Dealing with NP-Completeness

Note: We will resume talking about optimization problems, rather than yes/no questions.

What to do?

- Give up
- Solve small instances
- Look for special structure that makes your problem easy (e.g. planar graphs, each variable in at most 2 clauses, ...)
- Run an exponential time algorithm that might do well on some instances (e.g. branch-and-bound, integer programming, constraint programming)
- Heuristics – algorithms that run for a limited amount of time and return a solution that is hopefully close to optimal, but with no guarantees
- Approximation Algorithms – algorithms that run in polynomial time and give a guarantee on the quality of the solution returned
Heuristics

• Simple algorithms like “add max degree vertex to the vertex cover”
• Metaheuristics are popular
  – Greedy
  – Local search
  – tabu search
  – simulated annealing
  – genetic algorithms
Approximation Algorithms

Set up: We have a minimization problem \( X \), inputs \( I \), algorithm \( A \).

- \( OPT(I) \) is the value of the optimal solution on input \( I \).
- \( A(I) \) is the value returned when running algorithm \( A \) on input \( I \).

Def: Algorithm \( A \) is an \( \rho \)-approximation algorithm for Problem \( X \) if, for all inputs \( I \)

- \( A \) runs in polynomial time
- \( A(I) \leq \rho OPT(I) \).

Note: \( \rho \geq 1 \), small \( \rho \) is good.
A 2-approximation for Vertex Cover

Problem Definition:

- Put a subset of vertices into vertex cover $VC$.
- Requirement: For every edge $(u,v)$, either $u$ or $v$ (or both) is in $VC$.
- Goal: minimize number of vertices in $VC$.

Basic Approach: while some edge is still not covered

- Pick an arbitrary uncovered edge $(u,v)$.
- Add either $u$ or $v$ to the vertex cover.
- We have to make a choice: do we add $u$ or $v$? It matters a lot!
- Solution: cover both.

The Algorithm: While there exists an uncovered edge:

1. pick an arbitrary uncovered edge $(u,v)$.
2. add both $u$ and $v$ to the vertex cover $VC$. 
Analysis

The Algorithm: While there exists an uncovered edge:
1. Pick an arbitrary uncovered edge \((u,v)\).
2. Add both \(u\) and \(v\) to the vertex cover \(VC\).

\(VC\) is a vertex cover: the algorithm only terminates when all edges are covered

Solution value:
- Let \((u_1,v_1),(u_2,v_2),\ldots,(u_k,v_k)\) be edges picked in step 1 of the algorithm
- \(|VC| = 2k\)

Claim: \(OPT \geq k\)
- The edges \((u_1,v_1),(u_2,v_2),\ldots,(u_k,v_k)\) are disjoint.
- For each edge \((u_i,v_i)\), any vertex cover must contain \(u_i\) or \(v_i\).

Conclusion: \(k \leq OPT \leq |VC| \leq 2k\)
In other words: \(OPT \leq |VC| \leq 2OPT\).

We have a 2-approximation algorithm.
Methodology

Lower bound: Given an instance $I$, a lower bound, $LB(I)$ is an “easily-computed” value such that $LB(I) \leq OPT(I)$.

Methodology

- Compute a lower bound $LB(I)$.
- Give an algorithm $A$, that computes a solution to the optimization problem on input $I$ with a guarantee that $A(I) \leq \rho LB(I)$ for some $\rho \geq 1$.
- Conclude that $A(I) \leq \rho OPT(I)$.
Euler Tour

• Give an even-degree graph $G$, an Euler Tour is a (non-simple) cycle that visits each edge exactly once.

• Every even-edge graph has an Euler tour.

• You can find one in linear time.
Travelling Salesman Problem

Variant: We will consider the symmetric TSP with triangle-inequality.

- Complete graph where each edge \((a, b)\) has non-negative weight \(w(a, b)\)

- **Symmetric**: \(w(a, b) = w(b,a)\)

- **Triangle Inequality**: \(w(a,b) \leq w(a,c) + w(c,b)\)

- **Objective**: find cycle \(v_1, v_2, ..., v_n, v_1\) that goes through all vertices and has minimum weight.

Notes:

- Without triangle inequality, you cannot approximate TSP (unless P=NP)
- Asymmetric version is harder to approximate.
Approximating TSP

Find a convenient lower bound: minimum spanning tree!

\( \text{MST}(I) \leq \text{OPT}(I) \)

- A minimum spanning tree doubled is an even degree graph \( GG \), and therefore has an Euler tour of total length \( GG(I) \), with \( GG(I) = 2\text{MST}(I) \).
- Because of triangle inequality, we can “shortcut” the Euler tour \( GG \) to find a tour with \( \text{TSP}(I) \leq GG(I) \)

Combining, we have

\( \text{MST}(I) \leq \text{OPT}(I) \leq \text{TSP}(I) \leq GG(I) = 2\text{MST}(I) \)

- 2-approximation for TSP
- 3/2-approximation is possible.
- If points are in the plane, there exists a polynomial time approximation scheme, an algorithm that, for any fixed \( \epsilon > 0 \) returns a tour of length at most \( (1 + \epsilon)\text{OPT}(I) \) in polynomial time. (The dependence on \( \epsilon \) can be large).
**MAX-3-SAT**

**Definition**  Given a boolean CNF formula with 3 literals per clause. We want to satisfy the maximum possible number of clauses.

**Note:** We have to invert definition of approximation, want to find \( \rho A(I) \geq OPT(I) \)

**Algorithm**
- Randomly set each variable to true with probability \( \frac{1}{2} \).
Analysis

Find an upper bound: \( OPT(I) \leq m \) (duh)

Algorithm:

- Let \( Y \) be the number of clauses satisfied.
- Let \( m \) be the number of clauses. (\( m \geq OPT(I) \)).
- Let \( Y_i \) be the i.r.v representing the \( i \) th clause being satisfied.
- \( Y = \sum_{i=1}^{m} Y_i \).
- \( E[Y] = \sum_{i=1}^{m} E[Y_i] \).

- What is \( E[Y_i] \), the probability that the \( i \) th clause is true?
  - The only way for a clause to be false is for all three literals to be false.
  - The probability a clause is false is therefore \( (1/2)^3 = 1/8 \).
  - Probability a clause is true is therefore \( 1 - 1/8 = 7/8 \).

- Finishing, \( E[Y_i] = 7/8 \).
- \( E[Y] = (7/8)m \)
- \( E[Y] = (7/8)m \geq (7/8)OPT(I) \)

Conclusion 7/8 -approximation algorithm.
Approximation Lower Bounds:

**Standard NP-completeness:** Assuming $P \neq NP$, there is no polynomial time algorithm for max 3-sat

**Can Prove:** Assuming $P \neq NP$, there is no polynomial time algorithm that achieves a $7/8 + .00001$ approximation to max 3-sat.

Simple algorithm is sometimes the best one:

- **Max 3-sat:** 7/8-approximation algorithm is optimal
- **Vertex Cover:** 2-approximation algorithm is optimal assuming popular conjecture (unique games conjecture).

**Note:** Not all approximation algorithms are simple!

**Note:** Sometimes NO constant approximation is possible.

**Note:** For many problems, do not have matching upper and lower bounds on approximation ratio.
Proving an Approximation Lower Bound

Example: TSP without triangle inequality not possible to approximate.

Claim: There is no 10-approximation for TSP (assuming $P \neq NP$).

- Reduction from Hamiltonian Cycle.
- Let $G$ be a graph with $n$ vertices.
- Will Show: poly-time algorithm for 10-approximation to TSP implies poly-time algorithm to determine if $G$ has a Hamiltonian cycle.
- Reduction: Form a complete graph $G'$ where $w(u, v) = 1$ if $(u, v) \in G$ and $w(u, v) = 20n$ otherwise.
- Let $OPT(G')$ be minimum traveling salesman cost for $G'$.
- Claim: if $G$ has a Hamiltonian cycle then $OPT(G') = n$.
- Claim: if $G$ has no Hamiltonian cycle then $OPT(G') \geq 20n$.
- TSP approximation: Our TSP algorithm is a 10-approximation:
  \[ OPT(G') \leq TSP(G') \leq 10OPT(G') \]
- Reduction Complete: G has a Hamiltonian cycle if and only if $TSP(G') \leq 10n$
Reductions do Not Preserve Approximation

**Exact Algorithms:** A polynomial time algorithm for vertex cover implies a polynomial time algorithm for maximum clique.

**Approximation Algorithms:** A poly-time algorithm for 2-approximate vertex cover does NOT imply a poly-time algorithm for 2-approximate maximum clique.
Min Vertex Cover and Max Clique

**Def:** Let $G'$ be the complement graph of $G$: edges are replaced by non-edges.

**Review:** $\text{MaxClique}(G) = n - \text{MinVertexCover}(G')$

**Not Approximation Preserving:**

- Say we want an approximation to MaxClique($G$)
- Can we use our 2-approximation to MinVertexCover?
- Let $n = 1000$
- Compute a 2-approximation to MinVertexCover($G'$). Say we learn:
  
  $$450 \leq \text{MinVertexCover}(G') \leq 900$$

- **Conclusion:** $100 \leq \text{MaxClique}(G) \leq 550$
- **Quality:** Not a 2-approximation!

**Lower Bound:** There is no good approximation to maximum clique (assuming $P \neq NP$).
Set Cover

An instance \((X, \mathcal{F})\) of the set-covering problem consists of a finite set \(X\) and a family \(\mathcal{F}\) of subsets of \(X\), such that every element of \(X\) belongs to at least one subset in \(\mathcal{F}\):

\[
X = \bigcup_{S \in \mathcal{F}} S.
\]

We say that a subset \(S \in \mathcal{F}\) covers its elements. The problem is to find a minimum-size subset \(C \subseteq \mathcal{F}\) whose members cover all of \(X\):

\[
X = \bigcup_{S \in C} S.
\]
Greedy Algorithm

**Greedy-Set-Cover**($X, \mathcal{F})$

1. $U \leftarrow X$
2. $C \leftarrow \emptyset$
3. while $U \neq \emptyset$
   4. do select an $S \in \mathcal{F}$ that maximizes $|S \cap U|$
5. $U \leftarrow U - S$
6. $C \leftarrow C \cup \{S\}$
7. return $C$

**Claim:** If the optimal set cover has $k$ elements, then $C$ has at most $k \log n$ elements.
Claim: If the optimal set cover has $k$ sets, then $C$ has at most $k \log n$ sets.

Proof:

- Optimal set cover has $k$ sets.
- One of the sets must therefore cover at least $\frac{n}{k}$ of the elements.
- First greedy step must therefore choose a set that covers at least $\frac{n}{k}$ of the elements.
- After first greedy step, the number of uncovered elements is at most $n - \frac{n}{k} = n(1 - \frac{1}{k})$.
Iterate argument

- On remaining uncovered elements, one set in optimal must cover at least a \( \frac{1}{k} \) fraction of the remaining elements.

- So after two steps, the number of uncovered elements is at most

\[
 n \left(1 - \frac{1}{k}\right)^2
\]

So after \( j \) iterations, the number of uncovered elements is at most

\[
 n \left(1 - \frac{1}{k}\right)^j \leq ne^{-j/k}
\]

When \( j = k \ln n \), the number of uncovered elements is at most

\[
 ne^{-j/k} = ne^{-k\ln n/k} = ne^{-\ln n} = n/n = 1
\]

Therefore, the algorithm stops after choosing at most \( k \ln n \) sets (without knowing \( k \)).