MAX BISECTION

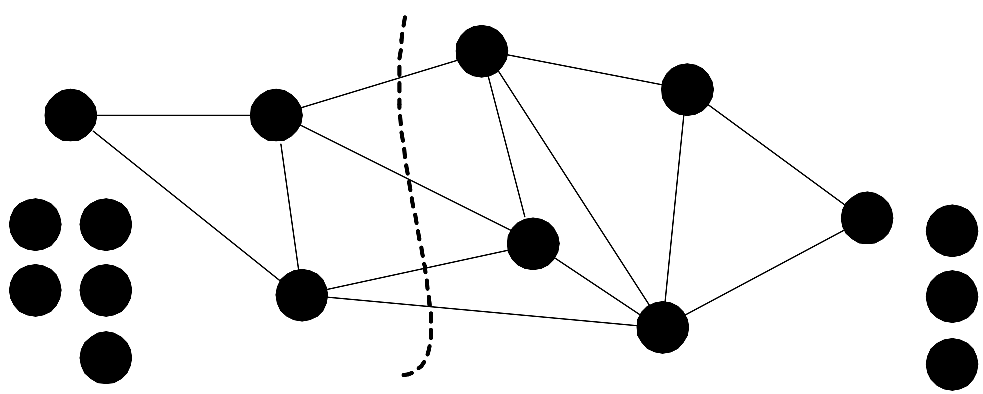
- MAX CUT becomes MAX BISECTION if we require that $|S| = |V - S|$.
- It has many applications, especially in VLSI layout.

MAX BISECTION Is NP-Complete

- We shall reduce the more general MAX CUT to MAX BISECTION.
- Add $|V| = n$ **isolated nodes** to G to yield G'.
- G' has $2n$ nodes.
- G''s goal K is identical to G's
  - As the new nodes have no edges, they contribute nothing to the cut.
- This completes the reduction.

The Proof (concluded)

- Every cut $(S, V - S)$ of $G = (V, E)$ can be made into a bisection by appropriately allocating the new nodes between S and $V - S$.
- Hence each cut of G can be made a cut of G' of the same size, and vice versa.



BISECTION WIDTH

- BISECTION WIDTH is like MAX BISECTION except that it asks if there is a bisection of size *at most* K (sort of MIN BISECTION).
- Unlike MIN CUT, BISECTION WIDTH is NP-complete.
- We reduce MAX BISECTION to BISECTION WIDTH.
- Given a graph $G = (V, E)$, where $|V|$ is even, we generate the complement of G.
- Given a goal of K, we generate a goal of $n^2 - K$.[a]

[a] $|V| = 2n$.

The Proof (concluded)

- To show the reduction works, simply notice the following easily verifiable claims.
  - A graph $G = (V, E)$, where $|V| = 2n$, has a bisection of size K if and only if the complement of G has a bisection of size $n^2 - K$.
  - So G has a bisection of size $\geq K$ if and only if its complement has a bisection of size $\leq n^2 - K$.

HAMILTONIAN PATH Is NP-Complete[a]

Theorem 42 *Given an* undirected *graph, the question whether it has a Hamiltonian path is NP-complete.*

---

[a] Karp (1972).

A Hamiltonian Path at IKEA, Covina, California?

TSP (D) Is NP-Complete

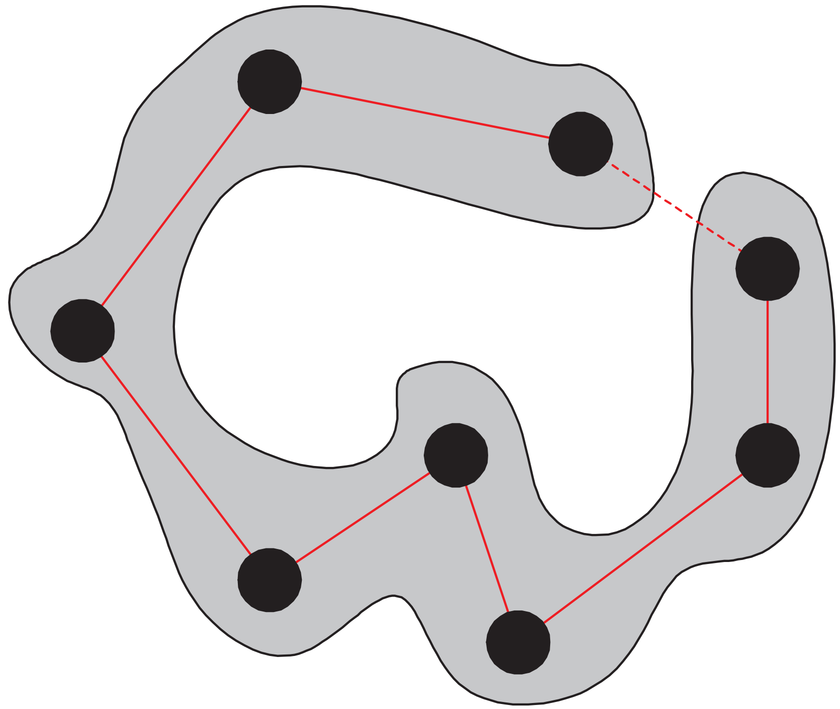
Corollary 43 TSP (D) *is NP-complete.*

- Consider a graph G with n nodes.
- Create a weighted complete graph G' with the same nodes as from G follows.
- Set $d_{ij} = 1$ on G' if $[\, i, j\,] \in G$ and $d_{ij} = 2$ on G' if $[\, i, j\,] \notin G$.
  - Note that G' is a complete graph.
- Set the budget $B = n + 1$.
- This completes the reduction.

TSP (D) Is NP-Complete (continued)

- Suppose G' has a tour of distance at most $n + 1$.[a]
- Then that tour on G' must contain at most one edge with weight 2.
- If a tour on G' contains 1 edge with weight 2, remove that edge to arrive at a Hamiltonian path for G.
- Suppose, on the other hand, a tour on G' contains no edge with weight 2.
- Then remove any edge to arrive at a Hamiltonian path for G.

[a] A tour is a cycle, not a path.



TSP (D) Is NP-Complete (concluded)

- On the other hand, suppose G has a Hamiltonian path.
- Then there is a tour on G' containing at most one edge with weight 2.
  - Start with a Hamiltonian path and then close the loop.
- 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 on G' if and only if G has a Hamiltonian path.

Random TSP

- Suppose each distance d_{ij} is picked uniformly and independently from the interval $[\,0, 1\,]$.
- It is known that the total distance of the shortest tour has a mean value of $\beta\sqrt{n}$ for some positive β.
- In fact, the total distance of the shortest tour can be away from the mean by more than t with probability at most $e^{-t^2/(4n)}$![a]

[a]Dubhashi and Panconesi (2012).

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?[a]
- 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?).

[a] k is *not* part of the input; k is part of the problem statement.

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 that can be colored with colors $\{0, 1, 2\}$ if and only if all the clauses can be NAE-satisfied.

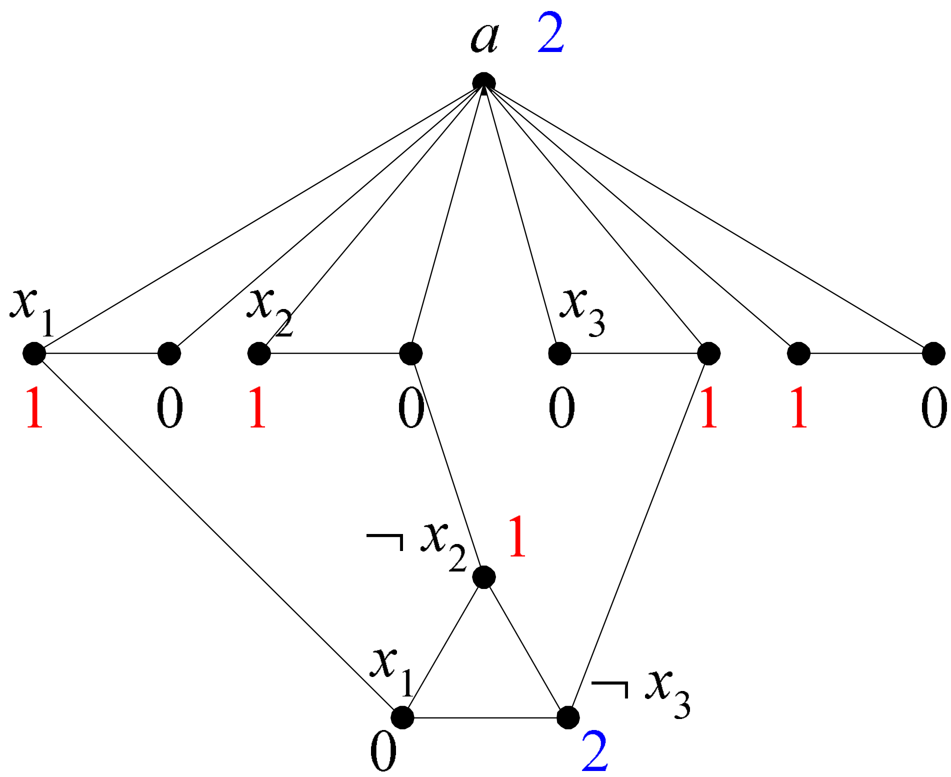
[a] Karp (1972).

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.[a]

[a] Alternative proof: There is an edge between $\neg c_{ij}$ and the node that represents the jth literal of C_i. Contributed by Mr. Ren-Shuo Liu (`D98922016`) on October 27, 2009.

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 are dealing with those triangles with the "a" node, not the clause triangles yet.
- 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 are dealing with those triangles with the "a" node, not the clause triangles.

The Proof (continued)

- 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 Proof (concluded)

- 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.

Algorithms for 3-COLORING and the Chromatic Number $\chi(G)$

- Assume G is 3-colorable.
- There is an algorithm to find a 3-coloring in time $O(3^{n/3}) = 1.4422^n$.[a]
- It has been improved to $O(1.3289^n)$.[b]

---

[a] Lawler (1976).

[b] Beigel and Eppstein (2000).

Algorithms for 3-COLORING and the Chromatic Number $\chi(G)$ (concluded)

- The **chromatic number** $\chi(G)$ is the smallest number of colors needed to color a graph G.
- There is an algorithm to find $\chi(G)$ in time $O((4/3)^{n/3}) = 2.4422^n$.[a]
- It can be improved to $O((4/3 + 3^{4/3}/4)^n) = O(2.4150^n)$[b] and $2^n n^{O(1)}$.[c]
- Computing $\chi(G)$ cannot be easier than 3-COLORING.[d]

---

[a] Lawler (1976).
[b] Eppstein (2003).
[c] Koivisto (2006).
[d] Contributed by Mr. Ching-Hua Yu (`D00921025`) on October 30, 2012.

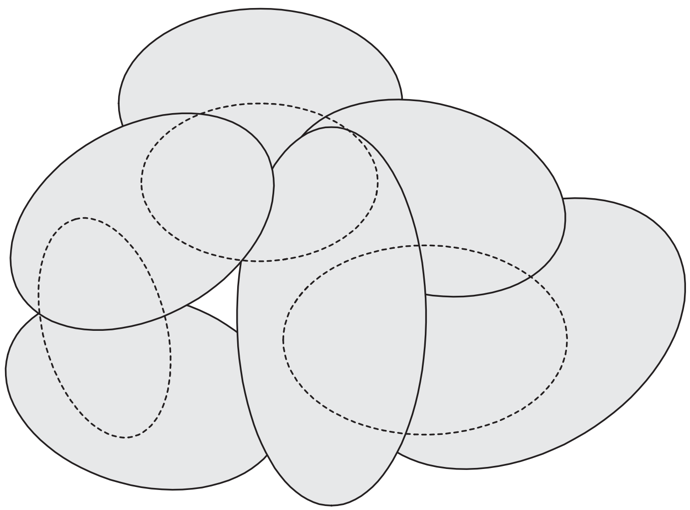
TRIPARTITE MATCHING

- 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.

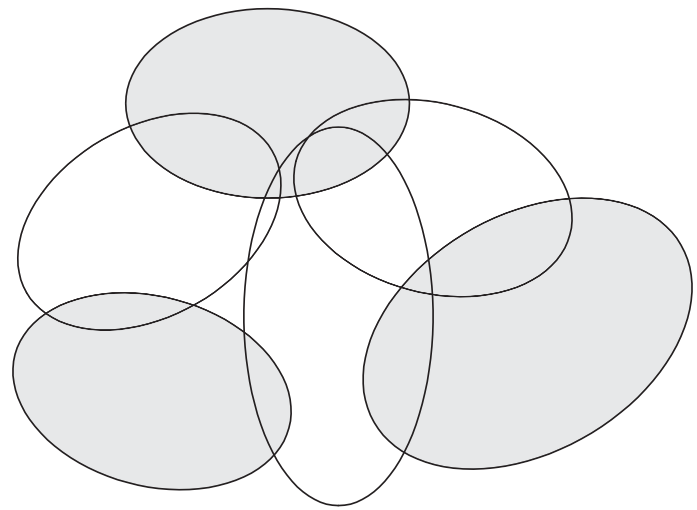
Theorem 44 (Karp (1972)) TRIPARTITE MATCHING *is NP-complete.*

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 (so have U as their union).



SET COVERING



SET PACKING

Related Problems (concluded)

Corollary 45 (Karp (1972)) SET COVERING, SET PACKING, *and* EXACT COVER BY 3-SETS *are all NP-complete.*

- SET COVERING can be used to prove that the influence maximization problem in social networks is NP-complete.[a]

[a] Kempe, Kleinberg, and Tardos (2003).

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[a]

- KNAPSACK $\in$ NP: Guess an S and check the constraints.
- We shall reduce EXACT COVER BY 3-SETS to KNAPSACK, in which $v_i = w_i$ for all i and $K = W$.
- The simplified KNAPSACK now asks if a subset of $v_1, v_2, \ldots, v_n$ adds up to exactly K.[b]
  - Picture yourself as a radio DJ.

---

[a] Karp (1972).

[b] This problem is called SUBSET SUM.

The Proof (continued)

- The primary differences between the two problems are:[a]
  - Sets vs. numbers.
  - Union vs. addition.
- 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.

[a] Thanks to a lively class discussion on November 16, 2010.

The Proof (continued)

- Think of a set as a bit vector in $\{0,1\}^{3m}$.
  - `110010000` means the set $\{1,2,5\}$.
  - `001100010` means the set $\{3,4,8\}$.
- Our goal is

$$\overbrace{\texttt{1}\,\texttt{1}\cdots\texttt{1}}^{3m}.$$

The Proof (continued)

- A bit vector can also be seen as a binary *number*.
- Set union resembles addition:

$$\begin{array}{rr} & \texttt{001100010} \\ + & \texttt{110010000} \\ \hline & \texttt{111110010} \end{array}$$

which denotes the set $\{1, 2, 3, 4, 5, 8\}$, as desired.

The Proof (continued)

- Trouble occurs when there is *carry*:

$$
\begin{array}{rr}
 & \texttt{010000000} \\
+ & \texttt{010000000} \\
\hline
 & \texttt{100000000}
\end{array}
$$

  which denotes the set $\{1\}$, not the desired $\{2\}$.

The Proof (continued)

- Or consider

$$
\begin{array}{rr}
 & \texttt{001100010} \\
+ & \texttt{001110000} \\
\hline
 & \texttt{011010010}
\end{array}
$$

which denotes the set $\{2, 3, 5, 8\}$, not the desired $\{3, 4, 5, 8\}$.[a]

[a]Corrected by Mr. Chihwei Lin (D97922003) on January 21, 2010.

The Proof (continued)

- Carry may also lead to a situation where we obtain our solution $\mathtt{1\,1} \cdots \mathtt{1}$ with more than m sets in F.
- For example,

$$
\begin{array}{rl}
 & \mathtt{000100010} \\
 & \mathtt{001110000} \\
 & \mathtt{101100000} \\
+ & \mathtt{000001101} \\
\hline
 & \mathtt{111111111}
\end{array}
$$

- But the correct answer, $\{1, 3, 4, 5, 6, 7, 8, 9\}$, is *not* an exact cover.

The Proof (continued)

- 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.[b]
- 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.

[b] You cannot map $\cup$ to $\vee$ because KNAPSACK requires $+$.

The Proof (continued)

- Set v_i to be the integer corresponding to the bit vector encoding S_i in base $n+1$:

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

- Now in base $n+1$, if there is a set S such that $\sum_{i \in S} v_i = \overbrace{1\,1 \cdots 1}^{3m}$, then every position must be contributed by exactly one v_i and $|S| = m$.

- Finally, set

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

The Proof (continued)

- For example, the case on p. 399 becomes

$$
\begin{array}{rr}
 & \texttt{000100010} \\
 & \texttt{001110000} \\
 & \texttt{101100000} \\
+ & \texttt{000001101} \\
\hline
 & \texttt{102311111}
\end{array}
$$

  in base 6.

- It does not meet the goal.

The Proof (continued)

- Suppose F admits an exact cover, say $\{S_1, S_2, \ldots, S_m\}$.
- Then picking $S = \{1, 2, \ldots, m\}$ clearly results in

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

  - It is important to note that the meaning of addition (+) is independent of the base.[a]
  - It is just regular addition.
  - But an S_i may give rise to different integer v_i's in Eq. (3) on p. 401 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

$$\sum_{i \in S} v_i = \overbrace{1\,1 \cdots 1}^{3m}$$

  in base $n+1$.

- The no-carry property implies that $|S| = m$ and

$$\{ S_i : i \in S \}$$

  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 (continued)

- Our reduction produces

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

An Example (concluded)

- Note $v_1 + v_3 + v_5 = K$ because

$$\begin{array}{rr} & 101100000 \\ & 010011000 \\ + & 000000111 \\ \hline & 111111111 \end{array}$$

- 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 46 BIN PACKING *is NP-complete.*

BIN PACKING (concluded)

- But suppose $a_1, a_2, \ldots, a_N$ are randomly distributed between 0 and 1.
- Let B be the smallest number of unit-capacity bins capable of holding them.
- Then B can differ from its average by more than t with probability at most $2e^{-2t^2/N}$.[a]

[a] Dubhashi and Panconesi (2012).

INTEGER PROGRAMMING

- INTEGER PROGRAMMING asks whether a system of linear inequalities with integer coefficients has an integer solution.
- In contrast, 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.
  - The operations in Ax are standard matrix operations.
- This shows INTEGER PROGRAMMING is NP-hard.
- Many NP-complete problems can be expressed as an INTEGER PROGRAMMING problem.

[a] Karp (1972).

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 or special cases.
  - The INDEPENDENT SET proof (p. 341) and the KNAPSACK proof (p. 393).
  - SAT to 2SAT (easier by p. 322).
  - CIRCUIT VALUE to MONOTONE CIRCUIT VALUE (equally hard by p. 294).

---

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

Easier or Harder? (concluded)

- Adding restrictions on the allowable *solutions* (the solution space) may make a problem harder, equally hard, or easier.
- It is problem dependent.
  - MIN CUT to BISECTION WIDTH (harder by p. 368).
  - LINEAR PROGRAMMING to INTEGER PROGRAMMING (harder by p. 410).
  - SAT to NAESAT (equally hard by p. 335) and MAX CUT to MAX BISECTION (equally hard by p. 366).
  - 3-COLORING to 2-COLORING (easier by p. 377).

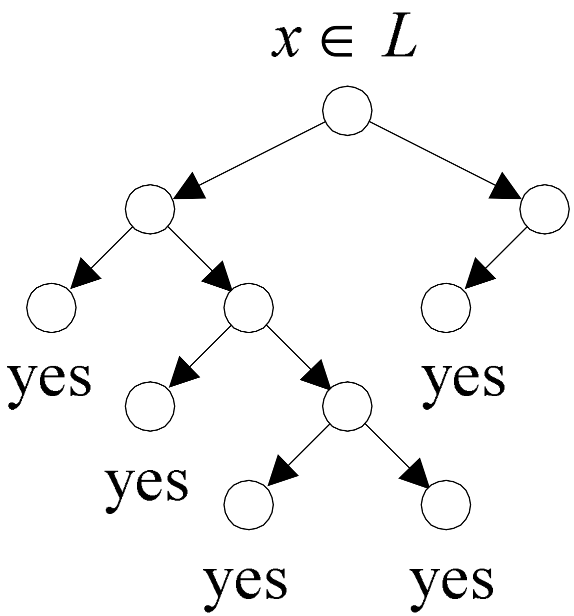
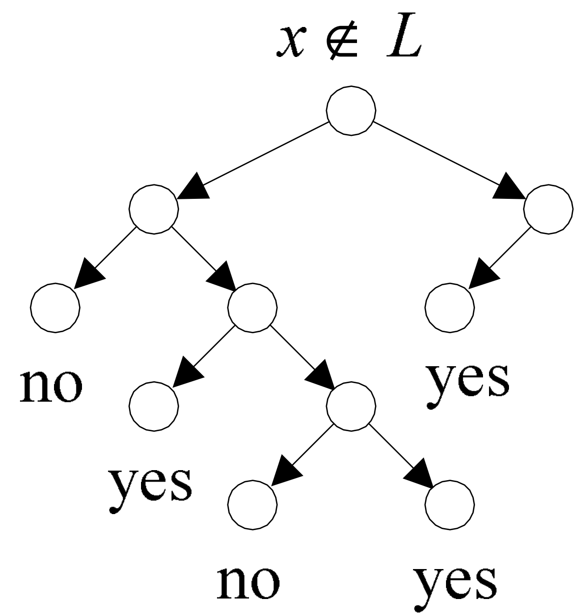
coNP and Function Problems

coNP

- NP is the class of problems that have succinct certificates (recall Proposition 35 on p. 306).
- By definition, coNP is the class of problems whose complement is in NP.
- 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.
- Note that if we swap "yes" and "no" of M, the new algorithm M' decides $\bar{L} \in$ NP in the classic sense (p. 94).


$x \in L$
yes
yes
yes
yes
yes

$x \notin L$
no
yes
no
yes
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. 153).

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.
  - SAT COMPLEMENT is the complement of SAT.
  - The disqualification is a truth assignment that satisfies it.
- HAMILTONIAN PATH COMPLEMENT $\in$ coNP.
  - The disqualification is a Hamiltonian path.

Some coNP Problems (concluded)

- OPTIMAL TSP (D) $\in$ coNP.
  - OPTIMAL TSP (D) asks if the optimal tour has a total distance of B, where B is an input.[a]
  - The disqualification is a tour with a length $< B$.

---

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

A Nondeterministic Algorithm for SAT COMPLEMENT

ϕ is a boolean formula with n variables.

for $i = 1, 2, \ldots, n$ **do**
    Guess $x_i \in \{0, 1\}$; {Nondeterministic choice.}
end for
{Verification:}
if $\phi(x_1, x_2, \ldots, x_n) = 1$ **then**
    "no";
else
    "yes";
end if

Analysis

- The algorithm decides language $\{\phi : \phi \text{ is unsatisfiable}\}$.
  - The computation tree is a complete binary tree of depth n.
  - Every computation path corresponds to a particular truth assignment out of 2^n.
  - ϕ is unsatisfiable iff every truth assignment falsifies ϕ.
  - But every truth assignment falsifies ϕ iff every computation path results in "yes."

An Alternative Characterization of coNP

Proposition 47 *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. 305, we assume $|\,y\,| \leq |\,x\,|^k$ for some k.)

- $\bar{L} = \{x : \exists y\, (x, y) \in \neg R\}$.
- Because $\neg R$ remains polynomially balanced, $\bar{L} \in$ NP by Proposition 35 (p. 306).
- Hence $L \in$ coNP by definition.

coNP-Completeness

Proposition 48 *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 $R(x) \in L$.
- Equivalently, $x \notin L'$ if and only if $R(x) \notin L$ (the law of transposition).

coNP Completeness (concluded)

- So $x \in \bar{L}'$ if and only if $R(x) \in \bar{L}$.
- R is a reduction from $\bar{L}'$ to $\bar{L}$.
- This shows $\bar{L}$ is coNP-hard.
- But $\bar{L} \in$ coNP.
- This shows $\bar{L}$ is coNP-complete.

Some coNP-Complete Problems

- SAT COMPLEMENT is coNP-complete.
- 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. $\text{P} = \text{NP} = \text{coNP}$.
2. $\text{NP} = \text{coNP}$ but $\text{P} \neq \text{NP}$.
3. $\text{NP} \neq \text{coNP}$ and $\text{P} \neq \text{NP}$.
   - This is the current "consensus."[a]

[a] Carl Gauss (1777–1855), "I could easily lay down a multitude of such propositions, which one could neither prove nor dispose of."