Aalto University
School of Science

CS-E4530 Computational Complexity Theory

Lecture 8: NP-Complete Problems

Aalto University
School of Science
Department of Computer Science

Spring 2017

NP-COMPLETE PROBLEMS

- Characterising **NP**
- Variants of satisfiability
- Graph-theoretic problems
- Colouring problems
- Sets and numbers
- Pseudopolynomial algorithms

(C. Papadimitriou: *Computational Complexity*, Chapter 9)

1. Characterising NP

- The complexity class **NP** can be characterised also without any reference to nondeterministic Turing machines.
- In light of the alternative characterisation **NP** can be seen as the class of problems having *succinct certificates*. ¡

Definition

1. A relation $R \subseteq \Sigma^* \times \Sigma^*$ is *polynomially decidable* if there is a deterministic TM deciding the language $\{x;y \mid (x,y) \in R\}$ in polynomial time.
2. A relation R is *polynomially balanced* if $(x,y) \in R$ implies $|y| \leq |x|^k$ for some $k \geq 1$.

Proposition

Let $L \subseteq \Sigma^*$ be a language. Now $L \in \mathbf{NP}$ iff there is a polynomially balanced and polynomially decidable relation R such that $L = \{x \in \Sigma^* \mid (x,y) \in R \text{ for some } y \in \Sigma^*\}$.

Proof

($\Leftarrow$) Assume there is such a relation R. Then L is decided by a NTM that on input x, guesses a y of length at most $|x|^k$ and uses the machine for R to decide in polynomial time whether $(x,y) \in R$.

($\Rightarrow$) Assume that $L \in \mathbf{NP}$, i.e. there is a NTM N deciding L in time $|x|^k$ for some k.

Define a relation R as follows: $(x,y) \in R$ iff y is the encoding of an accepting computation of N on input x. Now R is polynomially

- balanced (each computation of N is polynomially bounded) and
- decidable (since it can be checked in linear time whether y encodes an accepting computation of N on x).
- As N decides L, $L = \{x \mid (x,y) \in R \text{ for some } y\}$.

Succinct certificates

A problem is in $\mathbf{NP}$ if any "yes" instance x of the problem has at least one *succinct certificate*, or polynomial witness, y. $\mathbf{NP}$ contains a huge number of practically important, natural computational problems:

- A typical problem is to construct a mathematical object satisfying certain specifications (path, solution of equations, routing, VLSI layout,...). This is the certificate.
- The decision version of the problem is to determine whether at least one such object exists for the input.
- The object is not very large compared to the input.
- The specifications of the object are simple enough to be checkable in polynomial time.

Boundary between $\mathbf{NP}$ and $\mathbf{P}$

- Most problems arising in computational practice are in $\mathbf{NP}$.
- Computational complexity theory provides tools to study which problems in $\mathbf{NP}$ belong to $\mathbf{P}$ and which (probably) do not.
- $\mathbf{NP}$-completeness is a basic tool in this respect:
  Showing that a problem is $\mathbf{NP}$-complete implies that the problem is among the least likely to be in $\mathbf{P}$.
  (If an $\mathbf{NP}$-complete problem is in $\mathbf{P}$, then $\mathbf{NP} = \mathbf{P}$.)
- A quote from Papadimitriou:
  There is nothing wrong with trying to prove that $\mathbf{P} = \mathbf{NP}$ by developing a polynomial-time algorithm for an $\mathbf{NP}$-complete problem.
  The point is that without an $\mathbf{NP}$-completeness proof we would be trying the same thing *without knowing it!*

NP-completeness and algorithm design techniques

When a problem is known to be NP-complete, further efforts are usually directed to:

(i) Attacking special cases

(ii) Approximation algorithms

(iii) Studying average case performance

(iv) Randomised algorithms

(v) (Exponential) algorithms that are practical for small instances

(vi) Local search methods

2. Variants of Satisfiability

- $\mathbf{NP}$-complete problems typically remain difficult even when restricted to quite simple inputs. But often they also have interesting special cases in $\mathbf{P}$.
- Hence it is relevant to find the borderlines, for a given problem, between subproblems that are in $\mathbf{P}$ and that are $\mathbf{NP}$-complete.
- A basic technique to find difficult subproblems of a given $\mathbf{NP}$-complete problem A is to look at the set of instances produced by a reduction R from some other $\mathbf{NP}$-complete problem B to A.
- Next we consider variants of SAT such as

  $$3\text{SAT},\ 2\text{SAT},\ \text{MAX2SAT},\ \text{and NAESAT}$$

  and analyse their computational complexities.

kSAT problems

Definition

kSAT, where $k \geq 1$ is an integer, is the set of Boolean expressions $\phi \in$ SAT (in CNF) whose every clause has exactly k literals.

Proposition

3SAT is $\mathbf{NP}$*-complete.*

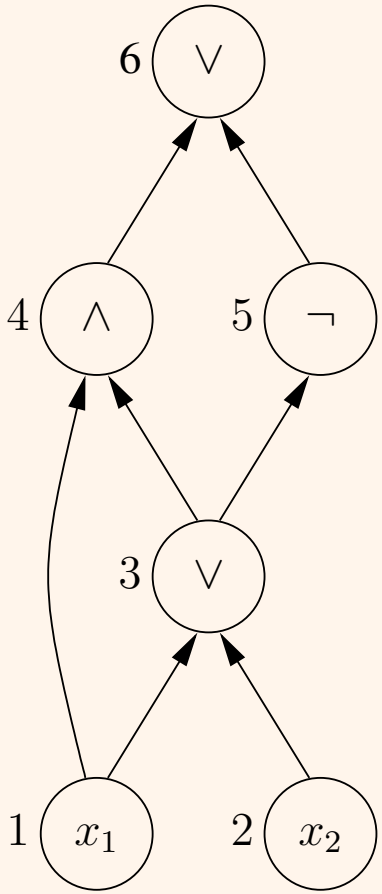
Proof

- 3SAT is in $\mathbf{NP}$ as a special case of SAT which is in $\mathbf{NP}$.
- CIRCUIT SAT was shown to be $\mathbf{NP}$-complete and a reduction from CIRCUIT SAT to SAT has already been presented.
- Consider now the clauses in the reduction. They have all at most 3 literals. Each clause with one or two literals can be modified to an equivalent clause with exactly 3 literals by duplicating literals.
- Hence, we can reduce CIRCUIT SAT to 3SAT.

Example

3SAT is **NP**-complete: reduction from CIRCUIT SAT to 3SAT

Circuit C:



The corresponding CNF formula $R(C)$:

$$(g_6 \vee g_6 \vee g_6) \wedge$$
$$(\neg g_6 \vee g_4 \vee g_5) \wedge (g_6 \vee \neg g_4 \vee \neg g_4) \wedge (g_6 \vee \neg g_5 \vee \neg g_5) \wedge$$
$$(\neg g_5 \vee \neg g_3 \vee \neg g_3) \wedge (g_5 \vee g_3 \vee g_3)$$
$$(g_4 \vee \neg g_1 \vee \neg g_3) \wedge (\neg g_4 \vee g_1 \vee g_1) \wedge (\neg g_4 \vee g_3 \vee g_3) \wedge$$
$$(\neg g_3 \vee g_1 \vee g_2) \wedge (g_3 \vee \neg g_1 \vee \neg g_1) \wedge (g_3 \vee \neg g_2 \vee \neg g_2) \wedge$$
$$(g_2 \vee \neg x_2 \vee \neg x_2) \wedge (\neg g_2 \vee x_2 \vee x_2) \wedge$$
$$(g_1 \vee \neg x_1 \vee \neg x_1) \wedge (\neg g_1 \vee x_1 \vee x_1)$$

Narrowing NP-complete languages

- An **NP**-complete languages can sometimes be narrowed down by *transformations* which eliminate certain features of the language but still preserve **NP**-completeness.
- The following result is a typical example.

Proposition

$3SAT$ remains $\mathbf{NP}$-complete even if each variable is restricted to appear at most three times in a Boolean expression $\phi \in 3SAT$ and each literal at most twice in ϕ.

Proof

This is shown by a reduction where any instance ϕ of 3SAT is rewritten to eliminate the forbidden features.

- Consider a variable x appearing $k \geq 3$ times in ϕ.
  - (i) Replace the first occurrence of x in ϕ by x_1, the second by x_2, and so on where $x_1, \ldots, x_k$ are new variables.
  - (ii) Add clauses $(\neg x_1 \vee x_2), (\neg x_2 \vee x_3), \ldots, (\neg x_k \vee x_1)$ to ϕ.
- Let ϕ' be the expression ϕ modified systematically in this way.
- It follows that ϕ' has the desired syntactic properties.
- Now ϕ is satisfiable iff ϕ' is satisfiable:
  For each x appearing $k > 3$ times in ϕ, the truth values of $x_1, \ldots, x_k$ are the same in each truth assignment satisfying ϕ'.

Example

Original CNF formula ϕ for 3SAT:

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (x_1 \vee x_3 \vee \neg x_4)$$

The "sparse" CNF formula $R(\phi)$:

$$(x_{1,1} \vee x_2 \vee x_{3,1}) \wedge (x_{1,2} \vee x_{3,2} \vee x_4) \wedge (\neg x_{1,3} \vee \neg x_2 \vee \neg x_4) \wedge (x_{1,4} \vee x_{3,3} \vee \neg x_4) (\neg x_{1,1} \vee x_{1,2}) \wedge (\neg x_{1,2} \vee x_{1,3}) \wedge (\neg x_{1,3} \vee x_{1,4}) \wedge (\neg x_{1,4} \vee x_{1,1}) \wedge (\neg x_{3,1} \vee x_{3,2}) \wedge (\neg x_{3,2} \vee x_{3,3}) \wedge (\neg x_{3,3} \vee x_{3,1})$$

Boundary between P and NP-completeness

- The boundary is between 2SAT and 3SAT.
- For an instance ϕ of 2SAT, there is a polynomial time algorithm which is based on reachability in a graph associated with ϕ.

Definition

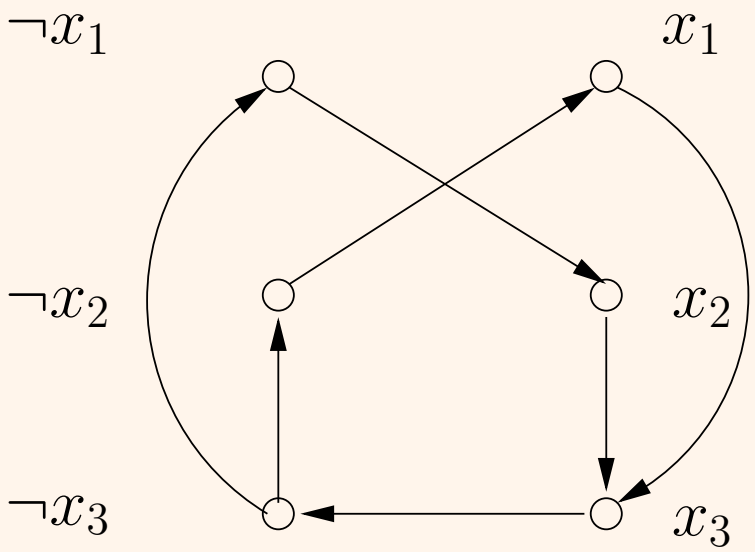
Let ϕ be an instance of 2SAT.
Define a graph $G(\phi)$ as follows:
– The vertices of $G(\phi)$ correspond to the variables of ϕ and their negations.
– For every clause $\alpha \vee \beta$ in ϕ, there are arcs $(\overline{\alpha}, \beta)$ and $(\overline{\beta}, \alpha)$ in $G(\phi)$.

Theorem

Let ϕ be an instance of 2SAT.
Then ϕ is unsatisfiable iff there is a variable x such that there are paths from x to $\neg x$ and from $\neg x$ to x in $G(\phi)$.

Example

- Consider the formula
  $\phi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee \neg x_3)$
- The graph $G(\phi)$:



- ϕ is unsatisfiable as there is a path from x_3 to $\neg x_3$ and from $\neg x_3$ to x_3 in $G(\phi)$.

The complexity of 2SAT—cont'd

Corollary

2SAT is in $\mathbf{NL}$ $(\subseteq \mathbf{P})$.

Proof

Since $\mathbf{NL}$ is closed under complement, it is sufficient to show that 2SAT COMPLEMENT is in $\mathbf{NL}$.

The reachability condition of the preceding theorem can be tested in logarithmic space non-deterministically by guessing a variable x and paths from x to $\neg x$ and back.

The complexity of 2SAT—cont'd

MAX2SAT is a generalisation of 2SAT:
INSTANCE: a Boolean expression ϕ in CNF (having at most two literals per clause) and an integer bound K.
QUESTION: Is there a truth assignment satisfying at least K clauses?

Theorem

MAX2SAT is $\mathbf{NP}$*-complete. (See pages 186–187 in the book.)*

The case of not-all-equal SAT (NAESAT)

For each $\phi \in$ NAESAT $\subseteq 3$SAT, there is a truth assignment so that the three literals in each clause of ϕ do not have the same truth value.

Theorem

NAESAT is $\mathbf{NP}$*-complete.*

Proof.

- CIRCUIT SAT was shown to be $\mathbf{NP}$-complete and a reduction R from CIRCUIT SAT to SAT has already been presented such that for a circuit C, $C \in$ CIRCUIT SAT iff $R(C) \in$ SAT.
- For all one- and two-literal clauses in the resulting set of clauses $R(C)$, add the same literal, say z, to make them 3-literal clauses.

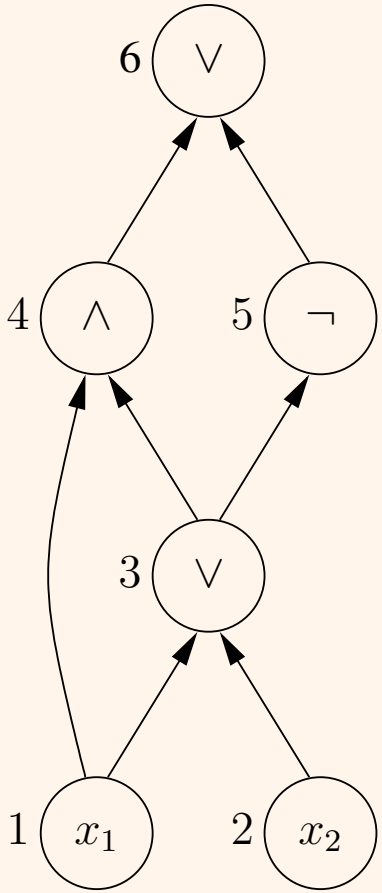
Claim: it holds for the resulting Boolean expression $R_z(C)$ in 3CNF:

$$C \in \text{CIRCUIT SAT } \textit{iff } R_z(C) \in \text{NAESAT}.$$

Example

Reduction from CIRCUIT SAT to NAESAT:

Circuit C:



The corresponding formula $R_z(C)$:

$$(g_6 \vee z \vee z) \wedge$$
$$(\neg g_6 \vee g_4 \vee g_5) \wedge (g_6 \vee \neg g_4 \vee z) \wedge (g_6 \vee \neg g_5 \vee z) \wedge$$
$$(\neg g_5 \vee \neg g_3 \vee z) \wedge (g_5 \vee g_3 \vee z)$$
$$(g_4 \vee \neg g_1 \vee \neg g_3) \wedge (\neg g_4 \vee g_1 \vee z) \wedge (\neg g_4 \vee g_3 \vee z) \wedge$$
$$(\neg g_3 \vee g_1 \vee g_2) \wedge (g_3 \vee \neg g_1 \vee z) \wedge (g_3 \vee \neg g_2 \vee z) \wedge$$
$$(g_2 \vee \neg x_2 \vee z) \wedge (\neg g_2 \vee x_2 \vee z) \wedge$$
$$(g_1 \vee \neg x_1 \vee z) \wedge (\neg g_1 \vee x_1 \vee z)$$

($\Rightarrow$) If C is satisfiable, then there is a truth assignment T satisfying $R(C)$. Let us then extend T for $R_z(C)$ by assigning $T(z) = \mathbf{false}$.
In no clause of $R_z(C)$ all literals are $\mathbf{true}$ in the extended assignment T (they cannot be all $\mathbf{false}$):

(i) Clauses for $\mathbf{true}/\mathbf{false}$/NOT/variable gates contain z that is $\mathbf{false}$.

(ii) For AND gates the clauses are: $(\neg g \vee h \vee z)$, $(\neg g \vee h' \vee z)$, $(g \vee \neg h \vee \neg h')$ where in the first two z is $\mathbf{false}$, and in the third all three cannot be true as then the first two would be not true.

(iii) The case of OR gates is similar.

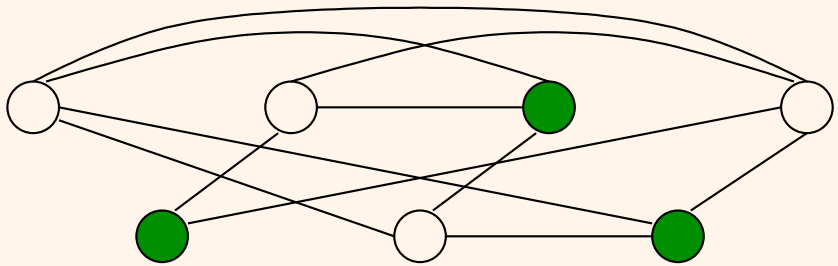
($\Leftarrow$) If a truth assignment T satisfies $R_z(C)$ in the sense of NAESAT, so does the complementary truth assignment $\overline{T}$.
Thus, z is $\mathbf{false}$ in either T or $\overline{T}$ which implies that $R(C)$ is satisfied by T or $\overline{T}$. Thus, C is satisfiable. $\square$

3. Graph-Theoretic Problems

- In this section, we will consider only undirected graphs $G = (V, E)$ and their properties.
- For instance, consider the problem of finding an *independent* subset I of V, i.e., a set I such that for all $i, j \in I$, $\{i, j\} \notin E$.

Example

A graph with an independent set of size 3 (illustrated with green)



Definition

INDEPENDENT SET:
INSTANCE: An undirected graph $G = (V, E)$ and an integer K.
QUESTION: Is there an independent set $I \subseteq V$ with $|I| = K$?

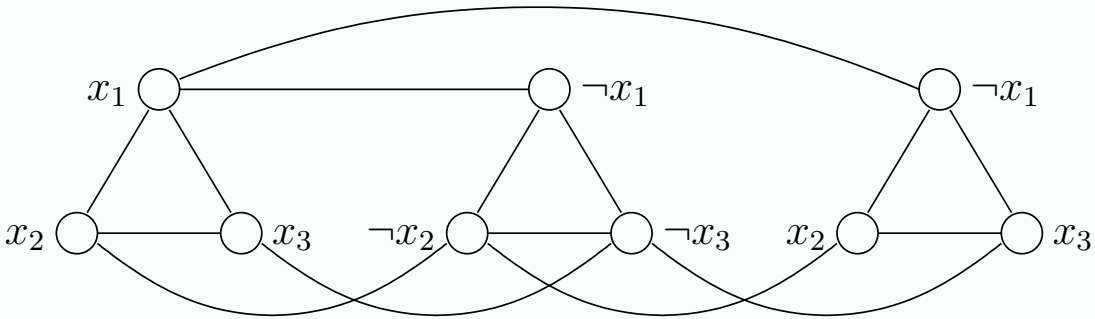
Theorem

INDEPENDENT SET is **NP***-complete.*

Proof

Reduction from 3SAT, see pages 188–190 in the book for details.

Example: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$ is reduced to



The subclass of graphs needed in the reduction implies the following:

Corollary

4-DEGREE INDEPENDENT SET is **NP***-complete.*

Graph problems: CLIQUE and VERTEX COVER

- The problems in graph theory are often closely related; suggesting even trivial reductions between problems.
- We now show that independent sets are closely related to cliques and vertex covers.

Definition

CLIQUE:

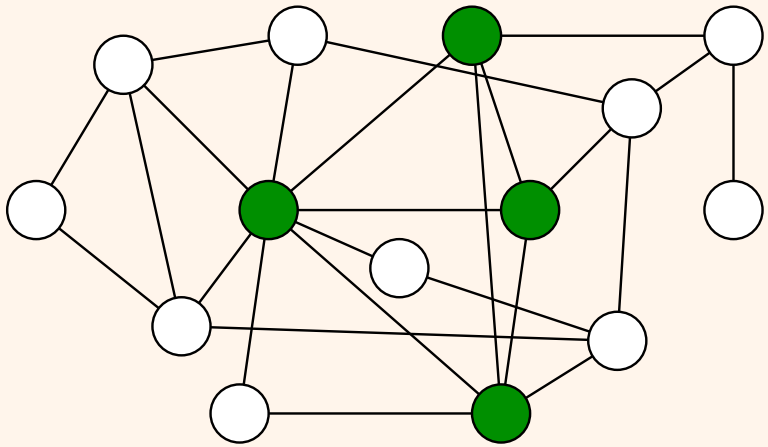
INSTANCE: An undirected graph $G = (V, E)$ and an integer K.

QUESTION: Is there a clique $C \subseteq V$ with $|C| = K$?

(A set $C \subseteq V$ is a clique iff for any two vertices $i, j \in C$, $\{i, j\} \in E$.)

Example

A graph with a clique of size 4 (shown in green):



Definition

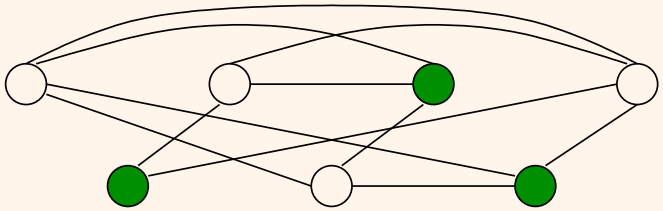
CLIQUE:
INSTANCE: An undirected graph $G = (V, E)$ and an integer K.
QUESTION: Is there a clique $C \subseteq V$ with $|C| = K$?
(A set $C \subseteq V$ is a clique iff for any two vertices $i, j \in C$, $\{i, j\} \in E$.)

- A set $I \subseteq V$ of vertices is an independent set of G iff it is a clique of the *complement* graph $\overline{G}$.

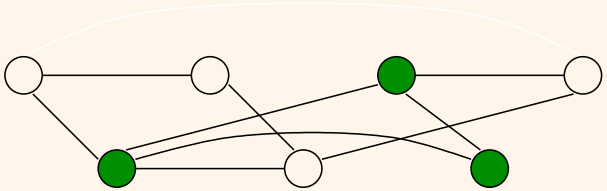
Example

Reduction from INDEPENDENT SET to CLIQUE illustrated:

$G; K$



$R(G; K) = \bar{G}; K$



Corollary

CLIQUE is $\mathbf{NP}$*-complete.*

Definition

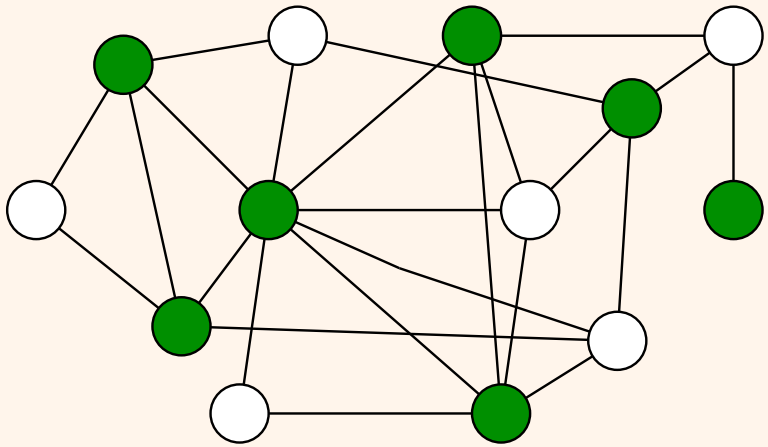
VERTEX COVER:

INSTANCE: An undirected graph $G = (V, E)$ and an integer B.

QUESTION: Is there a set $C \subseteq V$ with $|C| \leq B$ such that for every $\{i, j\} \in E$, either $i \in C$ or $j \in C$ (or both)?

Example

A graph with a vertex cover of size 7 (shown in green):



Definition

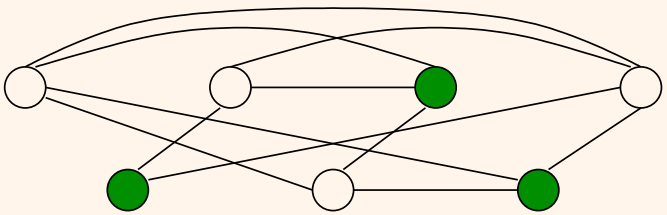
VERTEX COVER:
INSTANCE: An undirected graph $G = (V, E)$ and an integer B.
QUESTION: Is there a set $C \subseteq V$ with $|C| \leq B$ such that for every $\{i, j\} \in E$, either $i \in C$ or $j \in C$ (or both)?

- A set $I \subseteq V$ of vertices is an independent set of G iff $V - I$ is a vertex cover of G.

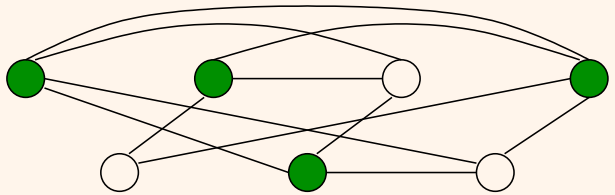
Example

Reduction from INDEPENDENT SET to VERTEX COVER illustrated:

$G; K$



$R(G; K) = G; |V| - |K|$



Corollary

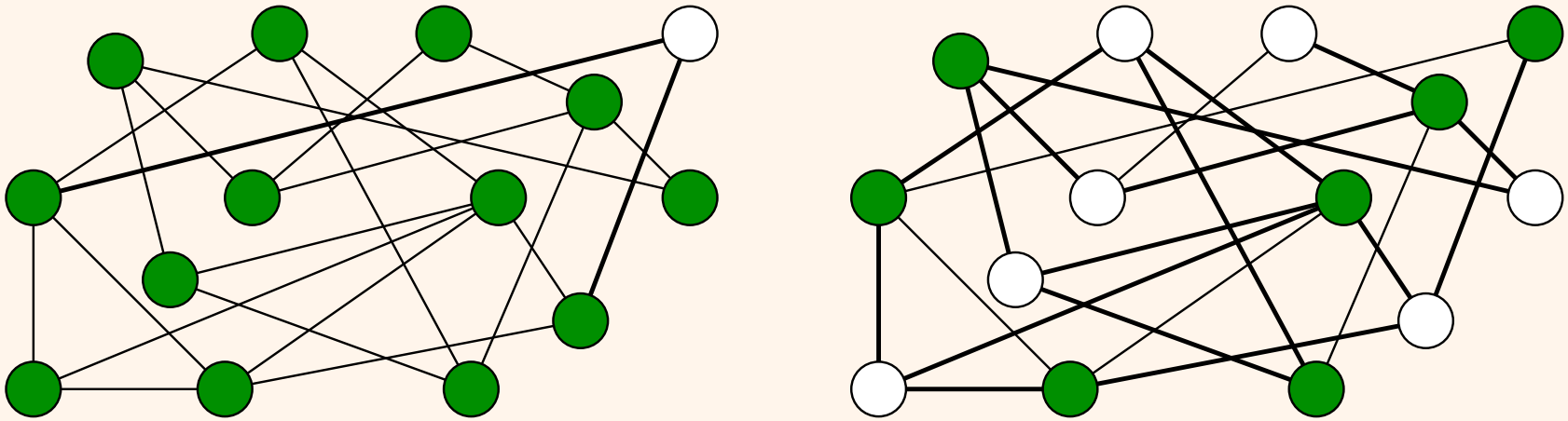
VERTEX COVER is $\mathbf{NP}$*-complete.*

Graph problems: MIN CUT and MAX CUT

- A cut in an undirected graph $G = (V, E)$ is a partition of the vertices into two nonempty sets S and $V - S$.
- The size of a cut is the number of edges between S and $V - S$.

Example

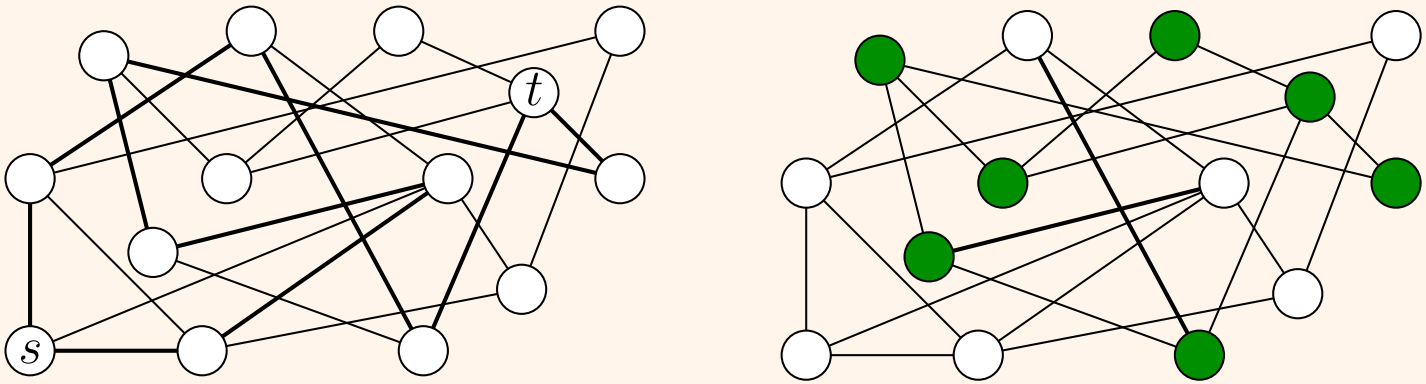
A graph and two cuts (of sizes 2 and 17, resp.):



- The problem of finding a cut with the smallest size is in $\mathbf{P}$:
  - (i) The size of the smallest cut that separates two given vertices s and t equals the maximum flow from s to t. ("Max-Flow/Min-Cut Thm".)
  - (ii) Minimum cut: find the maximum flow between a fixed s and all other vertices and choose the smallest value found.

Example

A maximum flow and cut of size 2:



- However, the problem of deciding whether there is a cut of a size at least K (MAX CUT) is much harder:

Theorem

MAX CUT is $\mathbf{NP}$-complete.

Reduction from NAESAT to MAX CUT

The $\mathbf{NP}$-completeness of MAX CUT is shown for graphs with multiple edges between vertices by a reduction from NAESAT.

- For a conjunction of clauses $\phi = C_1 \wedge \ldots \wedge C_m$, we construct a graph $G = (V, E)$ so that

  G has a cut of size $5m$ iff ϕ is satisfied in the sense of NAESAT.
- The vertices of G are $x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n$ where $x_1, \ldots, x_n$ are the variables in ϕ.
- The edges in G include a triangle $[\alpha, \beta, \gamma]$ for each clause $\alpha \vee \beta \vee \gamma$ and n_i copies of the edge $\{x_i, \neg x_i\}$ where n_i is the number of occurrences of x_i or $\neg x_i$ in the clauses.
- Now a cut $(S, V - S)$ of size $5m$ in G corresponds to a truth assignment satisfying ϕ in the sense of NAESAT.

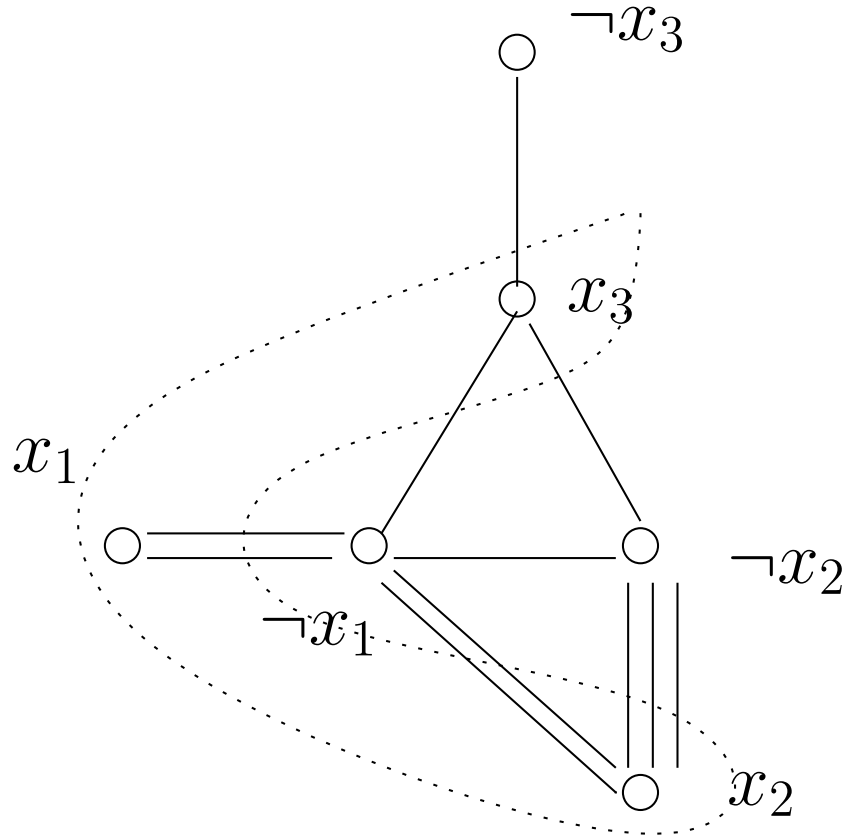
Example. Consider the conjunction of clauses ϕ:

$$(\neg x_1 \vee x_2 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

which is satisfied in the sense of NAESAT iff the graph G on the right obtained as the result of the reduction has a cut of size 5*2=10.
For instance,

$$(\{x_1, x_2, x_3\}, \{\neg x_1, \neg x_2, \neg x_3\})$$

is a cut of size 10 and it corresponds to a truth assignment $T(x_1) = T(x_2) = T(x_3) = \textbf{true}$ satisfying ϕ in the sense of NAESAT.



Correctness of the reduction

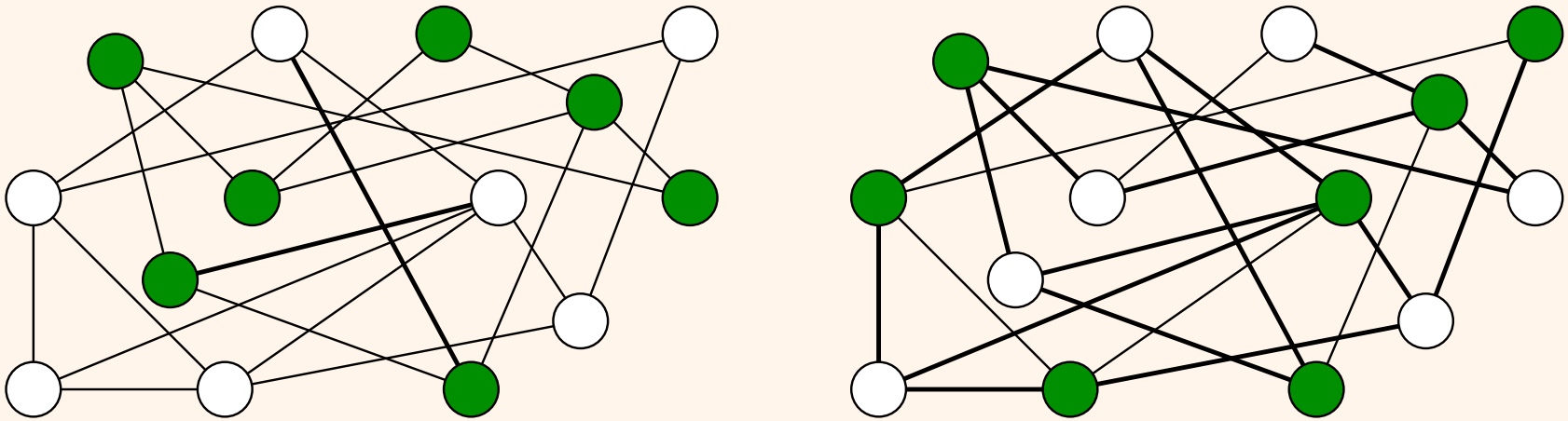
- It is easy to see that a satisfying truth assignment (in the sense of NAESAT) gives rise to a cut of size $5m$.
- Conversely, suppose there is a cut $(S, V - S)$ of size $5m$ or more.
- All variables can be assumed separate from their negations:
  If both $x_i, \neg x_i$ are on the same side, they contribute at most $2n_i$ edges to the cut (where n_i is the number of occurrences of x_i or $\neg x_i$ in the clauses).
  Hence, moving the one with fewer neighbours to the other side of the cut does not decrease the size of the cut.
- Let S be the set of true literals and $V - S$ those false.
- The total number of edges in the cut joining opposite literals is $3m$. The remaining $2m$ are coming from triangles meaning that all m triangles are cut, i.e. ϕ is satisfied in the sense of NAESAT. □

Graph problems: MAX BISECTION

- In many applications of graph partitioning, the sizes of S and $V - S$ cannot be arbitrarily small or large.
- MAX BISECTION is the problem of determining whether there is a cut $(S, V - S)$ with size of K or more such that $|S| = |V - S|$.

Example

Bisections with cut sizes of 2 and 17, respectively:



- Is MAX BISECTION easier than MAX CUT?

Lemma

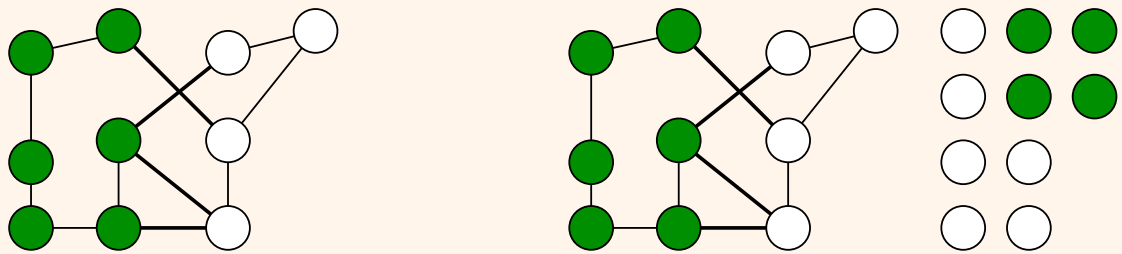
MAX BISECTION is $\mathbf{NP}$*-complete.*

Proof

Reducing MAX CUT to MAX BISECTION by modifying input:
Add $|V|$ disconnected new vertices to G. Now every cut of G can be made a bisection by appropriately splitting the new vertices.
Now $G = (V, E)$ has a cut $(S, V - S)$ with size of K or more iff the modified graph has a cut with size of K or more with $|S| = |V - S|$.

Example

Reducing MAX CUT to MAX BISECTION:



Graph problems: BISECTION WIDTH

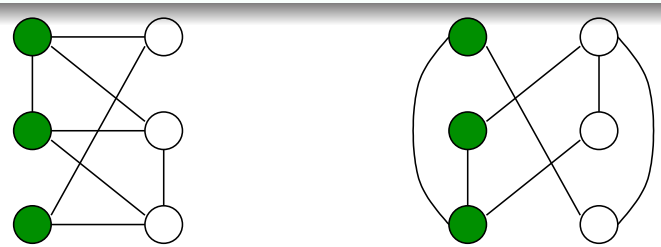
- The respective minimisation problem, i.e. MIN CUT with the bisection requirement, is $\mathbf{NP}$-complete, too.
  (Remember that MIN CUT $\in \mathbf{P}$).
- BISECTION WIDTH: is there a bisection of size K or less?

Theorem

BISECTION WIDTH is $\mathbf{NP}$-complete.

Proof

A reduction from MAX BISECTION. A graph $G = (V, E)$ where $|V| = 2n$ for some n has a bisection of size K or more iff the complement $\overline{G}$ has a bisection of size $n^2 - K$ or less.



General instructions on how to establish NP-completeness

Designing an **NP**-completeness proof for a given problem Q:

- Work on small instances of Q to develop gadgets/primitives.
- Look at known **NP**-complete problems.
- Design a reduction R **from** *a known* **NP***-complete problem* **to** Q.
- Typical ingredients of a reduction:

  choices + consistency + constraints.

- The key question is how to express these in Q?

Graph problems: HAMILTON PATH

Theorem

HAMILTON PATH is $\mathbf{NP}$*-complete.*

Proof.

- Reduction from 3SAT to HAMILTON PATH:
  given a formula ϕ in CNF with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ each with three literals, we construct a graph $R(\phi)$ that has a Hamilton path iff ϕ is satisfiable.
- *Choice gadgets* select a truth assignment for variables x_i.
- *Consistency gadgets* (XOR) enforce that all occurrences of x_i have the same truth value and all occurrences of $\neg x_i$ the opposite.
- *Constraint gadgets* guarantee that all clauses are satisfied.

Gadgets [Papadimitriou, 1994]

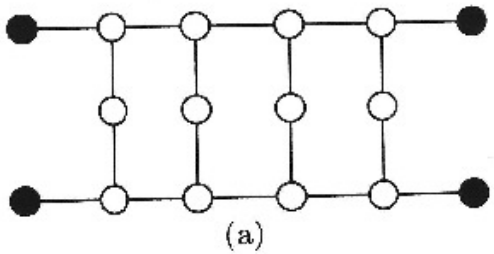


Figure 9-4. The choice gadget.

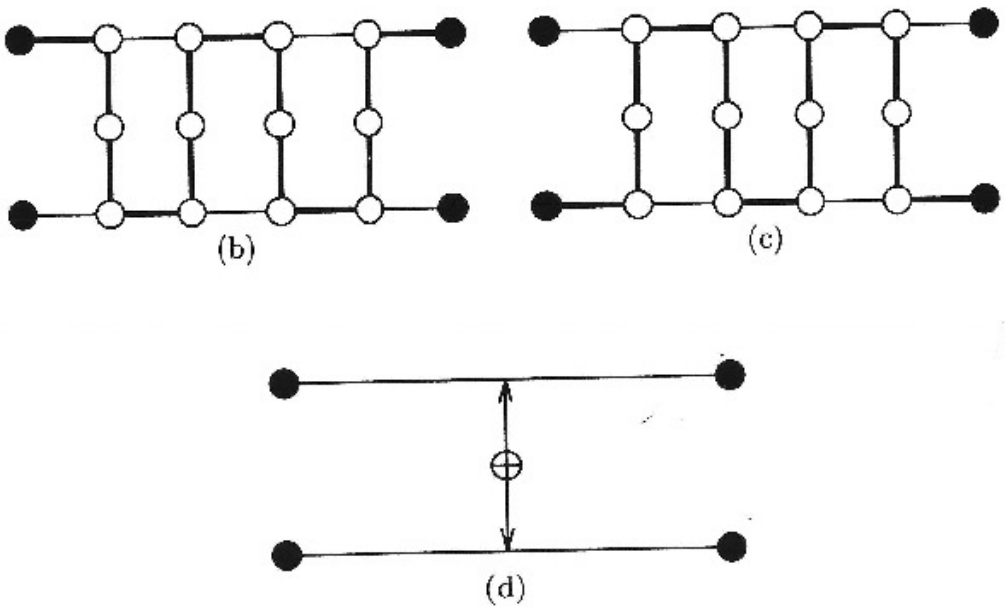


Figure 9-5. The consistency gadget.

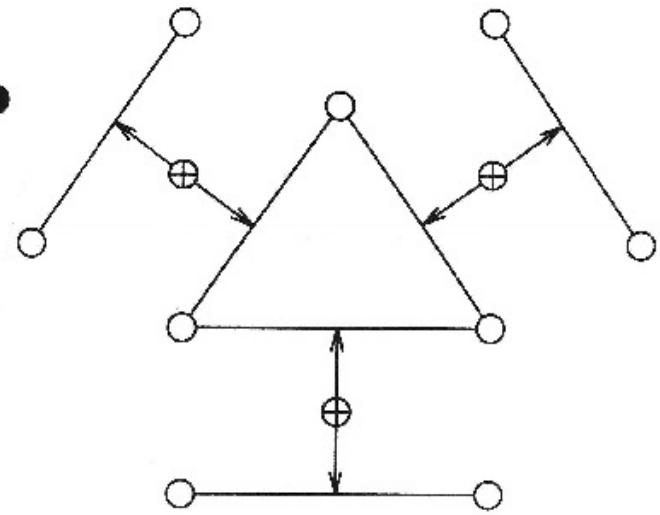


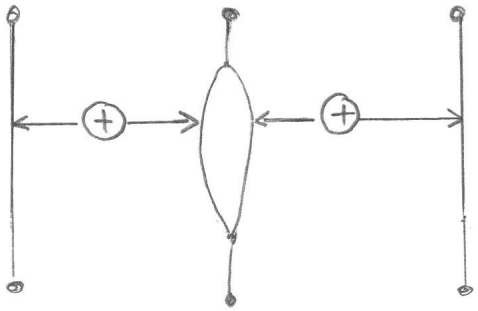
Figure 9-6. The constraint gadget.

Reduction from 3SAT to HAMILTON PATH

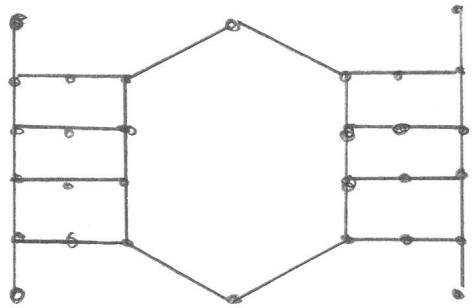
The graph $R(\phi)$ is constructed as follows:

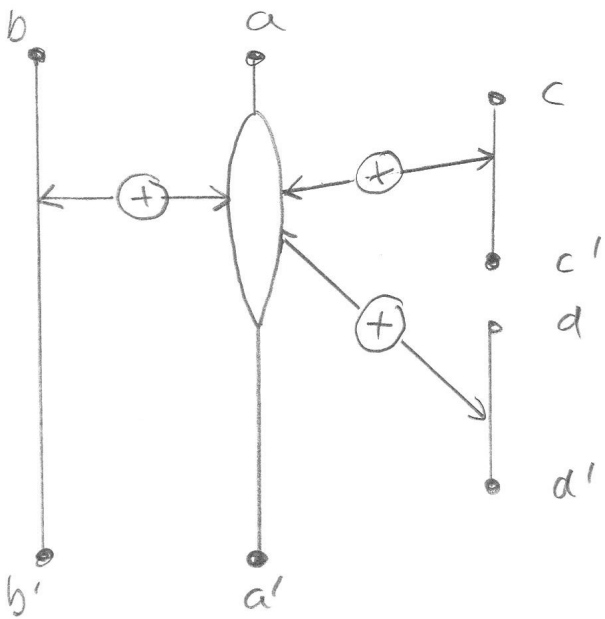
- The *choice gadgets* of variables x_i are connected in series.
- A *constraint gadget* (triangle) for each clause with an edge identified with each literal l in the clause.
  – If l is x_i, then XOR to **true** edge of choice gadget of x_i.
  – If it is $\neg x_i$, then XOR to **false** edge of choice gadget of x_i.
- All vertices of the triangles, the end vertex of choice gadgets and a new vertex 3 form a clique. Add a vertex 2 connected to 3.

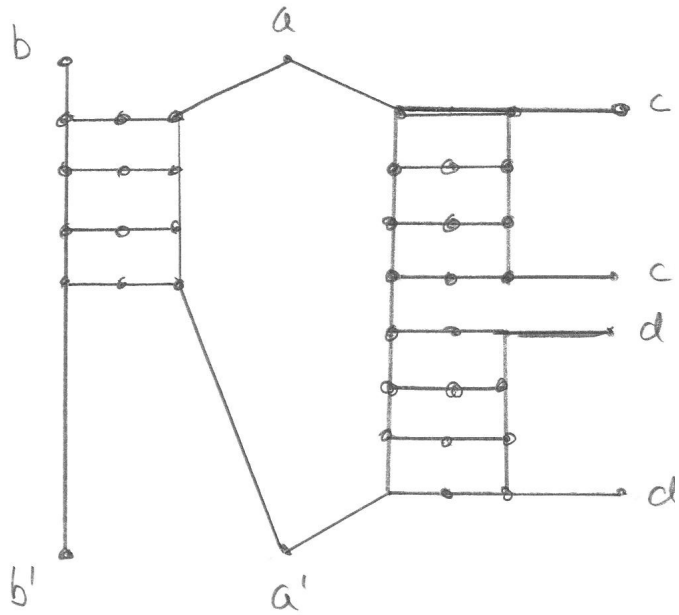
Basic idea: each side of the constraint gadget is traversed by the Hamilton path iff the corresponding literal is **false**. Hence, at least one literal in any clause is **true** since otherwise all sides for its triangle should be traversed which is impossible (implying no Hamilton path).



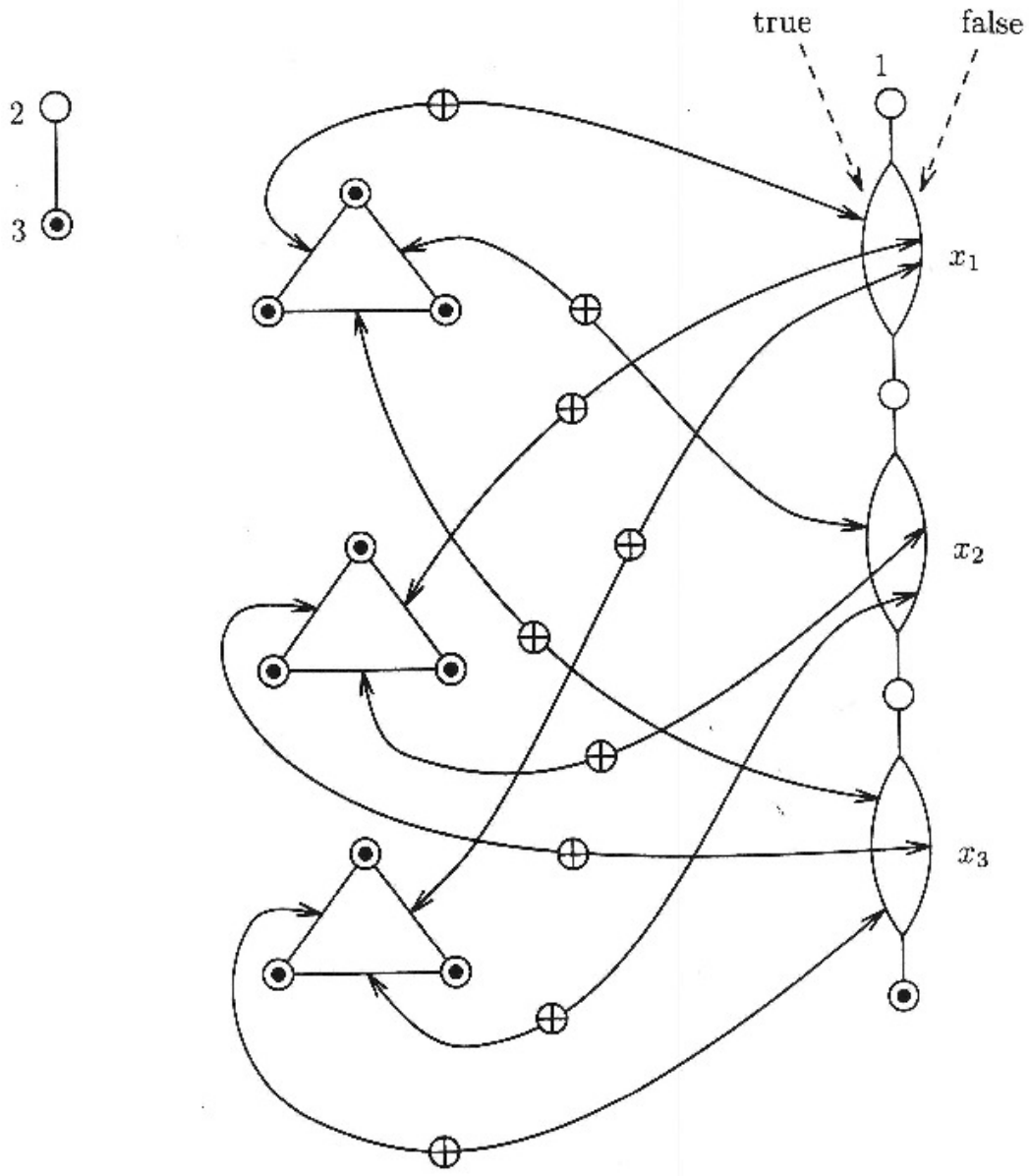
=




b
a
c
c'
d
d'
b'
a'


a
b
c
c
d
d
b'
a'

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$



[Papadimitriou, 1994]

Correctness of the reduction

- If ϕ is satisfiable, there is a Hamilton path:
  From a satisfying truth assignment, we construct a Hamilton path by starting at 1, traversing choice gadgets according to the truth assignment, the rest is a big clique for which a trivial path can be found leading to 3 and then to 2.
- If there is a Hamilton path, ϕ is satisfiable:
  The path starts at 1, makes a truth assignment, traverses the triangles in some order and ends up in 2. The truth assignment satisfies ϕ as there is no triangle where all sides are traversed, i.e., where all literals are **false**. □

Travelling salesperson (TSP) revisited

Corollary

TSP(D) is $\mathbf{NP}$*-complete.*

Proof: A reduction from HAMILTON PATH to TSP(D). Given a graph G with n vertices, construct a distance matrix d_{ij} and a budget B so that there is a tour of length at most B iff G has a Hamilton path.

- There are n cities and the distance $d_{ij} = 1$ if there is $\{i,j\} \in G$ and $d_{ij} = 2$ otherwise. The budget $B = n+1$.
- If there is a tour of length $n+1$ or less, then there is at most one pair $(\pi(i), \pi(i+1))$ in it with cost 2, i.e., a pair for which $\{\pi(i), \pi(i+1)\}$ is not an edge. Removing it gives a Hamilton path.
- If G has a Hamilton path, then its cost is $n-1$ and it can be made into a tour with additional cost of at most 2. $\square$

4. Colouring Problems

Consider the following problem:

k-COLOURING:
INSTANCE: An undirected graph $G = (V, E)$.
QUESTION: Is there an assignment of one of k colours to each of the vertices in V such that no two vertices i, j connected by an edge $\{i, j\} \in E$ have the same colour?

☞ Colouring with $k = 2$ colours is easy (in $\mathbf{P}$) but not when $k = 3$.

Determining the complexity of 3-COLOURING

Theorem

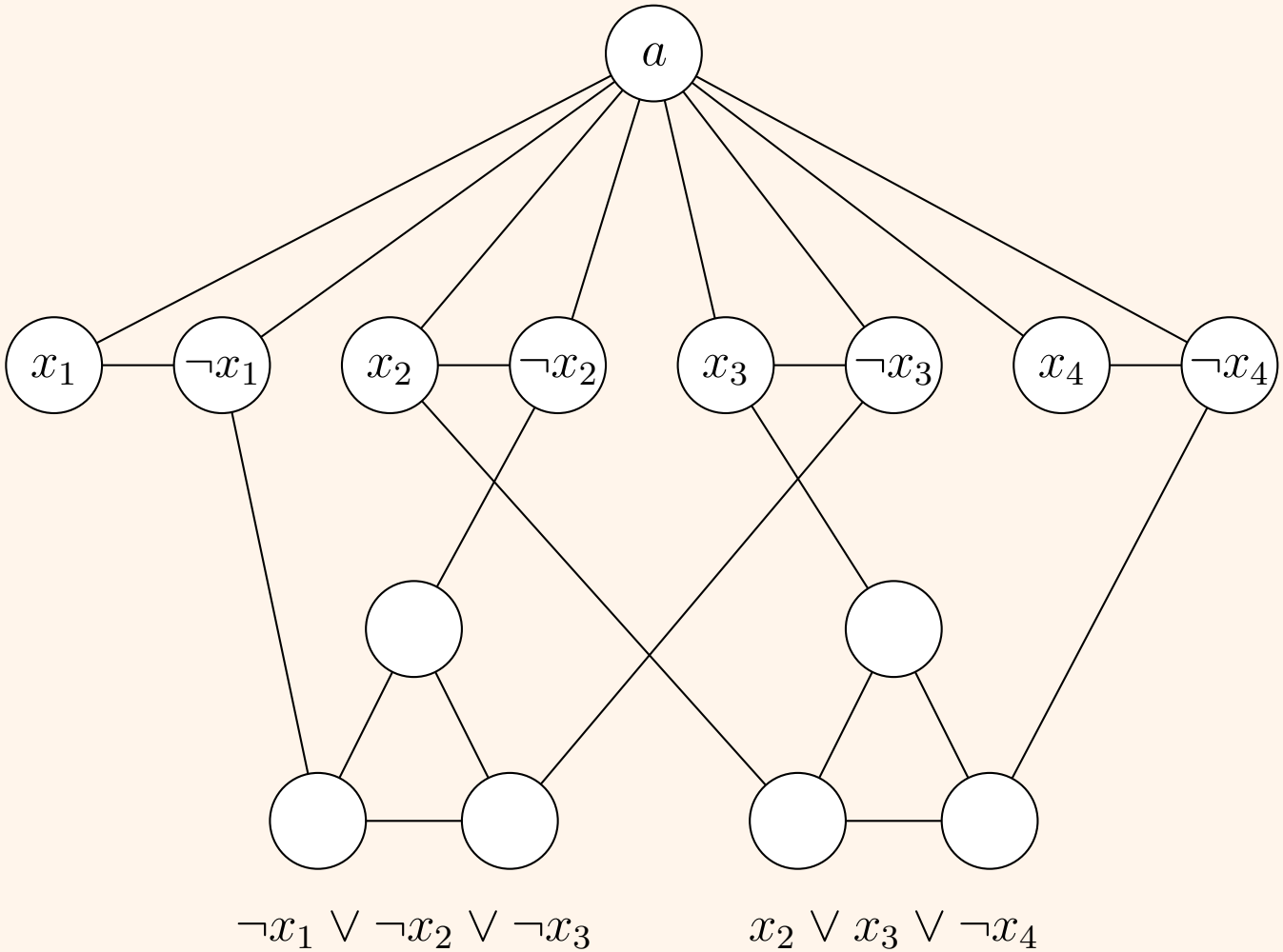
3-COLOURING is $\mathbf{NP}$*-complete.*

Proof. A reduction from NAESAT to 3-COLOURING.

- For a conjunction of clauses $\phi = C_1 \wedge \ldots \wedge C_m$ with variables $x_1, \ldots, x_n$, construct a graph $G(\phi)$ that can be coloured with $\{0,1,2\}$ iff there is a truth assignment satisfying all clauses in ϕ in the NAESAT sense.
- *Choice gadgets:* For each variable x_i, introduce a triangle $[a, x_i, \neg x_i]$, i.e. all triangles share a vertex a.
- *Constraints:* For each clause C_i, introduce a triangle $[C_{i1}, C_{i2}, C_{i3}]$ where each C_{ij} is further connected to the vertex representing the jth literal in C_i.

Example

Reducing NAESAT to 3-COLOURING:



Correctness of the reduction

($\Rightarrow$) Assume that ϕ is satisfied by T in the sense of NAESAT. Then we can extract a colouring for G from T as follows:

1. Vertex a is coloured with colour 2.
2. If $T(x_i) = \textbf{true}$, then colour x_i with 1 and $\neg x_i$ with 0, else vice versa.
3. From each $[C_{i1}, C_{i2}, C_{i3}]$, colour two literals having opposite truth values with 0 (**true**) and 1 (**false**). Colour the third with 2.

($\Leftarrow$) Assume that G can be coloured with $\{0, 1, 2\}$ and a has colour 2. This induces a truth assignment T via the colours of the vertices x_i: if the colour is 1, then $T(x_i) = \textbf{true}$ else $T(x_i) = \textbf{false}$.

Suppose that T assigns all literals of some clause C_i to true/false. Then colour 1/0 cannot be used for colouring $[C_{i1}, C_{i2}, C_{i3}]$, a contradiction.

Thus, ϕ is satisfied in the sense of NAESAT. $\square$

5. Sets and Numbers

Definition

TRIPARTITE MATCHING:
INSTANCE: Three sets B (boys), G (girls), and H (houses) each containing n elements, and a ternary relation $T \subseteq B \times G \times H$.
QUESTION: Is there a set of n triples in T no two of which have a component in common?

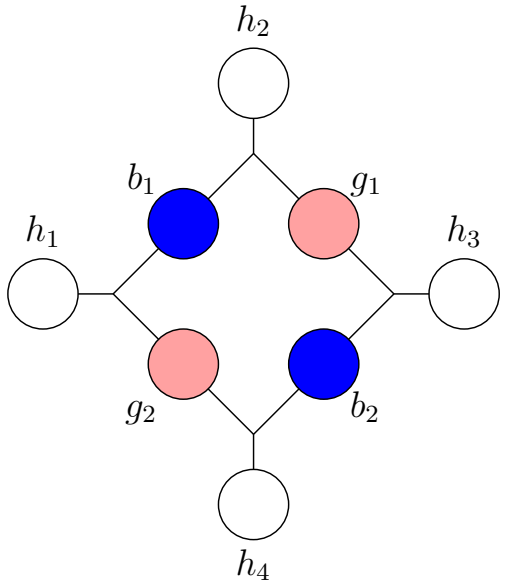
Theorem

TRIPARTITE MATCHING is $\mathbf{NP}$*-complete.*

Proof. By reduction from 3SAT. Each variable x has a combined choice and consistency gadget, and each clause c a dedicated pair of boy b_c and girl g_c, together with three triples (b_c, g_c, h_l) where h_l ranges over the three houses corresponding to the occurrences of literals in the clause (appearing in the combined gadgets).

The combined gadget for choice and consistency

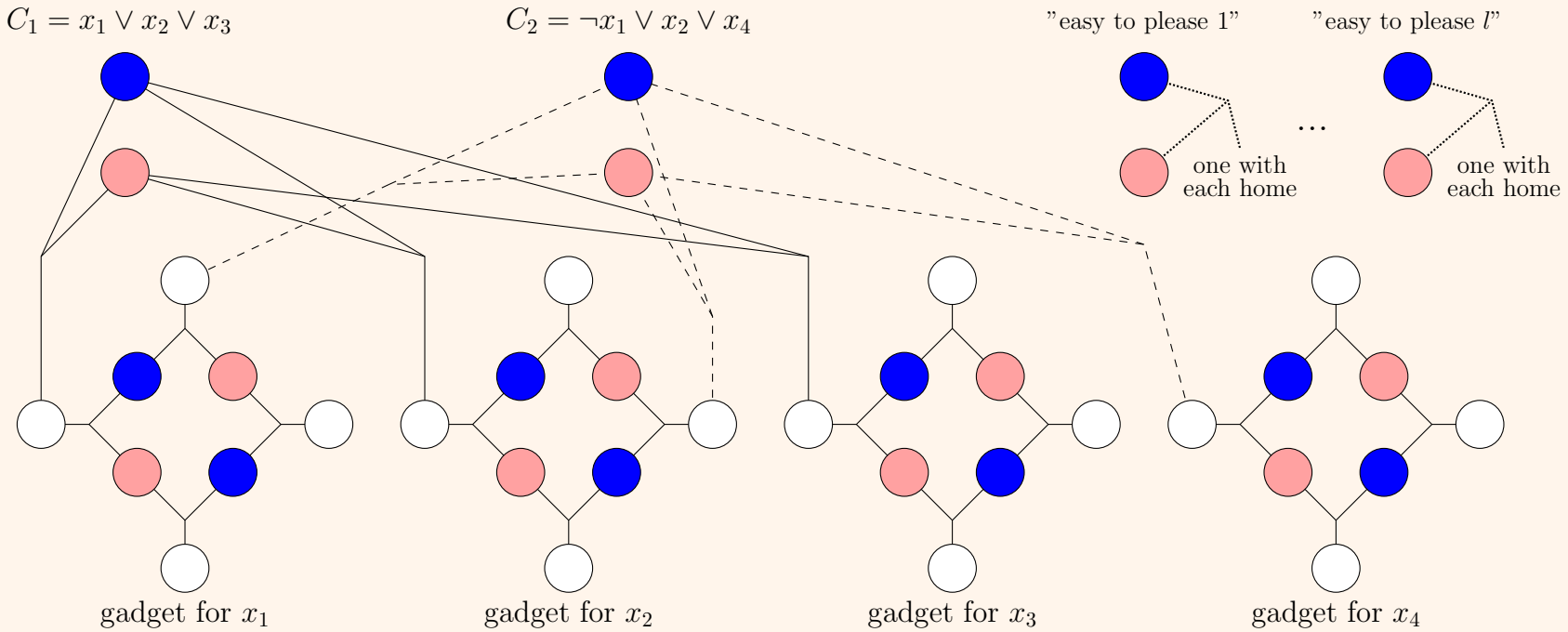
The gadget for a variable x involves k boys, k girls and $2k$ houses forming a "k-circle", where k is either the number of occurrences of x or its negation whichever is larger. (Recall that k can be assumed to equal 2.) The case $k = 2$ is given alongside.



- Occurrences of x in the clauses are connected to the odd houses h_{2i-1} in the variable gadget for x and those of $\neg x$ to the even houses h_{2i}.
- Exactly two kinds of matchings in the variable gadget for x are possible:
  – "$T(x) = \textbf{true}$": each b_i with g_i and h_{2i}.
  – "$T(x) = \textbf{false}$": each b_i with g_{i-1} (g_k if $i = 1$) and h_{2i-1}.

Example

Reducing 3SAT to TRIPARTITE MATCHING:



Correctness of the reduction

- Note that a "$T(x) = \textbf{true}$" matching in the variable gadget for x leaves the odd houses unoccupied, and a "$T(x) = \textbf{false}$" matching respectively the even houses.
- For a clause c, the dedicated b_c and g_c can be matched to a house h in a variable gadget for x that is left unoccupied when x is assigned a truth values satisfying c.
- One more detail needs to be settled: there are now more houses H than boys B and girls G (but $|B| = |G|$).
- Solution: add $l = |H| - |B|$ new boys and l new girls. The ith new girl participates in $|H|$ triples containing the ith new boy and each house.
- Now a tripartite matching exists iff the set of clauses is satisfiable.

Other problems involving sets — classifications obtained by generalisation

Definition

EXACT COVER BY 3-SETS:
INSTANCE: A family $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set U such that $|U| = 3m$ for some integer m and $|S_i| = 3$ for all i.
QUESTION: Is there a subfamily of m sets in F that are disjoint and have U as their union?

Corollary

EXACT COVER BY 3-SETS is $\mathbf{NP}$*-complete.*

Proof sketch

TRIPARTITE MATCHING can be reduced to EXACT COVER BY 3-SETS by noticing that it is a special case where U is partitioned in three sets B, G, H with $|B| = |G| = |H|$ and $F = \{\{b, g, h\} \mid (b, g, h) \in T\}$.

Example

TRIPARTITE MATCHING:
$B = \{b_1, ..., b_n\}, G = \{g_1, ..., g_n\},$
$H = \{h_1, ..., h_n\},$
$T = \{(b_1, g_2, h_1), (b_1, g_2, h_2), ...\}$

EXACT COVER BY 3-SETS:
$U = \{b_1, ..., b_n, g_1, ..., g_n, h_1, ..., h_n\}$
$F = \{\{b_1, g_2, h_1\}, \{b_1, g_2, h_2\}, ...\}$

Definition

SET COVER:
INSTANCE: A family $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set U and an integer B.
QUESTION: Is there a subfamily of B sets in F whose union is U?

Corollary

SET COVER is $\mathbf{NP}$*-complete.*

Proof sketch

EXACT COVER BY 3-SETS can be reduced to SET COVER as a special case where the universe has $3m$ elements, all sets in F have 3 elements and the budget $B = m$.

Example

EXACT COVER BY 3-SETS:
$U = \{e_1, \ldots, e_{93}\}$
$F = \{\{e_1, e_7, e_{11}\}, \{e_2, e_7, e_{11}\}, \ldots\}$

SET COVER:
$U = \{e_1, \ldots, e_{93}\}$
$F = \{\{e_1, e_7, e_{11}\}, \{e_2, e_7, e_{11}\}, \ldots\}$
$B = 31$

Definition

SET PACKING:
INSTANCE: A family $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set U and an integer K.
QUESTION: Is there a subfamily of K pairwise disjoint sets in F?

Corollary

SET PACKING is $\mathbf{NP}$*-complete.*

Proof sketch

EXACT COVER BY 3-SETS can be reduced to SET PACKING as a special case where the universe has $3m$ elements, all sets in F have 3 elements and limit $K = m$.

Example

EXACT COVER BY 3-SETS:
$U = \{e_1, \ldots, e_{93}\}$
$F = \{\{e_1, e_7, e_{11}\}, \{e_2, e_7, e_{11}\}, \ldots\}$

SET PACKING:
$U = \{e_1, \ldots, e_{93}\}$
$F = \{\{e_1, e_7, e_{11}\}, \{e_2, e_7, e_{11}\}, \ldots\}$
$K = 31$

A number problem: INTEGER PROGRAMMING

INSTANCE: a system of linear inequalities with integer coefficients.

QUESTION: Is there an integer solution of the system?

Corollary

INTEGER PROGRAMMING is $\mathbf{NP}$*-complete.*

Proof

SET COVER is reducible to INTEGER PROGRAMMING:

Given a family $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set $U = \{u_1, \ldots, u_m\}$ and a integer B, construct a system:

$$0 \leq x_1 \leq 1, \ldots, 0 \leq x_n \leq 1 \qquad a_{11}x_1 + \cdots + a_{1n}x_n \geq 1$$
$$\Sigma_{i=1}^{n} x_i \leq B \qquad \vdots$$
$$a_{m1}x_1 + \cdots + a_{mn}x_n \geq 1$$

where $a_{ij} = 1$ if ith element of U is in the set S_j, otherwise $a_{ij} = 0$. (The idea: in a solution $x_i = 1$ if S_i in the cover and otherwise $x_i = 0$.)

Example

Reducing SET COVER to INTEGER PROGRAMMING

SET COVER:

$U = \{e_1, ..., e_{93}\}$

$F = \{S_1 = \{e_1, e_7, e_{11}\}, S_2 = \{e_2, e_7, e_{11}\}, ..., S_{50} = \{e_2, e_{12}\}\}$

$B = 35$

reduces to

INTEGER PROGRAMMING:

$$
\begin{aligned}
x_1 + 0 + ... + 0 &\geq 1 \quad \text{(for } e_1\text{)} \\
0 + x_2 + ... + x_{50} &\geq 1 \quad \text{(for } e_2\text{)} \\
&\vdots \\
x_1 + x_2 + ... + 0 &\geq 1 \quad \text{(for } e_7\text{)} \\
&\vdots
\end{aligned}
$$

$0 \leq x_1 \leq 1, ..., 0 \leq x_{50} \leq 1$

$\sum_{i=1}^{50} x_i \leq 35$

Further problems involving numbers

1. LINEAR PROGRAMMING (i.e. INTEGER PROGRAMMING where non-integer solutions are allowed) is in $\mathbf{P}$.
2. KNAPSACK:
   INSTANCE: A set of n items with each item i having a value v_i and a weight w_i (both positive integers) and integers W and K.
   QUESTION: Is there a subset S of the items such that $\Sigma_{i \in S} w_i \leq W$ but $\Sigma_{i \in S} v_i \geq K$?

Theorem

KNAPSACK is $\mathbf{NP}$-complete.

Proof. We show that a simple special case of KNAPSACK is $\mathbf{NP}$-complete where $v_i = w_i$ for all i and $W = K$:
INSTANCE: A set of integers $w_1, \ldots, w_n$ and an integer K.
QUESTION: Is there a subset S of the integers with $\Sigma_{i \in S} w_i = K$?

Reduction from EXACT COVER BY 3-SETS

The reduction is based on the set $U = \{1, 2, \ldots, 3m\}$ and the sets $S_1, \ldots, S_n$ given as bit vectors $\{0,1\}^{3m}$ and $K = 2^{3m} - 1$. Then the task is to find a subset of bit vectors that sum to K.

$$
\begin{array}{lccccc}
\rightarrow & 0 & 1 & \ldots & 0 & 0 \\
 & 1 & 0 & \ldots & 0 & 0 \\
 & \vdots & & & & \\
\rightarrow & 0 & 0 & \ldots & 1 & 1 \\
\hline
 & 1 & 1 & \ldots & 1 & 1
\end{array}
$$

- This does not quite work because of the carry bit, but the problem can be circumvented by using $n+1$ as the base rather than 2.
- Now each S_i corresponds to $w_i = \Sigma_{j \in S_i} (n+1)^{3m-j}$.
- Then a set of these integers w_i adds up to $K = \sum_{j=0}^{3m-1} (n+1)^j$ iff there is an exact cover among $\{S_1, S_2, \ldots, S_n\}$. $\square$

Example

Reducing EXACT COVER BY 3-SETS to KNAPSACK

EXACT COVER BY 3-SETS:

$U = \{e_1, ..., e_6\}$

$F = \{S_1 = \{e_1, e_4, e_6\}, S_2 = \{e_1, e_3, e_6\}, S_3 = \{e_2, e_3, e_5\}\}$

reduces to

KNAPSACK:

Integers

$$w_1 = 1 \cdot 4^{6-6} + 0 \cdot 4^{6-5} + 1 \cdot 4^{6-4} + 0 \cdot 4^{6-3} + 0 \cdot 4^{6-2} + 1 \cdot 4^{6-1} = 1041$$
$$w_2 = 1 \cdot 4^{6-6} + 0 \cdot 4^{6-5} + 0 \cdot 4^{6-4} + 1 \cdot 4^{6-3} + 0 \cdot 4^{6-2} + 1 \cdot 4^{6-1} = 1089$$
$$w_3 = 0 \cdot 4^{6-6} + 1 \cdot 4^{6-5} + 0 \cdot 4^{6-4} + 1 \cdot 4^{6-3} + 1 \cdot 4^{6-2} + 0 \cdot 4^{6-1} = 324$$
$$K = 4^0 + 4^1 + 4^2 + 4^3 + 4^4 + 4^5 = 1365$$

6. Pseudopolynomial Algorithms

Proposition

Any instance of KNAPSACK can be solved in $O(nW)$ time where n is the number of items and W is the weight limit.

Proof

- Define $V(w,i)$: the largest value attainable be selecting some among the first i items so that their total weight is exactly w.
- Each $V(w,i)$ with $w = 1,\ldots,W$ and $i = 1,\ldots,n$ can be computed by

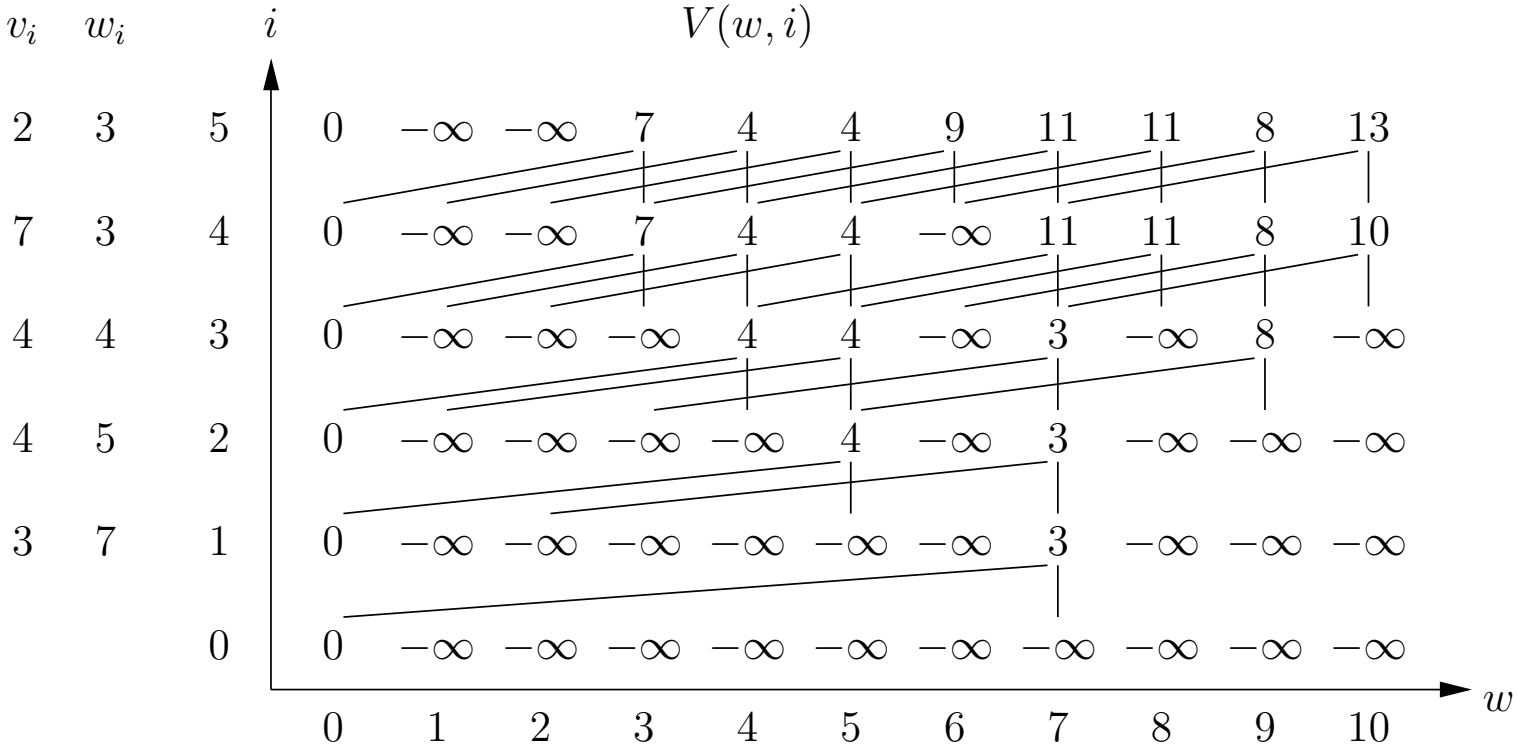
$$V(w,i+1) = \max\{V(w,i), v_{i+1} + V(w - w_{i+1}, i)\}$$

  where $V(w,i) = -\infty$ if $w \leq 0$, $V(0,i) = 0$ for all i, and $V(w,0) = -\infty$ if $w \geq 1$.
- For each entry this can be done in constant number of steps and there are nW entries. Hence, the algorithm runs in $O(nW)$ time.
- An instance is answered "yes" iff there is an entry $V(w,i) \geq K$.

Pseudopolynomial algorithm for KNAPSACK: example

Items $\{(v_1 = 3, w_1 = 7), (v_2 = 4, w_2 = 5), (v_3 = 4, w_3 = 4), (v_4 = 7, w_4 = 3), (v_5 = 2, w_5 = 3)\}$

weight limit $W = 10$, capacity limit $K = 12$



Strong NP-completeness

- The preceding algorithm is not polynomial w.r.t. the length of the input (which is $O(n \log W)$) but exponential ($W = 2^{\log W}$).
- An algorithm where the time bound is polynomial in the integers in the input (not their logarithms) is called *pseudopolynomial*.
- A problem is called **strongly NP-complete** if the problem remains **NP**-complete even if any instance of length n is restricted to contain integers of size (i.e. "value") at most $p(n)$, for a polynomial p.
  ☞ Strongly **NP**-complete problems cannot have pseudopolynomial algorithms (unless $\mathbf{P} = \mathbf{NP}$).
- SAT, MAX CUT, TSP(D), HAMILTON PATH, ... are strongly **NP**-complete but KNAPSACK is not.

Yet another number problem: BIN PACKING

Definition

BIN PACKING
INSTANCE: N positive integers $a_1, \ldots, a_N$ (items) and integers C (capacity) and B (number of bins).
QUESTION: Is there a partition of the numbers into B subsets such that for each subset S, $\Sigma_{a_i \in S} a_i \leq C$?

- BIN PACKING is strongly $\mathbf{NP}$-complete:
  Even if the integers are restricted to have polynomial values (w.r.t. the length of input), BIN PACKING remains $\mathbf{NP}$-complete.
  For the proof, see the pages 204–205 in Papadimitriou's book.
- Any pseudopolynomial algorithm for BIN PACKING would yield a polynomial algorithm for all problems in $\mathbf{NP}$ implying $\mathbf{P} = \mathbf{NP}$.

Learning Objectives

- The concept of $\mathbf{NP}$-completeness and its characterisations in terms of succinct certificates.
- You should know basic techniques to prove problems $\mathbf{NP}$-complete and be able to construct such proofs on your own.
- A basic repertoire of $\mathbf{NP}$-complete problems (related with satisfiability, graphs, sets, and numbers) to be used in further $\mathbf{NP}$-completeness proofs.
- The definition of strong $\mathbf{NP}$-completeness and awareness of number problems which are (not) strongly $\mathbf{NP}$-complete.