NP-completeness

1. The complexity class P
   a. Instance encodings
   b. Formal languages
2. The complexity class NP
   a. NP-completeness and reducibility
   b. proofs of NP-completeness
3. Some NP-complete Problems
   a. SAT
   b. 3-SAT
   c. Clique
   d. Partition
   e. k-coloring
   f. 3D-Matching

1a. The complexity class P

An abstract problem is a relation on the sets of problem instances and problem solutions.

Example 1 *The* SHORTEST PATH *problem:*
Instance: *A simple graph G and two vertices u, v.*
Output: *A shortest path $u \rightsquigarrow v$ in G (it such exists).*

Decision problems:
A solution is of the form "Y" or "N".

Example 2
Instance: *A simple graph G, two vertices u, v, and $k > 0$.*
Question: *$\exists u \rightsquigarrow v$ in G of length $\leq k$?*

Optimization problems:
Some function should be minimized or maximized.

Any "discrete optimization problem" can be formulated as a decision problem.

Remark 1
Optimization problem is "easily solvable" $\Rightarrow$ corresponding decision problem is also "easily solvable".

Optimization problem is "hardly solvable" $\Rightarrow$ corresponding decision problem is also "hardly solvable".

An encoding is a mapping of the set of abstract object into the set of binary strings.

Any algorithm that "solves" an abstract decision problem works with an encoding of this problem. We call a problem with encoded instance concrete problem.

Definition 1 *We say that an algorithm solves a problem in time $O(T(n))$ if for any instance encoding of length n the algorithm computes a solution in time $O(T(n))$.*

Definition 2 *A concrete problem with instance encoding of size n is solvable in polynomial time, if there exists an algorithm for solving the problem in time $O(n^k)$ for come constant k (independent on n).*

Definition 3 *The complexity class P consists of the concrete decision problems solvable in polynomial time.*

Let f be a mapping $f : \{0,1\}^* \mapsto \{0,1\}^*$. f is said to be computable in polynomial time, if there exists an algorithm that for any $x \in \{0,1\}^*$ constructs a sequence $f(x)$ in polynomial time.

Let I be the set of all problem instances. We call two encodings e_1 and e_2 polynomially equivalent, if there exist computable in polynomial time functions f_{12} and f_{21} such that for any $i \in I$ one has: $f_{12}(e_1(i)) = e_2(i)$ and $f_{21}(e_2(i)) = e_1(i)$.

Lemma 1 *Let Q be an abstract decision problem and e_1, e_2 be polynomially equivalent encodings of the set $I = \{i\}$. Then $Q(e_1(i)) \in P \iff Q(e_2(i)) \in P$.*

1b. A formal language framework

Let $\Sigma = \{0, 1\}$. A language L is a subset of Σ^*. Any decision problem Q can be represented as the following language:

$$L = \{x \in \Sigma^* \mid Q(x) = 1\}.$$

Definition 4 *An algorithm A accepts $x \in \Sigma^*$, if its output $A(x) = 1$. The algorithm A rejects a string $x \in \Sigma^*$ if $A(x) = 0$.*

The set $L = \{x \in \Sigma^* \mid A(x) = 1\}$ is the language accepted by algorithm A.

A language L is decided by an algorithm A if for any $x \in \Sigma^*$ either A accepts x or A rejects x.

Definition 5 *A language L is accepted by algorithm A in in polynomial time, if any $x \in L$ with $|x| = n$ is accepted by A in time $O(n^k)$.*

The language L is decided in polynomial time by an algorithm A, if any $x \in \Sigma^*$ with $|x| = n$ is decided by A in time $O(n^k)$.

Further definitions for the class P:

$$P = \{L \subseteq \Sigma^* \mid \exists A \text{ which decides } L \text{ in polynomial time}\}.$$

Theorem 1 (Theorem 34.2, p.977)

$$P = \{L \subseteq \Sigma^* \mid L \textit{ is accepted by a polyn.-time algorithm}\}.$$

2a. The complexity class NP

Let x and y be binary strings.

Definition 6 *A verification algorithm is an algorithm with two parameters. We say A verifies a string x if $\exists y$ such that $A(x, y) = 1$.*

An algorithm A verifies a language L if:

$$L = \{x \in \Sigma^* \mid \exists y \in \Sigma^* \text{ with } A(x, y) = 1\}.$$

Definition 7 *The complexity class NP is the set of all languages L for which there exists a polynomial-time verification algorithm A and a constant c such that:*

$$L = \{x \in \Sigma^* \mid \exists y \text{ with } |y| = O(|x|^c),\ A(x, y) = 1\}.$$

Obviously, P $\subseteq$ NP. The principal question is whether P $\neq$ NP.

Definition 8 *A language L_1 is called polynomial-time reducible to a language L_2 if there exists a polynomial-time computable function $f : \Sigma^* \mapsto \Sigma^*$ such that for all $x \in \Sigma^*$ one has:*

$$x \in L_1 \qquad \Longleftrightarrow \qquad f(x) \in L_2.$$

(denotation $L_1 \leq_P L_2$).

We write $L_1 \equiv L_2$ if $L_1 \leq_P L_2$ and $L_2 \leq_P L_1$.

Lemma 2 *Let $L_1, L_2 \subseteq \Sigma^*$ be languages and $L_1 \leq_P L_2$. $L_2 \in P$ implies $L_1 \in P$.*

Definition 9 *Let $L \subseteq \Sigma^*$ be a language.*

1. *L is called NP-hard if $L' \leq_P L$ for any language $L' \in$ NP.*
2. *The language L is called L NP-complete if L is NP-hard and $L \in$ NP (denotation $L \in \mathrm{NPC}$).*

Theorem 2

1. *If some NP-complete problem is solvable in polynomial time then P=NP.*
2. *If some problem of NP is not solvable in polynomial time then no other NP-complete problem is solvable in polynomial time.*

Proof.

1. Let $L \in$ NPC and $L \in$ P.
   $\Rightarrow L' \leq_P L$ for any problem $L' \in$ NP (Definition 9).
   $\Rightarrow L' \in$ P (Lemma 2).
2. Assume $\exists L \in$ NP with $L \notin$ P.
   Let $L' \in$ NPC. $\Rightarrow L \leq_P L'$ (Definition 9).
   Now if $L' \in$ P then $L \in$ P (Lemma 2), a contradiction. $\square$

2b. Proofs of NP-completeness

Lemma 3 *Let L be a language such that $L' \leq_P L$ for some $L' \in$ NPC. Then L is NP-hard. If additionally, $L \in$ NP, then $L \in$ NPC.*

Proof. $L' \in$ NPC $\Rightarrow L'' \leq_P L'$ for any $L'' \in$ NP.
Furthermore, since $L' \leq_P L \Rightarrow L'' \leq_P L$
$\Rightarrow L$ is NP-hard.
$\Rightarrow L \in$ NPC if $L \in$ NP.

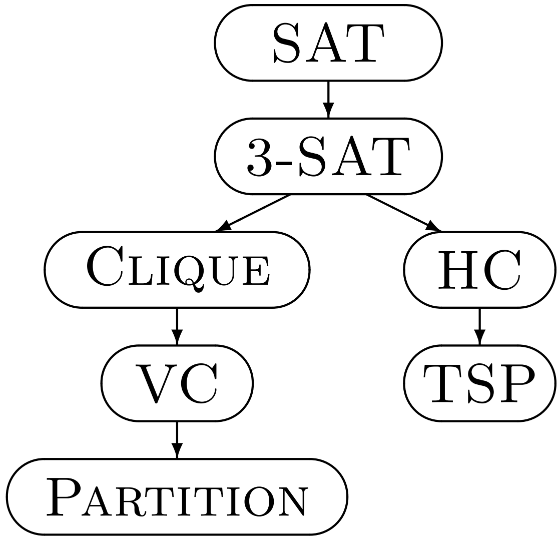
To prove NP-completeness:

1. Show: $L \in$ NP.
2. Choose an appropriate language L' (problem) for which it is know that it is NP-complete.
3. Design an algorithm that computes a function f mapping every instance x of L' to an instance $f(x)$ for L.
4. Prove that $x \in L' \Leftrightarrow f(x) \in L$ for any $x \in \{0, 1\}^*$.
5. Show that the function f is polynomial-time computable.

Theorem 3 (Cook).
One has:

$$\mathrm{SAT} \in \text{NPC}.$$

3. Some NP-complete problems



SATISFIABILITY (SAT):

Instance: Boolean formula F.

Question: Is F satisfiable?

3-SAT:

Similar to SAT, but each clause in the formula has 3 literals.

CLIQUE:

Instance: Graph G and $k \in \mathbb{N}$.

Question: Does G contain a k-clique?

HAM-CYCLE (HC):

Instance: Graph $G = (V, E)$.

Question: Does G contain a simple cycle of length $|V|$?

VERTEX-COVER (VC):

Instance: Graph G and $k \in \mathbb{N}$.

Question: Is there a set $C \subset V$ of size k such that any edge of G is incident to some vertex of C?

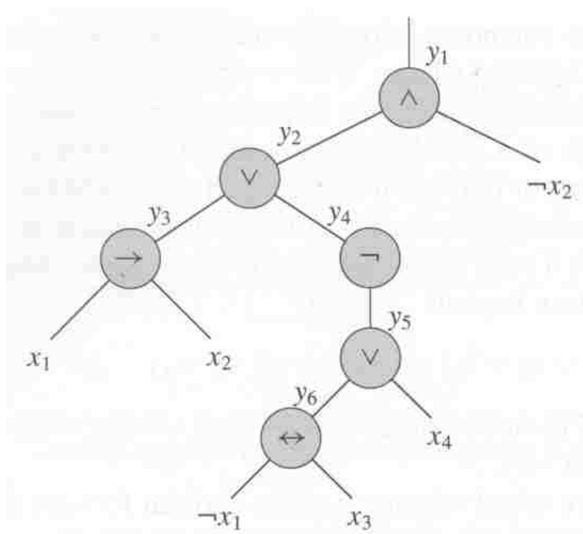
Theorem 4 $\quad$ $\mathrm{3\text{-}SAT} \in \mathsf{NPC}$.

Proof. We show $\mathrm{SAT} \le_P \mathrm{3\text{-}SAT}$.
Given a boolean formula f (instance for SAT), we construct an instance f' for $\mathrm{3\text{-}SAT}$.

Step 1. For any "internal subformula" we create a new variable y_i.

Example 3

$$f = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$$



$$\begin{aligned} f' = y_1 \ &\wedge\ (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\ &\wedge\ (y_2 \leftrightarrow (y_3 \vee y_4)) \\ &\wedge\ (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \\ &\wedge\ (y_4 \leftrightarrow \neg y_5) \\ &\wedge\ (y_5 \leftrightarrow (y_6 \vee x_4)) \\ &\wedge\ (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3)) \end{aligned}$$

<u>Step 2.</u> Write every clause C_i in f' in CNF and obtain a formula f''.

y_1	y_2	x_2	$(y_1 \leftrightarrow (y_2 \wedge \neg x_2))$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	1

$C_i = (\neg y_1 \vee \neg y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee x_2) \wedge (y_1 \vee \neg y_2 \vee x_2)$.

<u>Step 3.</u> Expand every clause C_i in f'' to make it depending on exactly 3 variables and obtain a formula f'''.

- $C_i = l_1 \vee l_2 \vee l_3 \Rightarrow C_i' := C_i \in f'''$.
- $C_i = l_1 \vee l_2 \Rightarrow C_i' := (l_1 \vee l_2 \vee p) \wedge (l_1 \vee l_2 \vee \neg p)$.
- $C_i = l \Rightarrow C_i' := (l \vee p \vee q) \wedge (l \vee \neg p \vee q) \wedge (l \vee p \vee \neg q) \wedge (l \vee \neg p \vee \neg q)$.

The formula f is satisfiable $\Longleftrightarrow$ f''' is satisfiable.

The formula f''' is constructible in polynomial time.

Therefore, $3\text{-SAT} \in$ NP. $\square$

CLIQUE:
Instance: A graph G and $k \in I\!N$.
Question: Does G contain a clique of size k ?

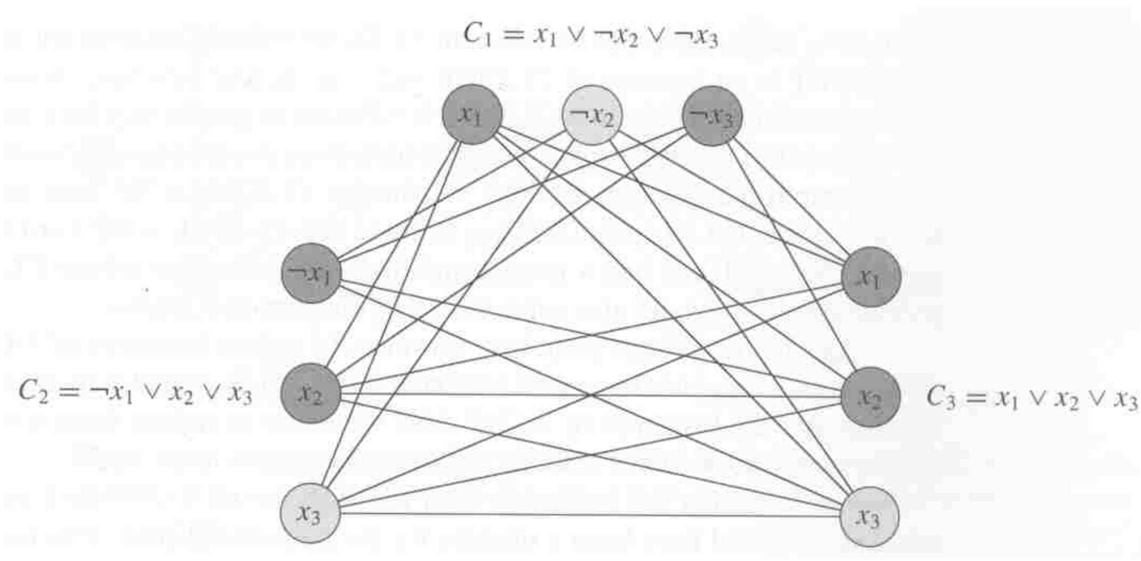
Theorem 5 CLIQUE $\in$ NPC.

Proof. Obviously, CLIQUE $\in$ NP.

We show: $3\text{-SAT} \leq_P$ CLIQUE. Let $f = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ be an instance for 3-SAT with $C_i = l_1^i \vee l_2^i \vee l_3^i$.

Construct a graph $G = (V, E)$ with $V = \{v_1^i, v_2^i, v_3^i \mid i = 1, ..., k\}$ and $(v_r^i, v_s^j) \in E$ iff $i \neq j$ and $l_r^i \neq \neg l_s^j$.

Example:



$$f = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3).$$

The formula f is satisfiable $\Longleftrightarrow$ G contains a clique with k vertices.

The graph G is constructible in polynomial time. $\square$

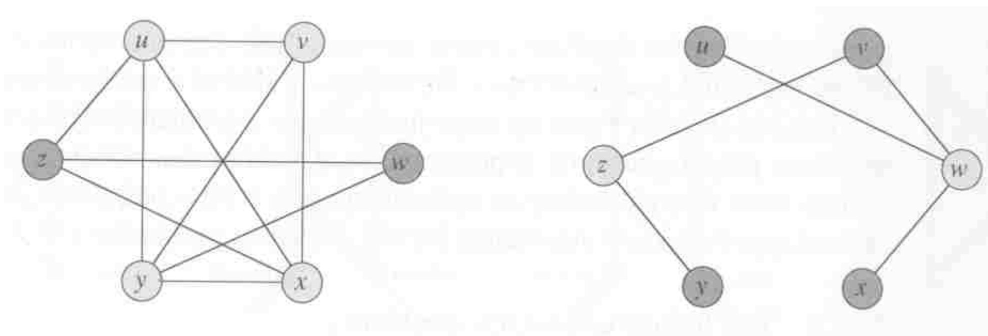
VERTEX COVER (VC):
Instance: A graph $G = (V, E)$ and $k \in I\!N$.
Question: Is there a subset $C \subset V$ with $|C| = k$ s.t. each edge of G is incident to some vertex of C?

Theorem 6 $\quad$ $\mathrm{VC} \in$ NPC.

Proof. Obviously, $\mathrm{VC} \in$ NP.
We show: $\textsc{Clique} \leq_P \mathrm{VC}$.

For a graph $G = (V, E)$ we define its complement $\overline{G} = (V, \overline{E})$.

Then G has a clique of size k iff $\overline{G}$ has a VC of size $|V| - k$.



(a) (b)

Indeed:
If G has a k-clique $V' \subset V$ then $V \setminus V'$ is a VC.

On the other hand, if $\overline{G}$ has a VC V' of size $|V'| = |V| - k$, then $\forall u, v \in V$ if $(u, v) \in \overline{E}$ then $u \in V'$ or $v \in V'$.
The contraposition of this implication is:
$\forall u, v \in V$ if $u \notin V'$ and $v \notin V'$ then $(u, v) \in E$.
In other words, $V \setminus V'$ is a clique. $\square$

Partition:
Instance: A set $S = \{s\}$ of integers and $t \in I\!N$.
Question: Is there a subset $S' \subseteq S$ with $\sum_{s \in S'} s = t$?

Theorem 7 Partition $\in$ NPC.

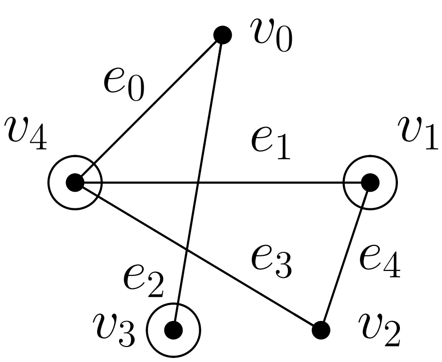
Proof. Obviously, Partition $\in$ NP.
We show: $\mathrm{VC} \leq_P$ Partition.

Let $G = (V, E)$ be an instance for VC with

$$V = \{v_0, ..., v_{n-1}\} \qquad \text{and} \qquad E = \{e_0, ..., e_{m-1}\}.$$

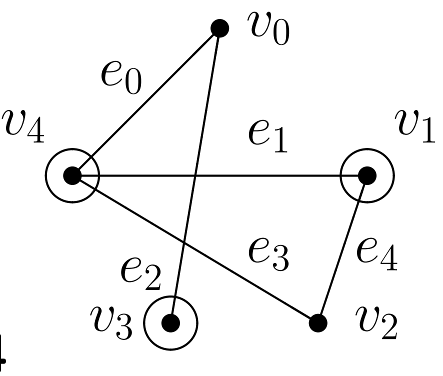
We represent G by its $n \times m$ incidence matrix $B = \{b_{ij}\}$, where

$$b_{ij} = \begin{cases} 1, & \text{if } e_i \text{ is incident to } v_j \\ 0, & \text{otherwise} \end{cases}$$



$$\begin{array}{c|ccccc|} B & e_4 & e_3 & e_2 & e_1 & e_0 \\ \hline v_0 & 0 & 0 & 1 & 0 & 1 \\ v_1 & 1 & 0 & 0 & 1 & 0 \\ v_2 & 1 & 1 & 0 & 0 & 0 \\ v_3 & 0 & 0 & 1 & 0 & 0 \\ v_4 & 0 & 1 & 0 & 1 & 1 \\ \hline \end{array}$$

Example 4



$$
\begin{array}{c|ccccc|}
B & e_4 & e_3 & e_2 & e_1 & e_0 \\
\hline
v_0 & 0 & 0 & 1 & 0 & 1 \\
v_1 & 1 & 0 & 0 & 1 & 0 \\
v_2 & 1 & 1 & 0 & 0 & 0 \\
v_3 & 0 & 0 & 1 & 0 & 0 \\
v_4 & 0 & 1 & 0 & 1 & 1 \\
\hline
\end{array}
$$

For $i = 0, ..., n-1$ and $j = 0, ..., m-1$ put:

$$x_i = 4^m + \sum_{j=0}^{m-1} b_{ij} 4^j, \qquad y_j = 4^j, \qquad t = k \cdot 4^m + \sum_{j=0}^{m-1} 2 \cdot 4^j$$

and extend the matrix B:

$$
\begin{array}{cccccccccccr}
 & x_0 & = & 1 & 0 & 0 & 1 & 0 & 1 & = & 1041 \\
\longrightarrow & x_1 & = & 1 & 1 & 0 & 0 & 1 & 0 & = & 1284 \\
 & x_2 & = & 1 & 1 & 1 & 0 & 0 & 0 & = & 1344 \\
\longrightarrow & x_3 & = & 1 & 0 & 0 & 1 & 0 & 0 & = & 1040 \\
\longrightarrow & x_4 & = & 1 & 0 & 1 & 0 & 1 & 1 & = & 1093 \\
\longrightarrow & y_0 & = & 0 & 0 & 0 & 0 & 0 & 1 & = & 1 \\
 & y_1 & = & 0 & 0 & 0 & 0 & 1 & 0 & = & 4 \\
\longrightarrow & y_2 & = & 0 & 0 & 0 & 1 & 0 & 0 & = & 16 \\
\longrightarrow & y_3 & = & 0 & 0 & 1 & 0 & 0 & 0 & = & 64 \\
\longrightarrow & y_4 & = & 0 & 1 & 0 & 0 & 0 & 0 & = & 256 \\
\hline
 & t & = & 3 & 2 & 2 & 2 & 2 & 2 & = & 3754
\end{array}
$$

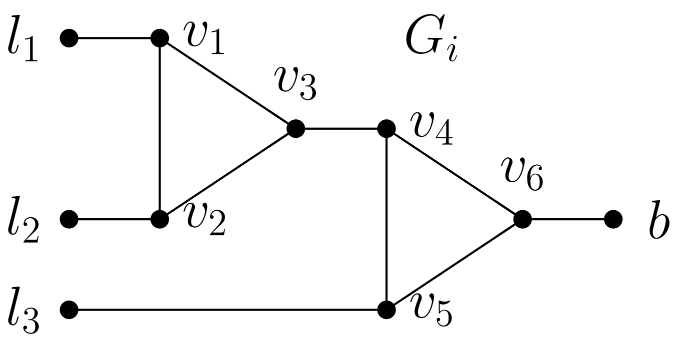
It holds: G has vertex cover of size $k \iff$
$\exists S' \subseteq S$ with $\sum_{s \in S'} s = t$. $\square$

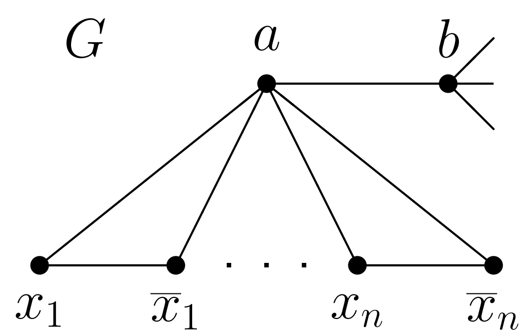
3-COLORING:
Instance: A graph $G = (V, E)$.
Question: Is G 3-colorable ?

Theorem 8 3-COLORING $\in$ NPC.

Proof. Obviously, 3-COLORING $\in$ NP.
We show: $3\text{-SAT} \leq_P 3$-COLORING.

Let $f = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ (here $f = f(x_1, ..., x_n)$) be an instance for 3-SAT. We construct for every clause $C_i = l_1 \vee l_2 \vee l_3$ a graph $G_i = (V_i, E_i)$, $1 \leq i \leq m$:





Assume the vertices l_1, l_2, l_3 are colored with color 0 or 1. Then v_6 can be colored with color 1 or 2 $\iff \exists\, l_i$, $1 \leq i \leq 3$ colored with 1.

We construct an instance $G = (V, E)$ for 3-COLORING:

$$V = \{a, b\} \bigcup_{i=1}^{m} V_i$$

$$E = \{(a, b)\} \cup \{(a, x_i),\ (a, \overline{x}_i),\ (x_i, \overline{x}_i) \mid 1 \leq i \leq n\} \bigcup_{i=1}^{m} E_i.$$

It holds: f is satisfiable $\iff G$ is 3-colorable. $\square$