TRIPARTITE MATCHING

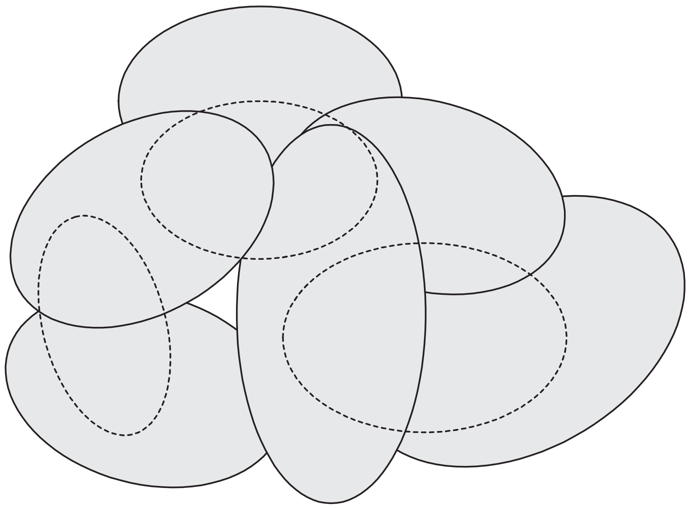
- We are given three sets B, G, and H, each containing n elements.
- Let $T \subseteq B \times G \times H$ be a ternary relation.
- TRIPARTITE MATCHING asks if there is a set of n triples in T, none of which has a component in common.
  - Each element in B is matched to a different element in G and different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

Theorem 43 (Karp, 1972) TRIPARTITE MATCHING *is NP-complete.*

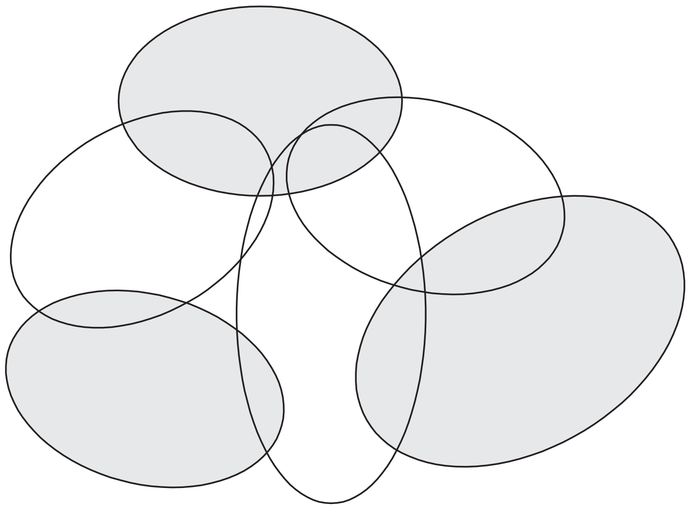
Related Problems

- 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.

Corollary 44 SET COVERING, SET PACKING, *and* EXACT COVER BY 3-SETS *are all NP-complete.*



SET COVERING



SET PACKING

INTEGER PROGRAMMING Is NP-Complete[a]

- INTEGER PROGRAMMING asks whether a system of linear inequalities with integer coefficients has an integer solution.
- Many NP-complete problems can be expressed as an INTEGER PROGRAMMING problem.
  - 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.

[a] Papadimitriou, 1981.

The KNAPSACK Problem

- There is a set of n items.
- Item i has value $v_i \in \mathbb{Z}^+$ and weight $w_i \in \mathbb{Z}^+$.
- 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.
  - Think of yourself as a radio DJ.
- We shall reduce EXACT COVER BY 3-SETS to it.

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 $\texttt{11} \cdots \texttt{1}$.

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 is there is *carry*.
  - $\texttt{001100010} + \texttt{001110000} = \texttt{010010010}$, which denotes the set $\{2, 5, 8\}$, not the desired $\{3, 4, 5, 8\}$.
- Carry also leads to a situation where we obtain our solution $\texttt{11} \cdots \texttt{1}$ with more than m sets in F.

The Proof (continued)

- – $\texttt{001100010} + \texttt{001110000} + \texttt{101100000} + \texttt{000001101} = \texttt{111111111}$.
  - But this "solution" $\{1, 3, 4, 5, 6, 7, 8, 9\}$ does not correspond to an exact cover.
  - Furthermore, it uses 4 sets instead of the required 3.
- 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 $n + 1$.

The Proof (continued)

- 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 $|S| = m$.
- Set v_i to be the $(n+1)$-ary number corresponding to the bit vector encoding S_i.
- Finally, set

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

The Proof (concluded)

- 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

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

- 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.

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.

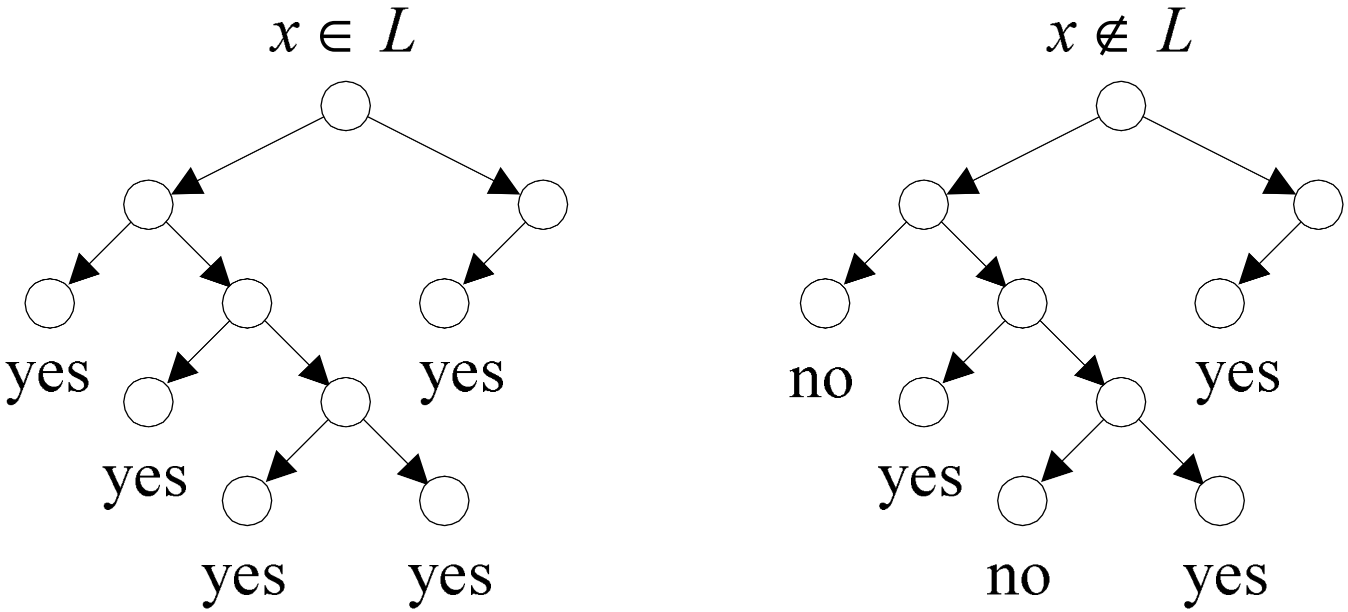
Theorem 45 BIN PACKING *is NP-complete.*

coNP

- NP is the class of problems that have succinct certificates (recall Proposition 35 on p. 240).
- coNP is 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.
- Clearly $\mathrm{P} \subseteq \mathrm{coNP}$.
- It is not known if $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$.
  - Contrast this with $\mathrm{R} = \mathrm{RE} \cap \mathrm{coRE}$.

coNP as Decision Problems

- 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.
- We can swap "yes" and "no" in the above definition without materially changing the coNP class (why?).


$x \in L$
yes
yes
yes
yes
yes
$x \notin L$
no
yes
yes
no
yes

An Alternative Characterization of coNP

Proposition 46 *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*

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

- $\bar{L} = \{x : (x, y) \in \neg R \text{ for some } y\}$.
- Because $\neg R$ remains polynomially balanced, $\bar{L} \in \mathrm{NP}$ by Proposition 35 (p. 240).
- Hence $L \in \mathrm{coNP}$ by definition.

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.

coNP Completeness

Proposition 47 *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

- $\mathrm{P} = \mathrm{NP} = \mathrm{coNP}$.
- $\mathrm{NP} = \mathrm{coNP}$ but $\mathrm{P} \neq \mathrm{NP}$.
- $\mathrm{NP} \neq \mathrm{coNP}$ and $\mathrm{P} \neq \mathrm{NP}$ (current "consensus").

coNP Completeness and NP Completeness

Proposition 48 *If a coNP-complete problem is in NP, then NP $=$ coNP.*

- Let $L \in$ NP be coNP-complete.
- Let NTM M decide L.
- For any $L' \in$ coNP, there is a reduction R from L' to L.
- $L' \in$ NP as it is decided by NTM $M(R(x))$.
  - Alternatively, NP is closed under complement.
- The other direction NP $\subseteq$ coNP is symmetric.

coNP Completeness and NP Completeness (concluded)

Similarly,

Proposition 49 *If a NP-complete problem is in coNP, then* $NP = coNP$.

Hence NP-complete problems are unlikely to be in coNP and coNP-complete problems are unlikely to be in NP.

The Primality Problem

- An integer p is **prime** if $p > 1$ and all positive numbers other than 1 and p itself cannot divide it.
- PRIMES asks if an integer N is a prime number.
- Dividing N by $2, 3, \ldots, \sqrt{N}$ is *not* efficient.
  - The length of N is only $\log N$, but $\sqrt{N} = 2^{0.5 \log N}$.
- A polynomial-time algorithm for PRIMES was not found until 2002 by Agrawal, Kayal, and Saxena!
- We will focus on efficient "probabilistic" algorithms for PRIMES (used in *Mathematica*, e.g.).

if $n = a^b$ for some $a, b > 1$ **then**
    return "composite";
end if
for $r = 2, 3, \ldots, n-1$ **do**
    if $\gcd(n, r) > 1$ **then**
        return "composite";
    end if
    if r is a prime **then**
        Let q be the largest prime factor of $r-1$;
        if $q \geq 4\sqrt{r} \log n$ and $n^{(r-1)/q} \neq 1 \bmod r$ **then**
            break; {Exit the for-loop.}
        end if
    end if
end for{$r-1$ has a prime factor $q \geq 4\sqrt{r}\log n$.}
for $a = 1, 2, \ldots, 2\sqrt{r}\log n$ **do**
    if $(x-a)^n \neq (x^n - a) \bmod (x^r - 1)$ in $Z_n[x]$ **then**
        return "composite";
    end if
end for
return "prime"; {The only place with "prime" output.}

DP

- DP $\equiv$ NP $\cap$ coNP is the class of problems that have succinct certificates and succinct disqualifications.
  - Each "yes" instance has a succinct certificate.
  - Each "no" instance has a succinct disqualification.
  - No instances have both.
- P $\subseteq$ DP.
- We will see that PRIMES $\in$ DP.
  - In fact, PRIMES $\in$ P as mentioned earlier.

Primitive Roots in Finite Fields

Theorem 50 (Lucas and Lehmer, 1927) *A number $p > 1$ is prime if and only if there is a number $1 < r < p$ (called the* **primitive root** *or* **generator***) such that*

1. $r^{p-1} = 1 \bmod p$, *and*
2. $r^{(p-1)/q} \neq 1 \bmod p$ *for all prime divisors q of $p - 1$.*

- The above theorem can be used to test efficiently primes of the form $2^m + 1$.
- We will prove the theorem later.

Pratt's Theorem

Theorem 51 (Pratt, 1975) PRIMES $\in$ *NP* $\cap$ *coNP.*

- PRIMES is in coNP because a succinct disqualification is a divisor.
- Suppose p is a prime.
- p's certificate includes the r in Theorem 50 (p. 324).
- Use recursive doubling to check if $r^{p-1} = 1 \bmod p$ in time polynomial in the length of the input, $\log_2 p$.
- We also need all *prime* divisors of $p - 1$: $q_1, q_2, \ldots, q_k$.
- Checking $r^{(p-1)/q_i} \neq 1 \bmod p$ is also easy.

The Proof (concluded)

- Checking $q_1, q_2, \ldots, q_k$ are all the divisors of $p-1$ is easy.
- We still need certificates for the primality of the q_i's.
- The complete certificate is recursive and tree-like:

$$C(p) = (r; q_1, C(q_1), q_2, C(q_2), \ldots, q_k, C(q_k)).$$

- $C(p)$ can also be checked in polynomial time.
- We next prove that $C(p)$ is succinct.

The Succinctness of the Certificate

Lemma 52 *The length of $C(p)$ is at most quadratic at $5\log_2^2 p$.*

- This claim holds when $p = 2$ or $p = 3$.
- In general, $p - 1$ has $k < \log_2 p$ prime divisors $q_1 = 2, q_2, \ldots, q_k$.
- $C(p)$ requires: 2 parentheses and $2k < 2\log_2 p$ separators (length at most $2\log_2 p$ long), r (length at most $\log_2 p$), $q_1 = 2$ and its certificate 1 (length at most 5 bits), the q_i's (length at most $2\log_2 p$), and the $C(q_i)$s.

The Proof (concluded)

- $C(p)$ is succinct because

$$
\begin{aligned}
|C(p)| &\leq 5\log_2 p + 5 + 5\sum_{i=2}^{k} \log_2^2 q_i \\
&\leq 5\log_2 p + 5 + 5\left(\sum_{i=2}^{k} \log_2 q_i\right)^2 \\
&\leq 5\log_2 p + 5 + 5\log_2^2 \frac{p-1}{2} \\
&< 5\log_2 p + 5 + 5(\log_2 p - 1)^2 \\
&= 5\log_2^2 p + 10 - 5\log_2 p \leq 5\log^2 p
\end{aligned}
$$

for $p \geq 4$.

Basic Modular Arithmetics[a]

- Let $m, n \in \mathbb{Z}^+$.
- $m|n$ means m divides n and m is n's **divisor**.
- We call the numbers $0, 1, \ldots, n-1$ the **residue** modulo n.
- The **greatest common divisor** of m and n is denoted $\gcd(m, n)$.
- The r in Theorem 50 (p. 324) is a primitive root of p.
- We now prove the existence of primitive roots and then Theorem 50.

[a]Carl Friedrich Gauss (1777–1855).

Euler's[a] Totient or Phi Function

- Let
$$\Phi(n) = \{m : 1 \leq m < n, \gcd(m, n) = 1\}$$
be the set of all positive integers less than n that are prime to n (Z_n^* is a more popular notation).
  - $\Phi(12) = \{1, 5, 7, 11\}$.
- Define **Euler's function** of n to be $\phi(n) = |\Phi(n)|$.
- $\phi(p) = p - 1$ for prime p, and $\phi(1) = 1$ by convention.
- Euler's function is not expected to be easy to compute without knowing n's factorization.

[a]Leonhard Euler (1707–1783).

Two Properties of Euler's Function

The inclusion-exclusion principle[a] can be used to prove the following.

Lemma 53 $\phi(n) = n \prod_{p \mid n} (1 - \frac{1}{p})$.

- If $n = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$ is the prime factorization of n, then

$$\phi(n) = n \prod_{i=1}^{t} \left(1 - \frac{1}{p_i} \right).$$

Corollary 54 $\phi(mn) = \phi(m)\phi(n)$ *if* $\gcd(m, n) = 1$.

[a] See my *Discrete Mathematics* lecture notes.

A Key Lemma

Lemma 55 $\sum_{m|n} \phi(m) = n$.

- Let $\prod_{i=1}^{\ell} p_i^{k_i}$ be the prime factorization of n and consider

$$\prod_{i=1}^{\ell} [\, \phi(1) + \phi(p_i) + \cdots + \phi(p_i^{k_i}) \,]. \tag{4}$$

- Equation (4) equals n because $\phi(p_i^k) = p_i^k - p_i^{k-1}$ by Lemma 53.
- Expand Eq. (4) to yield $\sum_{k'_1 \leq k_1, \ldots, k'_\ell \leq k_\ell} \prod_{i=1}^{\ell} \phi(p_i^{k'_i})$.

The Proof (concluded)

- By Corollary 54 (p. 331),

$$\prod_{i=1}^{\ell} \phi(p_i^{k_i'}) = \phi\left(\prod_{i=1}^{\ell} p_i^{k_i'}\right).$$

- Each $\prod_{i=1}^{\ell} p_i^{k_i'}$ is a unique divisor of $n = \prod_{i=1}^{\ell} p_i^{k_i}$.
- Equation (4) becomes

$$\sum_{m \mid n} \phi(m).$$