**Tackling Intractable Problems**

- Many important problems are NP-complete or worse.
- **Heuristics** have been developed to attack them.
- They are **approximation algorithms**.
- How good are the approximations?
  - We are looking for theoretically *guaranteed* bounds, not “empirical” bounds.
- Are there NP problems that cannot be approximated well (assuming NP ≠ P)?
- Are there NP problems that cannot be approximated at all (assuming NP ≠ P)?

**Optimization Problem and Threshold Language**

- Given a maximization (minimization) problem, its decision version, the **threshold language**, asks if the optimal cost is at least (at most, resp.) a given threshold.
- If the decision version is hard, the optimization problem cannot be easy.
  - Otherwise, we can solve the optimization problem first and then do a simple test.
- If the optimization problem is hard, its decision version is not expected to be easy.
  - Otherwise, we can often do a binary search to bracket the optimal cost.

**Some Definitions**

- Given an **optimization problem**, each problem instance \( x \) has a set of **feasible solutions** \( F(x) \).
- Each feasible solution \( s \in F(x) \) has a cost \( c(s) \in \mathbb{Z}^+ \).
- The **optimum cost** is \( \text{opt}(x) = \min_{s \in F(x)} c(s) \) for a minimization problem.
- It is \( \text{opt}(x) = \max_{s \in F(x)} c(s) \) for a maximization problem.

**Approximation Algorithms**

- Let algorithm \( M \) on \( x \) returns a feasible solution.
- \( M \) is an **\( \epsilon \)-approximation algorithm**, where \( \epsilon \geq 0 \), if for all \( x \),
  \[
  \frac{|c(M(x)) - \text{opt}(x)|}{\text{opt}(x), c(M(x))} \leq \epsilon.
  \]
  - For a minimization problem,
    \[
    \frac{c(M(x)) - \min_{s \in F(x)} c(s)}{c(M(x))} \leq \epsilon.
    \]
  - For a maximization problem,
    \[
    \frac{\max_{s \in F(x)} c(s) - c(M(x))}{\max_{s \in F(x)} c(s)} \leq \epsilon.
    \]
Lower and Upper Bounds

- For a minimization problem,
  \[ \min_{s \in F(x)} c(s) \leq c(M(x)) \leq \frac{\min_{s \in F(x)} c(s)}{1 - \epsilon}. \]
- So approximation ratio \( \frac{\min_{s \in F(x)} c(s)}{c(M(x))} \geq 1 - \epsilon. \)

- For a maximization problem,
  \[ (1 - \epsilon) \times \max_{s \in F(x)} c(s) \leq c(M(x)) \leq \max_{s \in F(x)} c(s). \]
- So approximation ratio \( \frac{c(M(x))}{\max_{s \in F(x)} c(s)} \geq 1 - \epsilon. \)
- The above are alternative definitions of \( \epsilon \)-approximation algorithms.

Approximation Thresholds

- The approximation threshold is the greatest lower bound of all \( \epsilon \geq 0 \) such that there is a polynomial-time \( \epsilon \)-approximation algorithm.
- The approximation threshold of an optimization problem can be anywhere between 0 (approximation to any desired degree) and 1 (no approximation is possible).
- If \( P = NP \), then all optimization problems in \( NP \) have approximation threshold 0.
- So we assume \( P \neq NP \) for the rest of the discussion.

Range Bounds

- \( \epsilon \) takes values between 0 and 1.
- For maximization problems, an \( \epsilon \)-approximation algorithm returns solutions within \( [1 - \epsilon] \times \text{OPT}, \text{OPT} \].
- For minimization problems, an \( \epsilon \)-approximation algorithm returns solutions within \( [\text{OPT}, \frac{1}{1-\epsilon}] \).
- For each NP-complete optimization problem, we shall be interested in determining the smallest \( \epsilon \) for which there is a polynomial-time \( \epsilon \)-approximation algorithm.
- Sometimes \( \epsilon \) has no minimum value.

NODE COVER

- NODE COVER seeks the smallest \( C \subseteq V \) in graph \( G = (V, E) \) such that for each edge in \( E \), at least one of its endpoints is in \( C \).
- A heuristic to obtain a good node cover is to iteratively move a node with the highest degree to the cover.
- This turns out to produce approximation ratio \( \frac{c(M(x))}{\text{OPT}(x)} = \Theta(\log n) \).
- It is not an \( \epsilon \)-approximation algorithm for any \( \epsilon < 1 \).
A 0.5-Approximation Algorithm
1: $C := \emptyset$;
2: while $E \neq \emptyset$ do
3: Delete an arbitrary edge $[u, v]$ from $E$;
4: Delete edges incident with $u$ and $v$ from $E$;
5: Add $u$ and $v$ to $C$; {Add 2 nodes to $C$ each time.}
6: end while
7: return $C$;

Analysis
- $C$ contains $|C|/2$ edges.
- No two edges of $C$ share a node.
- Any node cover must contain at least one node from each of these edges.
- This means that $\text{opt}(G) \geq |C|/2$.
- So $\frac{\text{opt}(G)}{|C|} \geq 1/2$.
- The approximation threshold is $\leq 0.5$.

Maximum Satisfiability
- Given a set of clauses, MaxSAT seeks the truth assignment that satisfies the most.
- Max2SAT is already NP-complete (p. 263).
- Consider the more general $k$-MaxGSAT for constant $k$.
  - Given a set of boolean expressions $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$ in $n$ variables.
  - Each $\phi_i$ is a general expression involving $k$ variables.
  - $k$-MaxGSAT seeks the truth assignment that satisfies the most expressions.
A Probabilistic Interpretation of an Algorithm

- Each $\phi_i$ involves exactly $k$ variables and is satisfied by $t_i$ of the $2^k$ truth assignments.
- A random truth assignment $\in \{0,1\}^n$ satisfies $\phi_i$ with probability $p(\phi_i) = t_i/2^k$.
- Hence a random truth assignment satisfies an expected number
  
  $$p(\Phi) = \sum_{i=1}^{m} p(\phi_i)$$

  of expressions $\phi_i$.

The Search Procedure (concluded)

- By our hill-climbing procedure,
  
  $$p(\Phi[x_1 = t_1, x_2 = t_2, \ldots, x_n = t_n])$$
  
  $$\geq \cdots$$
  
  $$\geq p(\Phi[x_1 = t_1, x_2 = t_2])$$
  
  $$\geq p(\Phi[x_1 = t_1])$$
  
  $$\geq p(\Phi).$$

- So at least $p(\Phi)$ expressions are satisfied by truth assignment $(t_1, t_2, \ldots, t_n)$.
- The algorithm is deterministic.

The Search Procedure

- Clearly
  
  $$p(\Phi) = \frac{1}{2} \{ p(\Phi[x_1 = \text{true}]) + p(\Phi[x_1 = \text{false}] \}.$$  

- Select the $t_i \in \{\text{true, false}\}$ such that $p(\Phi[x_1 = t_1])$ is the larger one.

- Note that $p(\Phi[x_1 = t_1]) \geq p(\Phi)$.

- Repeat with expression $\Phi[x_1 = t_1]$ until all variables $x_i$ have been given truth values $t_i$ and all $\phi_i$ either true or false.

Approximation Analysis

- The optimum is at most the number of satisfiable $\phi_i$, i.e., those with $p(\phi_i) > 0$.

- Hence the ratio of algorithm’s output vs. the optimum is
  
  $$\frac{p(\Phi)}{\sum_{p(\phi_i) > 0} 1} \geq \frac{\sum_i p(\phi_i) \geq 0 p(\phi_i)}{\sum_{p(\phi_i) > 0} 1} \geq \min_{p(\phi_i) > 0} p(\phi_i).$$

- The heuristic is a polynomial-time $\epsilon$-approximation algorithm with $\epsilon = 1 \cdot \min_{p(\phi_i) > 0} p(\phi_i)$.

- Because $p(\phi_i) \geq 2^{-k}$, the heuristic is a polynomial-time $\epsilon$-approximation algorithm with $\epsilon = 1 - 2^{-k}$.
Back to MAXSAT

- In MAXSAT, the $\phi_i$'s are clauses.
- Hence $p(\phi_i) \geq 1/2$, which happens when $\phi_i$ contains a single literal.
- And the heuristic becomes a polynomial-time $\epsilon$-approximation algorithm with $\epsilon = 1/2$.\(^a\)
- If the clauses have $k$ distinct literals, $p(\phi_i) = 1 - 2^{-k}$.
- And the heuristic becomes a polynomial-time $\epsilon$-approximation algorithm with $\epsilon = 2^{-k}$.
  - This is the best possible for $k \geq 3$ unless $P = NP$.

\(^a\)Johnson (1974).

A 0.5-Approximation Algorithm for MAX CUT

1: $S := \emptyset$
2: while $\exists v \in V$ whose switching sides results in a larger cut do
3: $S := S \cup \{v\}$
4: end while
5: return $S$

- A 0.12-approximation algorithm exists.\(^a\)
- 0.059-approximation algorithms do not exist unless NP = ZPP.

\(^a\)Goemans and Williamson (1995).

MAX CUT Revisited

- The NP-complete MAX CUT seeks to partition the nodes of graph $G = (V, E)$ into $(S, V - S)$ so that there are as many edges as possible between $S$ and $V - S$ (p. 284).
- Local search starts from a feasible solution and performs “local” improvements until none are possible.

Analysis

- Optimal cut
- Heuristic cut
Analysis (continued)

- Partition $V = V_1 \cup V_2 \cup V_3 \cup V_4$, where our algorithm returns $(V_1 \cup V_2, V_3 \cup V_4)$ and the optimum cut is $(V_1 \cup V_3, V_2 \cup V_4)$.
- Let $e_{ij}$ be the number of edges between $V_i$ and $V_j$.
- Because no migration of nodes can improve the algorithm's cut, for each node in $V_1$, its edges to $V_1 \cup V_2$ are outnumbered by those to $V_3 \cup V_4$.
- Considering all nodes in $V_1$ together, we have $2e_{11} + e_{12} \leq e_{13} + e_{14}$, which implies
  $$e_{12} \leq e_{13} + e_{14}.$$ 

Approximability, Unapproximability, and Between

- **KNAPSACK**, **NODE COVER**, **MAXSAT**, and **MAX CUT** have approximation thresholds less than 1.
  - **KNAPSACK** has a threshold of 0.
  - But **NODE COVER** and **MAXSAT** have a threshold larger than 0.
- The situation is maximally pessimistic for **TSP**: It cannot be approximated unless $P = NP$.
  - The approximation threshold of **TSP** is 1.
  - The threshold is $1/3$ if the **TSP** satisfies the triangular inequality.
  - The same holds for **INDEPENDENT SET**.

Analysis (concluded)

- Similarly,
  $$e_{12} \leq e_{23} + e_{24}$$
  $$e_{34} \leq e_{23} + e_{13}$$
  $$e_{34} \leq e_{14} + e_{24}$$
- Adding all four inequalities, dividing both sides by 2 and adding the inequality $e_{14} + e_{23} \leq e_{14} + e_{23} + e_{13} + e_{24}$, we obtain
  $$e_{12} + e_{34} + e_{14} + e_{23} \leq 2(e_{13} + e_{14} + e_{23} + e_{24}).$$
- The above says our solution is at least half the optimum.

Unapproximability of TSP\(^a\)

**Theorem 72** The approximation threshold of **TSP** is 1 unless $P = NP$.

- Suppose there is a polynomial-time $\epsilon$-approximation algorithm for **TSP** for some $\epsilon < 1$.
- We shall construct a polynomial-time algorithm for the NP-complete **HAMILTONIAN CYCLE**.
- Given any graph $G = (V, E)$, construct a **TSP** with $|V|$ cities with distances
  $$d_{ij} = \begin{cases} 
  1, & \text{if } \{i, j\} \in E \\
  \left\lceil \frac{|V|}{2} \right\rceil, & \text{otherwise}
  \end{cases}$$

\(^a\)Sahni and Gonzalez (1976).
The Proof (concluded)
- Run the alleged approximation algorithm on this instance.
- Suppose a tour of cost $|V|$ is returned.
  - This tour must be a Hamiltonian cycle.
- Suppose a tour with at least one edge of length $\frac{|V|}{\varepsilon}$ is returned.
  - The total length of this tour is $> \frac{|V|}{\varepsilon}$.
  - Because the algorithm is $\varepsilon$-approximate, the optimum is at least $1 - \varepsilon$ times the returned tour’s length.
- The optimum tour has a cost exceeding $|V|$.
- Hence $G$ has no Hamiltonian cycles.

KNAPSACK Has an Approximation Threshold of Zero\(^a\)

**Theorem 73** For any $\varepsilon$, there is a polynomial-time $\varepsilon$-approximation algorithm for KNAPSACK.
- We have $n$ weights $w_1, w_2, \ldots, w_n$, a weight limit $W$, and $n$ values $v_1, v_2, \ldots, v_n$.
- We must find an $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i$ is the largest possible.
- Let $V = \max\{v_1, v_2, \ldots, v_n\}$.

\(^a\)Barva and Kim (1975).

The Proof (continued)
- For $0 \leq i \leq n$ and $0 \leq v \leq nV$, define $W(i, v)$ to be the minimum weight attainable by selecting some among the $i$ first items, so that their value is exactly $v$.
- Start with $W(0, v) = \infty$ for all $v$.
- Then
  \[ W(i + 1, v) = \min\{W(i, v), W(i, v - v_i + 1) + w_i + 1\} \]
- Finally, pick the largest $v$ such that $W(n, v) \leq W$.
- The running time is $O(n^2V)$, not polynomial time.
- Key idea: Limit the number of precision bits.
The Proof (concluded)

• Hence

\[ \sum_{i \in S'} v_i \geq \sum_{i \in S} v_i \quad n2^b. \]

• Because \( V \) is a lower bound on \( \omega^* \) (if, without loss of

  generality, \( w_i \leq W \)), the relative deviation from the

  optimum is at most \( n2^b/V \).

• By truncating the last \( b = \lceil \log_2 \frac{V}{n} \rceil \) bits of the values,

  the algorithm becomes \( \epsilon \)-approximate.

• The running time is then \( O(n^2V/b) = O(n^3/\epsilon) \), a

  polynomial in \( n \) and \( \epsilon \).