Coping With NP-Completeness

Special Cases

Average Case

Approximation Algorithms

Intelligent Brute Force

Heuristics

Special Cases

2-CNF Sat:

Use resolution:

$(x \vee y \vee z) \wedge (\bar{z} \vee w \vee \bar{v})$

resolve to

$(x \vee y \vee w \vee \bar{v})$

Sat iff $\square$ (the empty clause)

<u>cannot</u> be obtained by resolution

Eliminate one variable at a time by resolving it in all possible ways

For 3-Sat, or general sat, clauses can get arbitrarily long: 3^n

2-sat

$(x \vee \bar{z}) \wedge (z \vee y)$ gives $(x \vee y)$

Resolution preserves 2-sat

$O(n^2)$ possible clauses

$\therefore O(n^3)$ time

Like transitive closure,

all-pairs shortest paths

Faster: Formula → Graph

$(x \vee y) \Rightarrow \bar{x} \to y$ (edges)

$x \leftarrow \bar{y}$

Literals are vertices

Satisfiable iff no literal and its negation are in the same strong component. (Why?)

$O(m+n)$ where m = # clauses

n = # literals

(Can propagate single-literal clauses first, or use $x \equiv x \vee x \Rightarrow \bar{x} \to x$)

Special cases

Min vertex cover for a bipartite graph

Find a maximum matching.

Search for augmenting paths from free vertices on A side. Let reached vertices be S, others $\bar{S}$.

Let $C = (B \cap S) \cup (A \cap \bar{S})$

This is a minimum vertex cover.

$B \cap S$: all matched in S $\quad\quad$ $\bar{S}$

S $\quad\quad$ $A \cap \bar{S}$:

all matched in $\bar{S}$

No matched $B \cap S$ to $A \cap \bar{S}$ edges

No unmatched $A \cap S$ to $B \cap \bar{S}$ edges

$\Rightarrow (B \cap S) \cup (A \cap \bar{S})$ is a vertex cover

of size = maximum matching

$\Rightarrow$ minimum (every edge of a matching

must be covered)

Approximation

General vertex cover

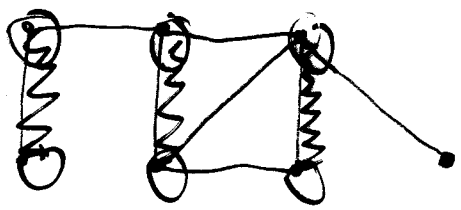
Find a maximal matching (no new edges can be added)

C = both ends of all matched edges

covers since maximal matching

2-approximation

O(m)-time



Approximation

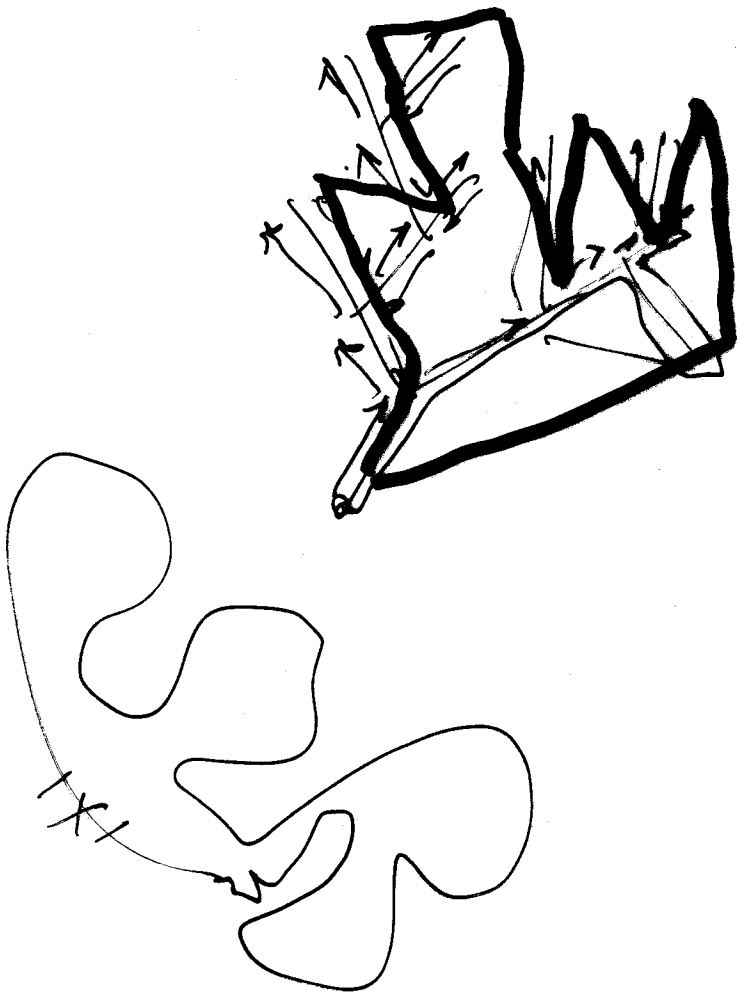
Minimum tour (TSP) with Δ inequality:

$$d(x,y) + d(y,z) \geq d(x,z)$$

$\Rightarrow$ given any tour with repeats, can find a tour no longer by dropping repeats

Find a minimum spanning tree, build a tour as a depth-first traversal (each edge used twice), delete repeated vertices.

2-approximation



1 + tree

+ tree +

edge

1.5 approximation

Find an MST T

odd-degree vertices is even

Find a min-cost perfect matching on odd degree vertices P

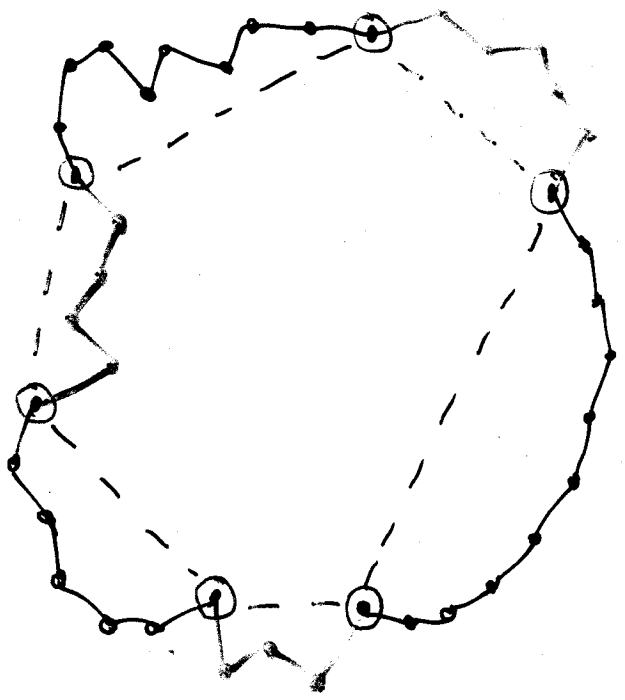
$T \cup P$ has all vertices of even degree:

Find an Eulerian tour, delete repeated vertices.

If R is a min-cost tour $|T| \leq |R|$,

$|P| \leq |R|/2 \Rightarrow |T+P| \leq 1.5|R|$

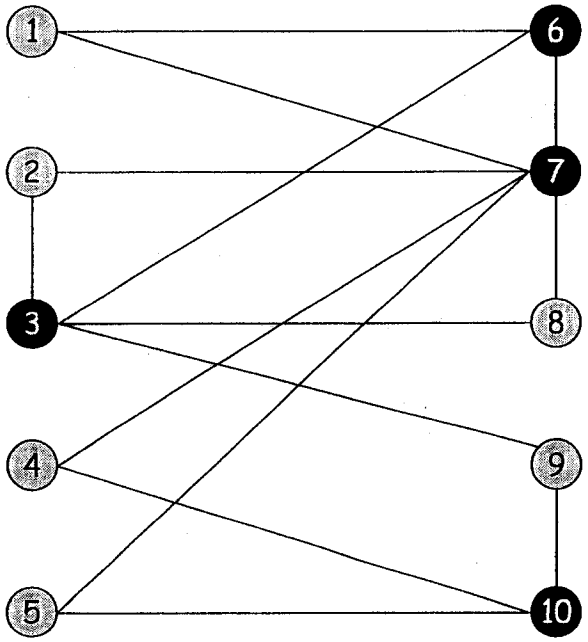
($|\ |$ denotes cost)



R decomposes into two ways of pairing odd-degree vertices, gives to matchings of odd-degree vertices.

Vertex Cover

VERTEX COVER: Given a graph $G = (V, E)$ and an integer k, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge (u, v) either $u \in S$, or $v \in S$, or both.



$k = 4$
$S = \{ 3, 6, 7, 10 \}$

Finding Small Vertex Covers

Q. What if k is small?

Brute force. $O(k\,n^{k+1})$.

- Try all $C(n, k) = O(n^k)$ subsets of size k.
- Takes $O(k\,n)$ time to check whether a subset is a vertex cover.

Goal. Limit exponential dependency on k, e.g., to $O(2^k\,k\,n)$.

Ex. $n = 1{,}000$, $k = 10$.
Brute. $k\,n^{k+1} = 10^{34} \Rightarrow$ infeasible.
Better. $2^k\,k\,n = 10^7 \Rightarrow$ feasible.

Remark. If k is a constant, algorithm is poly-time; if k is a small constant, then it's also practical.

Finding Small Vertex Covers

Claim. Let u-v be an edge of G. G has a vertex cover of size $\leq k$ iff at least one of $G - \{u\}$ and $G - \{v\}$ has a vertex cover of size $\leq k-1$.

delete v and all incident edges

Pf. $\Rightarrow$

- Suppose G has a vertex cover S of size $\leq k$.
- S contains either u or v (or both). Assume it contains u.
- $S - \{u\}$ is a vertex cover of $G - \{u\}$.

Pf. $\Leftarrow$

- Suppose S is a vertex cover of $G - \{u\}$ of size $\leq k-1$.
- Then $S \cup \{u\}$ is a vertex cover of G. ▪

Claim. If G has a vertex cover of size k, it has $\leq k(n-1)$ edges.

Pf. Each vertex covers at most n-1 edges. ▪

Finding Small Vertex Covers: Algorithm

Claim. The following algorithm determines if G has a vertex cover of size $\leq k$ in $O(2^k kn)$ time.

```
boolean Vertex-Cover(G, k) {
   if (G contains no edges)   return true
   if (G contains ≥ kn edges) return false

   let (u, v) be any edge of G
   a = Vertex-Cover(G - {u}, k-1)
   b = Vertex-Cover(G - {v}, k-1)
   return a or b
}
```

Pf.

- Correctness follows previous two claims.
- There are $\leq 2^{k+1}$ nodes in the recursion tree; each invocation takes $O(kn)$ time. ▪

Finding Small Vertex Covers: Recursion Tree

$$T(n,k) \le \begin{cases} cn & \text{if } k = 1 \\ 2T(n,k-1) + ckn & \text{if } k > 1 \end{cases} \Rightarrow T(n,k) \le 2^k ckn$$