CS 341: Algorithms

Lecture 22: NP-completeness, continued

Éric Schost

based on lecture notes by many other CS341 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Fall 2024

the Traveling Salesman Problem

Traveling salesman is NP-complete

Definition: TSP

- **input:** n, nonnegative integer "distances" $d_{u,v}$, $1 \leq u < v \leq n$, integer K
- **output:** is there a Hamiltonian cycle C in the complete graph G_n on n vertices with $\sum_{\{u,v\} \in C} d_{u,v} \leq K$
- **NP** (certificate? certifier? runtime? correctness?)

Claim

HamiltonianCycle $\leq_P$ TSP

Proof: given $G = (V, E)$, set $n = |V|$

- reduction: $d_{u,v} = 1$ if $\{u, v\}$ in E, $d_{u,v} = 2$ otherwise, and $K = n$
- Hamiltonian cycle in G_n with $\sum_{\{u,v\} \in C} d_{u,v} \leq n \iff d_{u,v} = 1$ for all $\{u, v\}$ on C $\iff$ all $\{u, v\}$ on C are in E

Remark: Euclidean traveling salesman

Definition: EUCLIDEANTSP

- **input:** $(x_u, y_u)_{1 \leq u \leq n}$ integers
- **output:** same as above, with $d_{u,v} = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$ (not necessarily integers)

Unknown if NP

- **open:** how to test efficiently if a sum of square roots of integers is $\leq K$

NP-hard

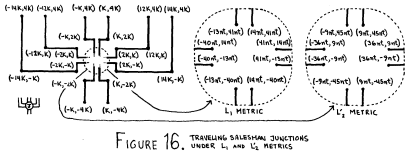


FIGURE 16. TRAVELING SALESMAN JUNCTIONS UNDER L_1 AND L'_1 METRICS

Colorability

kColorable

Definition:

- **input:** symmetric graph G, integer k
- **output:** can we color all vertices of G with $\boldsymbol{k}$ colors, with adjacent vertices having **different** colors?
- **NP**

Remark

- $\boldsymbol{k = 1}$: G 1-colorable iff no edge
- $\boldsymbol{k = 2}$: G 2-colorable iff bipartite

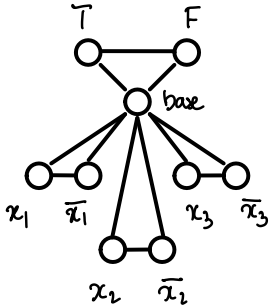
kColorable is NP-complete

Claim

$3\text{SAT} \leq_P \text{kColorable}$

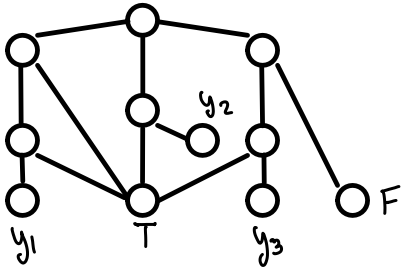
Starting the construction

- given a formula in 3CNF, e.g. $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2)$
- build a graph with vertices x_i and $\overline{x_i}$, T, F, and a base
- 3-coloring $\iff$ variable assigments (x_i true if the same color as T)



3-colorability gadget

For any clause $\boldsymbol{y_1} \vee \boldsymbol{y_2} \vee \boldsymbol{y_3}$, attach graph below to the previous construction

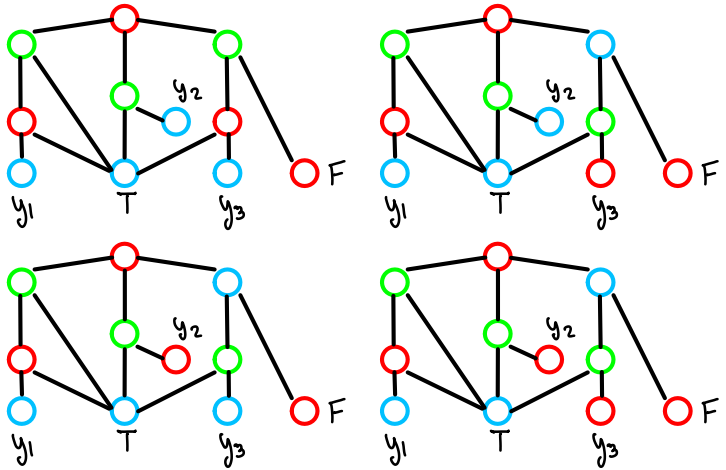


Claim

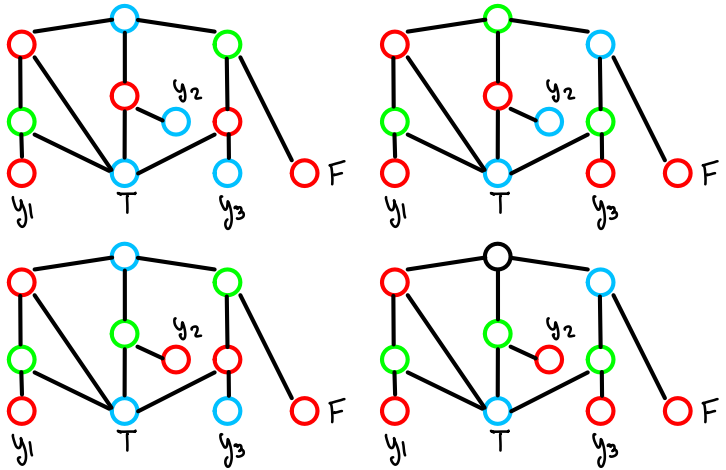
Assuming we can use colors T or F for y_1, y_2, y_3, this graph is 3-colorable **iff** at least one of y_1, y_2, y_3 is colored T

$\Longrightarrow$ new graph 3-coloriable **iff** all clauses can simultaneously be satisfied

Case discussion 1/2



Case discussion 2/2



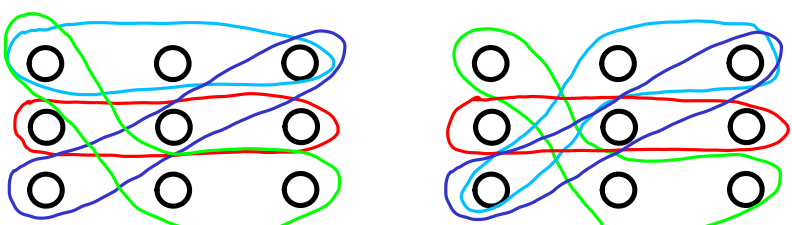
Perfect 3D matchings

Definition

3D matching

- **input:** 3 sets X, Y, Z of size n and a family of **hyperedges** $E \subset X \times Y \times Z$ (don't require $|E| = n$)
- **output:** is there a **perfect matching** (n hyperedges that cover X, Y and Z)? each x_i, and each y_j, and each z_k, is in a unique hyperedge
- **NP**

Examples



Definition

3D matching

- **input:** 3 sets X, Y, Z of size n and a family of **hyperedges** $E \subset X \times Y \times Z$ (don't require $|E| = n$)
- **output:** is there a **perfect matching** (n hyperedges that cover X, Y and Z)? each x_i, and each y_j, and each z_k, is in a unique hyperedge
- **NP**

Remark: 2D version

- **input:** 2 sets X, Y of size n and a family of **edges** $E \subset X \times Y$
- **output:** is there a **perfect matching** (n edges that cover X, Y)?
- this is testing if a bipartite graph has a perfect matching
- in **P** via max-flow

3DMatchings is NP-complete

(this is yet another question where 2 is easy and 3 looks hard)

Claim

3SAT $\leq_P$ 3DMatching

- **given:** a formula F in 3CNF, with s clauses $C_1, \ldots, C_s$
- **want:** an instance $H = (X, Y, Z, E)$ of 3DMatching such that F satisfiable iff H admits a perfect 3D matching
- reduction must be polynomial time

The variable gadget

build **one fidget-spinner-thing per variable** x_i, $i = 1, \ldots, n$.

Vertices (not done)

- $2s$ **core vertices** $v_{i,1}, \ldots, v_{i,2s}$ — only used in the gadget
- $2s$ **tip vertices** $z^T_{i,1}, z^F_{i,1}, \ldots, z^T_{i,s}, z^F_{i,s}$ — will connect to clause vertices

Partition

- $\boldsymbol{v_{i,1}, v_{i,3}, v_{i,5}, \ldots}$ in $\boldsymbol{X}$ — $\boldsymbol{ns}$ so far
- $\boldsymbol{v_{i,2}, v_{i,4}, v_{i,6}, \ldots}$ in $\boldsymbol{Y}$ — $\boldsymbol{ns}$ so far
- all tips in $\boldsymbol{Z}$ — $\boldsymbol{2ns}$ (done)

Hyperedges: for $i = 1, \ldots, n,$, $j = 1, \ldots, s$

- $\{v_{i,\mathbf{2j-1}},\ v_{i,\mathbf{2j}},\ z^T_{i,j}\}$
- $\{v_{i,\mathbf{2j+1}},\ v_{i,\mathbf{2j}},\ z^F_{i,j}\}$

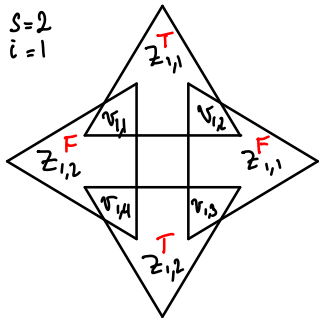
(not done)

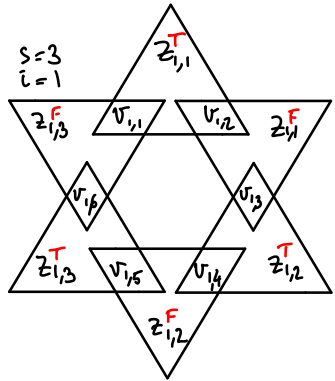
The variable gadget

build **one fidget-spinner-thing per variable** x_i, $i = 1, \ldots, n$.

Vertices (not done)

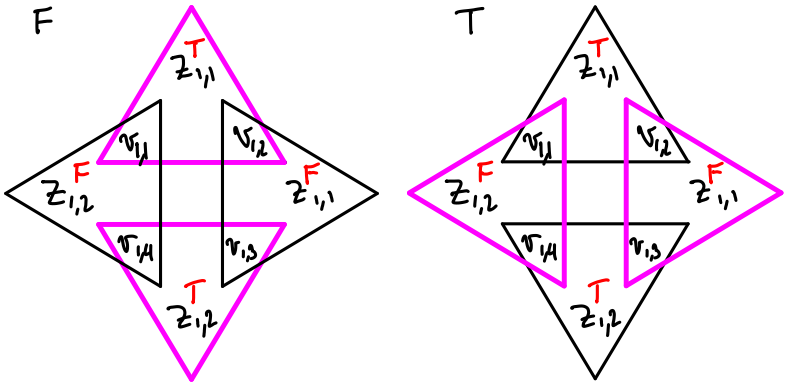
- $2s$ **core vertices** $v_{i,1}, \ldots, v_{i,2s}$ — only used in the gadget
- $2s$ **tip vertices** $z^T_{i,1}, z^F_{i,1}, \ldots, z^T_{i,s}, z^F_{i,s}$ — will connect to clause vertices





Covering the core vertices

- there will be no other hyperedge covering the core vertices $v_{i,j}$
- so only **two** ways to cover a gadget

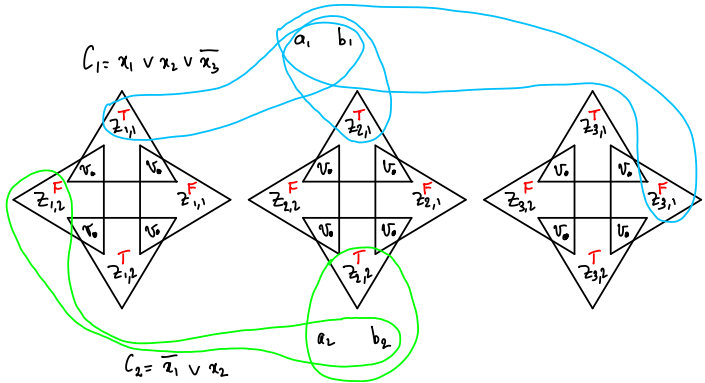


2^n coverings (2 possibilities per gadget) $\iff$ boolean values for the x_i's

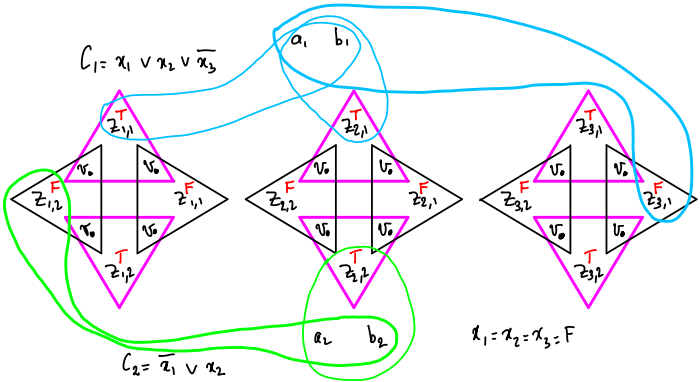
Using the clauses to (almost) finish the graph

For each clause C_j

- add two new vertices $\boldsymbol{a_j} \in \boldsymbol{X}$ and $\boldsymbol{b_j} \in \boldsymbol{Y}$
- for any literal $\boldsymbol{x_i}$ in C_j, add hyperedge $\{\boldsymbol{a_j}, \boldsymbol{b_j}, \boldsymbol{z^T_{i,j}}\}$
- for any literal $\overline{\boldsymbol{x_i}}$ in C_j, add hyperedge $\{\boldsymbol{a_j}, \boldsymbol{b_j}, \boldsymbol{z^F_{i,j}}\}$



Final adjustments



- we have $2ns = 12$ tips, with ns matched in pink and ns left
- in a perfect matching, each clause covers a tip, so $ns - s = 4$ tips left
- add $ns - s$ dummy vertices $\boldsymbol{d_k} \in \boldsymbol{X}$, $\boldsymbol{e_k} \in \boldsymbol{Y}$
- add **all** hyperedges $\{d_k,\ e_k,\ z^T_{i,j}\}$ and $\{d_k,\ e_k,\ z^F_{i,j}\}$ (that's $(ns - s)(2ns)$ of them)

F satisfiable iff perfect 3D matching

If F is satisfiable

- cover gadgets for $x_1, \ldots, x_n$ according to their truth value
- pick **exactly** one true literal per clause C_j
  - if x_i, take hyperedge $\{a_j, b_j, z^T_{i,j}\}$ $\quad z^T_{i,j}$ still free
  - if $\overline{x_i}$, take hyperedge $\{a_j, b_j, z^F_{i,j}\}$ $\quad z^F_{i,j}$ still free
- match all remaining tips with pairs of dummy vertices

If perfect 3D matching

- matching gives truth values
- for each clause C_j, we picked a hyperedge $\{a_j, b_j, z^T_{i,j}\}$, resp. $\{a_j, b_j, z^F_{i,j}\}$
- the corresponding x_i is T, resp. F
- this makes C_j satisfied either way

Subset sum and knapsack

Subset sum and Knapsack are NP-complete

Subset sum

- **given:** positive integers $a_1, \ldots, a_n$ and K
- **want:** is there a subset S of $\{1, \ldots, n\}$ with $\sum_{i \in S} a_i = K$
- **NP**

Claim

3DMatching $\leq_P$ SubsetSum $\leq_P$ Knapsack (this is already known)

- **given:** sets X, Y, Z of size n, m hyperedges $E \subset X \times Y \times Z$
- **want:** integers $a_1, \ldots, a_s, K$ s.t. perfect 3D matching iff $\sum_{i \in S} a_i = K$ for some $S \in \{1, \ldots, n\}$
- reduction must be polynomial time

From 3D matchings to vectors

we define $\boldsymbol{m}$ 0/1 vectors (one per hyperedge) of size $3n$.

- jth hyperedge $= \{x_{\boldsymbol{u}}, y_{\boldsymbol{v}}, z_{\boldsymbol{w}}\}$, $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ in $\{1, \ldots, n\}$
- jth vector given by

$$v_j = \Big[0 \cdots 0\,1\,0 \cdots 0 \;\; 0 \cdots 0\,1\,0 \cdots 0 \;\; 0 \cdots 0\,1\,0 \cdots 0\Big]$$

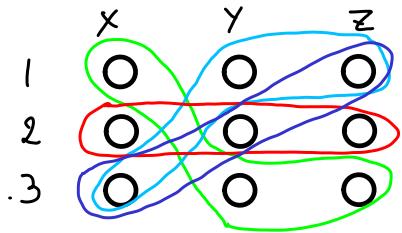
$$\uparrow_{u} \qquad \uparrow_{v+n} \qquad \uparrow_{w+2n}$$

Observation

there is a perfect 3D matching iff there is a subset of $\{v_1, \ldots, v_m\}$ that adds up to

$$\Big[1\,1\,1\,1\,1\,1 \cdots \cdots \cdots 1\,1\,1\,1\,1\,1\Big]$$

Example



$$v_1 = [\,1\,0\,0 \quad 0\,0\,1 \quad 0\,0\,1\,]$$
$$v_2 = [\,0\,1\,0 \quad 0\,1\,0 \quad 0\,1\,0\,]$$
$$v_3 = [\,0\,0\,1 \quad 1\,0\,0 \quad 1\,0\,0\,]$$
$$v_4 = [\,0\,0\,1 \quad 0\,1\,0 \quad 1\,0\,0\,]$$

From vectors to numbers

we define $\boldsymbol{m}$ integers (one per hyperedge) of bit-size polynomial in n, m

- set $\boldsymbol{b = m + 1}$ (large enough)
- given

$$v_j = \Big[0 \cdots 0\ 1\ 0 \cdots 0 \;\; 0 \cdots 0\ 1\ 0 \cdots 0 \;\; 0 \cdots 0\ 1\ 0 \cdots 0\Big]$$

$$\uparrow_{u} \qquad \uparrow_{v+n} \qquad \uparrow_{w+2n}$$

  define

$$\boldsymbol{a_j = b^u + b^{v+n} + b^{w+2n} = v_j \cdot [\ b\ b^2\ \cdots\ b^{3n}\ ]}$$

  (vector $v_j \leftrightarrow$ digits of a_j in base b)
- $a_j \le (m+1)^{3n+1}$ so $\log_2(a_j) \in (mn)^{O(1)}$

End of the proof

set $K = b + b^2 + \cdots + b^{3n}$ (so $\log(K) \in (nm)^{O(1)}$)

Claim

for S subset of $\{1, \ldots, m\}$, $\sum_{i \in S} v_i = [1 \ \cdots \ 1] \iff \sum_{i \in S} a_i = K$

1. we always have

$$\sum_{i \in S} a_i = \left(\sum_{i \in S} v_i \right) \cdot [\ b\ b^2\ \cdots\ b^{3n}\ ] = [\ c_1\ c_2\ \cdots\ c_{3n}\ ] \cdot [\ b\ b^2\ \cdots\ b^{3n}\ ], \quad 0 \leq c_j \leq m$$

2. if $\sum_{i \in S} v_i = [1 \ \cdots \ 1]$, $\sum_{i \in S} a_i = \sum_{j=1}^{3n} b^j = K$
3. if $\sum_{i \in S} a_i = K$,

$$\sum_{j=1}^{3n} b^j = \sum_{j=1}^{3n} c_j b^j$$

so $c_j = 1$ for all j **because $c_j < b$**, and $\sum_{i \in S} v_i = [1 \ \cdots \ 1]$