CMSC 451: More NP-completeness Results

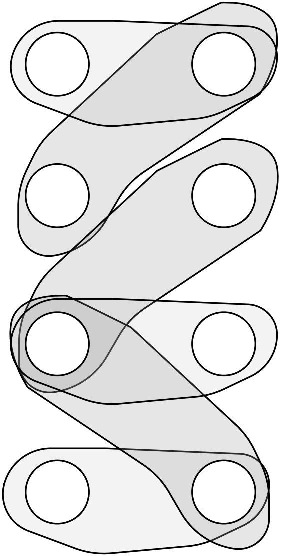
Slides By: Carl Kingsford

Department of Computer Science
University of Maryland, College Park

Based on Sect. 8.5,8.7,8.9 of *Algorithm Design* by Kleinberg & Tardos.

Three-Dimensional Matching

Two-Dimensional Matching



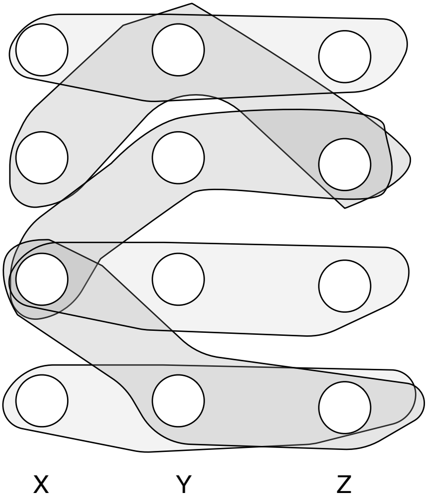
Recall '2-d matching':

Given sets X and Y, each with n elements, and a set E of pairs $\{x, y\}$,

Question: is there a choice of pairs such that every element in $X \cup Y$ is paired with some other element?

Usually, we thought of edges instead of pairs: $\{x, y\}$, but they are really the same thing.

Three-Dimensional Matching



Given: Sets X, Y, Z, each of size n, and a set $T \subset X \times Y \times Z$ of order triplets.

Question: is there a set of n triplets in T such that each element is contained in exactly one triplet?

3DM Is NP-Complete

Theorem

Three-dimensional matching (aka 3DM) is NP-complete

Proof. 3DM is in NP: a collection of n sets that cover every element exactly once is a certificate that can be checked in polynomial time.

Reduction from 3-SAT. We show that:

$$\text{3-SAT} \leq_P \text{3DM}$$

In other words, if we could solve 3DM, we could solve 3-SAT.

3-SAT $\leq_P$ 3DM

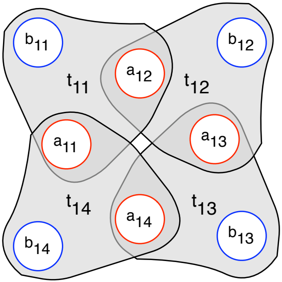
3SAT instance: $x_1, \ldots, x_n$ be n boolean variables, and $C_1, \ldots, C_k$ clauses.

We create a gadget for each variable x_i:

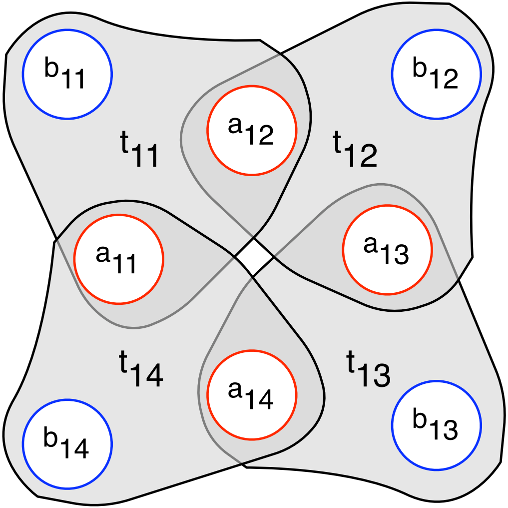
$A_i = \{a_{i1}, \ldots, a_{i,2k}\}$ *core*

$B_i = \{a_{i1}, \ldots, a_{i,2k}\}$ *tips*

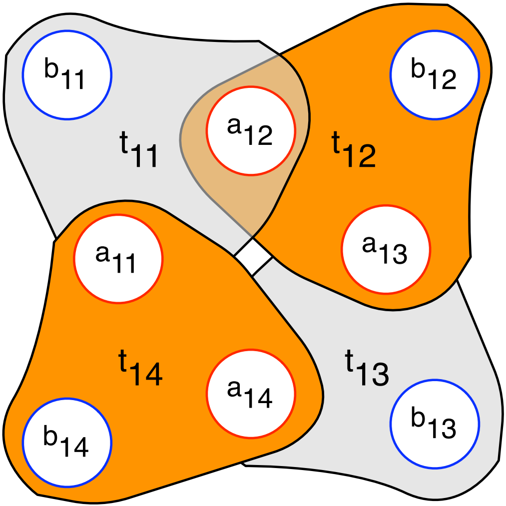
$t_{ij} = (a_{ij}, a_{i,j+1}, b_{ij})$ *TF triples*



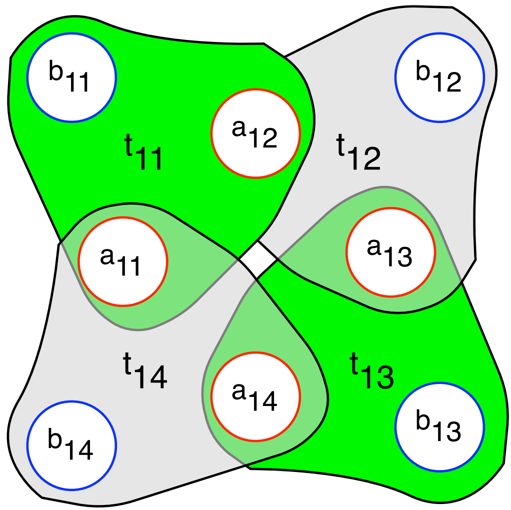
Gadget Encodes True and False



Gadget Encodes True and False

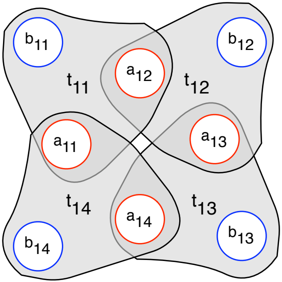


Gadget Encodes True and False



How “choice” is encoded

- We can only either use the even or odd “wings” of the gadget.
- In other words, if we use the even wings, we leave the odd tips uncovered (and vice versa).
- Leaving the odd tips free for gadget i means setting x_i to **false**.
- Leaving the odd tips free for gadget i means setting x_i to **true**.



Clause Gadgets

Need to encode constraints between the tips that ensure we satisfy all the clauses.

We create a gadget for each clause $C_j = \{t_1, t_2, t_3\}$

$$P_j = \{c_j, c_j'\} \quad \textit{Clause core}$$

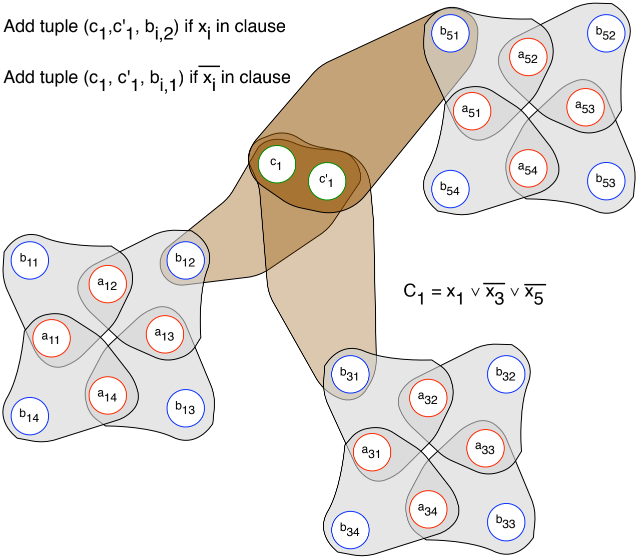
We will hook up these two clause core nodes with some tip nodes depending on whether the clause asks for a variable to be true or false.

See the next slide.

Clause Gadget Hookup

Add tuple $(c_1, c'_1, b_{i,2})$ if x_i in clause

Add tuple $(c_1, c'_1, b_{i,1})$ if $\overline{x_i}$ in clause



$C_1 = x_1 \vee \overline{x_3} \vee \overline{x_5}$

Clause Gadgets

Since only clause tuples (brown) cover c_j, c'_j, we have to choose exactly one of them for every clause.

We can only choose a clause tuple (c_j, c'_j, b_{ij}) if we haven't chosen a TF tuple that already covers b_{ij}.

Hence, we can satisfy (cover) the clause (c_j, c'_j) with the term represented by b_{ij} only if we "set" x_i to the appropriate value.

That's the basic idea. Two technical points left...

Details

Need to cover all the tips:

Even if we satisfy all the clauses, we might have extra tips left over. We add a clean up gadget (q_i, q_i', b) for every tip b.

Can we partition the sets?

$$X = \{a_{ij} : j \text{ even}\} \cup \{c_j\} \cup \{q_i\}$$
$$Y = \{a_{ij} : j \text{ odd}\} \cup \{c_j'\} \cup \{q_i'\}$$
$$Z = \{b_{ij}\}$$

Every set we defined uses 1 element from each of X, Y, Z.

Proof

If there is a satisfying assignment,

We choose the odd / even wings depending on whether we set a variable to **true** or **false**. At least 1 free tip for a term will be available to use to cover each clause gadget. We then use the clean up gadgets to cover all the rest of the tips.

If there is a 3D matching,

We can set variable x_i to **true** or **false** depending on whether it's even or odd wings were chosen. Because $\{c_j, c_j'\}$ were covered, we must have correctly chosen one even/odd wing that will satisfy this clause.

Subset Sum

Subset Sum

Subset Sum Problem

Given n natural numbers $w_1, \ldots, w_n$ and a number W, is there a subset of $w_1, \ldots, w_n$ that adds up exactly to W?

We saw a $O(nW)$ dynamic programming algorithm for this problem earlier in the semester.

But this is pseudo-polynomial! Even problems with pseudo-polynomial algorithms can be **NP**-complete.

Reason: W is actually exponential in the input size, $O(\log W)$.

Subset Sum is **NP**-complete

Theorem

Subset Sum is **NP***-complete.*

Proof. (1) Subset Sum is in **NP**: a certificate is the set of numbers that add up to W.

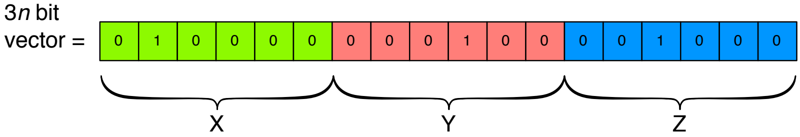
(2) 3-DM $\leq_P$ Subset Sum.

Instance of 3-DM: Let X, Y, Z be sets of size n and let $T \subseteq X \times Y \times Z$ be a set of tuples.

We encode this 3-DM instance into a instance of Subset Sum.

Bit Vectors

Encode each tuple $(x, y, z) \subseteq X \times Y \times Z$ as a bit vector:



Each tuple $t \in T$ corresponds to a number

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

for some base d.

Union $\equiv$ to Sum

For 3DM we want to choose a set of tuples that includes every element exactly once.

$t_1 \cup t_2$ corresponds to $w_{t_1} + w_{t_2}$:

t_1 =	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
t_2 =	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
$t_1 + t_2$ =	0	1	0	1	0	0	0	1	0	1	0	0	0	0	2	0	0	0

Goal: all ones

Set W equal to the number represented by the all 1s vector:

$$W = \sum_{i=0}^{3n-1} d^i$$

What base d should we use?

Want to avoid carries. Let m be the number of tuples in T.

Set d equal to $1 + m \implies$ Can't have any carries.

Proof

If T contains a 3-dimensional matching,

Then $t_1, \ldots, t_n$ then $w_{t_1} + \cdots + w_{t_n}$ contains a 1 in every position and equals W.

If $w_{t_1} + \cdots + w_{t_k} = W$,

Then $k = n$, and each of the $3n$ positions is covered by one 1 digit, and hence each element is covered by exactly 1 tuple.

Polynomially bounded numbers

If W is bounded by a polynomial function of n, then we can solve Subset Sum in polynomial time: $O(nW)$.

Other Complexity Classes

Asymmetry of **NP**

Suppose B is an efficient certifier for an NP problem.

Problems in **NP** have `yes`-instances with efficient certifiers:

> Instance I is a `yes` instance $\iff$ there is a short certificate C such that $B(I, C) =$ `yes`.

Negation:

> Instance I is a `no` instance $\iff$ for all short C, we have $B(I, C) =$ `no`.

I.e. we have short proofs for `yes`-instances, but not necessarily for `no`-instances.

Example

How would you convince me that G does not have an Hamiltonian cycle?

Co-NP

Recall that decision problems are really sets of strings.

For every decision problem X there is a complementary problem $\bar{X}$:

$$I \in \bar{X} \iff I \notin X.$$

That is, $\bar{X}$ contains those instances that X does not.

Characterization of $\bar{X}$:

Instance $I \in \bar{X} \iff$ for all short certificates C, $B(I, C) =$ `no`.

Open Question

Def. A problem $\bar{X}$ is in co-**NP** iff the complementary problem X belongs to **NP**.

- These are the problems that have efficient "no" certificates.
- Does **NP** $=$ co-**NP**? We don't know.

Theorem

If **NP** $\neq$ *co-***NP***, then* **P** $\neq$ **NP***.*

Proof. Contrapositive: **P** $=$ **NP** $\implies$ **NP** $=$ co-**NP**.

Since **P** is closed under complementation, if **P** $=$ **NP**, then **NP** $=$ co-**NP**.

Good Characterizations?

Consider the set: $\mathbf{NP} \cap \text{co-}\mathbf{NP}$.

These are the problems that have short "yes" proofs and short "no" proofs.

Any problem in $\mathbf{P}$ is in both $\mathbf{NP}$ and $\text{co-}\mathbf{NP}$, so $\mathbf{P} \subseteq \mathbf{NP} \cap \text{co-}\mathbf{NP}$.

Open Question: Does $\mathbf{P} = \text{co-}\mathbf{NP}$?

Summary of NP-complete problems

We've seen NP-completeness proofs for many problems:

- Independent Set
- Vertex Cover
- Set Cover
- 3-Dimensional matching
- Graph Coloring and 3-Coloring
- SAT and 3-SAT
- Hamiltonian Path and Cycle
- Traveling Salesman
- Subset Sum