Phase Transitions

- P or NP classification based entirely on worst-case analysis
- Problem banished from P if one instance requires exponential solution time
- Overconstrained and underconstrained are easy
- Really hard problems occur on boundary between these regions
- Probability of a Hamiltonian Circuit as average connectivity varies
- Almost fully connected almost always has HC
- With connectivity of just over 2, almost never has HC

Graph Coloring

4 colors, connectivity increases
3 colors, connectivity increases

Traveling Salesman

Approximation Algorithms

- NP-Complete problems require exponential running times to find optimal solutions
- If the problem instance is small, then you can wait
- If not, a near-optimal solution in polynomial-time may be acceptable

---

Ratio Bound

How does the approximate solution compare to the optimal solution?

An approximation algorithm has a _______________ p(n) if for any input of size n, the cost C of the approximate solution is within a factor p(n) of the cost C* of the optimal solution:

$$1 \leq p(n) \leq \max(\frac{C}{C*}, \frac{C*}{C})$$

where C/C* is used for minimization problems, and C*/C for maximization problems.

Alternatively, an approximation algorithm has a ___________________________

$$\frac{\mid C \; - \; C* \mid}{C*} \; \leq \; \epsilon(n)$$

where ϵ(n) $\leq$ p(n) - 1.

Approximation Scheme

An approximation ________ is an approximation algorithm that takes as input an instance of the problem and a value $\epsilon > 0$, such that the algorithm has a relative error bound ϵ.

The approximation scheme is ____________ if it runs in time polynomial in the size n of the input.

The approximation scheme is ____________________ if it runs in time polynomial in $1/\epsilon$ and n.

$$T(n) = f(1/\epsilon, n)$$

Vertex-Cover Problem

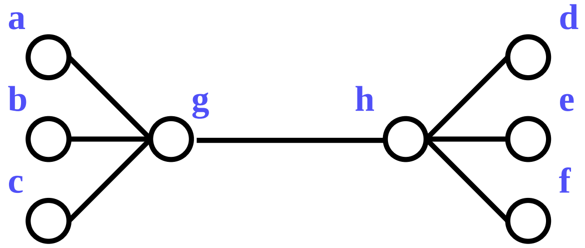
Approximation algorithm:

Keep grabbing edges whose vertices are not already in the cover

```
Approx-Vertex-Cover(G)          ; G = (V, E)
   C = {}
   E' = E
   while E' ≠ {}
      (u,v) = arbitrary edge from E'
      C = C ∪ {u,v}
      remove from E' every edge incident on u or v
   return C
```

This algorithm takes _____ time.

Example



Optimal = {g, h}

Approximate = {a, g, d, h}

The size of the approximate vertex cover is never more than twice the size of the optimal vertex cover.

Theorem 37.1

Approx-Vertex-Cover has a ratio bound of 2.

Proof:

The approximate solution C is a vertex cover.

Let A be the set of edges chosen by the algorithm. Since each such edge's endpoints were not in C at the time, |C| = 2|A|. An optimal cover must have at least |A| vertices, |C*| ≥ |A|. Thus |C*| ≥ 1/2|C| and $\frac{|C|}{|C*|}$ ≤ 2 = p.

Traveling Salesman Problem

Triangle Inequality

c(u,w) ≤ c(u,v) + c(v,w)

(generally satisfied)

An approximation algorithm with ratio bound p = 2 exists for TSPs exhibiting triangle inequality.

```
Approx-TSP-Tour(G, c)          ; G = (V, E), c = edge costs
    select root vertex r in V
    T = MST-Prim(G, c, r)
    L = list of vertices in preorder traversal of T
    return cycle with vertices ordered as in L
```

Theorem 37.2

For TSP with triangle inequality, Approx-TSP-Tour is an approximation algorithm with a ratio bound of 2.

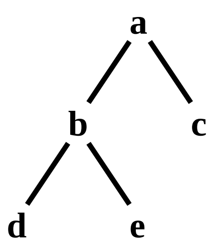
If H is approximate tour, $c(H) \leq 2c(H^*)$

Proof:

If H^* is the optimal tour and T is a MST(G), then $c(T) \leq c(H^*)$.

Consider a **full walk** W of a MST with cost c(W).

Example



$$W = a\ \underline{b}\ d\ \underline{b}\ e\ b\ a\ c\ a$$

$$c(W) = 2c(T) \longrightarrow c(W) \leq 2c(H^*)$$

W is not a tour, but by triangle inequality, we can change $w \rightarrow x \rightarrow w \rightarrow y$ to $w \rightarrow x \rightarrow y$, without increasing cost to yield approximate tour H.

$$c(H) \leq c(W) \leq 2c(H^*)$$

Theorem 37.3

If $P \neq NP$, there is no poly-time approximation algorithm with ratio bound $p \geq 1$ for the general TSP (i.e., no triangle inequality).

Proof:

If such an algorithm A exists, then we can use A to solve the Hamiltonian Cycle problem, which is NP-Complete, in polynomial time.

From graph G for Hamiltonian Cycle problem construct complete graph G' = (V,E'), where edges appearing in G have cost 1, and remaining edges have cost p$|V|$ + 1.

If HC in G, then there is a tour of cost $|V|$ in G', and A must return it to satisfy its ratio bound of p. If no HC in G, the TSP tour costs at least

$$(p|V| + 1) + (|V| - 1) > p|V|.$$

Thus, can determine if HC in G based on whether TSP tour cost is $|V|$. But, unless P = NP, such an algorithm cannot exist, because it solves an NP-complete problem in polynomial time.

Set-Covering Problem

Algorithm:

- Greedy approach
- Grab the set covering the largest number of uncovered members

```
Greedy-Set-Cover(X, F)
    U = X       ; uncovered
    C = {}
    while U ≠ {}
        select S ∈ F maximizing | S ∩ U |
        U = U - S
        C = C ∪ {S}
    return C
```

Corollary 37.5

Greedy-Set-Cover has a ratio bound of (ln|x| + 1).
Proof in book.

Subset-Sum Problem

Algorithm:

- Variation
- Return largest sum $\leq$ t of elements in S

```
Exact-Subset-Sum(S, t)          ; S = {x_1, .., x_n}
   n = |S|
   L_0 = ⟨0⟩
   for i = 1 to n
      L_i = Merge-Lists(L_{i-1}, L_{i-1} + x_i)
      remove from L_i elements > t
   return largest element in L_n
```

Example

S = $\{1, 2, 3\}$
$L_0 = \langle 0 \rangle$
$L_1 = \langle 0, 1 \rangle$
$L_2 = \langle 0, 1, 2, 3 \rangle$
$L_3 = \langle 0, 1, 2, 3, 4, 5, 6 \rangle$

Analysis

- L could double in size after each iteration

- Final merge could take 2^n steps
- Exponential running time

Fully Poly-Time Approximation Scheme

- Returned value is largest < t to within some percentage error
- Let ϵ = error bound, $0 < \epsilon < 1$
- Trim L of all values whose relative error is no more then δ away from a value in L

```
Trim(L, δ)          ; L = < y_1, .., y_m > sorted in non-decreasing order
    m = | L |
    L' = ⟨y_1⟩
    last = y_1
    for i = 2 to m
        if last < (1 - δ)y_i
        then append y_i on end of L'
            last = y_i
    return L'
```

```
Approx-Subset-Sum(S, t, ε)
    n = |S|
    L_0 = ⟨0⟩
    for i = 1 to n
        L_i = Merge-Lists(L_{i-1}, L_{i-1} + x_i)
        L_i = Trim(L_i, ε/n)
```

        remove from L_i elements $>$ t
    return largest value in L_n

Error passed to Trim is $\frac{\epsilon}{n}$ to prevent too much inaccuracy after repeated trimmings.

Theorem 37.6

Approx-Subset-Sum is a fully poly-time approximation scheme for the subset-sum problem.

Approximation Algorithms