Lecture 8: NP-complete problems

Consider HAMILTON PATH: "Is there a path that visits each node *exactly* once?"

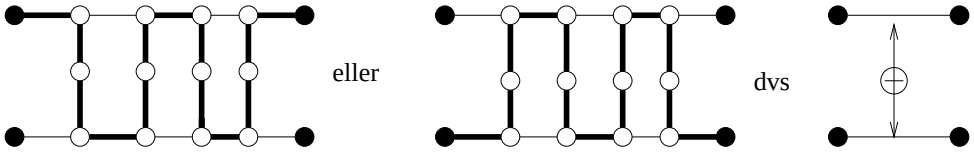
HAMILTON PATH is **NP**-complete

Proof: Reduction from 3SAT. Given 3SAT instance φ construct HAM PATH graph $R(\varphi)$.

We shall express the three properties of 3SAT as graph elements:

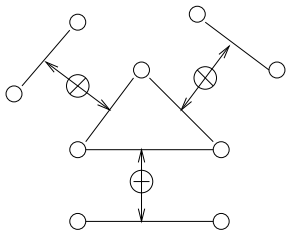
1. `choice` between true and false

2. `consistency` : all x have the same value, and $\neg x$ the opposite value



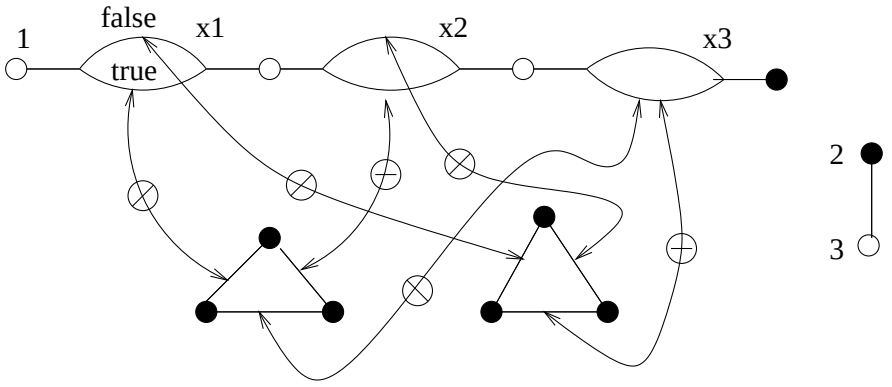
propagates choice of value to all occurrences. This xor gadget can be traversed in two ways.

3. `restriction` of at most three literals/clause



Suppose we have made sure that each side of a triangle is traversed by a Hamilton path iff the corresponding literal is false. Then at least one literal is true, since otherwise all three sides are traversed and there is no Hamilton path.

Example: $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$ gives graph where nodes $\bullet$ makes up a large clique:



Hamilton path ($1 \to 2 \to 3$) exists $\Leftrightarrow$ φ is satisfiable

$\Rightarrow$) From node 1, one of two parallel edges for each choice must be traversed. If a choice edge is connected to an xor gadget these nodes will be visited. After the chain of choices have been made, the path continues to one of the triangle nodes. Provided that at least one side of each triangle does not have to be traversed we have have a Hamilton path, i.e. provided an xor gadget has been visited during the choice setting phase, which corresponds to an assignment that satisfies φ.

$\Leftarrow$) Similarly, a satisfying assignment gives a Hamilton path that starts in node 1. $\square$

TSP(D) is **NP**-complete

Proof: Reduction from HAMILTON PATH. Given graph G with n nodes let $B = n + 1$ and define edge weights:

$$d_{ij} = \begin{cases} 1 & \text{if edge } [i,j] \in G \\ 2 & \text{otherwise} \end{cases}$$

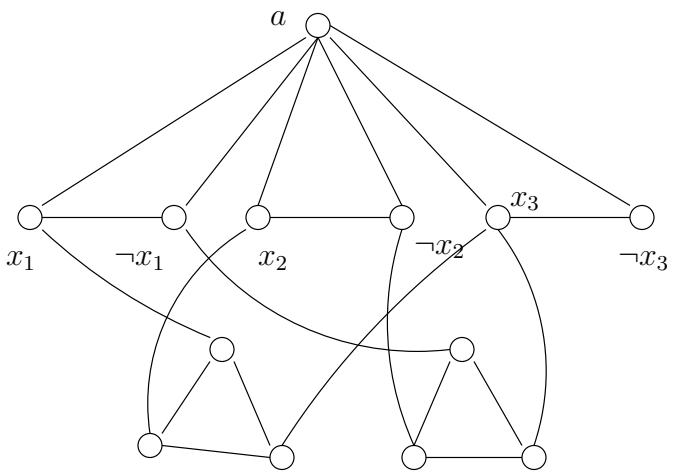
A tour in the complete graph with edge sum $\leq n + 1$ exists $\Leftrightarrow$ A Hamilton path in G exists. At most one edge $\notin G$ can be part of a tour with n edges. $\square$

3-COLORING: Color graph nodes with three colors so that adjacent nodes have different colors.

3-COLORING is **NP**-complete

Proof: Reduction from NAESAT. Clauses $C_i = (\alpha \vee \beta \vee \gamma)$ in NAESAT-instance φ correspond to triangles $[\alpha, \beta, \gamma]$ in graph G. A triangle node has edges to its literal node x_i, and for all possible variables there are triangles $[a, x_i, \neg x_i]$, all with a common node a.

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



If G is 3-colorable, then color a with "2", and let each x_i and $\neg x_i$ be colored "0" (true) or "1" (false).

How can a clause triangle be colored?

If all its literals are true, then the triangle cannot be colored with three colors. Neither can it be colored with three colors if all literals are false. Otherwise it can, using color "2" as well.

If there is a satisfying assignment (in NAESAT), then color a with "2" and variable triangles according to the assignment. In clause triangles color two nodes with different truth values (which exist if the clause is satisfiable), i.e. with "0" and "1", and color the third node "2". □

We can generalize bipartite matching (which $\in$ **P**) to tripartite matching. Given three sets connect one from each to unique triples. By reduction from 3SAT one can prove:

TRIPARTITE MATCHING (3-MATCHING) is **NP**-complete

SET COVERING: Given $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set U, and a budget B, are there B sets in F whose union is U?

EXACT COVER BY 3-SETS: Given is $F = \{S_1, \ldots, S_n\}$ of subsets of U, such that $|U| = 3m$ for some m, and $|S_i| = 3,\ \forall i$. Are there m disjoint S_i whose union is U?

SET COVERING and EXACT COVER BY 3-SETS are **NP**-complete

Proof: Both belong to **NP** and are generalizations of 3-MATCHING. □

INTEGER PROGRAMMING looks for a solution in integers for a system of linear inequalities with n variables and integer coefficients.

Example: $x_1 + 2x_2 \leq 3,\ 3x_2 + x_3 \leq 4$, with object function $2x_1 + x_2 + 3x_3$ to maximize (or decide if $\geq K$ for yes/no answer).

INTEGER PROGRAMMING is **NP**-complete (since **NP**-complete problems can be formulated as instances of INTEGER PROGRAMMING; we give example on tutorial 4).

But LINEAR PROGRAMMING (where the solutions to the system of equations do not have to be integers) $\in$ **P**, a much publicized result (Khachiyan, 1979 and Karmarkar, 1984).

Consider KNAPSACK (special case of INTEGER PROGRAMMING): Given n elements with values v_i and weights w_i (positive integers), and a maximum capacity W. Find a subset S of elements such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i$ is maximized.

Yes/no version: Is there a choice such that $\sum w_i \leq W$ and $\sum v_i \geq K$?

KNAPSACK is **NP**-complete

Proof: Restrict to special case: $v_i = w_i$ and $K = W$, i.e. look for $\sum w_i = K$.

Reduction from EXACT COVER BY 3-SETS: Given $\{S_1, \ldots S_n\},\ |S_i| = 3$, are there disjoint sets which cover $\{1, \ldots, 3m\}$? First, think of S_i as bit vectors in $\{0,1\}^{3m}$. These can be viewed as binary numbers, so that set union is similar to integer addition (Figure 9.10).

Problem: Adding bit vectors can result in carry which does not correspond to union of disjoint sets.

Solution: If S_i are vectors in base $(n+1)$ the values become unique. S_i then corresponds to

$$w_i = \sum_{j \in S_i} (n+1)^{3m-j}$$

For instance, $S_i = \{j_1, j_2, j_3\}$ corresponds to $(n+1)^{3m-j_1} + (n+1)^{3m-j_2} + (n+1)^{3m-j_3}$, where $j_k \in \{1, \ldots, 3m\}$.

This implies that $S_i \neq S_j + S_k,\ \forall i, j, k$, i.e. there is no carry when $\leq n$ of these numbers are added.

If a subset of these add to $K = \sum_{j=0}^{3m-1} (n+1)^j$ it is equivalent to that there is an exact cover among $\{S_1, \ldots S_n\}$. □

But KNAPSACK can be solved in $O(nW)$ time, by filling a table $V(1:W, 0:n)$.

Let $V(w,i)$ be the largest $\sum_j v_i$ of $1 \leq j \leq i$ so that $\sum_j w_i = w$.

Initialize $V(w,0) = 0,\ \forall w$, and compute $V(w,i+1) = \max\{V(w,i), v_{i+1} + V(w - w_{i+1}, i)\}$, where $V(w,i)$ means full knapsack without element $(i+1)$, while $(v_{i+1} + V(w - w_{i+1}, i))$ includes it plus a maximum subset of elements 1 to i fitting the remaining capacity of the knapsack.

If any table entry $\geq K$, the answer is yes. $\square$

Since nW is **not** a polynomial in the length of input, which is $\Theta(n \log W)$, the algorithm is said to be *pseudo-polynomial*.

The proof that KNAPSACK is **NP**-complete required exponentially large numbers in the reduction. Previous **NP**-complete problems we proved by reductions which only constructed polynomially small integers. These problems are said to be *strongly* **NP**-complete, and have no pseudopolynomial algorithms (unless $\mathbf{P} = \mathbf{NP}$).

Another strongly **NP**-complete problem is BIN PACKING: Given $n+2$ positive integers $a_1, \ldots, a_n$ (the elements), B (number of bins), and C (capacity of bin), do the elements fit in the B bins? By reduction from 3-MATCHING one can prove:

BIN PACKING is **NP**-complete