MAX BISECTION

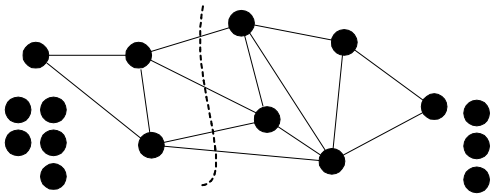
- MAX CUT becomes MAX BISECTION if we require that $|S| = |V - S|$.
- It has many applications, especially in VLSI layout.

MAX BISECTION Is NP-Complete

- We shall reduce the more general MAX CUT to MAX BISECTION.
- Add $|V|$ **isolated nodes** to G to yield G'.
- G' has $2 \times |V|$ nodes.
- As the new nodes have no edges, moving them around contributes nothing to the cut.

The Proof (concluded)

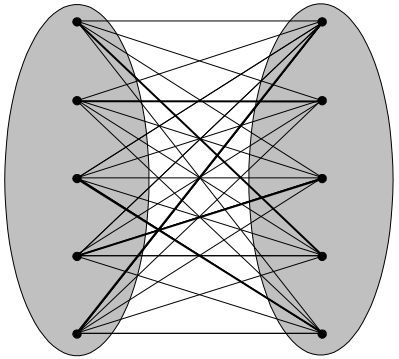
- Every cut $(S, V - S)$ of $G = (V, E)$ can be made into a bisection by appropriately allocating the new nodes between S and $V - S$.
- Hence each cut of G can be made a cut of G' of the same size, and vice versa.



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 remains NP-complete.
  - 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 $\leq n^2 - K$.

Illustration



HAMILTONIAN PATH Is NP-Complete[a]

Theorem 38 *Given an* undirected *graph, the question whether it has a Hamiltonian path is NP-complete.*

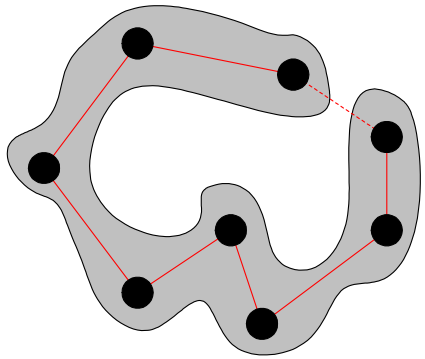
[a]Karp (1972).

TSP (D) Is NP-Complete

Corollary 39 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 $B = n + 1$.
- If G has no Hamiltonian paths, then every tour on the new graph must contain at least two edges with weight 2.
- The total cost is then at least $(n - 2) + 2 \cdot 2 = n + 2$.
- There is a tour of length B or less if and only if G has a Hamiltonian path.

Hamiltonian Path and TSP Tour



Graph Coloring

- k-COLORING asks if the nodes of a graph can 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?).

3-COLORING Is NP-Complete[a]

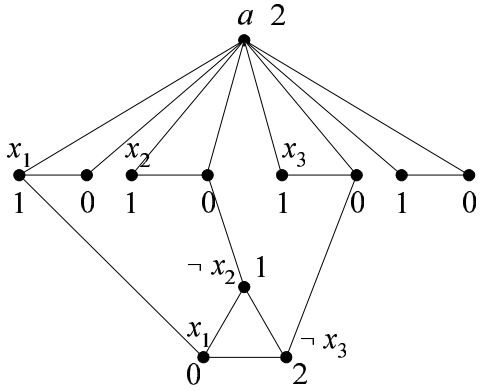
- We will reduce NAESAT to 3-COLORING.
- We are given a set of clauses $C_1, C_2, \ldots, C_m$ each with 3 literals.
- The boolean variables are $x_1, x_2, \ldots, x_n$.
- We shall construct a graph G such that it can be colored with colors $\{0, 1, 2\}$ if and only if all the clauses can be NAE-satisfied.

[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.

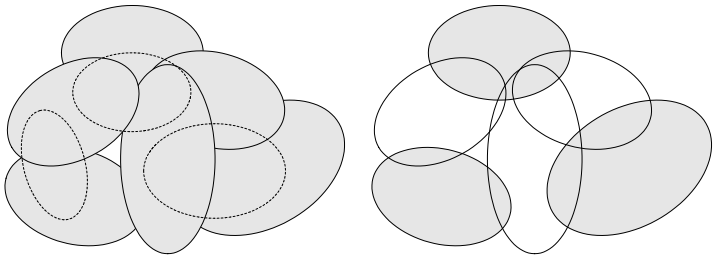
TRIPARTITE MATCHING

- 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 40 (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 41 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 $\{w_1, w_2, \ldots, w_n\}$ 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.
  - `001100010` + `110010000` = `111110010`, which denotes the set $\{1, 2, 3, 4, 5, 8\}$, as desired.
- Trouble occurs when there is *carry*.
  - `001100010` + `001110000` = `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 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 $\sum_{v_i \in S} v_i = \overbrace{11\cdots1}^{3m}$, then every bit position must be contributed by exactly one v_i and $|S| = m$.
- Finally, set

$$K = \sum_{j=0}^{3m-1} (n+1)^j = \overbrace{11\cdots1}^{3m} \quad \text{(base } n+1\text{)}.$$

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

$$v_1 + v_2 + \cdots + v_m = \overbrace{11\cdots1}^{3m}.$$

  - It is important to note that the meaning of addition $(+)$ is independent of the base.[a]
  - It is just regular addition.

[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 $\sum_{v_i \in S} v_i = \overbrace{11\cdots1}^{3m}$ in base $n + 1$.
- The no-carry property implies that $|S| = m$ and $\{S_i : v_i \in S\}$ is an exact cover.

BIN PACKINGS

- 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 42 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).

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. 278). and the KNAPSACK proof (p. 319).
  - SAT to 2SAT (easier by p. 265).
  - CIRCUIT VALUE to MONOTONE CIRCUIT VALUE (equally hard by p. 244).

[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. 301).
  - LINEAR PROGRAMMING to INTEGER PROGRAMMING (harder by p. 327).
  - SAT to NAESAT (equally hard by p. 273) and MAX CUT to MAX BISECTION (equally hard by p. 299).
  - 3-COLORING to 2-COLORING (easier by p. 306).