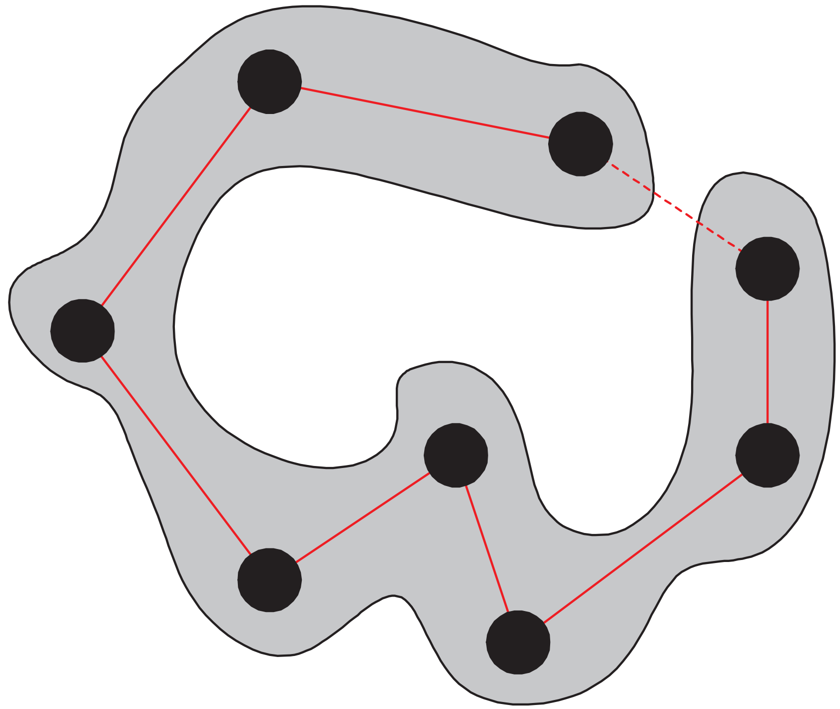
TSP (D) Is NP-Complete

Corollary 40 TSP (D) *is NP-complete.*

- Consider a graph G with n nodes.
- Define $d_{ij} = 1$ if $[\, i, j\,] \in G$ and $d_{ij} = 2$ if $[\, i, j\,] \notin G$.
- Set the budget $B = n + 1$.
- Suppose G has no Hamiltonian paths.
- Then every tour on the new graph must contain at least two edges with weight 2.
  - Otherwise, by removing up to one edge with weight 2, one obtains a Hamiltonian path, a contradiction.



TSP (D) Is NP-Complete (concluded)

- The total cost is then at least $(n-2)+2\cdot 2=n+2>B$.
- On the other hand, suppose G has Hamiltonian paths.
- Then there is a tour on the new graph containing at most one edge with weight 2.
- The total cost is then at most $(n-1)+2=n+1=B$.
- We conclude that there is a tour of length B or less if and only if G has a Hamiltonian path.

Graph Coloring

- k-COLORING: Can the nodes of a graph be colored with $\leq k$ colors such that no two adjacent nodes have the same color?
- 2-COLORING is in P (why?).
- But 3-COLORING is NP-complete (see next page).
- k-COLORING is NP-complete for $k \geq 3$ (why?).
- EXACT-k-COLORING asks if the nodes of a graph can be colored using exactly k colors.
- It remains NP-complete for $k \geq 3$ (why?).

3-COLORING Is NP-Complete[a]

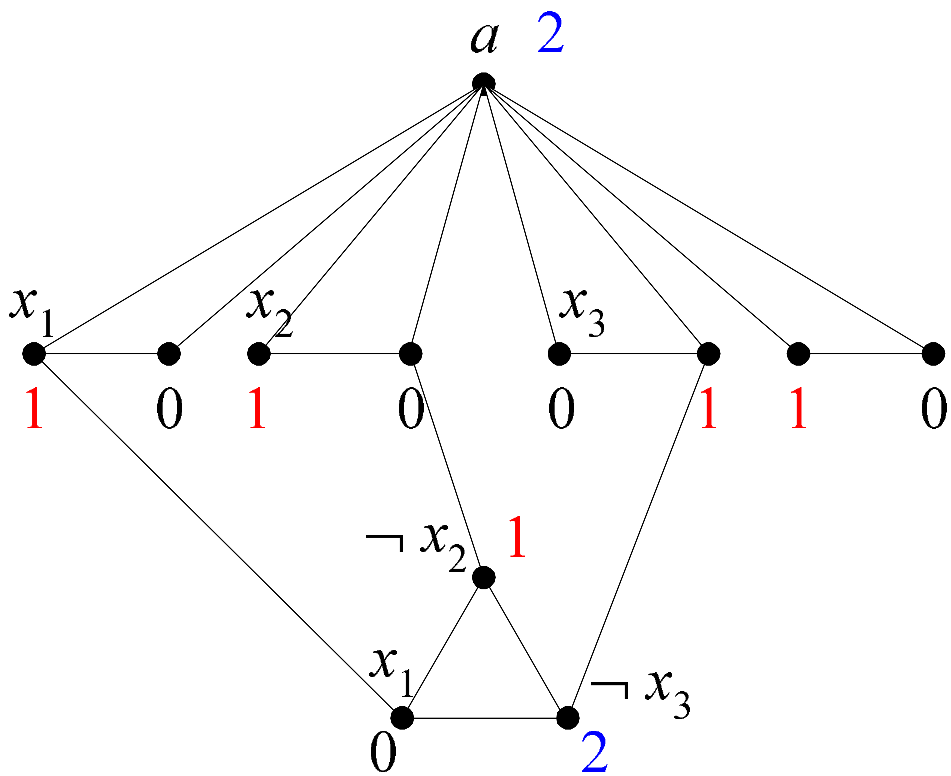
- We will reduce NAESAT to 3-COLORING.
- We are given a set of clauses $C_1, C_2, \ldots, C_m$ each with 3 literals.
- The boolean variables are $x_1, x_2, \ldots, x_n$.
- We shall construct a graph G such that it can be colored with colors $\{0, 1, 2\}$ if and only if all the clauses can be NAE-satisfied.

[a] Karp (1972).

The Proof (continued)

- Every variable x_i is involved in a triangle $[a, x_i, \neg x_i]$ with a common node a.
- Each clause $C_i = (c_{i1} \vee c_{i2} \vee c_{i3})$ is also represented by a triangle
$$[c_{i1}, c_{i2}, c_{i3}].$$
  - Node c_{ij} with the same label as one in some triangle $[a, x_k, \neg x_k]$ represent *distinct* nodes.
- There is an edge between c_{ij} and the node that represents the jth literal of C_i.

Construction for $\cdots \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge \cdots$



The Proof (continued)

Suppose the graph is 3-colorable.

- Assume without loss of generality that node a takes the color 2.
- A triangle must use up all 3 colors.
- As a result, one of x_i and $\neg x_i$ must take the color 0 and the other 1.

The Proof (continued)

- Treat 1 as `true` and 0 as `false`.[a]
  - We were dealing only with those triangles with the a node, not the clause triangles.
- The resulting truth assignment is clearly contradiction free.
- As each clause triangle contains one color 1 and one color 0, the clauses are NAE-satisfied.

[a]The opposite also works.

The Proof (continued)

Suppose the clauses are NAE-satisfiable.

- Color node a with color 2.
- Color the nodes representing literals by their truth values (color 0 for `false` and color 1 for `true`).
  - We were dealing only with those triangles with the a node, not the clause triangles.

The Proof (concluded)

- For each clause triangle:
  - Pick any two literals with opposite truth values.
  - Color the corresponding nodes with 0 if the literal is `true` and 1 if it is `false`.
  - Color the remaining node with color 2.
- The coloring is legitimate.
  - If literal w of a clause triangle has color 2, then its color will never be an issue.
  - If literal w of a clause triangle has color 1, then it must be connected up to literal w with color 0.
  - If literal w of a clause triangle has color 0, then it must be connected up to literal w with color 1.

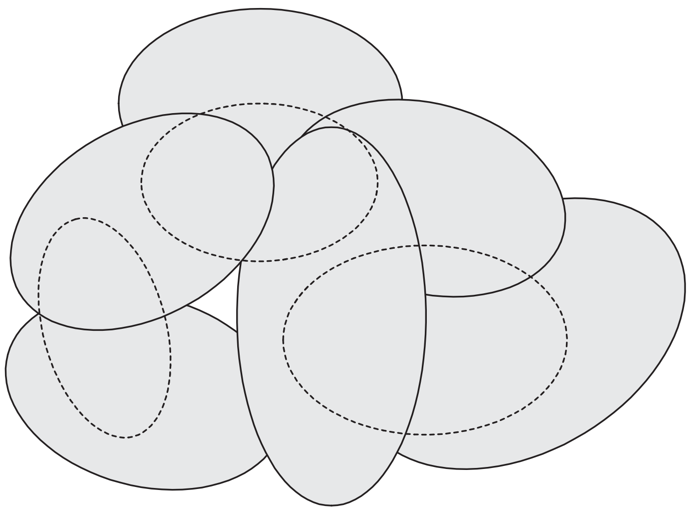
TRIPARTITE MATCHING

- We are given three sets B, G, and H, each containing n elements.
- Let $T \subseteq B \times G \times H$ be a ternary relation.
- TRIPARTITE MATCHING asks if there is a set of n triples in T, none of which has a component in common.
  - Each element in B is matched to a different element in G and different element in H.

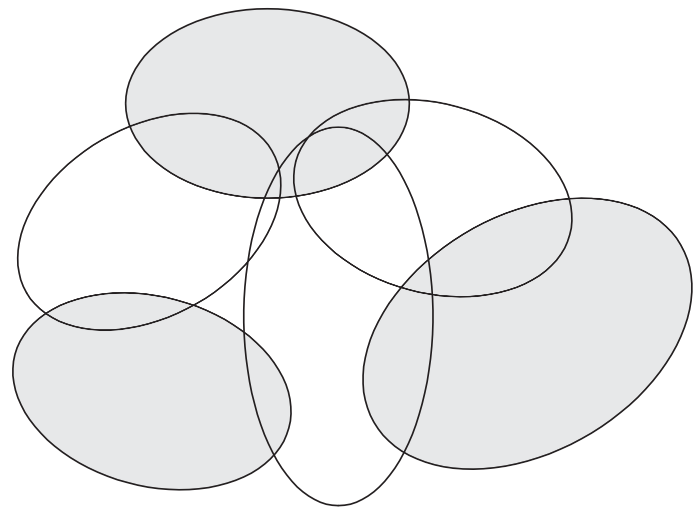
Theorem 41 (Karp (1972)) TRIPARTITE MATCHING *is NP-complete.*

Related Problems

- We are given a family $F = \{S_1, S_2, \ldots, S_n\}$ of subsets of a finite set U and a budget B.
- SET COVERING asks if there exists a set of B sets in F whose union is U.
- SET PACKING asks if there are B disjoint sets in F.
- Assume $|U| = 3m$ for some $m \in \mathbb{N}$ and $|S_i| = 3$ for all i.
- EXACT COVER BY 3-SETS asks if there are m sets in F that are disjoint and have U as their union.



SET COVERING



SET PACKING

Related Problems (concluded)

Corollary 42 SET COVERING, SET PACKING, *and* EXACT COVER BY 3-SETS *are all NP-complete.*

The KNAPSACK Problem

- There is a set of n items.
- Item i has value $v_i \in \mathbb{Z}^+$ and weight $w_i \in \mathbb{Z}^+$.
- We are given $K \in \mathbb{Z}^+$ and $W \in \mathbb{Z}^+$.
- KNAPSACK asks if there exists a subset $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i \geq K$.
  - We want to achieve the maximum satisfaction within the budget.

KNAPSACK Is NP-Complete

- KNAPSACK $\in$ NP: Guess an S and verify the constraints.
- We assume $v_i = w_i$ for all i and $K = W$.
- KNAPSACK now asks if a subset of $\{v_1, v_2, \ldots, v_n\}$ adds up to exactly K.
  - Picture yourself as a radio DJ.
  - Or a person trying to control the calories intake.
- We shall reduce EXACT COVER BY 3-SETS to KNAPSACK.

The Proof (continued)

- We are given a family $F = \{S_1, S_2, \ldots, S_n\}$ of size-3 subsets of $U = \{1, 2, \ldots, 3m\}$.
- EXACT COVER BY 3-SETS asks if there are m disjoint sets in F that cover the set U.
- Think of a set as a bit vector in $\{0, 1\}^{3m}$.
  - `001100010` means the set $\{3, 4, 8\}$, and `110010000` means the set $\{1, 2, 5\}$.
- Our goal is $\overbrace{\texttt{11} \cdots \texttt{1}}^{3m}$.

The Proof (continued)

- A bit vector can also be considered as a binary *number*.
- Set union resembles addition.
  - $\texttt{001100010} + \texttt{110010000} = \texttt{111110010}$, which denotes the set $\{1, 2, 3, 4, 5, 8\}$, as desired.
- Trouble occurs when there is *carry*.
  - $\texttt{001100010} + \texttt{001110000} = \texttt{010010010}$, which denotes the set $\{2, 5, 8\}$, not the desired $\{3, 4, 5, 8\}$.

The Proof (continued)

- Carry may also lead to a situation where we obtain our solution $\texttt{11}\cdots\texttt{1}$ with more than m sets in F.
  - $\texttt{001100010} + \texttt{001110000} + \texttt{101100000} + \texttt{000001101} = \texttt{111111111}$.
  - But this "solution" $\{1, 3, 4, 5, 6, 7, 8, 9\}$ does not correspond to an exact cover.
  - And it uses 4 sets instead of the required $m = 3$.[a]
- To fix this problem, we enlarge the base just enough so that there are no carries.
- Because there are n vectors in total, we change the base from 2 to $n + 1$.

[a] Thanks to a lively class discussion on November 20, 2002.

The Proof (continued)

- Set v_i to be the $(n+1)$-ary number corresponding to the bit vector encoding S_i.
- Now in base $n+1$, if there is a set S such that $\sum_{v_i \in S} v_i = \overbrace{11 \cdots 1}^{3m}$, then every bit position must be contributed by exactly one v_i and $|S| = m$.
- Finally, set

$$K = \sum_{j=0}^{3m-1} (n+1)^j = \overbrace{11 \cdots 1}^{3m} \quad \text{(base } n+1\text{)}.$$

The Proof (continued)

- Suppose F admits an exact cover, say $\{S_1, S_2, \ldots, S_m\}$.
- Then picking $S = \{v_1, v_2, \ldots, v_m\}$ clearly results in

$$v_1 + v_2 + \cdots + v_m = \overbrace{11 \cdots 1}^{3m}.$$

  - It is important to note that the meaning of addition (+) is independent of the base.[a]
  - It is just regular addition.
  - But a S_i may give rise to different v_i's under different bases.

---

[a] Contributed by Mr. Kuan-Yu Chen (`R92922047`) on November 3, 2004.

The Proof (concluded)

- On the other hand, suppose there exists an S such that $\sum_{v_i \in S} v_i = \overbrace{11 \cdots 1}^{3m}$ in base $n+1$.
- The no-carry property implies that $|S| = m$ and $\{S_i : v_i \in S\}$ is an exact cover.

An Example

- Let $m = 3$, $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and

$$\begin{aligned} S_1 &= \{1, 3, 4\}, \\ S_2 &= \{2, 3, 4\}, \\ S_3 &= \{2, 5, 6\}, \\ S_4 &= \{6, 7, 8\}, \\ S_5 &= \{7, 8, 9\}. \end{aligned}$$

- Note that $n = 5$, as there are 5 S_i's.

An Example (concluded)

- Our reduction produces

$$
\begin{aligned}
K &= \sum_{j=0}^{3\times 3-1} 6^j = \overbrace{11\cdots 1}^{3\times 3} \quad \text{(base 6)} = 2015539,\\
v_1 &= \texttt{101100000} = 1734048,\\
v_2 &= \texttt{011100000} = 334368,\\
v_3 &= \texttt{010011000} = 281448,\\
v_4 &= \texttt{000001110} = 258,\\
v_5 &= \texttt{000000111} = 43.
\end{aligned}
$$

- Note $v_1 + v_3 + v_5 = K$.
- Indeed, $S_1 \cup S_3 \cup S_5 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, an exact cover by 3-sets.

BIN PACKING

- We are given N positive integers $a_1, a_2, \ldots, a_N$, an integer C (the capacity), and an integer B (the number of bins).
- BIN PACKING asks if these numbers can be partitioned into B subsets, each of which has total sum at most C.
- Think of packing bags at the check-out counter.

Theorem 43 BIN PACKING *is NP-complete.*

INTEGER PROGRAMMING

- INTEGER PROGRAMMING asks whether a system of linear inequalities with integer coefficients has an integer solution.
  - LINEAR PROGRAMMING asks whether a system of linear inequalities with integer coefficients has a *rational* solution.

INTEGER PROGRAMMING Is NP-Complete[a]

- SET COVERING can be expressed by the inequalities $Ax \geq \vec{1}$, $\sum_{i=1}^{n} x_i \leq B$, $0 \leq x_i \leq 1$, where
  - x_i is one if and only if S_i is in the cover.
  - A is the matrix whose columns are the bit vectors of the sets $S_1, S_2, \ldots$.
  - $\vec{1}$ is the vector of 1s.
- This shows INTEGER PROGRAMMING is NP-hard.
- Many NP-complete problems can be expressed as an INTEGER PROGRAMMING problem.

[a]Papadimitriou (1981).

Christos Papadimitriou

Easier or Harder?[a]

- Adding restrictions on the allowable *problem instances* will not make a problem harder.
  - We are now solving a subset of problem instances.
  - The INDEPENDENT SET proof (p. 304) and the KNAPSACK proof (p. 349).
  - SAT to 2SAT (easier by p. 287).
  - CIRCUIT VALUE to MONOTONE CIRCUIT VALUE (equally hard by p. 262).

[a] Thanks to a lively class discussion on October 29, 2003.

Easier or Harder? (concluded)

- Adding restrictions on the allowable *solutions* may make a problem easier, as hard, or harder.
- It is problem dependent.
  - MIN CUT to BISECTION WIDTH (harder by p. 330).
  - LINEAR PROGRAMMING to INTEGER PROGRAMMING (harder by p. 359).
  - SAT to NAESAT (equally hard by p. 298) and MAX CUT to MAX BISECTION (equally hard by p. 328).
  - 3-COLORING to 2-COLORING (easier by p. 336).

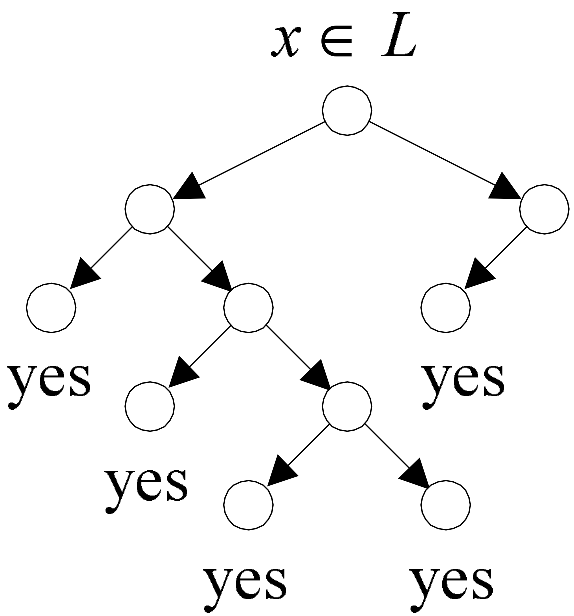
coNP and Function Problems

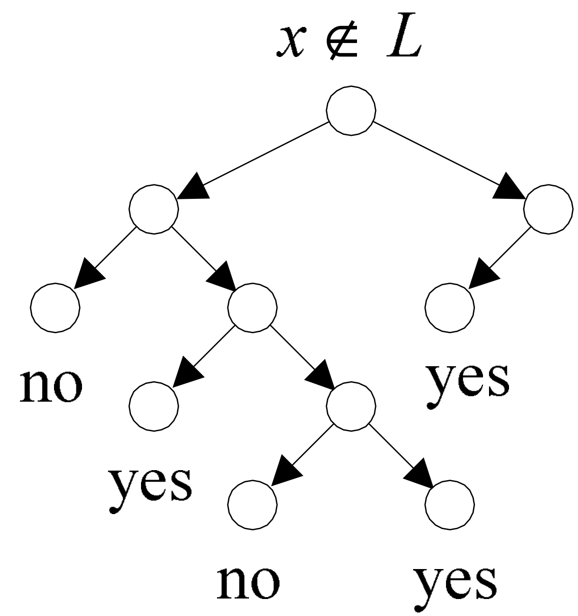
coNP

- By definition, coNP is the class of problems whose complement is in NP.
- NP is the class of problems that have succinct certificates (recall Proposition 31 on p. 271).
- coNP is therefore the class of problems that have succinct disqualifications:
  - A "no" instance of a problem in coNP possesses a short proof of its being a "no" instance.
  - Only "no" instances have such proofs.

coNP (continued)

- Suppose L is a coNP problem.
- There exists a polynomial-time nondeterministic algorithm M such that:
  - If $x \in L$, then $M(x) =$ "yes" for all computation paths.
  - If $x \notin L$, then $M(x) =$ "no" for some computation path.


$x \in L$
yes
yes
yes
yes
yes


$x \notin L$
no
yes
yes
no
yes

coNP (concluded)

- Clearly $\mathrm{P} \subseteq \mathrm{coNP}$.
- It is not known if

$$\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}.$$

  - Contrast this with

$$\mathrm{R} = \mathrm{RE} \cap \mathrm{coRE}$$

    (see Proposition 11 on p. 134).

Some coNP Problems

- VALIDITY $\in$ coNP.
  - If ϕ is not valid, it can be disqualified very succinctly: a truth assignment that does not satisfy it.
- SAT COMPLEMENT $\in$ coNP.
  - The disqualification is a truth assignment that satisfies it.
- HAMILTONIAN PATH COMPLEMENT $\in$ coNP.
  - The disqualification is a Hamiltonian path.
- OPTIMAL TSP (D) $\in$ coNP.[a]
  - The disqualification is a tour with a length $< B$.

[a] Asked by Mr. Che-Wei Chang (R95922093) on September 27, 2006.

An Alternative Characterization of coNP

Proposition 44 *Let $L \subseteq \Sigma^*$ be a language. Then $L \in coNP$ if and only if there is a polynomially decidable and polynomially balanced relation R such that*

$$L = \{x : \forall y \, (x, y) \in R\}.$$

(As on p. 270, we assume $|\, y \,| \leq |\, x \,|^k$ for some k.)

- $\bar{L} = \{x : (x, y) \in \neg R \text{ for some } y\}$.
- Because $\neg R$ remains polynomially balanced, $\bar{L} \in$ NP by Proposition 31 (p. 271).
- Hence $L \in$ coNP by definition.

coNP Completeness

Proposition 45 *L is NP-complete if and only if its complement $\bar{L} = \Sigma^* - L$ is coNP-complete.*

Proof ($\Rightarrow$; the $\Leftarrow$ part is symmetric)

- Let $\bar{L}'$ be any coNP language.
- Hence $L' \in$ NP.
- Let R be the reduction from L' to L.
- So $x \in L'$ if and only if $R(x) \in L$.
- So $x \in \bar{L}'$ if and only if $R(x) \in \bar{L}$.
- R is a reduction from $\bar{L}'$ to $\bar{L}$.

Some coNP-Complete Problems

- SAT COMPLEMENT is coNP-complete.
  - SAT COMPLEMENT is the complement of SAT.
- VALIDITY is coNP-complete.
  - ϕ is valid if and only if $\neg\phi$ is not satisfiable.
  - The reduction from SAT COMPLEMENT to VALIDITY is hence easy.
- HAMILTONIAN PATH COMPLEMENT is coNP-complete.

Possible Relations between P, NP, coNP

1. P = NP = coNP.
2. NP = coNP but P $\neq$ NP.
3. NP $\neq$ coNP and P $\neq$ NP.
   - This is current "consensus."