CS 344
LECTURE 21
APPROXIMATION ALGORITHMS
Let's say $P \neq NP$

And we have a hard problem to solve

What then?
We can use search techniques:

- backtracking
- branch-and-bound

(find optimum, but may take exponential time)
We can use approximation algorithms (faster, but usually don't find the optimum)
We can use heuristics

(no guarantees on running time or quality of result)
BACKTRACKING
Initial formula $\phi$

\[
\begin{align*}
& w = 0 \\ & w = 1
\end{align*}
\]
Setting $w = 0$ and $x = 1$ makes $(w \lor \overline{x})$ unsatisfied.
Setting $w = 0$ and $x = 0$ lets us simplify the clauses:

$$(y \lor z), (\overline{y}), (y \lor \overline{z})$$
Similarly, with $w = 0$ and $x = 1$, we get:

\[
(), (y \lor \bar{z})
\]

The empty clause indicates unsatisfiability
\[(w \lor x \lor y \lor z), (w \lor \overline{x}), (x \lor \overline{y}), (y \lor \overline{z}), (z \lor \overline{w}), (\overline{w} \lor \overline{z})\]

\[w = 0\]

\[(x \lor y \lor z), (\overline{x}), (x \lor \overline{y}), (y \lor \overline{z})\]

\[x = 0\]

\[(y \lor z), (\overline{y}), (y \lor \overline{z})\]

\[y = 0\]

\[(z), (\overline{z})\]

\[z = 0\]

\[()\]

\[z = 1\]

\[()\]

\[x = 1\]

\[(y \lor \overline{z})\]

\[y = 1\]

\[(z), (\overline{z})\]

\[z = 1\]

\[(x \lor y), ()\]

\[x = 1\]

\[(x \lor y), (y), ()\]
Backtracking requires a test procedure that returns:

- fail
- success
- uncertain
Start with some problem $P_0$
Let $S = \{P_0\}$, the set of active subproblems
Repeat while $S$ is nonempty:
  - choose a subproblem $P \in S$ and remove it from $S$
  - expand it into smaller subproblems $P_1, P_2, ..., P_k$
For each $P_i$:
  - If $\text{test}(P_i)$ succeeds: halt and announce this solution
  - If $\text{test}(P_i)$ fails: discard $P_i$
  - Otherwise: add $P_i$ to $S$
Announce that there is no solution
For optimization problems (say, minimization), we can use another technique
BRANCH-AND-BOUND

Start with some problem $P_0$
Let $S = \{P_0\}$, the set of active subproblems
bestsofar = $\infty$
Repeat while $S$ is nonempty:
  choose a subproblem (partial solution) $P \in S$ and remove it from $S$
  expand it into smaller subproblems $P_1, P_2, \ldots, P_k$
  For each $P_i$:
    If $P_i$ is a complete solution: update bestsofar
    else if lowerbound($P_i$) < bestsofar: add $P_i$ to $S$
return bestsofar
A partial tour goes from $a$ to $b$ and passes through nodes $S$

Denote this as $[a, S, b]$
Then we start at $[a, \{a\}, a]$
From a partial solution \([a, S, b]\), we can add edge \((b, x)\) to get \([a, S \cup \{x\}, x]\)
If we are at \([a, S, b]\), can we find a lower bound on the tour cost?

Cost must be at least the sum of:

- lightest edge from \(a\) to \(V - S\)
- lightest edge from \(b\) to \(V - S\)
- minimum spanning tree of \(V - S\)
APPROXIMATION ALGORITHMS

- $\text{OPT}(I)$: optimal value
- $A(I)$: our result
APPROXIMATION RATIO

\[ \alpha_A = \max_I \frac{A(I)}{\text{OPT}(I)} \]
VERTEX COVER
VERTEX COVER

Recall that a greedy approach gives us a $O(\log n)$ approximation
matching: subset of edges with no vertices in common
Any matching is a lower bound for vertex cover

$$|M| \leq \text{OPT}(I)$$
This approach produces a cover $S$ with $2|\mathcal{M}|$ vertices:

$$\alpha_A \leq 2$$

But we can find cases where this is exact:

$$\alpha_A = 2$$