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
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?
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:

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

In other words, if we could solve 3DM, we could solve 3-SAT.
**3-SAT \( \leq_P \) 3DM**

**3SAT instance:** Let \( x_1, \ldots, x_n \) 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.
Need to encode constraints between the tips that ensure we satisfy all the clauses.

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

$$P_j = \{c_j, c'_j\} \quad \text{Clause core}$$

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.
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

\[C_1 = x_1 \lor \overline{x_3} \lor \overline{x_5}\]
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...
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 \(X, Y, Z\).
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.