Algorithms and Complexity (AC): Reductions

Marie Schmidt & Tom van der Zanden

(Based on slides by Gerhard Woeginger and Jesper Nederlof)

Landelijk Netwerk Mathematische Besliskunde

NP-hardness: 3-SAT

3-SAT

Instance: a set of logical variables $X := \{x_1, \ldots, x_n\}$ and a set of clauses C of three literals over X

Question: does there exist a truth assignment for X that simultaneously satisfies all clauses in C?

Theorem

3-SAT is NP-hard (and NP-complete).

NP-hardness: 3-SAT

3-SAT

Instance: a set of logical variables $X := \{x_1, \ldots, x_n\}$ and a set of clauses C of three literals over X

Question: does there exist a truth assignment for X that simultaneously satisfies all clauses in C?

Theorem

3-SAT is NP-hard (and NP-complete).

Proof: We show SAT $\leq_p$ 3-SAT.

NP-hardness: 3-SAT

3-SAT

Instance: a set of logical variables $X := \{x_1, \ldots, x_n\}$ and a set of clauses C of three literals over X

Question: does there exist a truth assignment for X that simultaneously satisfies all clauses in C?

Theorem

3-SAT is NP-hard (and NP-complete).

Proof: We show SAT $\leq_p$ 3-SAT.
Let $I = (X, C)$ an instance of SAT. We construct the following instance (X', C') of 3-SAT:

- For each clause c_j we construct a set of variables X_j and additional clauses C_j (with 3 literals each)
- $X' := X \cup \bigcup_{j=1}^{|C|} X_j$, $C' := \bigcup_{j=1}^{|C|} C_j$

Construction of X_j and C_j from clause c_j
1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show:

1.: This transformation is polynomial.

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show:

1.: This transformation is polynomial. ✓

2.: If there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$, then there is a satisfying truth assignment for (X, C):

Construction of X_j and C_j from clause c_j
1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$
4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show:
1.: This transformation is polynomial. ✓
2.: If there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$,
then there is a satisfying truth assignment for (X, C):
Let t' be a truth assignment on X'.

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show:

1.: This transformation is polynomial. ✓

2.: If there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$, then there is a satisfying truth assignment for (X, C):

Let t' be a truth assignment on X'.

We define $t := t'_{|X}$.

Then all clauses in C are satisfied.

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show:

1.: This transformation is polynomial. ✓

2.: If there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$, then there is a satisfying truth assignment for (X, C):

Let t' be a truth assignment on X'.

We define $t := t'_{|X}$.

Then all clauses in C are satisfied. ✓

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$, $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show: 3. If there is a satisfying truth assignment for (X, C), then there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$:

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$, $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show: 3. If there is a satisfying truth assignment for (X, C), then there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$:

Define truth assignment t' as follows:

- $t'(x) := t(x)\ \forall x \in X$

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$, $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show: 3. If there is a satisfying truth assignment for (X, C), then there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$:

Define truth assignment t' as follows:

- $t'(x) := t(x)\ \forall x \in X$
- $t'(x) := \text{true}\ \forall x \in X_j$ with $|c_j| \in \{1, 2\}$

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$, $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show: 3. If there is a satisfying truth assignment for (X, C), then there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$:

Define truth assignment t' as follows:

- $t'(x) := t(x)\ \forall x \in X$
- $t'(x) := \text{true}\ \forall x \in X_j$ with $|c_j| \in \{1, 2\}$
- For c_j with $|c_j| > 3$ let l_j^i be the first literal with $t(l_j^i) = \text{true}$:

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$, $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show: 3. If there is a satisfying truth assignment for (X, C), then there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$:

Define truth assignment t' as follows:

- $t'(x) := t(x)\ \forall x \in X$
- $t'(x) := \text{true}\ \forall x \in X_j$ with $|c_j| \in \{1, 2\}$
- For c_j with $|c_j| > 3$ let l_j^i be the first literal with $t(l_j^i) = \text{true}$: If $i \in \{1, 2\}$ we set $t'(y_j^i) := \text{false}\ \forall i$

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$, $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show: 3. If there is a satisfying truth assignment for (X, C), then there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$:

Define truth assignment t' as follows:

- $t'(x) := t(x)\ \forall x \in X$
- $t'(x) := \text{true}\ \forall x \in X_j$ with $|c_j| \in \{1, 2\}$
- For c_j with $|c_j| > 3$ let l_j^i be the first literal with $t(l_j^i) = \text{true}$: If $i \in \{1, 2\}$ we set $t'(y_j^i) := \text{false}\ \forall i$
  If $i \in \{k-1, k\}$ we set $t'(y_j^i) := \text{true}\ \forall i < k-3$

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$, $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show: 3. If there is a satisfying truth assignment for (X, C), then there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$:

Define truth assignment t' as follows:

- $t'(x) := t(x)\ \forall x \in X$
- $t'(x) := \text{true}\ \forall x \in X_j$ with $|c_j| \in \{1, 2\}$
- For c_j with $|c_j| > 3$ let l_j^i be the first literal with $t(l_j^i) = \text{true}$: If $i \in \{1, 2\}$ we set $t'(y_j^i) := \text{false}\ \forall i$
  If $i \in \{k-1, k\}$ we set $t'(y_j^i) := \text{true}\ \forall i < k-3$
  If $i \in \{3, 4, \ldots, k-2\}$, set $t'(y_j^l) := \text{true}\ \forall\ l \leq i-2$ & $t'(y_j^l) = \text{true}\ \forall\ l \geq i-2$

Then all clauses in C' are satisfied.

Construction of X_j and C_j from clause c_j

1.) For $|c_j| = 1$ with $c_j = (l_j)$:

- $X_j = \{y_j^1, y_j^2\}$,
- $C_j = \{(l_j \vee y_j^1 \vee y_j^2), (l_j \vee \neg y_j^1 \vee \neg y_j^2), (l_j \vee \neg y_j^1 \vee y_j^2), (l_j \vee y_j^1 \vee \neg y_j^2)\}$

2.) For $|c_j| = 2$ with $c_j = (l_j^1 \vee l_j^2)$:

- $X_j = \{y_j\}$, $C_j = \{(l_j^1 \vee l_j^2 \vee y_j), (l_j^1 \vee l_j^2 \vee \neg y_j)\}$

3.) For $|c_j| = 3$: $X_j = \emptyset$, $C_j = \{c_j\}$

4.) For $|c_j| > 3$ with $c_j = (l_j^1 \vee l_j^2 \vee \ldots \vee l_j^k)$

- $X_j = \{y_j^1, y_j^2, \ldots, y_j^{k-3}\}$
- $C_j = \{(l_j^1 \vee l_j^2 \vee y_j^1), (\neg y_j^1 \vee l_j^3 \vee y_j^2), (\neg y_j^2 \vee l_j^4 \vee y_j^3), \ldots, (\neg y_j^{k-3} \vee l_j^{k-1} \vee l_j^k)$

To show: 3. If there is a satisfying truth assignment for (X, C), then there is a satisfying truth assignment for $(X', C') = (X \cup \bigcup_{j=1}^{|C|} X_j, \bigcup_{j=1}^{|C|} C_j)$:

Define truth assignment t' as follows:

- $t'(x) := t(x)\ \forall x \in X$
- $t'(x) := \text{true}\ \forall x \in X_j$ with $|c_j| \in \{1, 2\}$
- For c_j with $|c_j| > 3$ let l_j^i be the first literal with $t(l_j^i) = \text{true}$: If $i \in \{1, 2\}$ we set $t'(y_j^i) := \text{false}\ \forall i$
  If $i \in \{k-1, k\}$ we set $t'(y_j^i) := \text{true}\ \forall i < k-3$
  If $i \in \{3, 4, \ldots, k-2\}$, set $t'(y_j^l) := \text{true}\ \forall\ l \leq i-2$ & $t'(y_j^l) = \text{true}\ \forall\ l \geq i-2$

Then all clauses in C' are satisfied. ✓

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof:

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

Idea: Let (X, C) with $X = x_1, \ldots, x_n$ and $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of SAT.

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

Idea: Let (X, C) with $X = x_1, \ldots, x_n$ and $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of SAT.
Define matrix $A \in \{-1, 0, 1\}^{m \times n}$ to encode the clauses, and b to check the truth assignment. Use decision variabless $y_j \in \{0, 1\}$ to indicate if $t(x_j) = \mathit{true}$.

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

Idea: Let (X, C) with $X = x_1, \ldots, x_n$ and $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of SAT.
Define matrix $A \in \{-1, 0, 1\}^{m \times n}$ to encode the clauses, and b to check the truth assignment. Use decision variabless $y_j \in \{0, 1\}$ to indicate if $t(x_j) = \mathit{true}$.
Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases}$$

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

Idea: Let (X, C) with $X = x_1, \ldots, x_n$ and $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of SAT.
Define matrix $A \in \{-1, 0, 1\}^{m \times n}$ to encode the clauses, and b to check the truth assignment. Use decision variabless $y_j \in \{0, 1\}$ to indicate if $t(x_j) = \mathit{true}$.
Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ \end{cases}$$

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

Idea: Let (X, C) with $X = x_1, \ldots, x_n$ and $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of SAT.
Define matrix $A \in \{-1, 0, 1\}^{m \times n}$ to encode the clauses, and b to check the truth assignment. Use decision variabless $y_j \in \{0, 1\}$ to indicate if $t(x_j) = \mathit{true}$.
Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ \end{cases}$$

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

Idea: Let (X, C) with $X = x_1, \ldots, x_n$ and $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of SAT.
Define matrix $A \in \{-1, 0, 1\}^{m \times n}$ to encode the clauses, and b to check the truth assignment. Use decision variabless $y_j \in \{0, 1\}$ to indicate if $t(x_j) = \mathit{true}$.
Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

Idea: Let (X, C) with $X = x_1, \ldots, x_n$ and $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of SAT.
Define matrix $A \in \{-1, 0, 1\}^{m \times n}$ to encode the clauses, and b to check the truth assignment. Use decision variabless $y_j \in \{0, 1\}$ to indicate if $t(x_j) = \mathit{true}$.
Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#\text{negated literals in } c_i$ -1.

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

Idea: Let (X, C) with $X = x_1, \ldots, x_n$ and $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of SAT.
Define matrix $A \in \{-1, 0, 1\}^{m \times n}$ to encode the clauses, and b to check the truth assignment. Use decision variabless $y_j \in \{0, 1\}$ to indicate if $t(x_j) = \mathit{true}$.
Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#\text{negated literals in } c_i$ -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

Idea: Let (X, C) with $X = x_1, \ldots, x_n$ and $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of SAT.
Define matrix $A \in \{-1, 0, 1\}^{m \times n}$ to encode the clauses, and b to check the truth assignment. Use decision variabless $y_j \in \{0, 1\}$ to indicate if $t(x_j) = \mathit{true}$.
Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#\text{negated literals in } c_i$ -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.
To show:
This transformation is polynomial ✓

NP-hardness: Integer programming

Integer programming (ILP) - Decision version

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

ILP is NP-hard (and NP-complete).

Proof: by reduction from SAT.

Idea: Let (X, C) with $X = x_1, \ldots, x_n$ and $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of SAT.
Define matrix $A \in \{-1, 0, 1\}^{m \times n}$ to encode the clauses, and b to check the truth assignment. Use decision variabless $y_j \in \{0, 1\}$ to indicate if $t(x_j) = \mathit{true}$.
Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#\text{negated literals in } c_i$ -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.
To show:
This transformation is polynomial ✓
There is a satisfying truth assignment for $(X, C) \Leftrightarrow$ there is a vector y fulfilling $Ay \leq b$

Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#\text{negated literals in } c_i$ -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

To show: There is a satisfying truth assignment for $(X, C) \Rightarrow$ there is a vector y fulfilling $Ay \leq b$

Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#\text{negated literals in } c_i$ -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

To show: There is a satisfying truth assignment for $(X, C) \Rightarrow$ there is a vector y fulfilling $Ay \leq b$

Let t be a satisfying truth assignment for (X, C).

Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#\text{negated literals in } c_i$ -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

To show: There is a satisfying truth assignment for $(X, C) \Rightarrow$ there is a vector y fulfilling $Ay \leq b$

Let t be a satisfying truth assignment for (X, C). We set $y_j := \begin{cases} 1 & \text{if } t(x_j) = \text{true} \\ 0 & \text{otherwise} \end{cases}$

Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#\text{negated literals in } c_i$ -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

To show: There is a satisfying truth assignment for $(X, C) \Rightarrow$ there is a vector y fulfilling $Ay \leq b$

Let t be a satisfying truth assignment for (X, C). We set $y_j := \begin{cases} 1 & \text{if } t(x_j) = \text{true} \\ 0 & \text{otherwise} \end{cases}$

Then for each $c_i \in C$ we have

$$(a_{i\cdot}) \cdot y = \sum_{j=\mathbf{1}}^{n} a_{ij} y_j = \underbrace{\sum_{x_j \text{ occurs in } c_i} -y_j}_{\leq \mathbf{0}} + \underbrace{\sum_{\neg x_j \text{ occurs in } c_i} y_j}_{\leq |\{j : \neg x_j \text{ occurs in } c_i\}|} \leq |\{j : \neg x_j \text{ occurs in } c_i\}| - \mathbf{1}$$

Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#\text{negated literals in } c_i$ -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

To show: There is a satisfying truth assignment for $(X, C) \Rightarrow$ there is a vector y fulfilling $Ay \leq b$

Let t be a satisfying truth assignment for (X, C). We set $y_j := \begin{cases} 1 & \text{if } t(x_j) = \text{true} \\ 0 & \text{otherwise} \end{cases}$

Then for each $c_i \in C$ we have

$$(a_{i\cdot}) \cdot y = \sum_{j=\mathbf{1}}^{n} a_{ij} y_j = \underbrace{\sum_{x_j \text{ occurs in } c_i} -y_j}_{\leq \mathbf{0}} + \underbrace{\sum_{\neg x_j \text{ occurs in } c_i} y_j}_{\leq |\{j : \neg x_j \text{ occurs in } c_i\}|} \leq |\{j : \neg x_j \text{ occurs in } c_i\}| - \mathbf{1}$$

✓

Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#\text{negated literals in } c_i$ -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

To show: There is a satisfying truth assignment for $(X, C) \Leftarrow$ there is a vector y fulfilling $Ay \leq b$

Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#$negated literals in c_i -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

To show: There is a satisfying truth assignment for $(X, C) \Leftarrow$ there is a vector y fulfilling $Ay \leq b$
Let y be a vector with $Ay \leq b$.
Set $t(x_j) = \begin{cases} \text{true} & \text{if } y_j = 1 \\ \text{false} & \text{otherwise.} \end{cases}$

Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#$negated literals in c_i -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

To show: There is a satisfying truth assignment for $(X, C) \Leftarrow$ there is a vector y fulfilling $Ay \leq b$
Let y be a vector with $Ay \leq b$.
Set $t(x_j) = \begin{cases} \text{true} & \text{if } y_j = 1 \\ \text{false} & \text{otherwise.} \end{cases}$
Then for $c_i = l_1 \vee l_2 \vee \ldots \vee l_k$ we have

$$\sum_{x_j \text{ occurs in } c_i} -y_j + \sum_{\neg x_j \text{ occurs in } c_i} y_j |\{j : \neg x_j \text{ occurs in } c_i\}| - 1$$

Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#$negated literals in c_i -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

To show: There is a satisfying truth assignment for $(X, C) \Leftarrow$ there is a vector y fulfilling $Ay \leq b$
Let y be a vector with $Ay \leq b$.
Set $t(x_j) = \begin{cases} \text{true} & \text{if } y_j = 1 \\ \text{false} & \text{otherwise.} \end{cases}$
Then for $c_i = l_1 \vee l_2 \vee \ldots \vee l_k$ we have

$$\sum_{x_j \text{ occurs in } c_i} -y_j + \sum_{\neg x_j \text{ occurs in } c_i} y_j |\{j : \neg x_j \text{ occurs in } c_i\}| - 1$$

$\Rightarrow$ There is a j with x_j in c_i and $y_i = 1$, or there is a j with $\neg x_j$ in c_i and y_j=0

Polynomial transformation:
Let (X, C) be an instance of SAT.
We define matrix A as

$$a_{ij} = \begin{cases} -1 & \text{if } x_j \text{ is in } c_i \\ 1 & \text{if } \neg x_j \text{ is in } c_i \\ 0 & x_j \text{ is not in } c_i, \end{cases}$$

and $b_i = \#$negated literals in c_i -1.
encode $y_j \in \{0, 1\}$ as $0 \leq y_j \leq 1$.

To show: There is a satisfying truth assignment for $(X, C) \Leftarrow$ there is a vector y fulfilling $Ay \leq b$
Let y be a vector with $Ay \leq b$.
Set $t(x_j) = \begin{cases} \text{true} & \text{if } y_j = 1 \\ \text{false} & \text{otherwise.} \end{cases}$
Then for $c_i = l_1 \vee l_2 \vee \ldots \vee l_k$ we have

$$\sum_{x_j \text{ occurs in } c_i} -y_j + \sum_{\neg x_j \text{ occurs in } c_i} y_j |\{j : \neg x_j \text{ occurs in } c_i\}| - 1$$

$\Rightarrow$ There is a j with x_j in c_i and $y_i = 1$, or there is a j with $\neg x_j$ in c_i and y_j=0
$\Rightarrow$ c_i is satisfied ✓

NP-hardness: Integer programming

Integer programming (ILP)

Instance: an integer matrix A; an integer vector b
Question: does there exist an integer vector y with $Ay \leq b$?

Theorem

SAT $\leq_p$ ILP, and therefore ILP is NP-hard (and NP-complete).

Consequence: Every problem in NP can be modelled as an ILP.

NP-hardness: Clique

Clique

Instance: a graph $G = (V, E)$; an integer k
Question: does G contain a clique of size (at least) k?

Theorem

CLIQUE is NP-hard (and NP-complete).

Proof: SAT is NP-hard and SAT $\leq_p$ CLIQUE.

NP-hardness: Independent set

Independent set (IS)

Instance: a graph $G = (V, E)$; an integer k
Question: does G contain an independent set of size (at least) k?
(a set of vertices that does not span any edge)

Theorem

IS is NP-hard (and NP-complete).

Proof:

NP-hardness: Independent set

Independent set (IS)

Instance: a graph $G = (V, E)$; an integer k
Question: does G contain an independent set of size (at least) k?
(a set of vertices that does not span any edge)

Theorem

IS is NP-hard (and NP-complete).

Proof: By reduction from CLIQUE:

NP-hardness: Independent set

Independent set (IS)

Instance: a graph $G = (V, E)$; an integer k
Question: does G contain an independent set of size (at least) k?
(a set of vertices that does not span any edge)

Theorem

IS is NP-hard (and NP-complete).

Proof: By reduction from CLIQUE:
Given an instance $(G = (V, E), k)$ of clique, construct the following instance of IS:
$V' := V$, $E' := \{\{i, j\} : i \neq j \in V, \{i, j\} \notin E\}$, $k' := k$.

NP-hardness: Independent set

Independent set (IS)

Instance: a graph $G = (V, E)$; an integer k
Question: does G contain an independent set of size (at least) k?
(a set of vertices that does not span any edge)

Theorem

IS is NP-hard (and NP-complete).

Proof: By reduction from CLIQUE:
Given an instance $(G = (V, E), k)$ of clique, construct the following instance of IS:
$V' := V$, $E' := \{\{i, j\} : i \neq j \in V, \{i, j\} \notin E\}$, $k' := k$.
Show:
$X \subset V$ is a clique in $G \Leftrightarrow X$ is an independent set in G'

NP-hardness: Exact cover

Exact cover (Ex-Cov)

Instance: a ground set X; subsets $S_1, \ldots, S_m$ of X

Question: do there exist some subsets S_i that form a partition of X?

Theorem

(Ex-Cov) is NP-hard (and NP-complete).

Proof:

NP-hardness: Exact cover

Exact cover (Ex-Cov)

Instance: a ground set X; subsets $S_1, \ldots, S_m$ of X

Question: do there exist some subsets S_i that form a partition of X?

Theorem

(Ex-Cov) is NP-hard (and NP-complete).

Proof: by reduction from IS.

NP-hardness: Exact cover

Exact cover (Ex-Cov)

Instance: a ground set X; subsets $S_1, \ldots, S_m$ of X

Question: do there exist some subsets S_i that form a partition of X?

Theorem

(Ex-Cov) is NP-hard (and NP-complete).

Proof: by reduction from IS.
Let (G, k) with $G = (V, E)$ be an instance of IS.

NP-hardness: Exact cover

Exact cover (Ex-Cov)

Instance: a ground set X; subsets $S_1, \ldots, S_m$ of X

Question: do there exist some subsets S_i that form a partition of X?

Theorem

(Ex-Cov) is NP-hard (and NP-complete).

Proof: by reduction from IS.
Let (G, k) with $G = (V, E)$ be an instance of IS.
Define an instance of (Ex-Cov) as follows: $X := E \cup \{1, \ldots, k\}$
and subsets
$S_{ih} := \{\{i, j\} : \{i, j\} \in E\} \cup \{h\}$ for $i \in V$, $h = 1, \ldots k$
$S_{\{i,j\}} := \{\{i, j\}\}$ for $\{i, j\} \in E$

NP-hardness: Exact cover

Exact cover (Ex-Cov)

Instance: a ground set X; subsets $S_1, \ldots, S_m$ of X

Question: do there exist some subsets S_i that form a partition of X?

Theorem

(Ex-Cov) is NP-hard (and NP-complete).

Proof: by reduction from IS.
Let (G, k) with $G = (V, E)$ be an instance of IS.
Define an instance of (Ex-Cov) as follows: $X := E \cup \{1, \ldots, k\}$
and subsets
$S_{ih} := \{\{i, j\} : \{i, j\} \in E\} \cup \{h\}$ for $i \in V$, $h = 1, \ldots k$
$S_{\{i,j\}} := \{\{i, j\}\}$ for $\{i, j\} \in E$
Show:
Polynomial transformation ✓
If S is a solution to (Ex-Cov), $i : S_{ih} \in S$ is an independent set of size k.
If $X = \{x_1, x_2, \ldots, x_k\} \subset V$ is an independent set, $\bigcup_{j=1}^{k} S_{x_j j} \cup \{\{i, j\} : i, j \notin X\}$ is a solution to (Ex-Cov).

NP-hardness: Exact cover

Let (G, k) with $G = (V, E)$ be an instance of IS.
Define an instance of (Ex-Cov) as follows: $X := E \cup \{1, \ldots, k\}$
and subsets
$S_{ih} := \{\{i, j\} : \{i, j\} \in E\} \cup \{h\}$ for $i \in V$, $h = 1, \ldots k$
$S_{\{i,j\}} := \{\{i, j\}\}$ for $\{i, j\} \in E$

Show:If S is a solution to (Ex-Cov), $i : S_{ih} \in S$ is an independent set of size k:

NP-hardness: Exact cover

Let (G, k) with $G = (V, E)$ be an instance of IS.
Define an instance of (Ex-Cov) as follows: $X := E \cup \{1, \ldots, k\}$
and subsets
$S_{ih} := \{\{i, j\} : \{i, j\} \in E\} \cup \{h\}$ for $i \in V$, $h = 1, \ldots k$
$S_{\{i,j\}} := \{\{i, j\}\}$ for $\{i, j\} \in E$

Show:If S is a solution to (Ex-Cov), $i : S_{ih} \in S$ is an independent set of size k:

For each $h = 1, \ldots, k$, there needs to be a $i(h)$ such that $S_{i(h)h)}$ is in the the partition.

NP-hardness: Exact cover

Let (G, k) with $G = (V, E)$ be an instance of IS.
Define an instance of (Ex-Cov) as follows: $X := E \cup \{1, \ldots, k\}$
and subsets
$S_{ih} := \{\{i, j\} : \{i, j\} \in E\} \cup \{h\}$ for $i \in V$, $h = 1, \ldots k$
$S_{\{i,j\}} := \{\{i, j\}\}$ for $\{i, j\} \in E$

Show:If S is a solution to (Ex-Cov), $i : S_{ih} \in S$ is an independent set of size k:

For each $h = 1, \ldots, k$, there needs to be a $i(h)$ such that $S_{i(h)h)}$ is in the the partition.
For any two $h \neq h'$, $i(h) \neq i(h')$ and vertices $i(h)$ and $i(h')$ are not adjacent.

NP-hardness: Exact cover

Let (G, k) with $G = (V, E)$ be an instance of IS.
Define an instance of (Ex-Cov) as follows: $X := E \cup \{1, \ldots, k\}$
and subsets
$S_{ih} := \{\{i, j\} : \{i, j\} \in E\} \cup \{h\}$ for $i \in V$, $h = 1, \ldots k$
$S_{\{i,j\}} := \{\{i, j\}\}$ for $\{i, j\} \in E$

Show:If S is a solution to (Ex-Cov), $i : S_{ih} \in S$ is an independent set of size k:

For each $h = 1, \ldots, k$, there needs to be a $i(h)$ such that $S_{i(h)h)}$ is in the the partition.
For any two $h \neq h'$, $i(h) \neq i(h')$ and vertices $i(h)$ and $i(h')$ are not adjacent.
$\Rightarrow \{i(h) : h = 1, \ldots, k\}$ is an independent set of size k in G. ✓

NP-hardness: Exact cover

Let (G, k) with $G = (V, E)$ be an instance of IS.
Define an instance of (Ex-Cov) as follows: $X := E \cup \{1, \ldots, k\}$ and subsets
$S_{ih} := \{\{i, j\} : \{i, j\} \in E\} \cup \{h\}$ for $i \in V$, $h = 1, \ldots k$
$S_{\{i,j\}} := \{\{i, j\}\}$ for $\{i, j\} \in E$
Show: If $X = \{x_1, x_2, \ldots, x_k\} \subset V$ is an independent set,
$\bigcup_{j=1}^{k} S_{x_j j} \cup \{\{i, j\} : i, j \notin X\}$ is a solution to (Ex-Cov).

NP-hardness: Subset Sum

Subset Sum (SS)

Instance: positive integers $a_1, \ldots, a_n$; a bound b

Question: does there exist an index set $J \subseteq \{1, \ldots, n\}$ with $\sum_{j \in J} a_j = b$?

Theorem

SS is NP-hard (and NP-complete).

Proof:

NP-hardness: Subset Sum

Subset Sum (SS)

Instance: positive integers $a_1, \ldots, a_n$; a bound b

Question: does there exist an index set $J \subseteq \{1, \ldots, n\}$ with $\sum_{j \in J} a_j = b$?

Theorem

SS is NP-hard (and NP-complete).

Proof: by reduction from Ex-Cov.

NP-hardness: Subset Sum

Subset Sum (SS)

Instance: positive integers $a_1, \ldots, a_n$; a bound b

Question: does there exist an index set $J \subseteq \{1, \ldots, n\}$ with $\sum_{j \in J} a_j = b$?

Theorem

SS is NP-hard (and NP-complete).

Proof: by reduction from Ex-Cov.
Let $(X = \{x_1, \ldots, x_m\}, \{S_1, \ldots, S_n\})$ be an instance of Ex-Cov.
Define numbers a_j as $a_j := \sum_{i=1}^{m} c_{ij} \cdot d_i$ with $c_{ij} = 1$ if $x_i \in S_j$ and $d_i = (n+1)^{i-1}$.
Set $b := \sum_{i=1}^{m} (n+1)^{i-1}$.

NP-hardness: Subset Sum

Subset Sum (SS)

Instance: positive integers $a_1, \ldots, a_n$; a bound b

Question: does there exist an index set $J \subseteq \{1, \ldots, n\}$ with $\sum_{j \in J} a_j = b$?

Theorem

SS is NP-hard (and NP-complete).

Proof: by reduction from Ex-Cov.
Let $(X = \{x_1, \ldots, x_m\}, \{S_1, \ldots, S_n\})$ be an instance of Ex-Cov.
Define numbers a_j as $a_j := \sum_{i=1}^{m} c_{ij} \cdot d_i$ with $c_{ij} = 1$ if $x_i \in S_j$ and $d_i = (n+1)^{i-1}$.
Set $b := \sum_{i=1}^{m} (n+1)^{i-1}$.

Show: 1.) This is a polynomial transformation.
2. & 3.) J is the index set of a solution to Ex-Cov $\Leftrightarrow$ J is the index set of a solution to SS.

Reduction of Ex-Cov to SS

Let $(X = \{x_1, \ldots, x_m\}, \{S_1, \ldots, S_n\})$ be an instance of Ex-Cov.
Define numbers a_j as $a_j := \sum_{i=1}^{m} c_{ij} \cdot d_i$ with $c_{ij} = 1$ if $x_i \in S_j$ and $d_i = (n+1)^{i-1}$.
Set $b := \sum_{i=1}^{m} (n+1)^{i-1}$.

Reduction of Ex-Cov to SS

Let $(X = \{x_1, \ldots, x_m\}, \{S_1, \ldots, S_n\})$ be an instance of Ex-Cov.
Define numbers a_j as $a_j := \sum_{i=1}^{m} c_{ij} \cdot d_i$ with $c_{ij} = 1$ if $x_i \in S_j$ and $d_i = (n+1)^{i-1}$.
Set $b := \sum_{i=1}^{m} (n+1)^{i-1}$.

Representation of Ex-Cov instance in table:

	x_1	x_2	$\ldots$	x_m	
S_1	1	0	$\ldots$	1	
S_2	1	1	$\ldots$	1	
$\ldots$					
S_n	0	1	$\ldots$	0	

✓

Reduction of Ex-Cov to SS

Let $(X = \{x_1, \ldots, x_m\}, \{S_1, \ldots, S_n\})$ be an instance of Ex-Cov.
Define numbers a_j as $a_j := \sum_{i=1}^{m} c_{ij} \cdot d_i$ with $c_{ij} = 1$ if $x_i \in S_j$ and $d_i = (n+1)^{i-1}$.
Set $b := \sum_{i=1}^{m} (n+1)^{i-1}$.

Representation of Ex-Cov instance in table:

	x_1	x_2	$\ldots$	x_m	
S_1	1	0	$\ldots$	1	
S_2	1	1	$\ldots$	1	
$\ldots$					
S_n	0	1	$\ldots$	0	

We have subsets $\{S_i :\in I\}$ that form a partition (thus: a YES-instance)

$\Leftrightarrow$ we can choose subset of rows (with indices in index set I) such that there is exactly one entry '1' in each column.

✓

Reduction of Ex-Cov to SS

Let $(X = \{x_1, \ldots, x_m\}, \{S_1, \ldots, S_n\})$ be an instance of Ex-Cov.
Define numbers a_j as $a_j := \sum_{i=1}^{m} c_{ij} \cdot d_i$ with $c_{ij} = 1$ if $x_i \in S_j$ and $d_i = (n+1)^{i-1}$.
Set $b := \sum_{i=1}^{m} (n+1)^{i-1}$.

Representation of Ex-Cov instance in table:

	x_1	x_2	$\ldots$	x_m	
S_1	1	0	$\ldots$	1	
S_2	1	1	$\ldots$	1	
$\ldots$					
S_n	0	1	$\ldots$	0	
	$\cdot 1$	$\cdot (n+1)$	$\ldots$	$\cdot (n+1)^{m-1}$	

We have subsets $\{S_i :\in I\}$ that form a partition (thus: a YES-instance)

$\Leftrightarrow$ we can choose subset of rows (with indices in index set I) such that there is exactly one entry '1' in each column.

✓

Reduction of Ex-Cov to SS

Let $(X = \{x_1, \ldots, x_m\}, \{S_1, \ldots, S_n\})$ be an instance of Ex-Cov.
Define numbers a_j as $a_j := \sum_{i=1}^{m} c_{ij} \cdot d_i$ with $c_{ij} = 1$ if $x_i \in S_j$ and $d_i = (n+1)^{i-1}$.
Set $b := \sum_{i=1}^{m} (n+1)^{i-1}$.

Representation of Ex-Cov instance in table:

	x_1	x_2	$\ldots$	x_m	
S_1	1	0	$\ldots$	1	$a_1 = 1 \cdot 1 + 0 \cdot (n+1) + \ldots + 1 \cdot (n+1)^{m-1}$
S_2	1	1	$\ldots$	1	$a_2 = 1 \cdot 1 + 1 \cdot (n+1) + \ldots + 1 \cdot (n+1)^{m-1}$
$\ldots$					
S_n	0	1	$\ldots$	0	$a_n = 0 \cdot 1 + 1 \cdot (n+1) + \ldots + 0 \cdot (n+1)^{m-1}$
	$\cdot 1$	$\cdot (n+1)$	$\ldots$	$\cdot (n+1)^{m-1}$	

We have subsets $\{S_i :\in I\}$ that form a partition (thus: a YES-instance)

$\Leftrightarrow$ we can choose subset of rows (with indices in index set I) such that there is exactly one entry '1' in each column.

✓

Reduction of Ex-Cov to SS

Let $(X = \{x_1, \ldots, x_m\}, \{S_1, \ldots, S_n\})$ be an instance of Ex-Cov.
Define numbers a_j as $a_j := \sum_{i=1}^{m} c_{ij} \cdot d_i$ with $c_{ij} = 1$ if $x_i \in S_j$ and $d_i = (n+1)^{i-1}$.
Set $b := \sum_{i=1}^{m} (n+1)^{i-1}$.

Representation of Ex-Cov instance in table:

	x_1	x_2	$\ldots$	x_m	
S_1	1	0	$\ldots$	1	$a_1 = 1 \cdot 1 + 0 \cdot (n+1) + \ldots + 1 \cdot (n+1)^{m-1}$
S_2	1	1	$\ldots$	1	$a_2 = 1 \cdot 1 + 1 \cdot (n+1) + \ldots + 1 \cdot (n+1)^{m-1}$
$\ldots$					
S_n	0	1	$\ldots$	0	$a_n = 0 \cdot 1 + 1 \cdot (n+1) + \ldots + 0 \cdot (n+1)^{m-1}$
	$\cdot 1$	$\cdot (n+1)$	$\ldots$	$\cdot (n+1)^{m-1}$	

We have subsets $\{S_i : \in I\}$ that form a partition (thus: a YES-instance)

$\Leftrightarrow$ we can choose subset of rows (with indices in index set I) such that there is exactly one entry '1' in each column.

$\Leftrightarrow$ we can choose a subset of numbers (with indices in index set I) such that $\sum_{i \in I} a_i = \sum_{i=1}^{m} (n+1)^{i-1} = b$
✓

NP-hardness: 2-Partition

2-PARTITION

Instance: positive integers $a_1, \ldots, a_n$ with $\sum_{i=1}^{n} a_i = 2A$.

Question: does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = A$?

Theorem

2-PARTITION is NP-hard (and thus NP-complete).

Proof:

NP-hardness: 2-Partition

2-PARTITION

Instance: positive integers $a_1, \ldots, a_n$ with $\sum_{i=1}^{n} a_i = 2A$.

Question: does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = A$?

Theorem

2-PARTITION is NP-hard (and thus NP-complete).

Proof: by reduction from SS.

NP-hardness: 2-Partition

2-PARTITION

Instance: positive integers $a_1, \ldots, a_n$ with $\sum_{i=1}^{n} a_i = 2A$.

Question: does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = A$?

Theorem

2-PARTITION is NP-hard (and thus NP-complete).

Proof: by reduction from SS.
Given an instance $(\{a_1, \ldots, a_n\}, b)$ of SS we construct an instance of 2PARTITION:

NP-hardness: 2-Partition

2-PARTITION

Instance: positive integers $a_1, \ldots, a_n$ with $\sum_{i=1}^{n} a_i = 2A$.

Question: does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = A$?

Theorem

2-PARTITION is NP-hard (and thus NP-complete).

Proof: by reduction from SS.
Given an instance $(\{a_1, \ldots, a_n\}, b)$ of SS we construct an instance of 2PARTITION:
we define $S := \sum_{i=1}^{n} a_i$
and consider the instance $\{a_1, \ldots, a_n, \underbrace{S + b}_{=:a_{n+1}}, \underbrace{2S - b}_{=:a_{n+2}}\}$.

NP-hardness: 2-Partition

2-PARTITION

Instance: positive integers $a_1, \ldots, a_n$ with $\sum_{i=1}^{n} a_i = 2A$.

Question: does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = A$?

Theorem

2-PARTITION is NP-hard (and thus NP-complete).

Proof: by reduction from SS.
Given an instance $(\{a_1, \ldots, a_n\}, b)$ of SS we construct an instance of 2PARTITION:
we define $S := \sum_{i=1}^{n} a_i$
and consider the instance $\{a_1, \ldots, a_n, \underbrace{S + b}_{=:a_{n+1}}, \underbrace{2S - b}_{=:a_{n+2}}\}$.

Note that $\sum_{i=1}^{n+2} a_i = a_1 + a_2 + \ldots, a_n + S + b + 2S - b = 4S$.

NP-hardness: 2-Partition

2-PARTITION

Instance: positive integers $a_1, \ldots, a_n$ with $\sum_{i=1}^{n} a_i = 2A$.

Question: does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = A$?

Theorem

2-PARTITION is NP-hard (and thus NP-complete).

Proof: by reduction from SS.
Given an instance $(\{a_1, \ldots, a_n\}, b)$ of SS we construct an instance of 2PARTITION:
we define $S := \sum_{i=1}^{n} a_i$
and consider the instance $\{a_1, \ldots, a_n, \underbrace{S + b}_{=:a_{n+1}}, \underbrace{2S - b}_{=:a_{n+2}}\}$.

Note that $\sum_{i=1}^{n+2} a_i = a_1 + a_2 + \ldots, a_n + S + b + 2S - b = 4S$.
The question (of the built 2-PARTITION instance) is thus: Is there a an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I'} a_i = 2S$?

NP-hardness: 2-Partition

2-PARTITION

Instance: positive integers $a_1, \ldots, a_n$ with $\sum_{i=1}^{n} a_i = 2A$.

Question: does there exist an index set $I \subseteq \{1, \ldots, n\}$ with $\sum_{i \in I} a_i = A$?

Theorem

2-PARTITION is NP-hard (and thus NP-complete).

Proof: by reduction from SS.
Given an instance $(\{a_1, \ldots, a_n\}, b)$ of SS we construct an instance of 2PARTITION:
we define $S := \sum_{i=1}^{n} a_i$
and consider the instance $\{a_1, \ldots, a_n, \underbrace{S + b}_{=:a_{n+1}}, \underbrace{2S - b}_{=:a_{n+2}}\}$.

Note that $\sum_{i=1}^{n+2} a_i = a_1 + a_2 + \ldots, a_n + S + b + 2S - b = 4S$.
The question (of the built 2-PARTITION instance) is thus: Is there a an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I'} a_i = 2S$?

We show: There is an index set $I \subset \{1, 2, \ldots, n\}$ with $\sum_{i \in I} a_i = b$
$\Leftrightarrow$ There is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$

NP-hardness: 2-Partition

We show: There is an index set $I \subset \{1, 2, \ldots, n\}$ with $\sum_{i \in I} a_i = b$
$\Leftrightarrow$ There is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$

$\Rightarrow$
Assume that there is an index set $I \subset \{a_1, a_2, \ldots, a_n\}$ such that $\sum_{i \in I} a_i = b$.

NP-hardness: 2-Partition

We show: There is an index set $I \subset \{1, 2, \ldots, n\}$ with $\sum_{i \in I} a_i = b$
$\Leftrightarrow$ There is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$

$\Rightarrow$
Assume that there is an index set $I \subset \{a_1, a_2, \ldots, a_n\}$ such that $\sum_{i \in I} a_i = b$.
Then for $I' := I \cup \{n+2\}$ we have $\sum_{i \in I'} a_i =$

NP-hardness: 2-Partition

We show: There is an index set $I \subset \{1, 2, \ldots, n\}$ with $\sum_{i \in I} a_i = b$
$\Leftrightarrow$ There is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$

$\Rightarrow$
Assume that there is an index set $I \subset \{a_1, a_2, \ldots, a_n\}$ such that $\sum_{i \in I} a_i = b$.
Then for $I' := I \cup \{n+2\}$ we have $\sum_{i \in I'} a_i = \sum_{i \in I} a_i + 2S - b = b + 2S - b = 2S$.
✓

NP-hardness: 2-Partition

We show: There is an index set $I \subset \{1, 2, \ldots, n\}$ with $\sum_{i \in I} a_i = b$
$\Leftrightarrow$ There is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$

$\Rightarrow$
Assume that there is an index set $I \subset \{a_1, a_2, \ldots, a_n\}$ such that $\sum_{i \in I} a_i = b$.
Then for $I' := I \cup \{n+2\}$ we have $\sum_{i \in I'} a_i = \sum_{i \in I} a_i + 2S - b = b + 2S - b = 2S$.
✓

$\Leftarrow$
Assume that there is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$:

NP-hardness: 2-Partition

We show: There is an index set $I \subset \{1, 2, \ldots, n\}$ with $\sum_{i \in I} a_i = b$
$\Leftrightarrow$ There is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$

$\Rightarrow$
Assume that there is an index set $I \subset \{a_1, a_2, \ldots, a_n\}$ such that $\sum_{i \in I} a_i = b$.
Then for $I' := I \cup \{n+2\}$ we have $\sum_{i \in I'} a_i = \sum_{i \in I} a_i + 2S - b = b + 2S - b = 2S$.
✓

$\Leftarrow$
Assume that there is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$:
This index set contains either $n+1$ or $n+2$, but not both, because
$a_{n+1} + a_{n+2} = S + b + 2S - b = 3S > 2S$. Let's (wlog) assume that $n+1 \in I'$.

NP-hardness: 2-Partition

We show: There is an index set $I \subset \{1, 2, \ldots, n\}$ with $\sum_{i \in I} a_i = b$
$\Leftrightarrow$ There is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$

$\Rightarrow$
Assume that there is an index set $I \subset \{a_1, a_2, \ldots, a_n\}$ such that $\sum_{i \in I} a_i = b$.
Then for $I' := I \cup \{n+2\}$ we have $\sum_{i \in I'} a_i = \sum_{i \in I} a_i + 2S - b = b + 2S - b = 2S$.
✓

$\Leftarrow$
Assume that there is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$:
This index set contains either $n+1$ or $n+2$, but not both, because
$a_{n+1} + a_{n+2} = S + b + 2S - b = 3S > 2S$. Let's (wlog) assume that $n+1 \in I'$.
Then we define $I := I' \setminus \{n+1\}$.

NP-hardness: 2-Partition

We show: There is an index set $I \subset \{1, 2, \ldots, n\}$ with $\sum_{i \in I} a_i = b$
$\Leftrightarrow$ There is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$

$\Rightarrow$
Assume that there is an index set $I \subset \{a_1, a_2, \ldots, a_n\}$ such that $\sum_{i \in I} a_i = b$.
Then for $I' := I \cup \{n+2\}$ we have $\sum_{i \in I'} a_i = \sum_{i \in I} a_i + 2S - b = b + 2S - b = 2S$.
✓

$\Leftarrow$
Assume that there is an index set $I' \subset \{1, 2, \ldots, n+2\}$ with $\sum_{i \in I} a_i = 2S$:
This index set contains either $n+1$ or $n+2$, but not both, because
$a_{n+1} + a_{n+2} = S + b + 2S - b = 3S > 2S$. Let's (wlog) assume that $n+1 \in I'$.
Then we define $I := I' \setminus \{n+1\}$. And have $\sum_{i \in I' \setminus \{n+1\}} a_i = 2S - (S - b) = b$ ✓

Vertex cover (VC)

Instance: a graph $G = (V, E)$; an integer k
Question: does G contain a vertex cover of size (at most) k?
(a set of vertices that touches every edge)

Theorem

VC is NP-hard (and thus NP-complete).

Proof:

Vertex cover (VC)

Instance: a graph $G = (V, E)$; an integer k
Question: does G contain a vertex cover of size (at most) k?
(a set of vertices that touches every edge)

Theorem

VC is NP-hard (and thus NP-complete).

Proof: by reduction from IS.
Let $(G = (V, E), l)$ be an instance of IS.

Vertex cover (VC)

Instance: a graph $G = (V, E)$; an integer k
Question: does G contain a vertex cover of size (at most) k?
(a set of vertices that touches every edge)

Theorem

VC is NP-hard (and thus NP-complete).

Proof: by reduction from IS.
Let $(G = (V, E), l)$ be an instance of IS.
We consider the instance $(G = (V, E), n - l)$ of Vertex Cover.

V^1 is an independent set of size l in $G \Leftrightarrow V^2 := V \setminus V^1$ is a vertex cover of size $n - l \in G$

NP-hardness: Hamiltonian cycle / TSP

Hamiltonian cycle (HC)

Instance: a graph (V, E)
Question: does this graph contain a Hamiltonian cycle?

Theorem

HC is NP-complete.

NP-hardness: Hamiltonian cycle / TSP

Hamiltonian cycle (HC)

Instance: a graph (V, E)
Question: does this graph contain a Hamiltonian cycle?

Theorem

HC is NP-complete.

Proof: Easy to see: in NP.
To show NP-hard: reduction from VC.

NP-hardness: Hamiltonian cycle / TSP

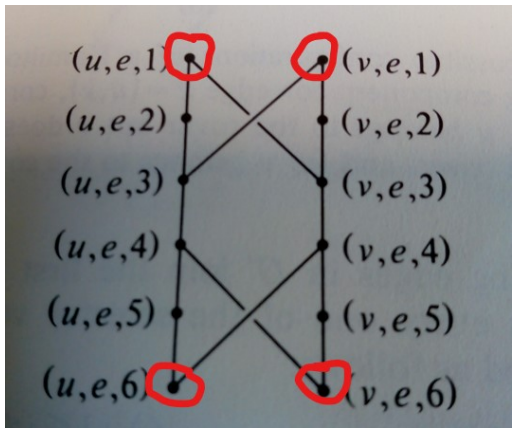
Given instance $G = (V, E), k$ of VC. Define $G' = (V', E')$:

NP-hardness: Hamiltonian cycle / TSP

Given instance $G = (V, E), k$ of VC. Define $G' = (V', E')$:
1.) 'selector vertices': $a_1, a_2, \ldots, a_k$

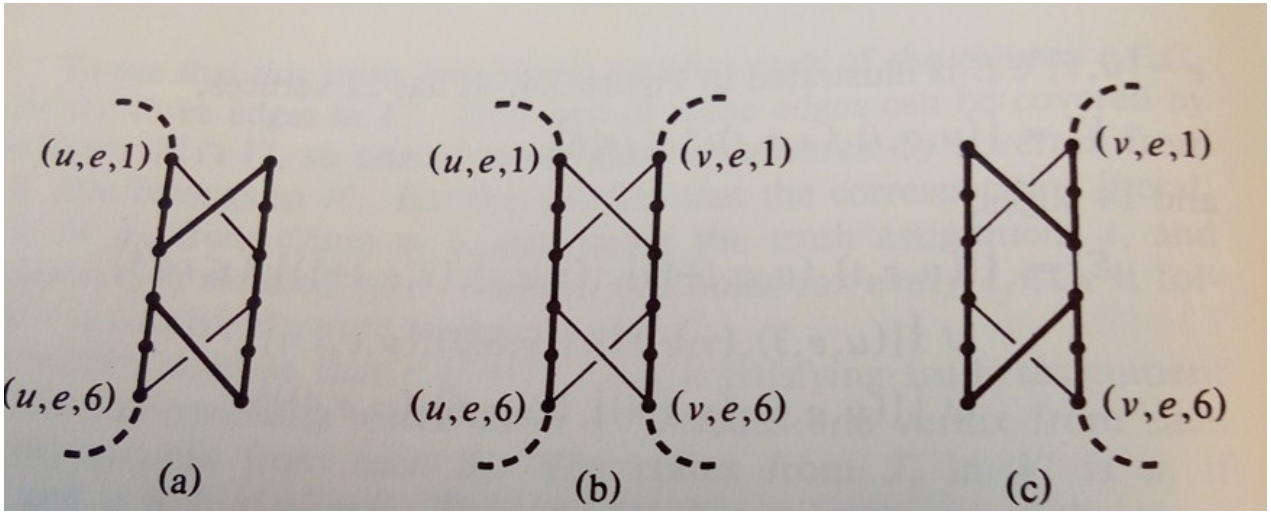
NP-hardness: Hamiltonian cycle / TSP

Given instance $G = (V, E), k$ of VC. Define $G' = (V', E')$:
1.) 'selector vertices': $a_1, a_2, \ldots, a_k$
2.) for each $e \in E$: 'cover-testing' component



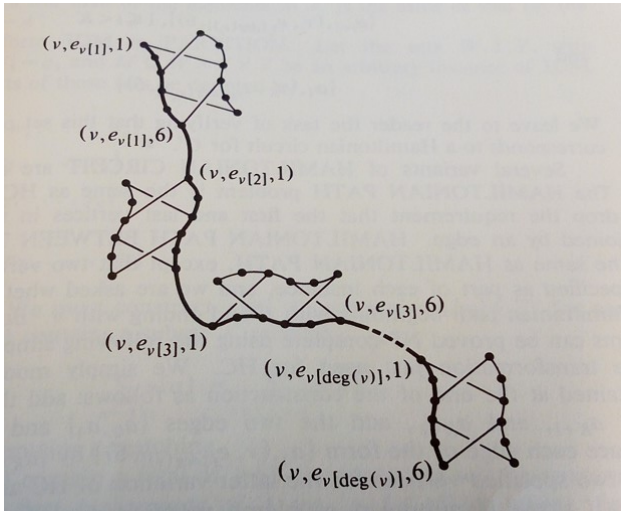
NP-hardness: Hamiltonian cycle / TSP

Given instance $G = (V, E), k$ of VC. Define $G' = (V', E')$:
1.) 'selector vertices': $a_1, a_2, \ldots, a_k$
2.) for each $e \in E$: 'cover-testing' component
Note: any Hamiltonian cycle needs to pass the edges in G' in one of the 3 shown configurations (or otherwise not all nodes would be visited exactly once):



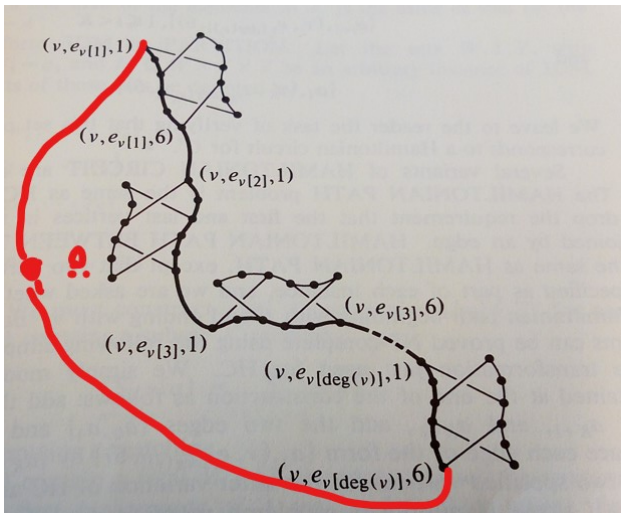
NP-hardness: Hamiltonian cycle / TSP

Given instance $G = (V, E), k$ of VC. Define $G' = (V', E')$:
1.) 'selector vertices': $a_1, a_2, \ldots, a_k$
2.) for each $e \in E$: 'cover-testing' component
Note: any Hamiltonian cycle needs to pass the edges in G' in one of the 3 shown configurations (or otherwise not all nodes would be visited exactly once):
3.) 'joining edges': for each node v denote by $v[1], v[2], \ldots, v[deg(v)]$ the incident edges of v:
Use $E'_v := \{(v, e_{v[i]}, 6), (v, e_{v[i+1]}) : 1 \leq i \leq deg(v)\}$ to build a path through the respective 'cover-testing components'



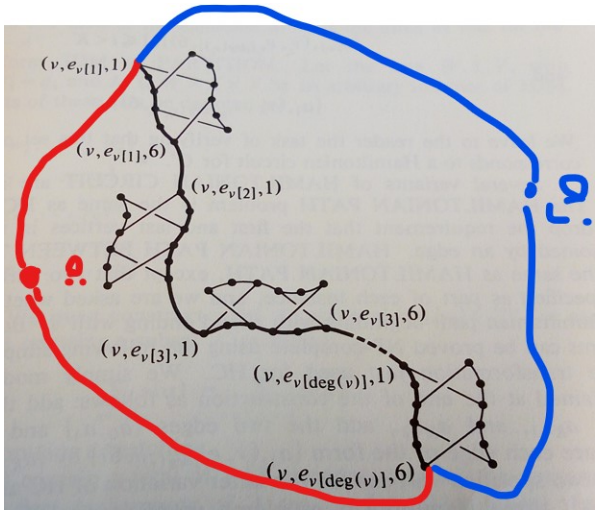
NP-hardness: Hamiltonian cycle / TSP

Given instance $G = (V, E), k$ of VC. Define $G' = (V', E')$:
1.) 'selector vertices': $a_1, a_2, \ldots, a_k$
2.) for each $e \in E$: 'cover-testing' component
Note: any Hamiltonian cycle needs to pass the edges in G' in one of the 3 shown configurations (or otherwise not all nodes would be visited exactly once):
3.) 'joining edges': for each node v denote by $v[1], v[2], \ldots, v[deg(v)]$ the incident edges of v:
Use $E'_v := \{(v, e_{v[i]}, 6), (v, e_{v[i+1]}) : 1 \leq i \leq deg(v)\}$ to build a path through the respective 'cover-testing components'
4.) more 'joining edges':
$E'' := \{(a_i, (v, e_{v[1]}, 1), (a_i, (v, e_{v[deg(v)]}, 6) : 1 \leq i \leq k, v \in V\}$



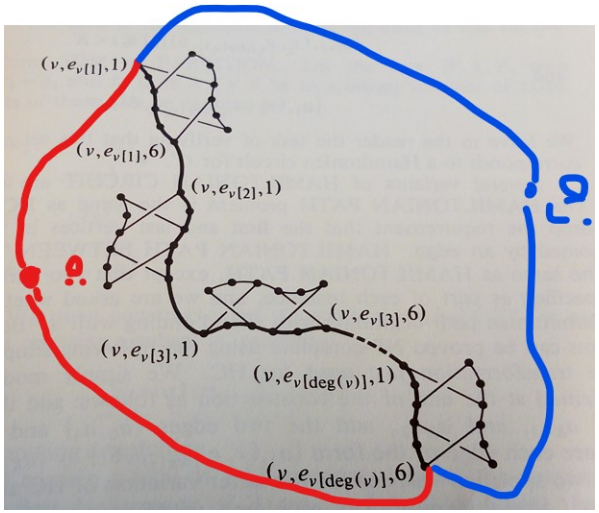
NP-hardness: Hamiltonian cycle / TSP

Given instance $G = (V, E), k$ of VC. Define $G' = (V', E')$:
1.) 'selector vertices': $a_1, a_2, \ldots, a_k$
2.) for each $e \in E$: 'cover-testing' component
Note: any Hamiltonian cycle needs to pass the edges in G' in one of the 3 shown configurations (or otherwise not all nodes would be visited exactly once):
3.) 'joining edges': for each node v denote by $v[1], v[2], \ldots, v[deg(v)]$ the incident edges of v:
Use $E'_v := \{(v, e_{v[i]}, 6), (v, e_{v[i+1]}) : 1 \leq i \leq deg(v)\}$ to build a path through the respective 'cover-testing components'
4.) more 'joining edges':
$E'' := \{(a_i, (v, e_{v[1]}, 1), (a_i, (v, e_{v[deg(v)]}, 6) : 1 \leq i \leq k, v \in V\}$



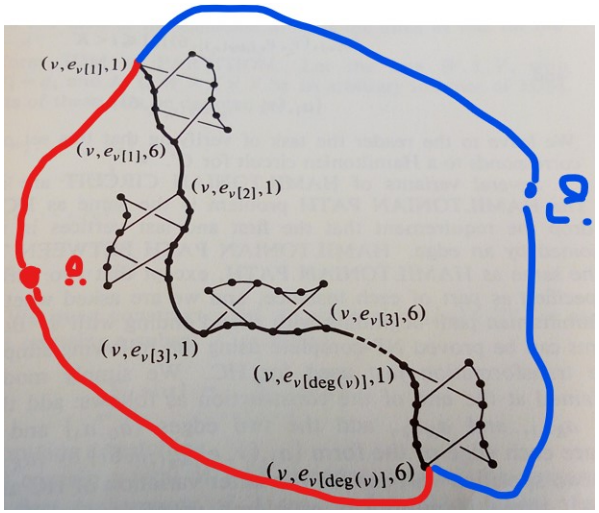
NP-hardness: Hamiltonian cycle / TSP

Given instance $G = (V, E), k$ of VC. Define $G' = (V', E')$:
1.) 'selector vertices': $a_1, a_2, \ldots, a_k$
2.) for each $e \in E$: 'cover-testing' component
Note: any Hamiltonian cycle needs to pass the edges in G' in one of the 3 shown configurations (or otherwise not all nodes would be visited exactly once):
3.) 'joining edges': for each node v denote by $v[1], v[2], \ldots, v[deg(v)]$ the incident edges of v:
Use $E'_v := \{(v, e_{v[i]}, 6), (v, e_{v[i+1]}) : 1 \leq i \leq deg(v)\}$ to build a path through the respective 'cover-testing components'
4.) more 'joining edges':
$E'' := \{(a_i, (v, e_{v[1]}, 1), (a_i, (v, e_{v[deg(v)]}, 6) : 1 \leq i \leq k, v \in V\}$
Transformation in polynomial time ✓



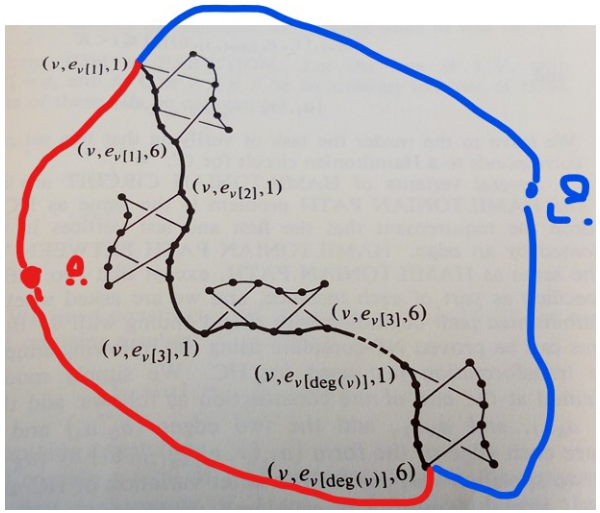
NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Rightarrow G$ has a vertex cover of size k:



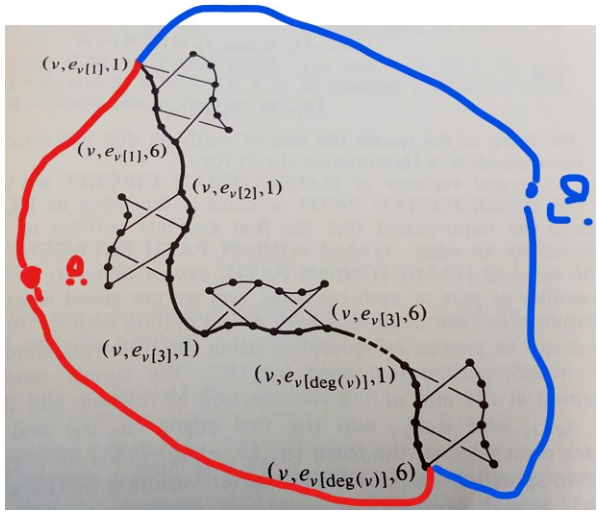
NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Rightarrow G$ has a vertex cover of size k:
Consider a subpath of the cycle that starts at an a_i and ends at an a_j ($j \neq i$) without passing a node a_k:



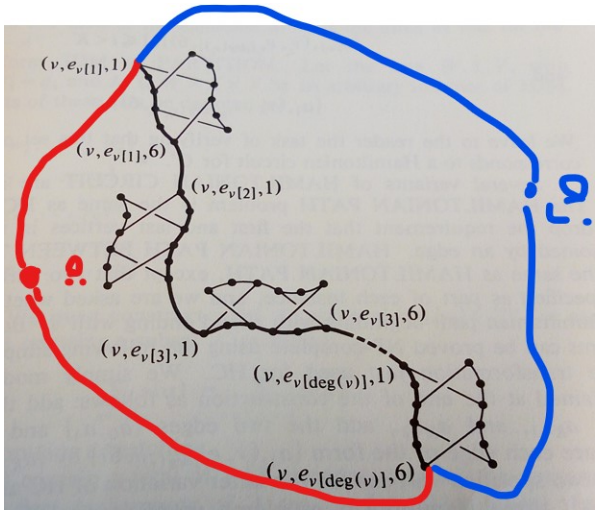
NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Rightarrow G$ has a vertex cover of size k:
Consider a subpath of the cycle that starts at an a_i and ends at an a_j ($j \neq i$) without passing a node a_k:
This subpath passes through a set of cover-testing components all associated with a specific v.



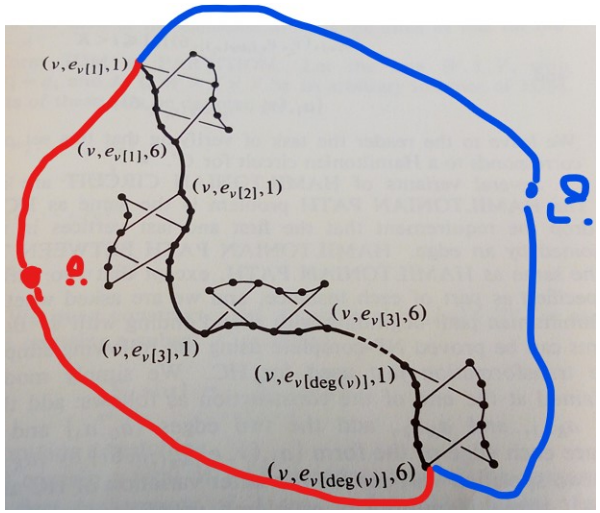
NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Rightarrow G$ has a vertex cover of size k:
Consider a subpath of the cycle that starts at an a_i and ends at an a_j ($j \neq i$) without passing a node a_k:
This subpath passes through a set of cover-testing components all associated with a specific v.
Therefore, the k vertices $\{a_1, a_2, \ldots, a_k\}$ divide the Hamiltonian circuit into k paths, each path i corresponding to a vertex v: the one that corresponds to the node $(v, e_{v[1}, 1)$ visited right after a_i.



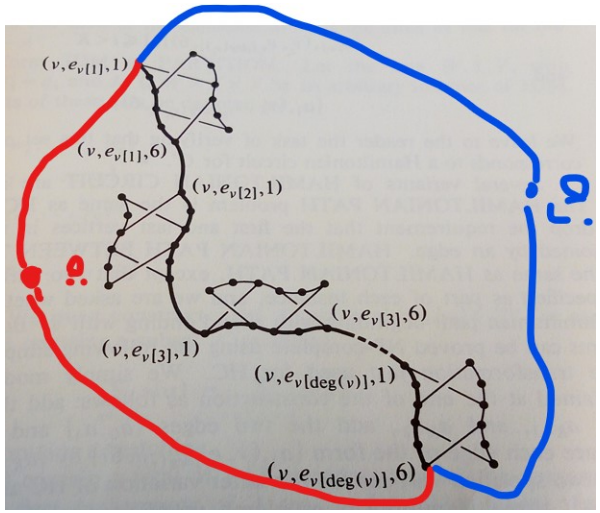
NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Rightarrow G$ has a vertex cover of size k:
Consider a subpath of the cycle that starts at an a_i and ends at an a_j $(j \neq i)$ without passing a node a_k:
This subpath passes through a set of cover-testing components all associated with a specific v.
Therefore, the k vertices $\{a_1, a_2, \ldots, a_k\}$ divide the Hamiltonian circuit into k paths, each path i corresponding to a vertex v: the one that corresponds to the node $(v, e_{v[1}, 1)$ visited right after a_i.
These vertices form a vertex cover:



NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Rightarrow G$ has a vertex cover of size k:
Consider a subpath of the cycle that starts at an a_i and ends at an a_j $(j \neq i)$ without passing a node a_k:
This subpath passes through a set of cover-testing components all associated with a specific v.
Therefore, the k vertices $\{a_1, a_2, \ldots, a_k\}$ divide the Hamiltonian circuit into k paths, each path i corresponding to a vertex v: the one that corresponds to the node $(v, e_{v[1}, 1)$ visited right after a_i.
These vertices form a vertex cover:
If there was an edge that was not covered in E, then its 'cover-testing component' would not be visited. ✓



NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Leftarrow G$ has a vertex cover of size k:

NP-hardness: Hamiltonian cycle / TSP

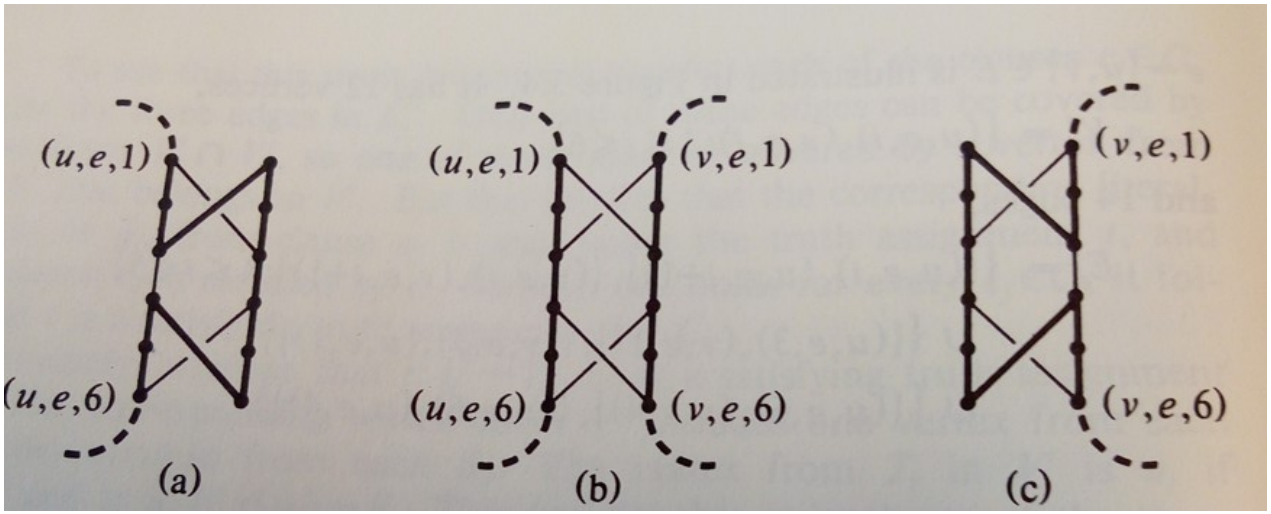
Show: G' has a Hamiltonian cycle $C \Leftarrow G$ has a vertex cover of size k:
Let $V^* = \{v_1, v_2, \ldots, v_k\}$ be a vertex cover (wlog $|V^*| = k$).

NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Leftarrow G$ has a vertex cover of size k:
Let $V^* = \{v_1, v_2, \ldots, v_k\}$ be a vertex cover (wlog $|V^*| = k$).
We include the following edges in C:

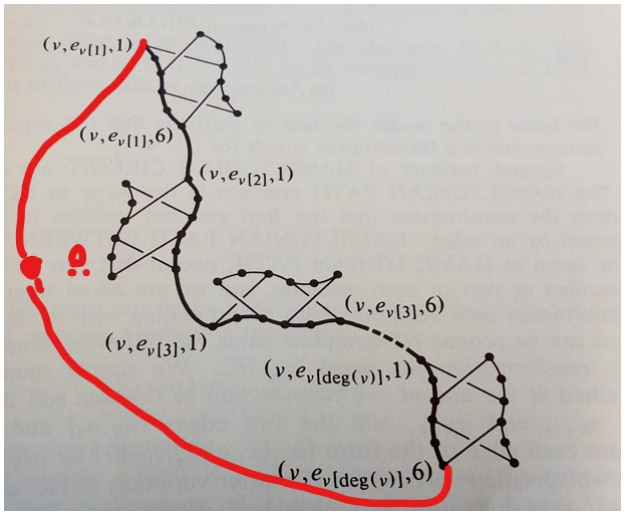
NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Leftarrow G$ has a vertex cover of size k:
Let $V^* = \{v_1, v_2, \ldots, v_k\}$ be a vertex cover (wlog $|V^*| = k$).
We include the following edges in C:
For each $\{u, v\} \in E$: if $\{u, v\} \cap E = \{u\}$: include edges as shown in (a)
if $\{u, v\} \cap E = \{u, v\}$: include edges as shown in (b)
if $\{u, v\} \cap E = \{v\}$: include edges as shown in (c)



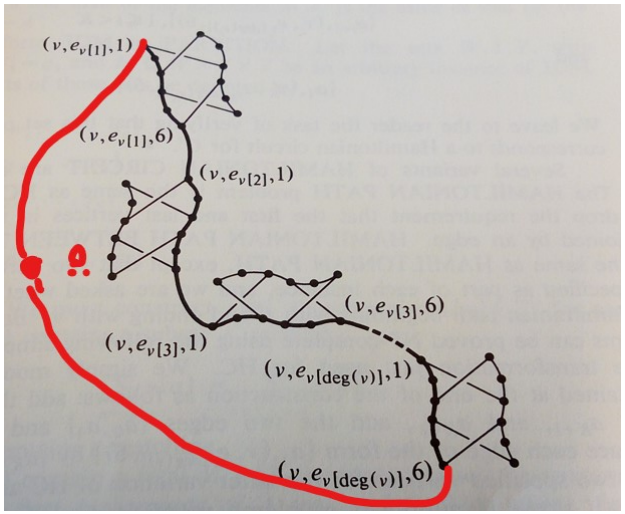
NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Leftarrow G$ has a vertex cover of size k:
Let $V^* = \{v_1, v_2, \ldots, v_k\}$ be a vertex cover (wlog $|V^*| = k$).
We include the following edges in C:
For each $\{u, v\} \in E$: if $\{u, v\} \cap E = \{u\}$: include edges as shown in (a)
if $\{u, v\} \cap E = \{u, v\}$: include edges as shown in (b)
if $\{u, v\} \cap E = \{v\}$: include edges as shown in (c)
For each $v_i \in V^*$: $\{a_i, (v_i, e_{v_i[1]}), 1\}, 1 \leq i \leq k$
$\{a_{i+1}, (v_i, e_{v_i[deg(v_i)]}), 1\}, 1 \leq i \leq k$



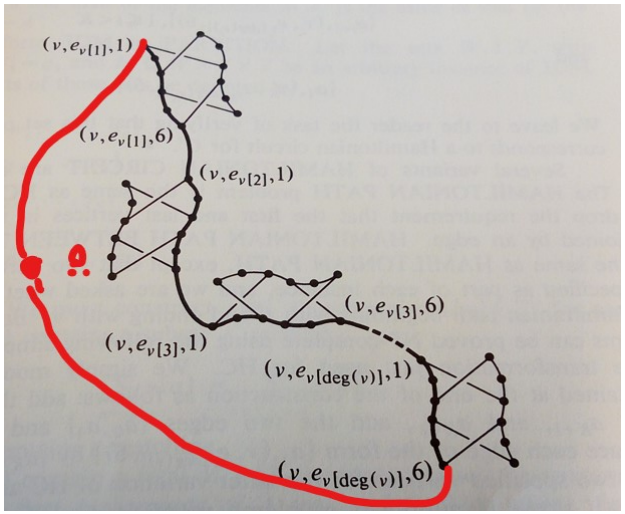
NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Leftarrow G$ has a vertex cover of size k:
Let $V^* = \{v_1, v_2, \ldots, v_k\}$ be a vertex cover (wlog $|V^*| = k$).
We include the following edges in C:
For each $\{u, v\} \in E$: if $\{u, v\} \cap E = \{u\}$: include edges as shown in (a)
if $\{u, v\} \cap E = \{u, v\}$: include edges as shown in (b)
if $\{u, v\} \cap E = \{v\}$: include edges as shown in (c)
For each $v_i \in V^*$: $\{a_i, (v_i, e_{v_i[1]}), 1\}, 1 \leq i \leq k$
$\{a_{i+1}, (v_i, e_{v_i[deg(v_i)]}), 1\}, 1 \leq i \leq k$
And: $\{a_1, (v_k, e_{v_k[deg(v_k)]}, 6)\}$



NP-hardness: Hamiltonian cycle / TSP

Show: G' has a Hamiltonian cycle $C \Leftarrow G$ has a vertex cover of size k:
Let $V^* = \{v_1, v_2, \ldots, v_k\}$ be a vertex cover (wlog $|V^*| = k$).
We include the following edges in C:
For each $\{u, v\} \in E$: if $\{u, v\} \cap E = \{u\}$: include edges as shown in (a)
if $\{u, v\} \cap E = \{u, v\}$: include edges as shown in (b)
if $\{u, v\} \cap E = \{v\}$: include edges as shown in (c)
For each $v_i \in V^*$: $\{a_i, (v_i, e_{v_i[1]}), 1\}, 1 \leq i \leq k$
$\{a_{i+1}, (v_i, e_{v_i[deg(v_i)]}), 1\}, 1 \leq i \leq k$
And: $\{a_1, (v_k, e_{v_k[deg(v_k)]}, 6)\}$
This forms a Hamiltonian cycle ✓



NP-hardness: Hamiltonian cycle / TSP

Theorem

TSP is NP-complete.

Proof: already seen: in NP and TSP$\leq_p$ HC.