COL758: Advanced Algorithms

Ragesh Jaiswal, CSE, IITD

Computational Intractability

Definition (NP)

A problem is said to be in NP iff there exists an efficient certification algorithm for the problem.

Definition (P)

A problem is said to be in P iff there exists an efficient algorithm that solves the problem.

Definition (NP-complete)

A problem X is said to be NP-complete iff the following two properties hold:

1. $X \in \text{NP}$.
2. For all $Y \in \text{NP}$, $Y \leq_p X$.

Theorem (Cook-Levin Theorem)

3-SAT is NP*-complete.*

Definition (NP-complete)

A problem X is said to be NP-complete iff the following two properties hold:

1. $X \in \text{NP}$.
2. For all $Y \in \text{NP}$, $Y \leq_p X$.

Theorem (Cook-Levin Theorem)

3-SAT is NP-complete.

Proof sketch

- Claim 1: CIRCUIT-SAT is NP-complete.
- Claim 2: CIRCUIT-SAT $\leq_p$ 3-SAT.

Theorem (Cook-Levin Theorem)

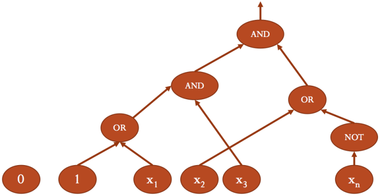
3-SAT is NP-*complete.*

Proof sketch

- Claim 1: CIRCUIT-SAT is NP-complete.
- Claim 2: CIRCUIT-SAT $\leq_p$ 3-SAT.

- Circuit: A directed acyclic graph where each node is either:
  - Constant nodes: Labeled 0/1
  - Input nodes: These denote the variables
  - Gates: AND, OR, and NOT

  There is a single output node.



Theorem (Cook-Levin Theorem)

3-SAT is NP*-complete.*

Proof sketch

- Claim 1: CIRCUIT-SAT is NP-complete.
- Claim 2: CIRCUIT-SAT $\leq_p$ 3-SAT.

- Circuit: A directed acyclic graph where each node is either:
  - Constant nodes: Labeled 0/1
  - Input nodes: These denote the variables
  - Gates: AND, OR, and NOT

  There is a single output node.

Problem

CIRCUIT-SAT: Given a circuit, determine if there is an input such that the output of the circuit is 1.

Theorem (Cook-Levin Theorem)

3-*SAT is* NP-*complete.*

Proof sketch

- Claim 1: CIRCUIT-SAT is NP-complete.
    - Fact: For every algorithm that runs in time polynomial in the input size n, there is an equivalent circuit of size polynomial in n.
- Claim 2: CIRCUIT-SAT $\leq_p$ 3-SAT.

- Circuit: A directed acyclic graph where each node is either:
    - Constant nodes: Labeled 0/1
    - Input nodes: These denote the variables
    - Gates: AND, OR, and NOT

  There is a single output node.

Problem

CIRCUIT-SAT: Given a circuit, determine if there is an input such that the output of the circuit is 1.

Theorem (Cook-Levin Theorem)

3-SAT is NP*-complete.*

Proof sketch

- Claim 1: CIRCUIT-SAT is NP-complete.
  - Fact: For every algorithm that runs in time polynomial in the input size n, there is an equivalent circuit of size polynomial in n.
  - Given an input instance s of any NP problem X, consider the equivalent circuit for the efficient certifier of X. The input gates of this circuit has s and t.
  - $s \in X$ if and only if this circuit is satisfiable.
- Claim 2: CIRCUIT-SAT $\leq_p$ 3-SAT.

Problem

CIRCUIT-SAT: Given a circuit, determine if there is an input such that the output of the circuit is 1.

Theorem (Cook-Levin Theorem)

3-SAT is NP*-complete.*

Proof sketch

- Claim 1: CIRCUIT-SAT is NP-complete.
    - Fact: For every algorithm that runs in time polynomial in the input size n, there is an equivalent circuit of size polynomial in n.
    - Given an input instance s of any NP problem X, consider the equivalent circuit for the efficient certifier of X. The input gates of this circuit has s and t.
    - $s \in X$ if and only if this circuit is satisfiable.
- Claim 2: CIRCUIT-SAT $\leq_p$ 3-SAT.
    - For any circuit, we can write an equivalent 3-SAT formula.

Problem

CIRCUIT-SAT: Given a circuit, determine if there is an input such that the output of the circuit is 1.

Definition (NP)

A problem X is said to be in NP iff there is an efficient certifier for X.

Definition (NP-complete)

A problem is said to be NP-complete iff the following two properties hold:

- $X \in \text{NP}$
- For all $Y \in \text{NP}$, $Y \leq_p X$

Theorem (Cook-Levin Theorem)

3-*SAT is* NP-*complete.*

Definition (NP-hard)

A problem X is said to be NP-hard iff the following property holds:

- $\cancel{X \in \text{NP}}$
- For all $Y \in \text{NP}$, $Y \leq_p X$

Theorem (Cook-Levin Theorem)

3-*SAT is* NP-*complete.*

- Claim 1: INDEPENDENT-SET, VERTEX-COVER, SET-COVER are also NP-complete.

Proof of Claim 1

- These problems are in NP.
- 3-SAT $\leq_p$ INDEPENDENT-SET $\leq_p$ VERTEX-COVER $\leq_p$ SET-COVER

Problem

TSP: Given a complete, weighted, directed graph G and an integer k, determine if there is a tour in the graph of total length at most k.

- Claim 1: TSP $\in$ NP
  - Proof sketch: A tour of length at most k is a certificate.

Problem

TSP: Given a complete, weighted, directed graph G and an integer k, determine if there is a tour in the graph of total length at most k.

- Claim 1: TSP $\in$ NP
  - Proof sketch: A tour of length at most k is a certificate.
- Claim 2: 3-SAT $\leq_p$ TSP

Proof of Claim 2

- Claim 2.1: 3-SAT $\leq_p$ HAMILTONIAN-CYCLE
- Claim 2.2: HAMILTONIAN-CYCLE $\leq_p$ TSP

Problem

HAMILTONIAN-CYCLE: Given an unweighted, directed graph, determine if there is a Hamiltonian cycle in the graph.

- Hamiltonian cycle: A cycle that visits each vertex exactly once.

Problem

TSP: Given a complete, weighted, directed graph G and an integer k, determine if there is a tour in the graph of total length at most k.

Problem

HAMILTONIAN-CYCLE: Given an unweighted, directed graph, determine if there is a Hamiltonian cycle in the graph.

- Hamiltonian cycle: A cycle that visits each vertex exactly once.

- Claim 2.2: HAMILTONIAN-CYCLE $\leq_p$ TSP

Proof of Claim 2.2

- Given an unweighted, directed graph G, construct the following complete, directed, weighted graph G':
  - For each edge (u, v) in G, give the weight of 1 to edge (u, v) in G'
  - For each pair (u, v) such that there is no edge from u to v in G, add an edge (u, v) with weight 2 in G'
- Claim 2.2.1: G has a Hamiltonian cycle if and only if G' has a tour of length at most n

Problem

TSP: Given a complete, weighted, directed graph G and an integer k, determine if there is a tour in the graph of total length at most k.

- Claim 1: TSP $\in$ NP
  - Proof sketch: A tour of length at most k is a certificate.
- Claim 2: 3-SAT $\leq_p$ TSP

Proof of Claim 2

- Claim 2.1: 3-SAT $\leq_p$ HAMILTONIAN-CYCLE
- Claim 2.2: HAMILTONIAN-CYCLE $\leq_p$ TSP

Problem

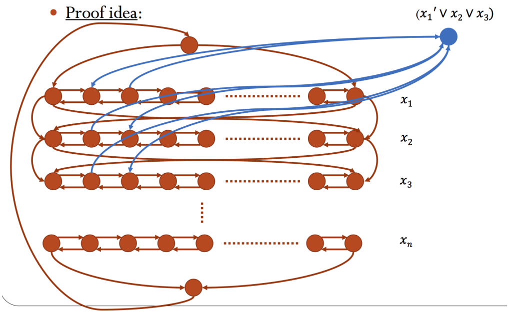
HAMILTONIAN-CYCLE: Given an unweighted, directed graph, determine if there is a Hamiltonian cycle in the graph.

- Hamiltonian cycle: A cycle that visits each vertex exactly once.

- Claim 2.1: 3-SAT $\leq_p$ HAMILTONIAN-CYCLE

Proof of Claim 2.1

- Given an instance of the 3-SAT problem (a formula Ω with n variables and m clauses), we need to create a directed graph G such that Ω is satisfiable if and only if G has a Hamiltonian cycle.

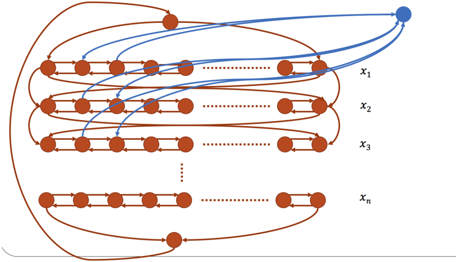


- Claim 2.1: 3-SAT $\leq_p$ HAMILTONIAN-CYCLE

Proof of Claim 2.1

- Given an instance of the 3-SAT problem (a formula Ω with n variables and m clauses), we need to create a directed graph G such that Ω is satisfiable if and only if G has a Hamiltonian cycle.
- Claim 2.1.1: If the 3-SAT formula is satisfiable, then there is a Hamiltonian cycle in the constructed graph.

- Proof idea:

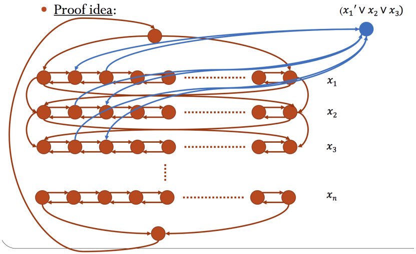


- Claim 2.1: 3-SAT $\leq_p$ HAMILTONIAN-CYCLE

Proof of Claim 2.1

- Given an instance of the 3-SAT problem (a formula Ω with n variables and m clauses), we need to create a directed graph G such that Ω is satisfiable if and only if G has a Hamiltonian cycle.
- Claim 2.1.1: If the 3-SAT formula is satisfiable, then there is a Hamiltonian cycle in the constructed graph.
- Claim 2.1.2: If the constructed graph has a Hamiltonian cycle, then the 3-SAT formula has a satisfying assignment.

- Proof idea:



Definition (Hamiltonian path)

A Hamiltonian path in any directed graph is a path that visits each vertex exactly once.

Problem

HAMILTONIAN-PATH: Given a directed graph G, determine if there is a Hamiltonian path in the graph.

Definition (Hamiltonian path)

A Hamiltonian path in any directed graph is a path that visits each vertex exactly once.

Problem

HAMILTONIAN-PATH: Given a directed graph G, determine if there is a Hamiltonian path in the graph.

- Claim 1: HAMILTONIAN-PATH is NP-complete.

Definition (Hamiltonian path)

A Hamiltonian path in any directed graph is a path that visits each vertex exactly once.

Problem

HAMILTONIAN-PATH: Given a directed graph G, determine if there is a Hamiltonian path in the graph.

- Claim 1: HAMILTONIAN-PATH is NP-complete.

Proof of Claim 1

- Claim 1.1: HAMILTONIAN-PATH $\in$ NP

Definition (Hamiltonian path)

A Hamiltonian path in any directed graph is a path that visits each vertex exactly once.

Problem

HAMILTONIAN-PATH: Given a directed graph G, determine if there is a Hamiltonian path in the graph.

- Claim 1: HAMILTONIAN-PATH is NP-complete.

Proof of Claim 1

- Claim 1.1: HAMILTONIAN-PATH $\in$ NP
  - A Hamiltonian path acts as a certificate.

Definition (Hamiltonian path)

A Hamiltonian path in any directed graph is a path that visits each vertex exactly once.

Problem

HAMILTONIAN-PATH: Given a directed graph G, determine if there is a Hamiltonian path in the graph.

- Claim 1: HAMILTONIAN-PATH is NP-complete.

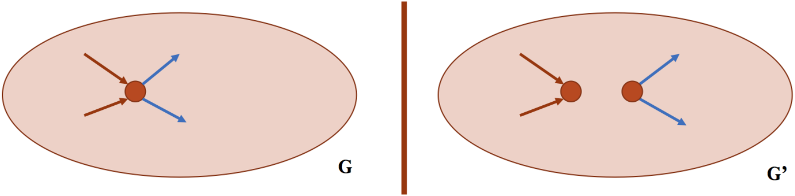
Proof of Claim 1

- Claim 1.1: HAMILTONIAN-PATH $\in$ NP
  - A Hamiltonian path acts as a certificate.
- Claim 1.2: HAMILTONIAN-PATH is NP-hard.
  - Claim 1.2.1: HAMILTONIAN-CYCLE $\leq_p$ HAMILTONIAN-PATH

- Claim 1.2.1: HAMILTONIAN-CYCLE $\leq_p$ HAMILTONIAN-PATH

Proof of Claim 1.2.1

- Consider the graph G' constructed from graph G.
- There is a Hamiltonian cycle in G if and only there is a Hamiltonian path in G'.



Definition (k-colorable)

A graph is said to be k-colorable is it is possible to assign one of k colors to each node such that for every edge (u, v), u and v are assigned different colors.

Problem

$\underline{k\text{-COLORING}}$: Given a graph G, determine if G is k-colorable.

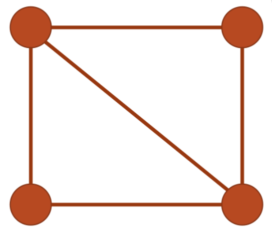


Figure: Is this graph 2-colorable?

Definition (k-colorable)

A graph is said to be k-colorable is it is possible to assign one of k colors to each node such that for every edge (u, v), u and v are assigned different colors.

Problem

<u>k-COLORING</u>: Given a graph G, determine if G is k-colorable.

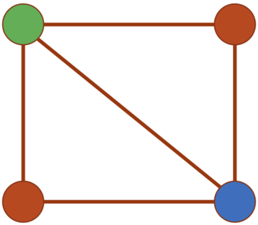


Figure: Is this graph 2-colorable? Yes

Definition (k-colorable)

A graph is said to be k-colorable is it is possible to assign one of k colors to each node such that for every edge (u, v), u and v are assigned different colors.

Problem

k-COLORING: Given a graph G, determine if G is k-colorable.

Problem

2-COLORING: Given a graph G, determine if G is 2-colorable.

- How hard is the 2-COLORING problem?

Definition (k-colorable)

A graph is said to be k-colorable is it is possible to assign one of k colors to each node such that for every edge (u, v), u and v are assigned different colors.

Problem

k-COLORING: Given a graph G, determine if G is k-colorable.

Problem

2-COLORING: Given a graph G, determine if G is 2-colorable.

- How hard is the 2-COLORING problem?
  - 2-COLORING $\in$ P since G is 2-colorable if and only if G is bipartite and we know an efficient algorithm for checking if a given graph is bipartite.

Definition (k-colorable)

A graph is said to be k-colorable is it is possible to assign one of k colors to each node such that for every edge (u, v), u and v are assigned different colors.

Problem

k-COLORING: Given a graph G, determine if G is k-colorable.

Problem

3-COLORING: Given a graph G, determine if G is 3-colorable.

- How hard is the 3-COLORING problem?

Definition (k-colorable)

A graph is said to be k-colorable is it is possible to assign one of k colors to each node such that for every edge (u, v), u and v are assigned different colors.

Problem

<u>k-COLORING</u>: Given a graph G, determine if G is k-colorable.

Problem

<u>3-COLORING</u>: Given a graph G, determine if G is 3-colorable.

- How hard is the 3-COLORING problem?
- <u>Claim 1</u>: 3-COLORING is NP-complete.

Problem

3-COLORING: Given a graph G, determine if G is 3-colorable.

- Claim 1: 3-COLORING is NP-complete.

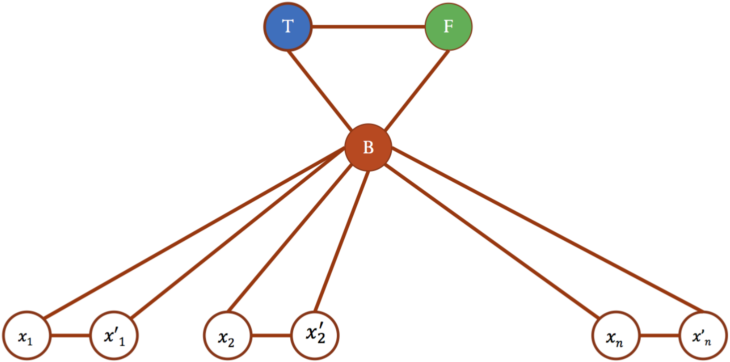
Proof of Claim 1

- Claim 1.1: 3-COLORING is in NP
  - A short certificate is a 3-coloring of the graph.
- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

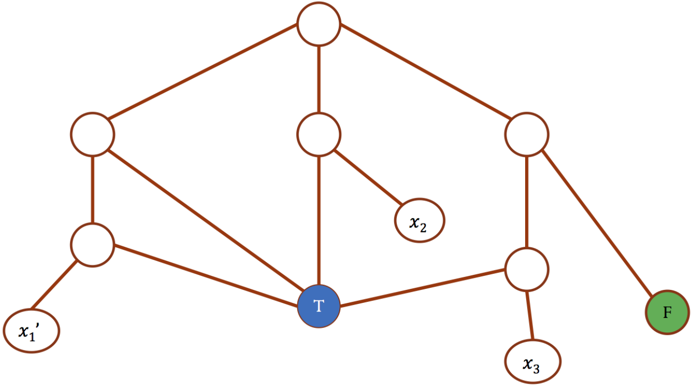
- Consider the following gadget. There is a bijection between colors and truth values.



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

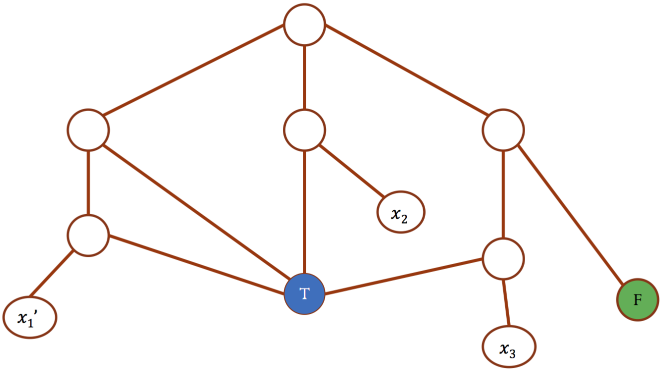
- How we encode a clause, say $(\bar{x}_1 \vee x_2 \vee x_3)$.



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

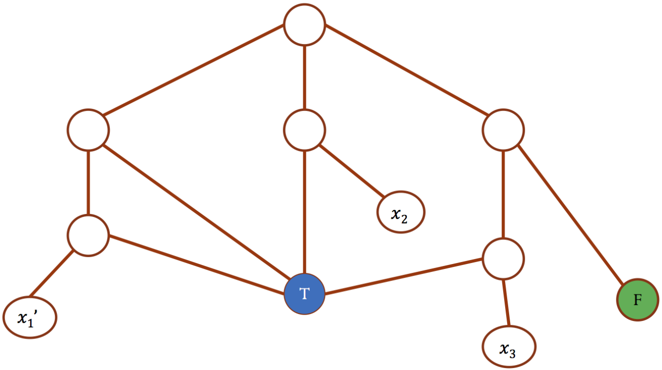
- Claim 1.2.1: There is no 3 coloring of the graph below with nodes $\bar{x}_1, x_2$, and x_3 assigned F color.



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

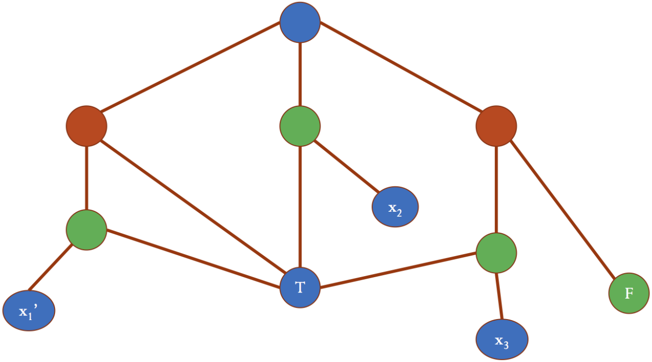
- Claim 1.2.2: There is a 3 coloring of the graph below with at least one of the nodes $\bar{x}_1, x_2$, and x_3 assigned T color.



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

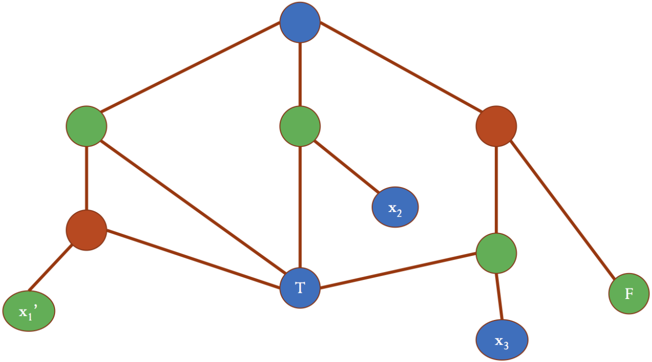
- Claim 1.2.2: There is a 3 coloring of the graph below with at least one of the nodes $\bar{x}_1, x_2$, and x_3 assigned T color.
  - $\bar{x}_1 : T, x_2 : T, x_3 : T$



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

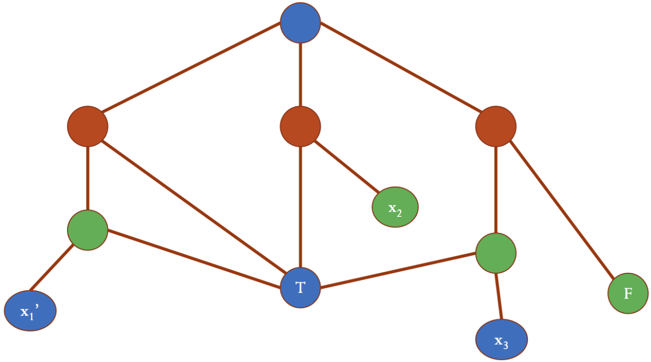
- Claim 1.2.2: There is a 3 coloring of the graph below with at least one of the nodes $\bar{x}_1, x_2$, and x_3 assigned T color.
  - $\bar{x}_1 : F, x_2 : T, x_3 : T$



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

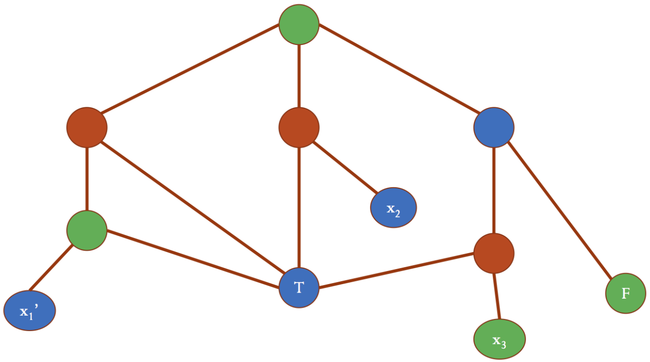
- Claim 1.2.2: There is a 3 coloring of the graph below with at least one of the nodes $\bar{x}_1, x_2$, and x_3 assigned T color.
  - $\bar{x}_1 : T, x_2 : F, x_3 : T$



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

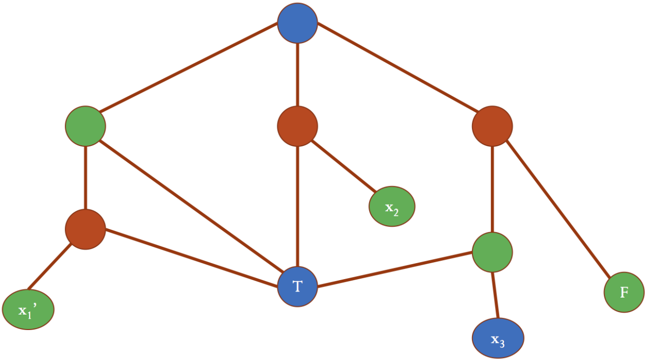
- Claim 1.2.2: There is a 3 coloring of the graph below with at least one of the nodes $\bar{x}_1, x_2$, and x_3 assigned T color.
  - $\bar{x}_1 : T, x_2 : T, x_3 : F$



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

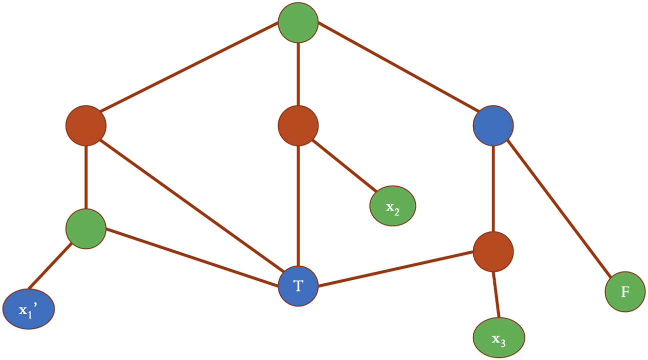
- Claim 1.2.2: There is a 3 coloring of the graph below with at least one of the nodes $\bar{x}_1, x_2$, and x_3 assigned T color.
  - $\bar{x}_1 : F, x_2 : F, x_3 : T$



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

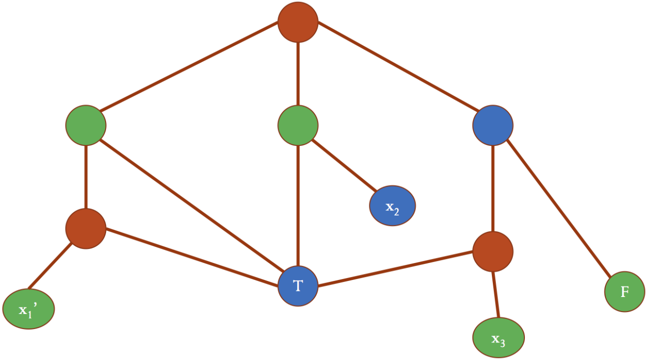
- Claim 1.2.2: There is a 3 coloring of the graph below with at least one of the nodes $\bar{x}_1, x_2$, and x_3 assigned T color.
  - $\bar{x}_1 : T, x_2 : F, x_3 : F$



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

- Claim 1.2.2: There is a 3 coloring of the graph below with at least one of the nodes $\bar{x}_1, x_2$, and x_3 assigned T color.
  - $\bar{x}_1 : F, x_2 : T, x_3 : F$



- Claim 1.2: 3-SAT $\leq_p$ 3-COLORING

Proof ideas for Claim 1.2

- Claim 1.2.3: The given formula is satisfiable if and only if the constructed graph has a 3 coloring.

SUBSET-SUM

Given natural numbers $w_1, ..., w_n$ and a target number W, determine if there is a subset S of $\{1, ..., n\}$ such that $\sum_{i \in S} w_i = W$.

SCHEDULING

Given n jobs with start time s_i and duration t_i and deadline d_i, determine if all the jobs can be scheduled on a single machine such that no deadlines are missed.

- Claim 1: SUBSET-SUM $\in$ NP

SUBSET-SUM

Given natural numbers $w_1, ..., w_n$ and a target number W, determine if there is a subset S of $\{1, ..., n\}$ such that $\sum_{i \in S} w_i = W$.

SCHEDULING

Given n jobs with start time s_i and duration t_i and deadline d_i, determine if all the jobs can be scheduled on a single machine such that no deadlines are missed.

- Claim 1: SUBSET-SUM $\in$ NP
- Claim 2: SCHEDULING $\in$ NP

SUBSET-SUM

Given natural numbers $w_1, ..., w_n$ and a target number W, determine if there is a subset S of $\{1, ..., n\}$ such that $\sum_{i \in S} w_i = W$.

SCHEDULING

Given n jobs with start time s_i and duration t_i and deadline d_i, determine if all the jobs can be scheduled on a single machine such that no deadlines are missed.

- Claim 1: SUBSET-SUM $\in$ NP
- Claim 2: SCHEDULING $\in$ NP
- Claim 3: SUBSET-SUM $\leq_p$ SCHEDULING

SUBSET-SUM

Given natural numbers $w_1, ..., w_n$ and a target number W, determine if there is a subset S of $\{1, ..., n\}$ such that $\sum_{i \in S} w_i = W$.

SCHEDULING

Given n jobs with start time s_i and duration t_i and deadline d_i, determine if all the jobs can be scheduled on a single machine such that no deadlines are missed.

- Claim 1: SUBSET-SUM $\in$ NP
- Claim 2: SCHEDULING $\in$ NP
- Claim 3: SUBSET-SUM $\leq_p$ SCHEDULING

Proof sketch for Claim 3

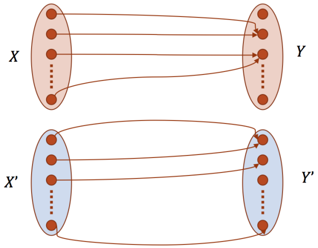
Given an instance of the subset sum problem $(\{w_1, ..., w_n\}, W)$, we construct the following instance of the Scheduling problem: $((0, w_1, S+1), ..., (0, w_n, S+1), (W, 1, W+1))$. We then argue that there is a subset that sums to W if and only if the $(n+1)$ jobs can be scheduled. Here $S = w_1 + ... + w_n$.

- Most of the polynomial-time reductions $X \leq_p Y$ that we have seen are of the following general nature: We give an efficient mapping from instances of X to instances of Y such that "yes" instances of X map to "yes" instances of Y and "no" instances of X map to "no" instances of Y.
- Such reductions have special name. They are called many-one reductions.

- Most of the polynomial-time reductions $X \leq_p Y$ that we have seen are of the following general nature: We give an efficient mapping from instances of X to instances of Y such that "yes" instances of X map to "yes" instances of Y and "no" instances of X map to "no" instances of Y.
- Such reductions have special name. They are called many-one reductions.

Many-one reduction

In order to show that $X \leq_p Y$ we design an efficient mapping f from the set of instances of X to set of instances of Y such that $s \in X$ iff $f(s) \in Y$.



3D-MATCHING

Given disjoint sets X, Y, and Z each of size n, and given a set T of triples (x, y, z), determine if there exist a subset of n triples in T such that each element of $X \cup Y \cup Z$ is contained in exactly one of these triples.

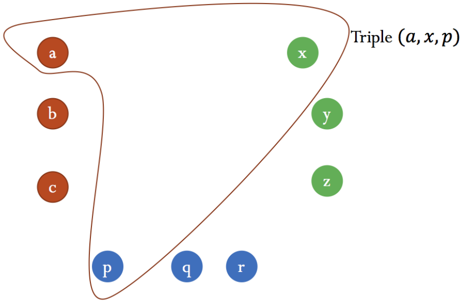


Figure: Let $T = \{(a, x, p), (a, y, p), (b, y, q), (c, z, r)\}$. Does there exist a 3D-Matching?

3D-MATCHING

Given disjoint sets X, Y, and Z each of size n, and given a set T of triples (x, y, z), determine if there exist a subset of n triples in T such that each element of $X \cup Y \cup Z$ is contained in exactly one of these triples.

- Claim 1: 3D-MATCHING $\in$ NP.

3D-MATCHING

Given disjoint sets X, Y, and Z each of size n, and given a set T of triples (x, y, z), determine if there exist a subset of n triples in T such that each element of $X \cup Y \cup Z$ is contained in exactly one of these triples.

- Claim 1: 3D-MATCHING $\in$ NP.
- Claim 2: 3D-MATCHING is NP-complete.

3D-MATCHING

Given disjoint sets X, Y, and Z each of size n, and given a set T of triples (x, y, z), determine if there exist a subset of n triples in T such that each element of $X \cup Y \cup Z$ is contained in exactly one of these triples.

- Claim 1: 3D-MATCHING $\in$ NP.
- Claim 2: 3D-MATCHING is NP-complete.
  - Claim 2.1: 3-SAT $\leq_p$ 3D-MATCHING.
  - Proof sketch of Claim 2.1: We will show an efficient many-one reduction.

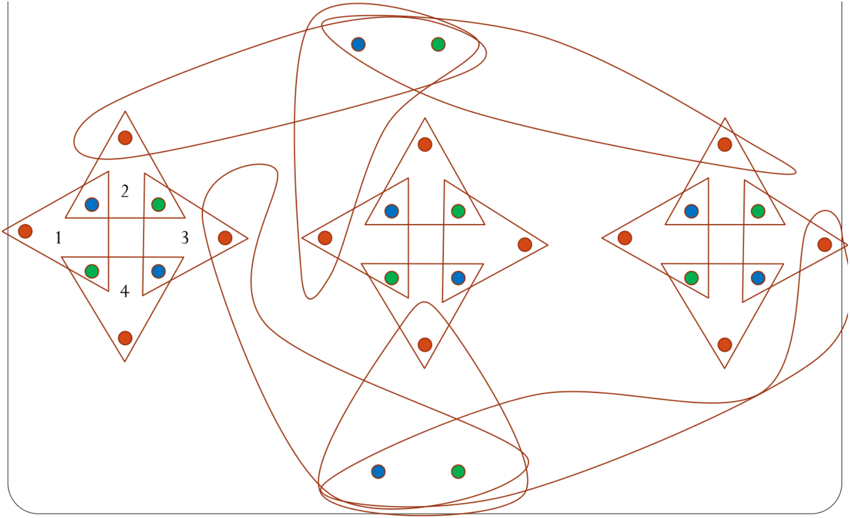


Figure: Example construction for $(x_1 \vee \bar{x}_2 \vee x_3), (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$

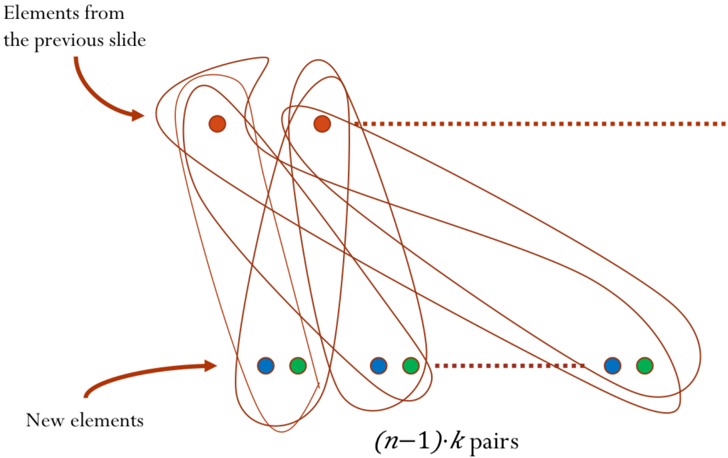


Figure: Example construction for $(x_1 \vee \bar{x}_2 \vee x_3), (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$. k denotes the number of clauses.

SUBSET-SUM

Given natural numbers $w_1, ..., w_n$ and a target number W, determine if there is a subset S of $\{1, ..., n\}$ such that $\sum_{i \in S} w_i = W$.

- Claim 1: SUBSET-SUM $\in$ NP.

SUBSET-SUM

Given natural numbers $w_1, ..., w_n$ and a target number W, determine if there is a subset S of $\{1, ..., n\}$ such that $\sum_{i \in S} w_i = W$.

- Claim 1: SUBSET-SUM $\in$ NP.
- Claim 2: SUBSET-SUM is NP-complete.

SUBSET-SUM

Given natural numbers $w_1, ..., w_n$ and a target number W, determine if there is a subset S of $\{1, ..., n\}$ such that $\sum_{i \in S} w_i = W$.

- Claim 1: SUBSET-SUM $\in$ NP.
- Claim 2: SUBSET-SUM is NP-complete.
  - Claim 2.1: 3D-MATCHING $\leq_p$ SUBSET-SUM.
  - Proof sketch: We will show an efficient many-one reduction. Given an instance (X, Y, Z, T) of the 3D-MATCHING problem, we construct an instance of the SUBSET-SET problem.
    - We first construct a $3n$-bit vector. Given a triple $t_i = (x_1, y_3, z_5)$, we construct the following vector v_i:

1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0

SUBSET-SUM

Given natural numbers $w_1, ..., w_n$ and a target number W, determine if there is a subset S of $\{1, ..., n\}$ such that $\sum_{i \in S} w_i = W$.

- Claim 1: SUBSET-SUM $\in$ NP.
- Claim 2: SUBSET-SUM is NP-complete.
  - Claim 2.1: 3D-MATCHING $\leq_p$ SUBSET-SUM.
  - Proof sketch: We will show an efficient many-one reduction. Given an instance (X, Y, Z, T) of the 3D-MATCHING problem, we construct an instance of the SUBSET-SET problem.
    - We first construct a $3n$-bit vector. Given a triple $t_i = (x_1, y_3, z_5)$, we construct the following vector v_i:

1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0

    - Let w_i be the value of v_i in base $(|T| + 1)$ and $W = \sum_{i=0}^{3n-1} (|T| + 1)^i$.
    - Claim 2.1.1: There is a 3D-Matching iff there is a subset $\{w_1, ..., w_{|T|}\}$ that sums to W.

End