Partitioning and Numerical Problems

1. The 3-Dimensional Matching Problem
   - a general version of bipartite matching
   - NP-completeness
   - reducing 3-SAT to 3-dimensional matching
2. Numerical Problems
   - the subset sum problem
   - reducing 3-dimensional matching to subset sum
3. Co-NP and the Asymmetry of NP
   - complementary problems
   - a partial taxonomy of hard problems

CS 401/MCS 401 Lecture 21
Computer Algorithms I
Jan Verschelde, 4 August 2025

Partitioning and Numerical Problems

1. The 3-Dimensional Matching Problem
   - a general version of bipartite matching
   - NP-completeness
   - reducing 3-SAT to 3-dimensional matching

2. Numerical Problems
   - the subset sum problem
   - reducing 3-dimensional matching to subset sum

3. Co-NP and the Asymmetry of NP
   - complementary problems
   - a partial taxonomy of hard problems

bipartite matching

We studied the Bipartite Matching Problem.

Given is a bipartite graph $G = (V, E)$, $V = X \cup Y$, $X \cap Y = \emptyset$.

A matching is a subset M of E such that every vertex occurs at most once in M.

A matching M is perfect if all vertices occur in M.

The bipartite matching problem asks

Does a given bipartite graph have a perfect matching?

The Ford-Fulkerson algorithm solves this problem efficiently.

the 3-dimensional matching problem

Given are the following:

- Three sets X, Y, and Z:
  1. each set has the same size $n = \#X = \#Y = \#Z$, and
  2. the sets are disjoint: $X \cap Y = \emptyset$, $X \cap Z = \emptyset$, $Y \cap Z = \emptyset$.
- A set T of ordered triples $T \subseteq X \times Y \times Z$.

The 3-dimensional matching problem asks

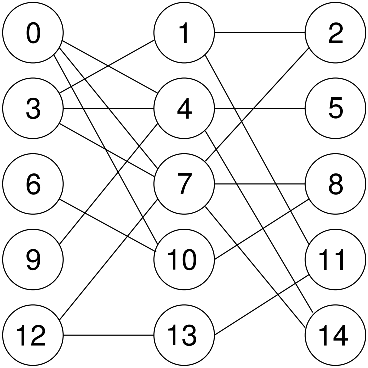
> Is there a set of n triples in T so each element of $X \cup Y \cup Z$ is contained *exactly once* in one of these triples?

Such a set of triples is *a perfect 3-dimensional matching*.

Example: $X = \{\text{instructors}\}$, $Y = \{\text{courses}\}$, $Z = \{\text{times}\}$.

an example of a 3-dimensional matching problem

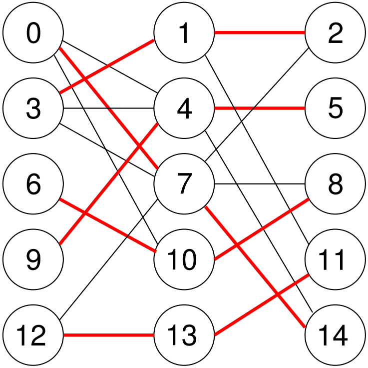
$X = \{ 0, 3, 6, 9, 12 \}$, $Y = \{ 1, 4, 7, 10, 13 \}$, $Z = \{ 2, 5, 8, 11, 14 \}$.



T is defined by the edges of the graph.

a solution to the example

$X = \{ 0, 3, 6, 9, 12 \}$, $Y = \{ 1, 4, 7, 10, 13 \}$, $Z = \{ 2, 5, 8, 11, 14 \}$.



A solution: $\{(0, 7, 14), (3, 1, 2), (6, 10, 8), (9, 4, 5), (12, 13, 11)\}$.

solve the 3-dimensional matching problem

The 3-dimensional matching problem is important.

Exercise 1: Solve the 3-dimensional matching problem by giving an algorithm to find a perfect matching in a tripartite graph.

What is the worst case running time of your algorithm?

the set cover problem

The set cover problem is a decision problem:

Given are $U = \{ 1, 2, \ldots, n \}$,
a collection $S_i \subseteq U$, $i = 1, 2, \ldots, m$,
and some number k.
Is there an index set $I = \{ i_1, i_2, \ldots, i_k \}$: $\displaystyle \bigcup_{j \in I} S_j = U$?

We proved that 3-SAT $\leq_P$ Set Cover. Set Cover is NP-Complete.

The 3-dimensional matching problem is a special case of set cover:

- $U = X \cup Y \cup Z$.
- We cover at most n sets from the given collection of triples.

Partitioning and Numerical Problems

1. The 3-Dimensional Matching Problem
   - a general version of bipartite matching
   - NP-completeness
   - reducing 3-SAT to 3-dimensional matching

2. Numerical Problems
   - the subset sum problem
   - reducing 3-dimensional matching to subset sum

3. Co-NP and the Asymmetry of NP
   - complementary problems
   - a partial taxonomy of hard problems

strategy to show a new problem is NP-complete

Given a new problem X, to show X is NP-complete:

1. Prove that $X \in \mathcal{NP}$.
   This consists in exhibiting an efficient certifier algorithm.
2. Choose a problem Y that is known to be NP-complete.
3. Prove that $Y \leq_P X$.
   The reformulation $Y \leq_P X$ implies that
   - we can solve Y by calling the black box solver for X, and
   - a polynomial number of elementary computational steps.

As the 3-dimensional matching problem is a special case of Set Cover, proving 3-Dimensional Matching $\leq_P$ Set Cover is not hard.

However, to show 3-Dimensional Matching is NP-complete,
we must show the other direction, that a known NP-complete problem can be reduced to the 3-Dimensional Matching problem.

3-dimensional matching is in $\mathcal{NP}$

Lemma (1)

The 3-Dimensional Matching Problem is in $\mathcal{NP}$.

Proof. A certificate for the 3-dimensional matching problem is a sequence of triples. The certifier algorithm takes on input this certificate and the input to the problem: the sets X, Y, Z, and the triples T. Let $n = \#X = \#Y = \#Z$ and $U = X \cup Y \cup Z$.
To verify the result:

1. We check that every point in U occurs in the certicate.
   This can be done with one scan through U and $3n$ steps, using a Boolean array of size $3n$.
2. We check that every triple in the certicate belongs to T.
   If $m = \#T$, then the check for one triple is $O(m)$.
   For all n triples, the check runs in $O(nm)$ time.

By the polynomial-time running time, the problem is in $\mathcal{NP}$. Q.E.D.

the 3-satisfiability problem is NP-complete

The 3-satisfiability problem, or 3-SAT for short is

> Given $X = \{ x_1, x_2, \ldots, x_n \}$ a set of Boolean variables, and
> C_1, C_2, ..., C_k, a set of clauses over X,
> where each clause C_i has exactly three terms,
> does there exist a satistying truth assignment?

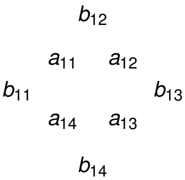
The example

$$(x \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (x \vee y \vee \overline{z})$$

is satisfied by $x = 1$, $\overline{y} = 1$, and $\overline{z} = 1$.

triples and gadgets

$T = \{(a_{11}, a_{12}, b_{12}), (a_{12}, a_{13}, b_{13}), (a_{13}, a_{14}, b_{14}), (a_{14}, a_{11}, b_{11})\}.$



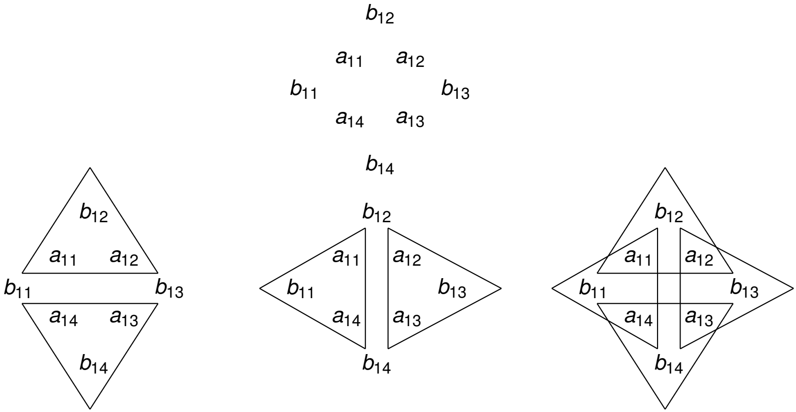
1. a_{11}, a_{12}, a_{13}, a_{14} are the *core* of the gadget;
2. b_{11}, b_{12}, b_{13}, b_{14} are the *tips* of the gadget.

A triple is even or odd if j of b_{ij} is even or odd.

To cover the core, use either all even or all odd triples.

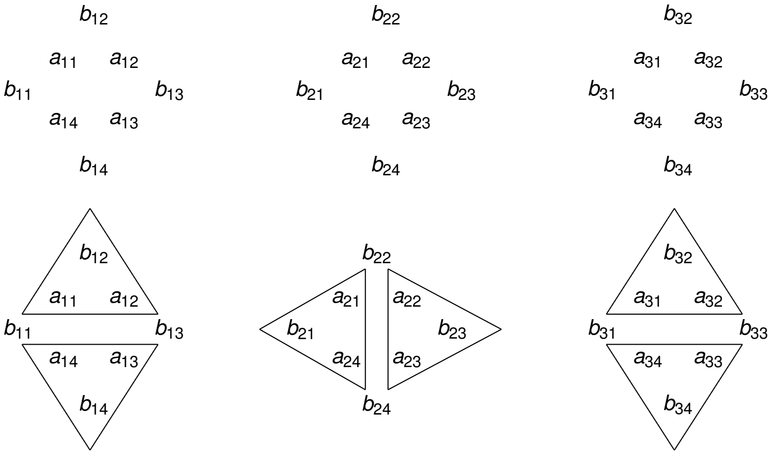
even/odd triples and 0/1 assignments

$$T = \{(a_{11}, a_{12}, b_{12}), (a_{12}, a_{13}, b_{13}), (a_{13}, a_{14}, b_{14}), (a_{14}, a_{11}, b_{11})\}.$$



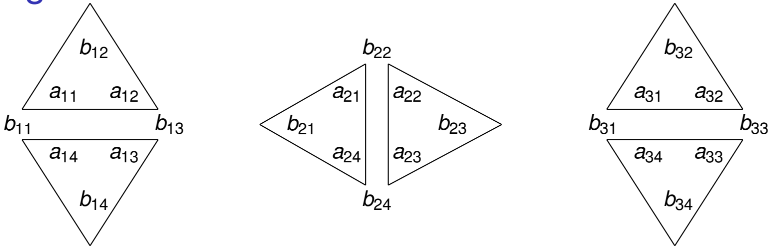
$x_1 = 0$ $x_1 = 1$

assigning to three variables in a clause



$x_1 = 0$ $x_2 = 1$ $x_3 = 0$

making a truth table



$x_1 = 0$ $\qquad$ $x_2 = 1$ $\qquad$ $x_3 = 0$

111: $(a_{11}, a_{14}, b_{11}), (a_{12}, a_{13}, b_{13}), (a_{21}, a_{24}, b_{21}), (a_{22}, a_{23}, b_{23}), (a_{31}, a_{34}, b_{31}), (a_{32}, a_{33}, b_{33})$

110: $(a_{11}, a_{14}, b_{11}), (a_{12}, a_{13}, b_{13}), (a_{21}, a_{24}, b_{21}), (a_{22}, a_{23}, b_{23}), (a_{31}, a_{32}, b_{32}), (a_{33}, a_{34}, b_{34})$

101: $(a_{11}, a_{14}, b_{11}), (a_{12}, a_{13}, b_{13}), (a_{21}, a_{22}, b_{22}), (a_{23}, a_{24}, b_{24}), (a_{31}, a_{34}, b_{31}), (a_{32}, a_{33}, b_{33})$

100: $(a_{11}, a_{14}, b_{11}), (a_{12}, a_{13}, b_{13}), (a_{21}, a_{22}, b_{22}), (a_{23}, a_{24}, b_{24}), (a_{31}, a_{32}, b_{32}), (a_{33}, a_{34}, b_{34})$

011: $(a_{11}, a_{12}, b_{12}), (a_{13}, a_{14}, b_{14}), (a_{21}, a_{24}, b_{21}), (a_{22}, a_{23}, b_{23}), (a_{31}, a_{34}, b_{31}), (a_{32}, a_{33}, b_{33})$

010: $(a_{11}, a_{12}, b_{12}), (a_{13}, a_{14}, b_{14}), (a_{21}, a_{24}, b_{21}), (a_{22}, a_{23}, b_{23}), (a_{31}, a_{32}, b_{32}), (a_{33}, a_{34}, b_{34})$

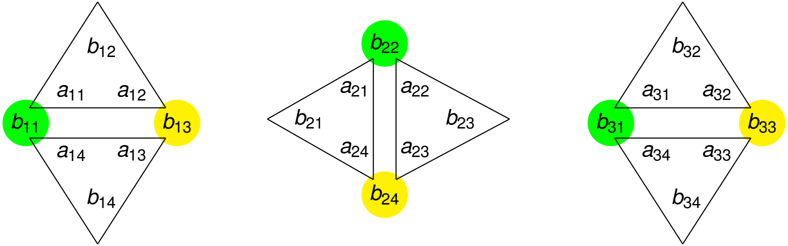
001: $(a_{11}, a_{12}, b_{12}), (a_{13}, a_{14}, b_{14}), (a_{21}, a_{22}, b_{22}), (a_{23}, a_{24}, b_{24}), (a_{31}, a_{34}, b_{31}), (a_{32}, a_{33}, b_{33})$

000: $(a_{11}, a_{12}, b_{12}), (a_{13}, a_{14}, b_{14}), (a_{21}, a_{22}, b_{22}), (a_{23}, a_{24}, b_{24}), (a_{31}, a_{32}, b_{32}), (a_{33}, a_{34}, b_{34})$

a clause in three variables

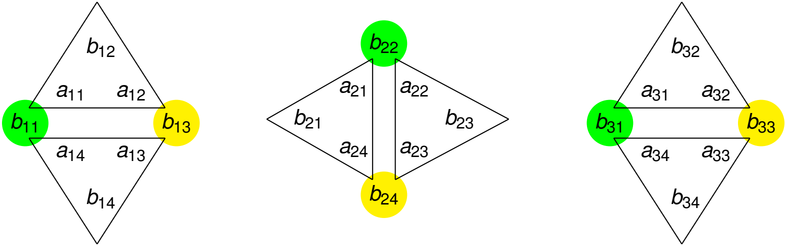
The clause $\overline{x}_1 \vee x_2 \vee \overline{x}_3$ is represented by the three triples: (p, p', b_{11}), (p, p', b_{22}), and (p, p', b_{31}).

For a perfect 3-dimensional matching corresponding to a satisfying assignment, add (q_1, q_1', b_{22}), (q_2, q_2', b_{31}), (q_3, q_3', b_{13}), (q_4, q_4', b_{24}), (q_5, q_5', b_{33}). In total, we have $8 = 2^3$ triples, for 3 variables.



$x_1 = 0$ $x_2 = 1$ $x_3 = 0$

an instance of 3-dimensional matching



- triples for the assignments of x_1, x_2, and x_3:
  (a_{11}, a_{12}, b_{12}), (a_{12}, a_{13}, b_{13}), (a_{13}, a_{14}, b_{14}), (a_{14}, a_{11}, b_{11}),
  (a_{21}, a_{22}, b_{22}), (a_{22}, a_{23}, b_{23}), (a_{23}, a_{24}, b_{24}), (a_{24}, a_{21}, b_{21}),
  (a_{31}, a_{32}, b_{32}), (a_{32}, a_{33}, b_{33}), (a_{33}, a_{34}, b_{34}), (a_{34}, a_{31}, b_{31}).
- triples for the clause: (p, p', b_{11}), (p, p', b_{22}), (p, p', b_{31}).
- triples to make the matching perfect: (q_1, q'_1, b_{22}), (q_2, q'_2, b_{31}), (q_3, q'_3, b_{13}), (q_4, q'_4, b_{24}), (q_5, q'_5, b_{33}).

a 3-dimensional matching problem

- triples for the assignments of x_1, x_2, and x_3: (a_{11}, a_{12}, b_{12}), (a_{12}, a_{13}, b_{13}), (a_{13}, a_{14}, b_{14}), (a_{14}, a_{11}, b_{11}), (a_{21}, a_{22}, b_{22}), (a_{22}, a_{23}, b_{23}), (a_{23}, a_{24}, b_{24}), (a_{24}, a_{21}, b_{21}), (a_{31}, a_{32}, b_{32}), (a_{32}, a_{33}, b_{33}), (a_{33}, a_{34}, b_{34}), (a_{34}, a_{31}, b_{31}).
- triples for the clause: (p, p', b_{11}), (p, p', b_{22}), (p, p', b_{31}).
- triples to make the matching perfect: (q_1, q_1', b_{22}), (q_2, q_2', b_{31}), (q_3, q_3', b_{13}), (q_4, q_4', b_{24}), (q_5, q_5', b_{33}).

Three sets X, Y, and Z:

- $X = \{a_{11}, a_{13}, a_{21}, a_{23}, a_{31}, a_{33}, p, q_1, q_2, q_3, q_4, q_5\}$
- $Y = \{a_{12}, a_{14}, a_{22}, a_{24}, a_{32}, a_{34}, p', q_1', q_2', q_3', q_4', q_5'\}$
- $Z = \{b_{11}, b_{12}, b_{13}, b_{14}, b_{21}, b_{22}, b_{23}, b_{24}, b_{31}, b_{32}, b_{33}, b_{34}\}$

We can verify that a perfect 3-dimensional matching corresponds to a satisfying truth assignment for the clause.

Partitioning and Numerical Problems

1. The 3-Dimensional Matching Problem
   - a general version of bipartite matching
   - NP-completeness
   - reducing 3-SAT to 3-dimensional matching
2. Numerical Problems
   - the subset sum problem
   - reducing 3-dimensional matching to subset sum
3. Co-NP and the Asymmetry of NP
   - complementary problems
   - a partial taxonomy of hard problems

reducing 3-SAT to 3-dimensional matching

Consider an arbitrary instance of the 3-SAT problem:

1. k clauses $C_j = c_{j1} \vee c_{j2} \vee c_{j3}$, $j = 1, 2, \ldots, k$,
2. n variables x_i, $c_{j1}, c_{j2}, c_{j3} \in \{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$.

For each variable, define the following:

- $A_i = \{a_{i1}, a_{i2}, \ldots, a_{i\,2k}\}$, $B_i = \{b_{i1}, b_{i2}, \ldots, b_{i\,2k}\}$,
- $t_{ij} = (a_{ij}, a_{i\,j+1}, b_{ij})$, for $j = 1, 2, \ldots, 2k$, where $j + 1 = 1$ if $j = 2k$.

For each clause C_j, add elements p_j, p'_j, and for each term t of C_j:

- if $t = x_i$, add the triple $(p_j, p'_j, b_{i\,2j})$, or otherwise
- if $t = \overline{x}_i$, add the triple $(p_j, p'_j, b_{i\,2j-1})$.

To cover all $2kn$ elements b_{ij}, we have nk covered by assignments to x_i and k covered by the clauses, so we still need $(n-1)k$ triples:

- add $2(n-1)$ elements q_i, q'_i,
- add $(n-1)k$ triples (q_i, q'_i, b), for every b not covered by a clause.

To an arbitrary instance of the 3-SAT problem:

1. k clauses $C_j = c_{j1} \vee c_{j2} \vee c_{j3}$, $j = 1, 2, \ldots, k$,
2. n variables x_i, $c_{j1}, c_{j2}, c_{j3} \in \{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$.

corresponds an instance of the 3-dimensional matching problem:

1. $2nk + 3k + (n-1)k$ triples T
2. X contains all a_{ij} with j odd, the p_j and the q_j elements
3. Y contains all a_{ij} with j even, the p'_j and the q'_j elements
4. Z contains all b_{ij} elements.

As $2nk + 3k + (n-1)k$ is a polynomial, we have a polynomial number of computational steps to reduce 3-SAT to 3-dimensional matching.

encoding a 3-SAT instance as 3-dimensional matching

Exercise 2: Encode

$$(x \vee \overline{y} \vee z) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (x \vee y \vee \overline{z})$$

as a 3-dimensional matching problem.

Demonstrate the interpretation of a truth assignment as a perfect matching in a tripartite graph, and vice versa.

3-SAT is equivalent to 3-dimensional matching

Lemma (2)

The 3-SAT problem is satisfiable if and only if there is a perfect 3-dimensional matching in the set of triples T.

Proof. If and only if means $\Rightarrow$ and $\Leftarrow$.

$\Rightarrow$ If we have a satifying truth statement, then the triples are chosen corresponding to the 0/1 values for each variable.

For each clause C_j, a triple containing p_j and p'_j is selected.

A choice of the remaining triples which contain q_i and q'_k gives a perfect 3-dimensional matching.

3-SAT is equivalent to 3-dimensional matching

Lemma (2)

The 3-SAT problem is satisfiable if and only if there is a perfect 3-dimensional matching in the set of triples T.

Proof. continued with the $\Leftarrow$ case.

$\Leftarrow$ If we have a perfect 3-dimensional matching, then the matching contains

- either all the even triples t_{ij},
- or all the odd triples t_{ij}

where the indices specify the 0/1 assignment of the variable x_i.

As we have a perfect matching, each p_j and p'_j corresponding to the clause C_j was selected to satisfy C_j.

Q.E.D.

NP-completeness

Theorem (3)

The 3-dimensional matching problem is NP-complete.

Proof. By Lemma (1), the 3-dimensional matching problem is in $\mathcal{NP}$.

An arbitrary 3-SAT problem can be reduced in polynomial time to a 3-dimensional matching problem. In particular, for k clauses in n variables, we construct $2nk + 3k + (n-1)k$ triples.

By Lemma (2), the reformulated 3-SAT problem can be solved by the 3-dimensional matching problem.

Q.E.D.

partitioning and graph coloring

A graph is k-colorable if k colors suffice to assign colors to all vertices so no two adjacent vertices have the same color.
Given a graph G and a natural number k, does G have a k-coloring?

Theorem (4)

A graph if 2-colorable if and only if it is bipartite.

Reducing 3-SAT to 3-coloring, the following can be proven:

Theorem (5)

3-coloring is NP-complete.

Partitioning and Numerical Problems

1. The 3-Dimensional Matching Problem
   - a general version of bipartite matching
   - NP-completeness
   - reducing 3-SAT to 3-dimensional matching
2. Numerical Problems
   - the subset sum problem
   - reducing 3-dimensional matching to subset sum
3. Co-NP and the Asymmetry of NP
   - complementary problems
   - a partial taxonomy of hard problems

CS 401/MCS 401 Lecture 21
Computer Algorithms I
Jan Verschelde, 4 August 2025

Partitioning and Numerical Problems

1. The 3-Dimensional Matching Problem
   - a general version of bipartite matching
   - NP-completeness
   - reducing 3-SAT to 3-dimensional matching

2. Numerical Problems
   - the subset sum problem
   - reducing 3-dimensional matching to subset sum

3. Co-NP and the Asymmetry of NP
   - complementary problems
   - a partial taxonomy of hard problems

the subset sum problem

A decision version of the subset sum problem is

> Given natural numbers w_1, w_2, ..., w_n and W,
> is there a subset $S \subseteq \{w_1, w_2, \ldots, w_n\}$ so that $\displaystyle\sum_{w \in S} w = W$?

With dynamic programming we obtained an $O(nW)$ algorithm, with a pseudo-polynomial running time.

But W is given as a bit sequence

$$c_k 2^k + \cdots + c_2 2^2 + c_1 2 + c_0, \quad c_i \in \{0, 1\}.$$

The size of W grows exponentionally with k, the size of the input.

solve the subset sum problem

The subset sum problem is important.

Exercise 3: Solve the subset sum problem by giving an algorithm to find a subset whose weights sums up to the given W.

What is the worst case running time of your algorithm?

subset sum is in $\mathcal{NP}$

Lemma (6)

The subset sum problem is in $\mathcal{NP}$.

Proof. The certificate consists of the subset $S \subseteq \{w_1, w_2, \ldots, w_n\}$.

The certifier computes $\displaystyle\sum_{w \in S} w$ and checks whether this sum equals W.

The computation of the sum and the check require $O(n)$ operations.

So we have an efficient certifier and subset sum is in $\mathcal{NP}$.

Q.E.D.

Partitioning and Numerical Problems

3-dimensional matching is NP-complete

Given are the following:

- Three sets X, Y, and Z:
  1. each set has the same size $n = \#X = \#Y = \#Z$, and
  2. the sets are disjoint: $X \cap Y = \emptyset$, $X \cap Z = \emptyset$, $Y \cap Z = \emptyset$.
- A set T of ordered triples $T \subseteq X \times Y \times Z$.

The 3-dimensional matching problem asks

> Is there a set of n triples in T so each element of $X \cup Y \cup Z$ is contained exactly once in one of these triples?

Such a set of triples is *a perfect three-dimensional matching*.

Example: $X = \{\text{instructors}\}$, $Y = \{\text{courses}\}$, $Z = \{\text{times}\}$.

reducing 3-dimensional matching to subset sum

To prove that subset sum is NP-complete, we will show that

$$\text{3-Dimensional Matching} \leq_P \text{Subset Sum}.$$

Any instance of the 3-dimensional matching problem is given by

- 3 sets X, Y, and Z, all of size n, and
- a set of m triples $T \subseteq X \times Y \times Z$.

A set can be represented as a bit vector

- bit i is 1 if the i-th element belongs to the set,
- bit i is 0 otherwise.

Any bit vector can be seen to hold the coefficients of a number.
A triple $t = (x_i, y_j, z_k)$ is represented by a bit vector of size $3n$:

1. for x_i, we have a 1 at position i,
2. for y_j, we have a 1 at position $n + j$,
3. for z_k, we have a 1 at position $2n + k$.

an example

triple t_i			x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4	z_1	z_2	z_3	z_4	w_i
x_1	y_2	z_3	1	0	0	0	0	1	0	0	0	0	1	0	100,001,000,010
x_2	y_4	z_2	0	1	0	0	0	0	0	1	0	1	0	0	10,000,010,100
x_1	y_1	z_1	1	0	0	0	1	0	0	0	1	0	0	0	100,010,001,000
x_2	y_2	z_4	0	1	0	0	0	1	0	0	0	0	0	1	10,001,000,001
x_4	y_3	z_4	0	0	0	1	0	0	1	0	0	0	0	1	100,100,001
x_3	y_1	z_2	0	0	1	0	1	0	0	0	0	1	0	0	1,010,000,100
x_3	y_1	z_3	0	0	1	0	1	0	0	0	0	0	1	0	1,010,000,010
x_3	y_1	z_1	0	0	1	0	1	0	0	0	1	0	0	0	1,010,001,000
x_4	y_4	z_4	0	0	0	1	0	0	0	1	0	0	0	1	100,010,001
															111,111,111,111

Observe:

We have a 3-dimensional matching if and only if the sum of some subset is $W = 111,111,111,111$.

constructing the weights

The m triples in T define bit vectors of size $3n$.
We defined $3nm$ bits, a polynomial number in the dimensions n and m of the 3-dimensional matching problem.

For base B, the triple $t = (x_i, y_j, z_k)$, defines the number

$$w_t = B^{i-1} + B^{n+j-1} + B^{2n+k-1}.$$

The adding of numbers corresponds to taking unions of triples, *provided there is no carry over when adding numbers*.

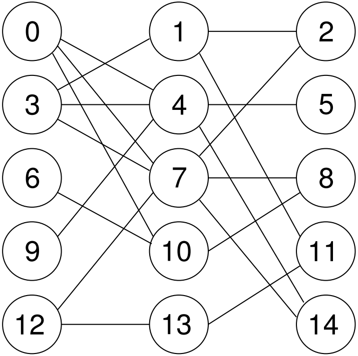
We prevent carry over, if the base is large enough. Take $B = m + 1$.

Define W as a number in base $m + 1$ with $3n$ digits, all equal to one:

$$W = \sum_{i=0}^{3n} (m+1)^i.$$

encode 3-dimensional matching into subset sum

Exercise 4: Consider the 3-dimensional matching problem, given by $X = \{ 0, 3, 6, 9, 12 \}$, $Y = \{ 1, 4, 7, 10, 13 \}$, $Z = \{ 2, 5, 8, 11, 14 \}$.



1. Construct the weights for the subset sum problem.
2. Demonstrate that the solutions to the two problems are equivalent.

3-dimensional matching is equivalent to subset sum

Lemma (7)

The set T of triples contains a perfect 3-dimensional matching if and only if there is a subset of the weights that adds up to W.

Proof. If and only if means $\Rightarrow$ and $\Leftarrow$.

- $\Rightarrow$ If we have a perfect 3-dimensional matching, then in the sum of the weights, there is a single one at each position in the $3n$-digit number, so the result equals W.
- $\Leftarrow$ If we have a subset of weights w_{t_1}, w_{t_2}, ..., w_{t_k} that adds up to W, then since each w_{t_ℓ} has three ones in its representation and there are no carry overs, so $k = n$.
  For each of the $3n$ positions, exactly one of the w_{t_ℓ} has a one in that position. Therefore, the t_1, t_2, ..., t_k define a perfect matching.

Q.E.D.

NP-completeness

Theorem (8)

The subset sum problem is NP-complete.

Proof. By Lemma (6), the subset sum problem is in $\mathcal{NP}$.

An arbitrary instance of the 3-dimensional matching problem can be reduced in polynomial time to a subset sum problem.

By Lemma (7), the reformulated 3-dimensional matching problem can be solved by the subset sum problem.

Q.E.D.

Scheduling with Release Times and Deadlines

We want to schedule n jobs on a single machine.

The input for each job i, $i = 1, 2, \ldots, n$, has three numbers:

1. r_i is the release time, when it is available for processing,
2. t_i is the processing duration, and
3. d_i is the deadline by which it must be completed.

Can we schedule all jobs so that each completes by its deadline?

The subset sum problem can be reduced in polynomial time to scheduling with release times and deadlines.

Theorem (9)

Scheduling with release times and deadlines is NP-complete.

Partitioning and Numerical Problems

1. The 3-Dimensional Matching Problem
   - a general version of bipartite matching
   - NP-completeness
   - reducing 3-SAT to 3-dimensional matching

2. Numerical Problems
   - the subset sum problem
   - reducing 3-dimensional matching to subset sum

3. Co-NP and the Asymmetry of NP
   - complementary problems
   - a partial taxonomy of hard problems

complementary problems

A problem X is in $\mathcal{NP}$ if there is an efficient certifier B:
$s \in X$ if and only if there is a short t so that $B(s, t) = \texttt{"yes"}$.

Negating the statement:
$s \not\in X$ if and only if for all short t so that $B(s, t) = \texttt{"no"}$.
In this case, there is no short proof available.

Definition (10)

Each problem X defines *the complementary problem* $\overline{X}$:
$s \in \overline{X}$ if and only if $s \not\in X$.

polynomial-time algorithms

Definition (10)

Each problem X defines *the complementary problem* $\overline{X}$:
$s \in \overline{X}$ if and only if $s \not\in X$.

Proposition (11)

If $X \in \mathcal{P}$ then $\overline{X} \in \mathcal{P}$.

Proof. If $X \in \mathcal{P}$, then an efficient algorithm A returns `"yes"` for all $s \in X$.

The algorithm $\overline{A}$ runs A and negates its result.

Thus $\overline{A}$ is an efficient algorithm to solve $\overline{X}$.

Q.E.D.

the class co-$\mathcal{NP}$

If $X \in \mathcal{NP}$, then $\overline{X} \in \mathcal{NP}$?

$X \in \mathcal{NP}$ implies we have an efficient certifier B:
for all s: $s \in X$ if and only if there is a short t so $B(s, t) =$"yes".

Inverting the statement does not lead to a $\overline{B}$,
because *there exists a t* becomes *for all t* in the negation.

Definition (12)

The class *co-$\mathcal{NP}$* is $\{$ problem $X \mid \overline{X} \in \mathcal{NP}$ $\}$.

Whether $\mathcal{NP}$ equals co-$\mathcal{NP}$ is not known.

we believe $\mathcal{NP} \neq$ co-$\mathcal{NP}$ because ...

Proposition (13)

$\mathcal{NP} \neq$ co-$\mathcal{NP}$ implies $\mathcal{P} \neq \mathcal{NP}$.

Proof. We proceed by contraposition.
Assuming $\mathcal{P} = \mathcal{NP}$ will lead to $\mathcal{NP} =$ co-$\mathcal{NP}$.

$$X \in \mathcal{NP} \Rightarrow X \in \mathcal{P} \Rightarrow \overline{X} \in \mathcal{P} \Rightarrow \overline{X} \in \mathcal{NP} \Rightarrow X \in \text{co-}\mathcal{NP}.$$

The above statement implies $\mathcal{NP} \subseteq$ co-$\mathcal{NP}$.

$$X \in \text{co-}\mathcal{NP} \Rightarrow \overline{X} \in \mathcal{NP} \Rightarrow \overline{X} \in \mathcal{P} \Rightarrow X \in \mathcal{P} \Rightarrow X \in \mathcal{NP}.$$

The above statement implies co-$\mathcal{NP} \subseteq \mathcal{NP}$.

$\mathcal{NP} \subseteq$ co-$\mathcal{NP}$ and co-$\mathcal{NP} \subseteq \mathcal{NP}$ implies $\mathcal{NP} =$ co-$\mathcal{NP}$.

Q.E.D.

Partitioning and Numerical Problems

1. The 3-Dimensional Matching Problem
   - a general version of bipartite matching
   - NP-completeness
   - reducing 3-SAT to 3-dimensional matching

2. Numerical Problems
   - the subset sum problem
   - reducing 3-dimensional matching to subset sum

3. Co-NP and the Asymmetry of NP
   - complementary problems
   - a partial taxonomy of hard problems

a partial taxonomy of hard problems

Circuit satisfiability was the first NP-complete problem.

Then we proved that 3-SAT is NP-complete.

1. packing and covering problems

   Independent Set, Set Packing, Vertex Cover, Set Cover
2. sequencing problems

   Hamiltonian Cycle, Traveling Salesman
3. partitioning problems

   3-Dimensional Matching, 3-Coloring
4. numerical problems

   Subset Sum, Scheduling with Release Times and Deadlines

a very short summary of the first four chapters

1. An algorithm is an ordered set of unambiguous, executable steps that define a terminating process.
   The Gale-Shapley algorithm solves the stable matching problem.

2. An algorithm is efficient if its worst case asymptotic cost is bounded by a polynomial in its input size. Implementing an algorithm determines running times of operating data structures.

3. Graph traversals have a running time of $O(n + m)$ for n vertices and m edges, which is also the cost of topological ordering of a directed acyclic graph.

4. Greedy algorithms solve optimization problems,
   such as interval scheduling, caching, shortest paths,
   spanning trees, via locally optimal rules.

a very short summary of the next four chapters

5. Divide and conquer methods applied to counting inversions, closest pairs of points, integer multiplication, convolutions, and matrix multiplications give subquadratic and subcubic times.
6. Dynamic programming derives a polynomial number of subproblems which joint solutions solve the original problem with a recurrence, applied to weighted interval scheduling, segmented least squares, sequence alignment, subset sum, knapsack, shortest paths in directed graphs with negative weights.
7. The Ford-Fulkerson algorithm computes maximum flow in pseudo-polynomial time and solves bipartite matching, circulation with demands, edge-disjoint paths, survey design, and airline scheduling.
8. If a polynomial time algorithm existed for an NP-complete problem, then all problems for which we have an efficient certifier could be solved in polynomial time.