Reduction Poly Time?

- The running time of the reduction (the algorithm to compute the 3SAT formula C', given the SAT formula C) is proportional to the size of C'
- rules for constructing C' are simple to calculate

Size of New Formula

- original clause with 1 literal becomes 4 clauses with 3 literals each
- original clause with 2 literals becomes 2 clauses with 3 literals each
- original clause with 3 literals becomes 1 clause with 3 literals
- original clause with k > 3 literals becomes k-2 clauses with 3 literals each
- So new formula is only a constant factor larger than the original formula

Recap: Proving NP-Completeness By Reduction

So now we have an initial NP-Complete problem (SAT)

To show L is NP-complete:

(1) Show L is in NP.

(2.a) Choose an appropriate known NP-complete language L'.

(2.b) Show $L' \leq_p L$: Describe an algorithm to compute a function f such that

- f is poly time
- f maps inputs for L' to inputs for L s.t. x is in L' if and only if f(x) is in L

SAT reduces to 3SAT

Show we can convert any SAT to 3SAT as follows:

- original clause with 1 literal becomes 4 clauses with 3 literals each
- original clause with 2 literals becomes 2 clauses with 3 literals each
- original clause with 3 literals becomes 1 clause with 3 literals
- original clause with k > 3 literals becomes k-2 clauses with 3 literals each

Note the above conversion isn't out of choice but necessity, i.e. we need 4 clauses and 2 additional literals for converting original clause with 1 literal, etc such that converted maps solutions to solutions and unsat to unsat

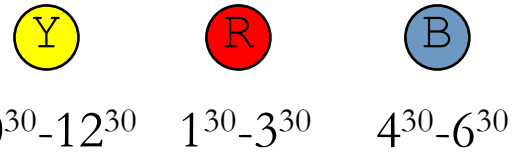
SAT reduces to 3SAT in polynomial time:

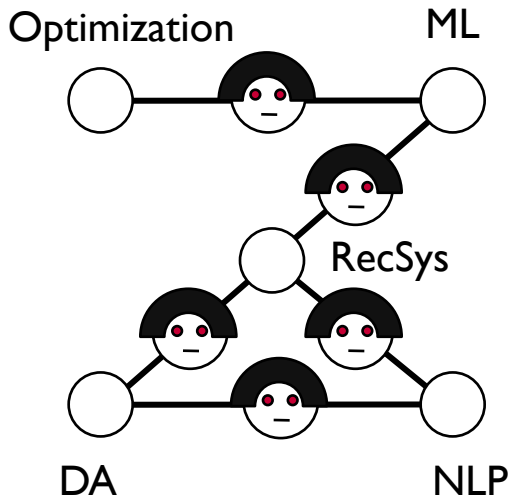
- So new formula is only a constant factor larger than the original formula

Example: Graph Coloring problem

Coloring

Suppose you need to schedule exams on same day in ***3 time slots***



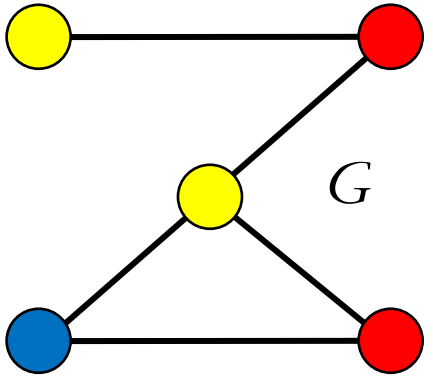


Some people are enrolled in ***multiple classes***, so there may be **conflicts**

Task: Schedule the exams so there are no conflicts

Coloring

This is called a valid 3-coloring

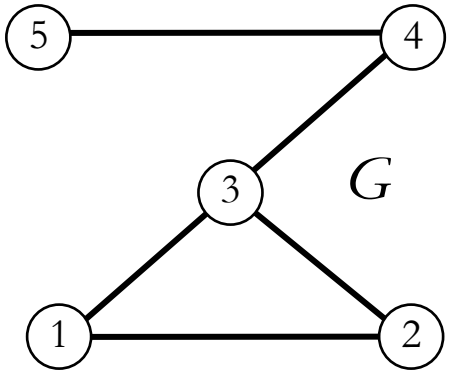


3COL = {⟨G⟩: G has a valid 3-coloring}

3COL is NP-complete

Coloring

- Step 1: 3COL is in NP



What is a **solution** for 3COL?

A solution *s* is a coloring of vertices
{1B,2R,3Y,4R,5B}

V := On input ⟨*G*, *s*⟩

For every edge {*u*, *v*} of *G*:

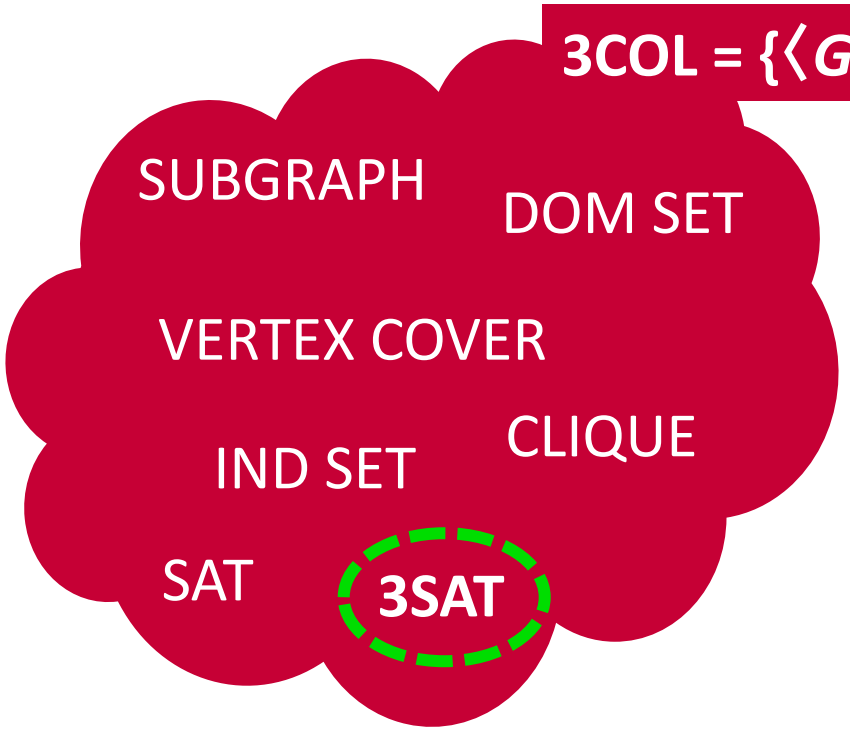
If *u* and *v* are assigned same color in *s*, *reject*.

Otherwise, *accept*.

Running time $= O(m)$

Coloring

- Step 2: Some NP-complete L ***reduces to*** 3COL



3COL does not look like any of these problems

In such a case it is a good idea to start with 3SAT

Coloring

- Step 3: Reduce 3SAT to 3COL

3SAT = {⟨f⟩: f is a 3CNF that has a satisfying assignment}

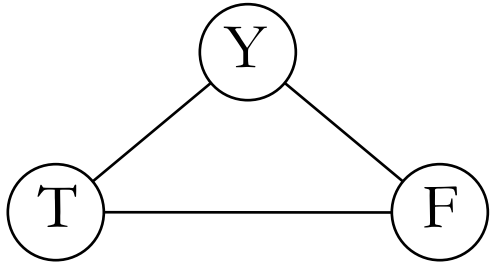
3COL = {⟨G⟩: G has a valid 3-coloring}

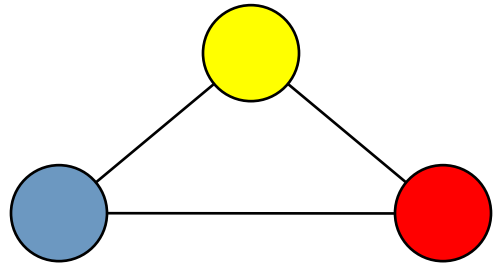
Idea: Assignment of ϕ ⟺ Coloring of G

Coloring -- Gadgets

3 Colors
T represents a colour (say Blue)
F represents a colour (say False)
Y represents a colour (say Yellow)

- Step 3: Reduce 3SAT to 3COL

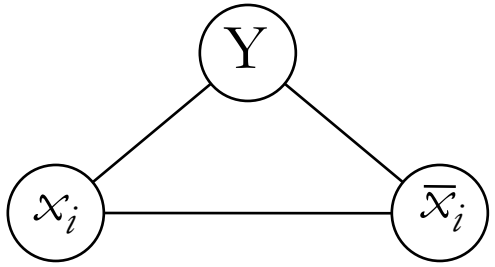




Part I: 3 special vertices T (true), F (false), and X

Variables in my 3SAT problem must be T (Blue) or False (Red)

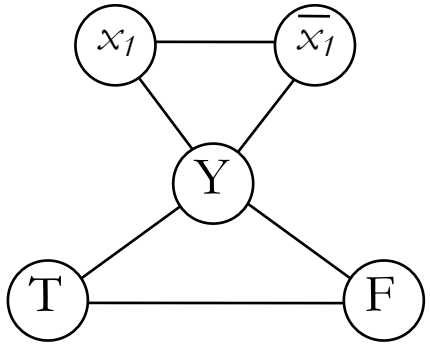
Coloring -- Gadgets



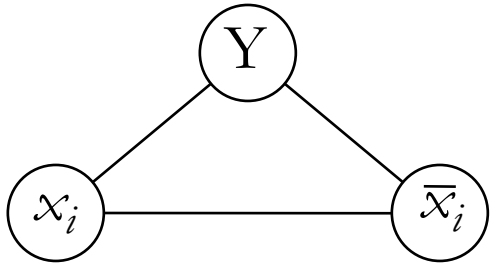
Either x_i has color of T
and $\bar{x}_i$ has color of F

Or x_i has color of F
and $\bar{x}_i$ has color of T

General Case → One variable:
Force a variable to take either T/F and it's negation to take the opposite



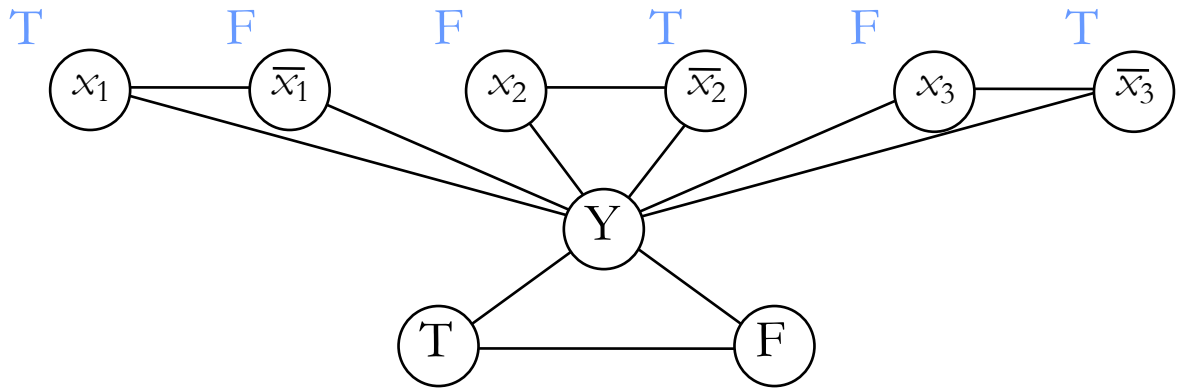
Coloring -- Gadgets



Either x_i has color of T and $\overline{x}_i$ has color of F

Or x_i has color of F and $\overline{x}_i$ has color of T

Part 2: For each variable x_i



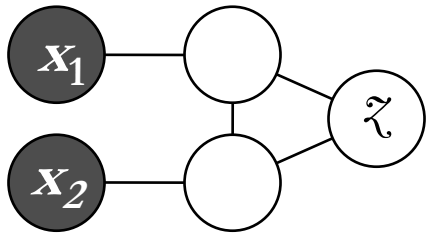
Example: $(x_1 \vee \overline{x}_1 \vee x_2) \wedge (x_1 \vee x_2 \vee \overline{x}_3)$

Still incomplete

Coloring

- To ***encode*** the **clauses** of ϕ, we need an **OR-*gadget***

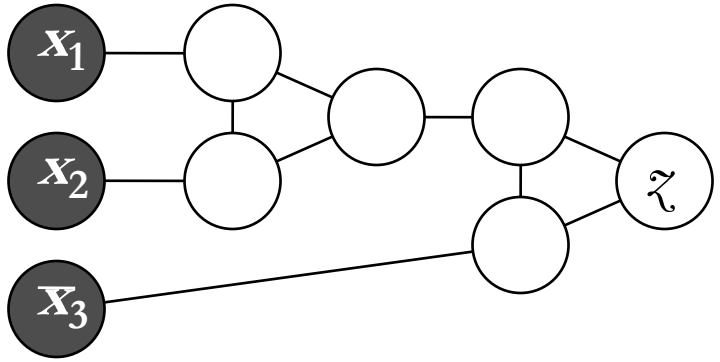
2 variables



If *x* or *y* is colored T, *z* can be colored T

If *x* and *y* are colored F, *z* must be colored F

3 variables

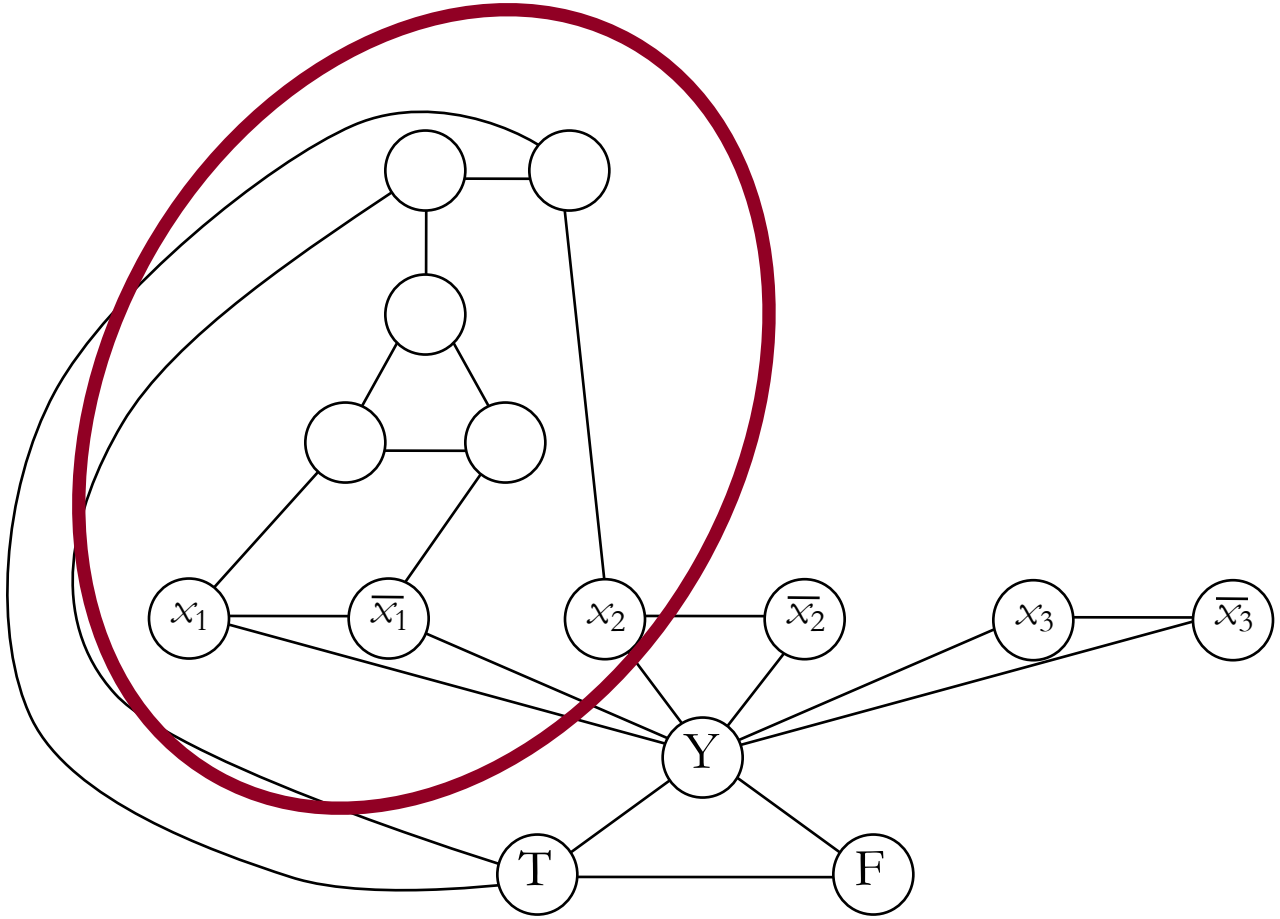


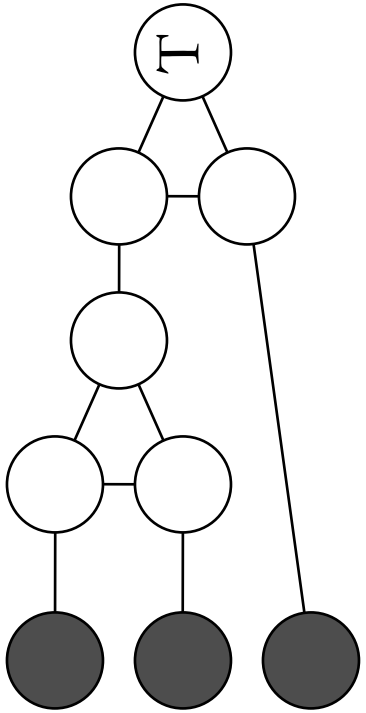
z can be colored T if and only if
at least one literal is colored T

Part 3: For each clause like $(x_1 \vee x_2 \vee \bar{x}_3)$

Coloring reduction: An example

OR-gadget

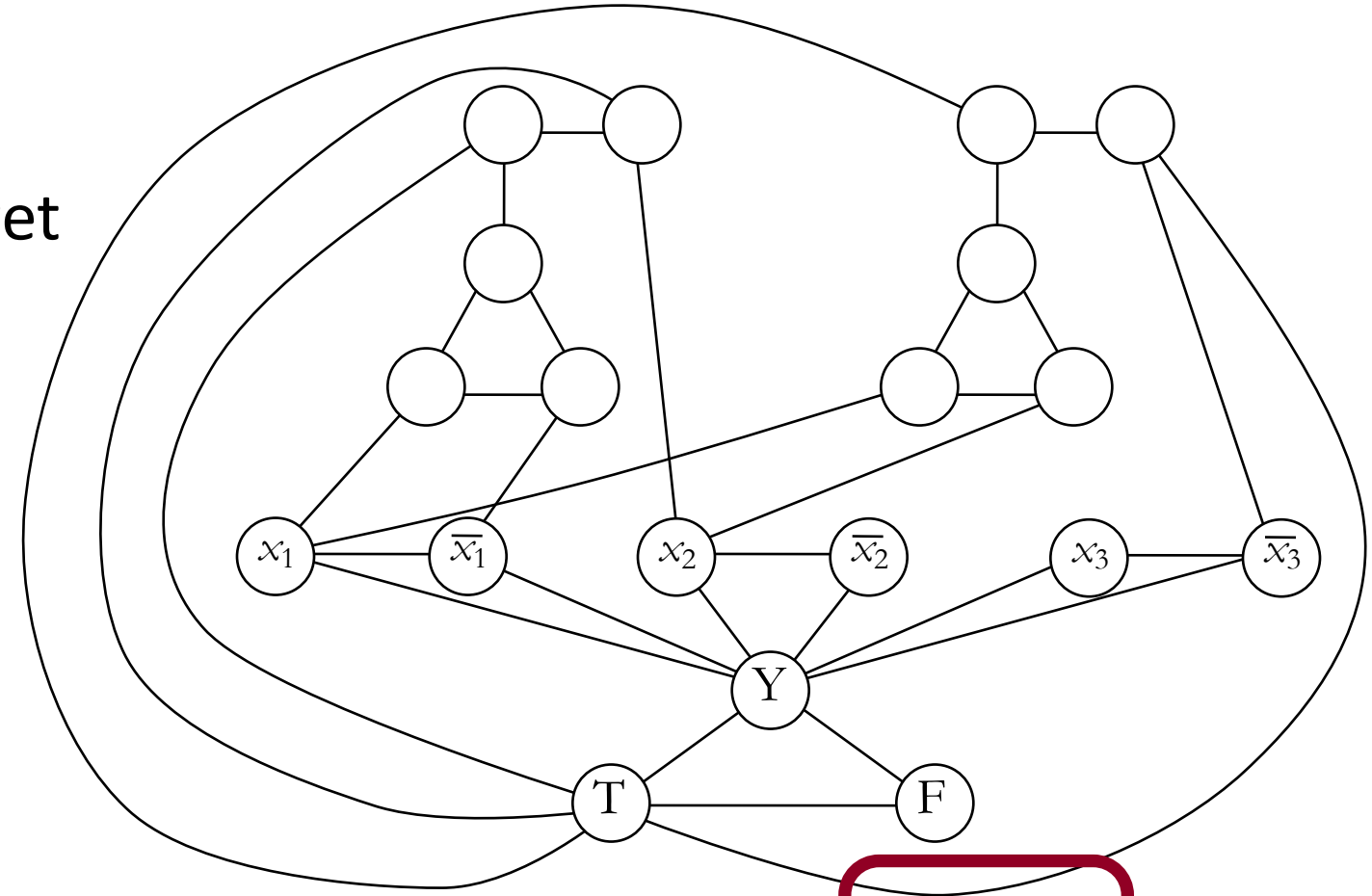


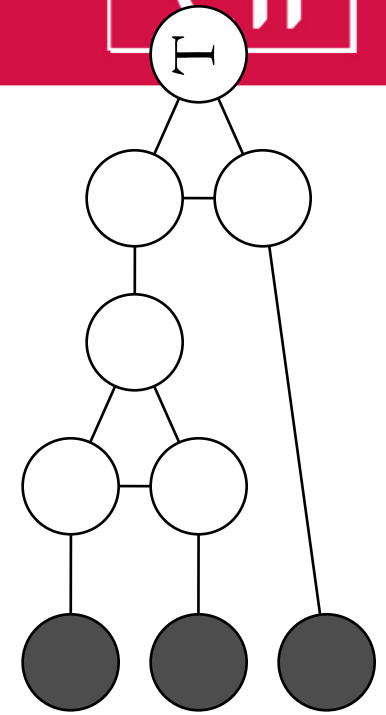


Example: $(x_1 \vee \overline{x_1} \vee x_2) \wedge (x_1 \vee x_2 \vee \overline{x_3})$

Coloring reduction: An example

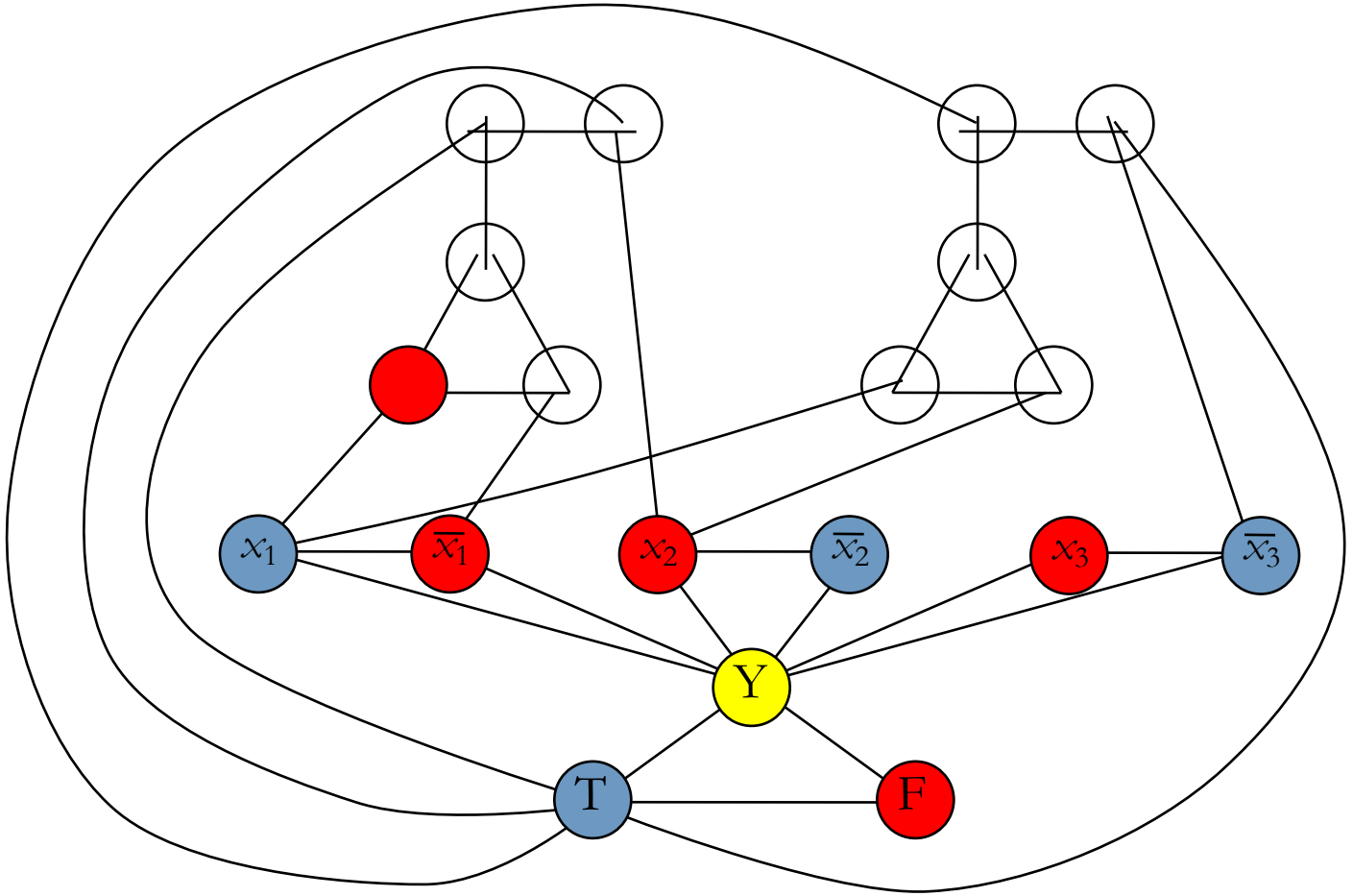
OR-gadget





Example: $(x_1 \vee \overline{x_1} \vee x_2) \wedge (x_1 \vee x_2 \vee \overline{x_3})$

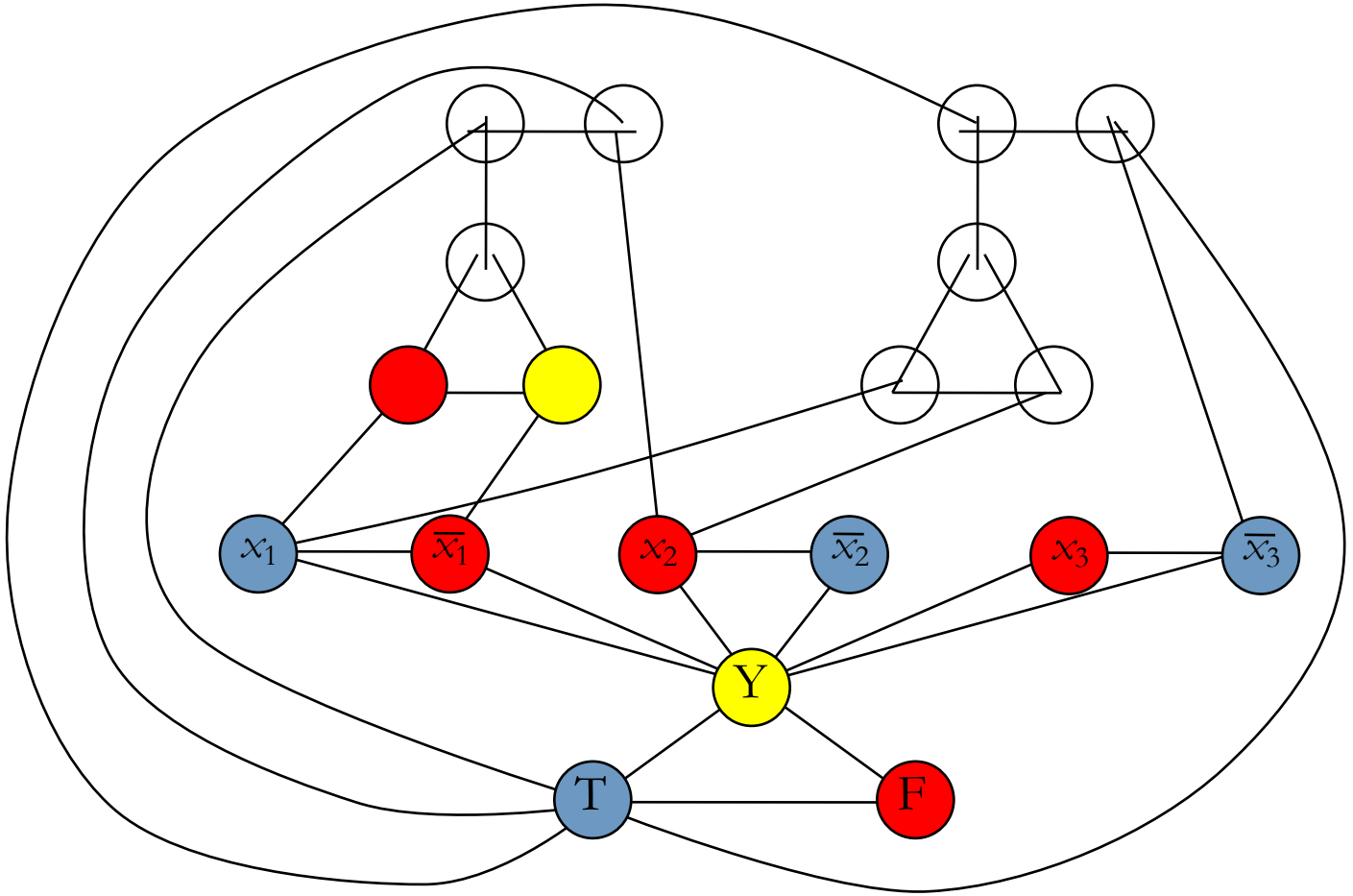
Coloring reduction: An example



Example: $(x_1 \vee \overline{x_1} \vee x_2) \wedge (x_1 \vee x_2 \vee \overline{x_3})$

T F F T F T

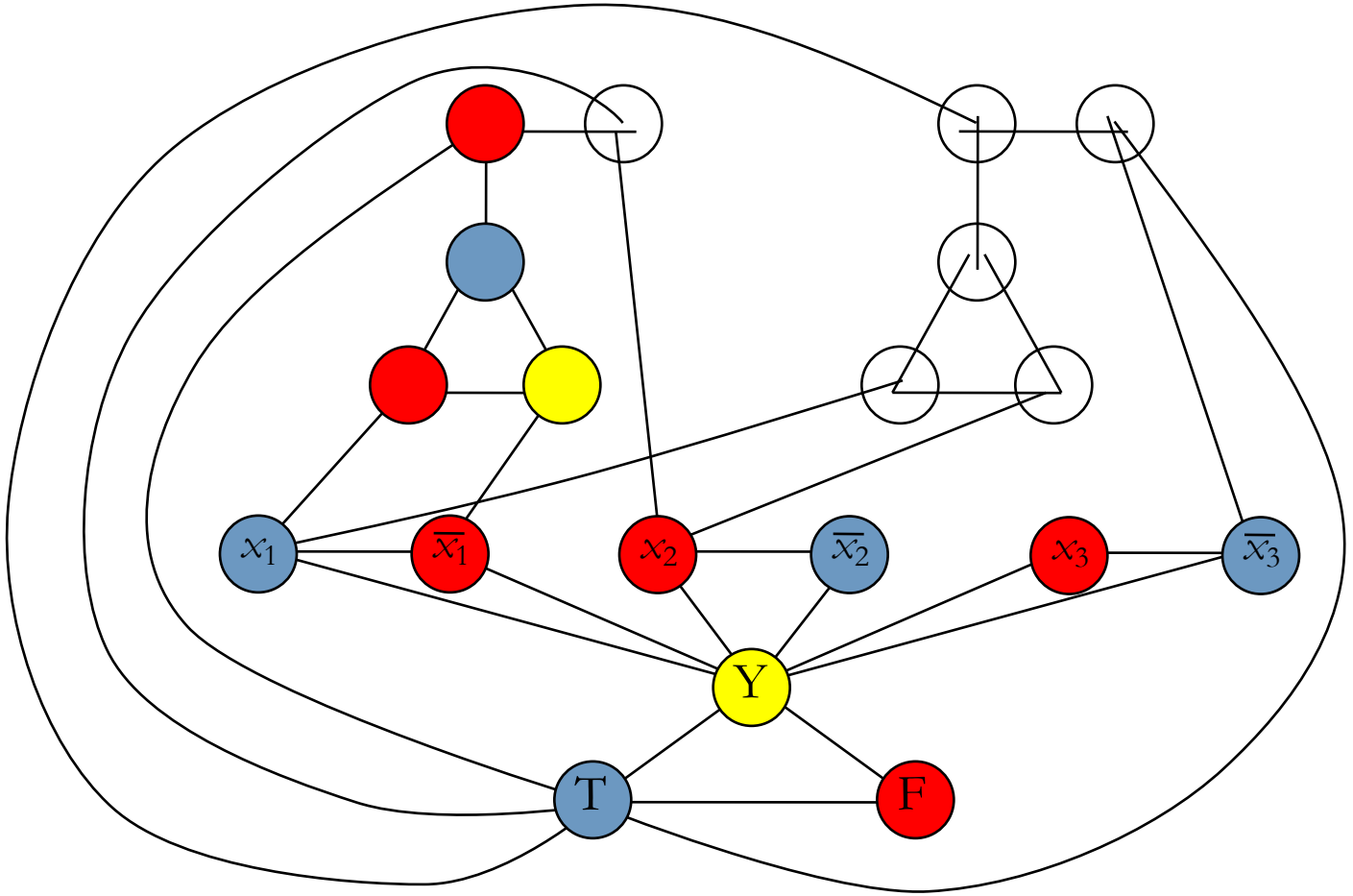
Coloring reduction: An example



Example: $(x_1 \vee \overline{x_1} \vee x_2) \wedge (x_1 \vee x_2 \vee \overline{x_3})$

T F F T F T

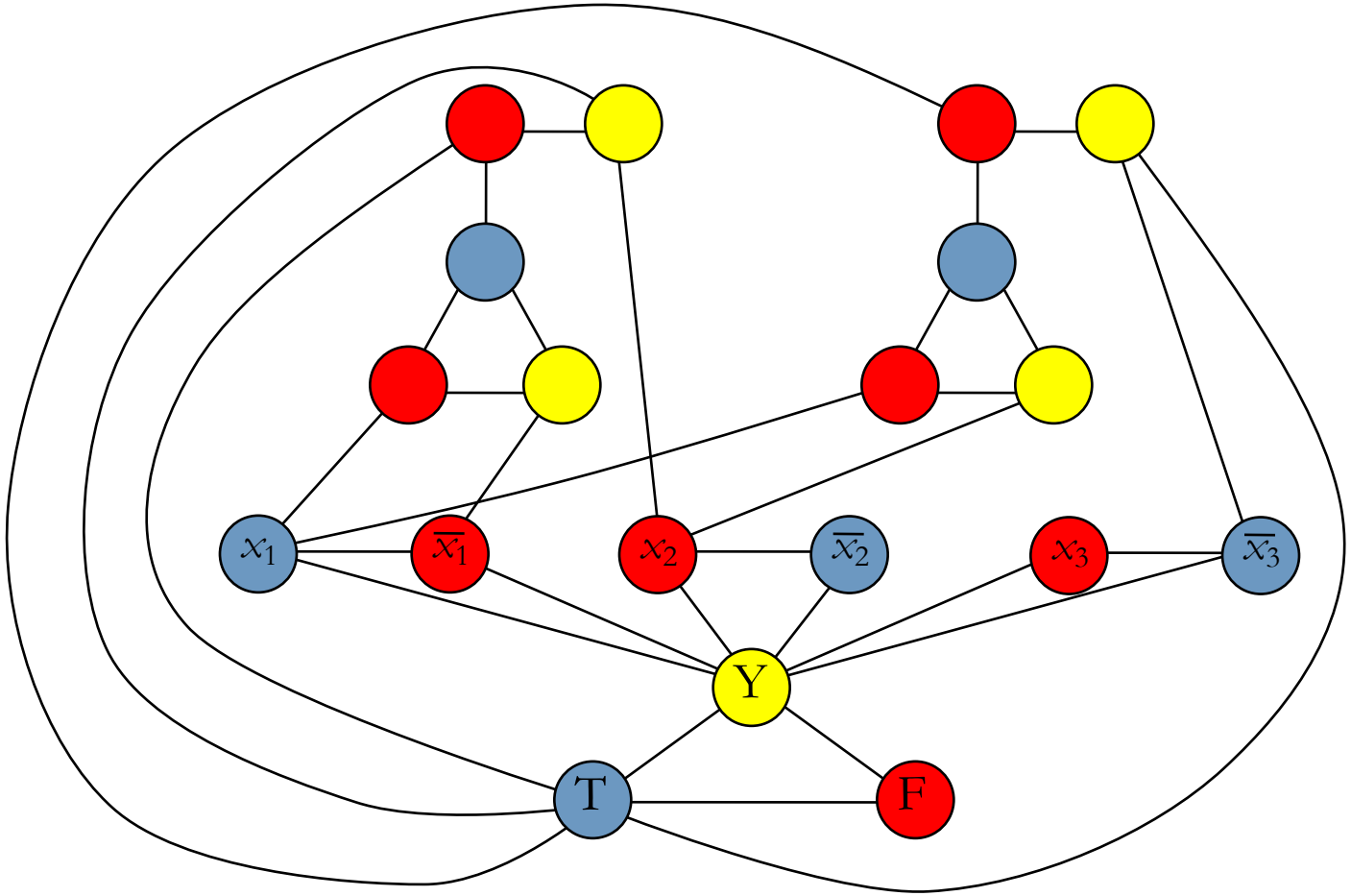
Coloring reduction: An example



Example: $(x_1 \vee \overline{x_1} \vee x_2) \wedge (x_1 \vee x_2 \vee \overline{x_3})$

T F F T F T

Coloring reduction: An example



Example: $(x_1 \vee \overline{x_1} \vee x_2) \wedge (x_1 \vee x_2 \vee \overline{x_3})$

T F F T F T

Coloring

- Step 3: Reduce 3SAT to 3COL

Running time = O($m + n$)
n = number of variables
m = number of clauses

R := On input $\langle f \rangle$, where f is a 3CNF:

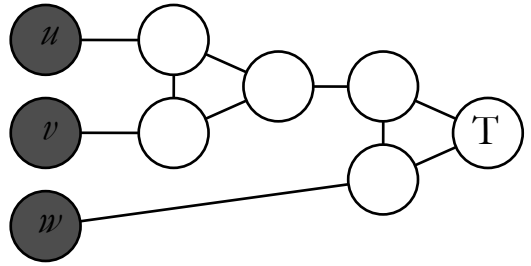
Construct the following graph G:

Add 3 special vertices T, F, X connected by a triangle.

For every variable x_i of f, add vertices x_i and $\overline{x}_i$ and include the triangle x_i, $\overline{x}_i$, X.

For every clause $u \vee v \vee w$ of f, connect u, v, w, and T using the gadget below.

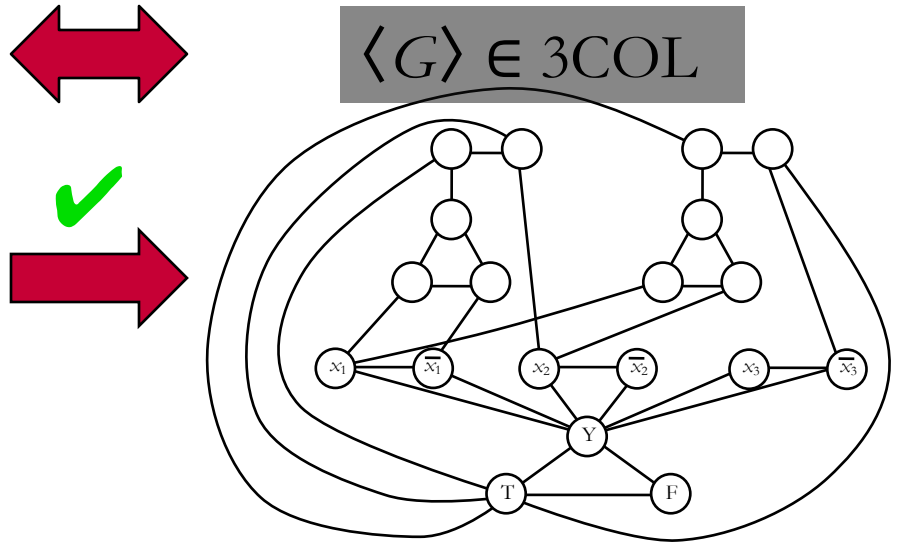
Output $\langle G \rangle$.



Coloring

$\langle f \rangle \in$ 3SAT

$(x_1 \vee \overline{x}_1 \vee x_2) \wedge (x_1 \vee x_2 \vee \overline{x}_3)$



f has a SAT assignment *a*

G has a valid 3-coloring

Each literal gets “color” as in *a*

Each **clause gadget** contains a true literal, so it can be colored

Coloring

- Step 4: Argue that the reduction is correct

$\langle f \rangle \in$ 3SAT

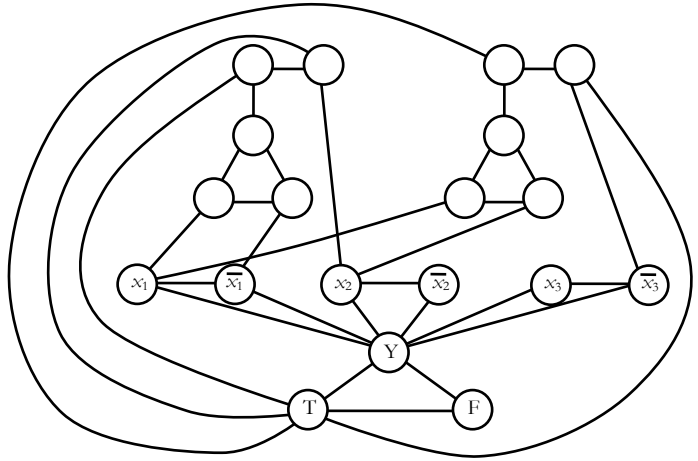
$(x_1 \vee \overline{x}_1 \vee x_2) \wedge (x_1 \vee x_2 \vee \overline{x}_3)$

ϕ has a SAT assignment

Each variable gets value as in G

$$x_i = \begin{cases} \text{true, if node } x_i \text{ has same color as T} \\ \text{false, if node } x_i \text{ has same color as F} \end{cases}$$

$\langle G \rangle \in$ 3COL



G has a valid 3-coloring c

Since each clause gadget is colored properly, each clause must contain a true literal