Approximability, Unapproximability, and Between

- KNAPSACK, NODE COVER, MAXSAT, and MAX CUT have approximation thresholds less than 1.
  - KNAPSACK has a threshold of 0 (see p. 586).
  - 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 (see p. 584).
  - 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.

Unapproximability of TSP[a]

Theorem 75 *The approximation threshold of* TSP *is 1 unless P = NP.*

- Suppose there is a polynomial-time ϵ-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

$$d_{ij} = \begin{cases} 1, & \text{if } \{\, i, j \,\} \in E \\ \frac{|V|}{1-\epsilon}, & \text{otherwise} \end{cases}$$

[a]Sahni and Gonzales (1976).

The Proof (concluded)

- Run the alleged approximation algorithm on this TSP.
- 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 $\frac{|V|}{1-\epsilon}$ is returned.
  - The total length of this tour is $> \frac{|V|}{1-\epsilon}$.
  - Because the algorithm is ϵ-approximate, the optimum is at least $1-\epsilon$ 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 76 *For any ϵ, there is a polynomial-time ϵ-approximation algorithm for* KNAPSACK.

- We have n weights $w_1, w_2, \ldots, w_n \in \mathbb{Z}^+$, a weight limit W, and n values $v_1, v_2, \ldots, v_n \in \mathbb{Z}^+$.[b]
- 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]Ibarra and Kim (1975).

[b]If the values are fractional, the result is slightly messier but the main conclusion remains correct. Contributed by Mr. Jr-Ben Tian (R92922045) on December 29, 2004.

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 $O(n^2 V)$, not polynomial time.
- Key idea: Limit the number of precision bits.

The Proof (continued)

- Given the instance $x = (w_1, \ldots, w_n, W, v_1, \ldots, v_n)$, we define the approximate instance

$$x' = (w_1, \ldots, w_n, W, v'_1, \ldots, v'_n),$$

where

$$v'_i = 2^b \left\lfloor \frac{v_i}{2^b} \right\rfloor.$$

- Solving x' takes time $O(n^2 V / 2^b)$.
- The solution S' is close to the optimum solution S:

$$\sum_{i \in S} v_i \geq \sum_{i \in S'} v_i \geq \sum_{i \in S'} v'_i \geq \sum_{i \in S} v'_i \geq \sum_{i \in S} (v_i - 2^b) \geq \sum_{i \in S} v_i - n 2^b.$$

The Proof (concluded)

- Hence

$$\sum_{i \in S'} v_i \geq \sum_{i \in S} v_i - n 2^b.$$

- Because V is a lower bound on OPT (if, without loss of generality, $w_i \leq W$), the relative deviation from the optimum is at most $n 2^b / V$.
- By truncating the last $b = \lfloor \log_2 \frac{\epsilon V}{n} \rfloor$ bits of the values, the algorithm becomes ϵ-approximate.
- The running time is then $O(n^2 V / 2^b) = O(n^3 / \epsilon)$, a polynomial in n and $1/\epsilon$.

Pseudo-Polynomial-Time Algorithms

- Consider problems with inputs that consist of a collection of integer parameters (TSP, KNAPSACK, etc.).
- An algorithm for such a problem whose running time is a polynomial of the input length and the *value* (not length) of the largest integer parameter is a **pseudo-polynomial-time algorithm**.[a]
- On p. 587, we presented a pseudo-polynomial-time algorithm for KNAPSACK that runs in time $O(n^2 V)$.
- How about TSP (D), another NP-complete problem?

[a]Garey and Johnson (1978).

No Pseudo-Polynomial-Time Algorithms for TSP (D)

- By definition, a pseudo-polynomial-time algorithm becomes polynomial-time if each integer parameter is limited to having a *value* polynomial in the input length.
- Corollary 39 (p. 304) showed that HAMILTONIAN PATH is reducible to TSP (D) with weights 1 and 2.
- As HAMILTONIAN PATH is NP-complete, TSP (D) cannot have pseudo-polynomial-time algorithms unless P = NP.
- TSP (D) is said to be **strongly NP-hard**.
- Many weighted versions of NP-complete problems are strongly NP-hard.

Polynomial-Time Approximation Scheme

- Algorithm M is a **polynomial-time approximation scheme** (**PTAS**) for a problem if:
  - For each $\epsilon > 0$ and instance x of the problem, M runs in time polynomial (depending on ϵ) in $|x|$.
    - ∗ Think of ϵ as a constant.
  - M is an ϵ-approximation algorithm for every $\epsilon > 0$.

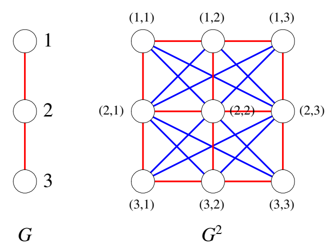
Fully Polynomial-Time Approximation Scheme

- A polynomial-time approximation scheme is **fully polynomial** (**FPTAS**) if the running time depends polynomially on $|x|$ and $1/\epsilon$.
  - Maybe the best result for a "hard" problem.
  - For instance, KNAPSACK is fully polynomial with a running time of $O(n^3/\epsilon)$ (p. 586).

Square of G

- Let $G = (V, E)$ be an undirected graph.
- G^2 has nodes $\{(v_1, v_2) : v_1, v_2 \in V\}$ and edges

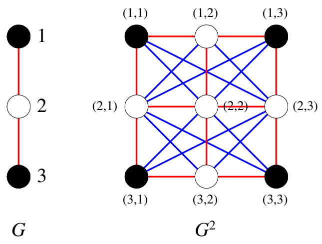
$$\{\{(u, u'), (v, v')\} : (u = v \wedge \{u', v'\} \in E) \vee \{u, v\} \in E\}.$$



Independent Sets of G and G^2

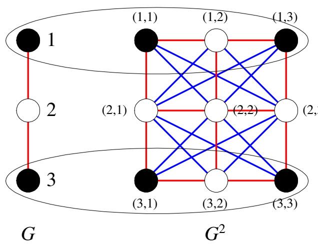
Lemma 77 *$G(V, E)$ has an independent set of size k if and only if G^2 has an independent set of size k^2.*

- Suppose G has an independent set $I \subseteq V$ of size k.
- $\{(u, v) : u, v \in I\}$ is an independent set of size k^2 of G^2.



The Proof (continued)

- Suppose G^2 has an independent set I^2 of size k^2.
- $U \equiv \{u : \exists v \in V\ (u, v) \in I^2\}$ is an independent set of G.



- $|\,U\,|$ is the number of "rows" that the nodes in I^2 occupy.

The Proof (concluded)[a]

- If $|\,U\,| \geq k$, then we are done.
- Now assume $|\,U\,| < k$.
- As the k^2 nodes in I^2 cover fewer than k "rows," there must be a "row" in possession of $> k$ nodes of I^2.
- Those $> k$ nodes will be independent in G as each "row" is a copy of G.

[a] Thanks to a lively class discussion on December 29, 2004.

Approximability of INDEPENDENT SET

- The approximation threshold of the maximum independent set is either zero or one (it is one!).

Theorem 78 *If there is a polynomial-time ϵ-approximation algorithm for* INDEPENDENT SET *for* any $0 < \epsilon < 1$, *then there is a polynomial-time approximation scheme.*

- Let G be a graph with a maximum independent set of size k.
- Suppose there is an $O(n^i)$-time ϵ-approximation algorithm for INDEPENDENT SET.

The Proof (continued)

- By Lemma 77 (p. 595), the maximum independent set of G^2 has size k^2.
- Apply the algorithm to G^2.
- The running time is $O(n^{2i})$.
- The resulting independent set has size $\geq (1-\epsilon)\, k^2$.
- By the construction in Lemma 77 (p. 595), we can obtain an independent set of size $\geq \sqrt{(1-\epsilon)\, k^2}$ for G.
- Hence there is a $(1-\sqrt{1-\epsilon})$-approximation algorithm for INDEPENDENT SET.

The Proof (concluded)

- In general, we can apply the algorithm to G^{2^ℓ} to obtain an $(1-(1-\epsilon)^{2^{-\ell}})$-approximation algorithm for INDEPENDENT SET.
- The running time is $n^{2^\ell i}$.[a]
- Now pick $\ell = \lceil \log \frac{\log(1-\epsilon)}{\log(1-\epsilon')} \rceil$.
- The running time becomes $n^{i \frac{\log(1-\epsilon)}{\log(1-\epsilon')}}$.
- It is an ϵ'-approximation algorithm for INDEPENDENT SET.

[a] It is not fully polynomial.

Comments

- INDEPENDENT SET and NODE COVER are reducible to each other (Corollary 37, p. 286).
- NODE COVER has an approximation threshold at most 0.5 (p. 569).
- But INDEPENDENT SET is unapproximable (see the textbook).
- INDEPENDENT SET limited to graphs with degree $\leq k$ is called k-DEGREE INDEPENDENT SET.
- k-DEGREE INDEPENDENT SET is approximable (see the textbook).

On P vs NP

Density[a]

The **density** of language $L \subseteq \Sigma^*$ is defined as

$$\mathrm{dens}_L(n) = |\{x \in L : |\,x\,| \leq n\}|.$$

- If $L = \{0,1\}^*$, then $\mathrm{dens}_L(n) = 2^{n+1} - 1$.
- So the density function grows at most exponentially.
- For a unary language $L \subseteq \{0\}^*$,

$$\mathrm{dens}_L(n) \leq n + 1.$$

  - Because $L \subseteq \{\epsilon, 0, 00, \ldots, \overbrace{00\cdots 0}^{n}, \ldots\}$.

[a] Berman and Hartmanis (1977).

Sparsity

- **Sparse languages** are languages with polynomially bounded density functions.
- **Dense languages** are languages with superpolynomial density functions.

Self-Reducibility for SAT

- An algorithm exploits **self-reducibility** if it reduces the problem to the same problem with a smaller size.
- Let ϕ be a boolean expression in n variables $x_1, x_2, \ldots, x_n$.
- $t \in \{0,1\}^j$ is a **partial** truth assignment for $x_1, x_2, \ldots, x_j$.
- $\phi[\,t\,]$ denotes the expression after substituting the truth values of t for $x_1, x_2, \ldots, x_t$ in ϕ.

An Algorithm for SAT with Self-Reduction

We call the algorithm below with empty t.

if $|\,t\,| = n$ **then**
    return $\phi[\,t\,]$;
else
    return $\phi[\,t0\,] \vee \phi[\,t1\,]$;
end if

The above algorithm runs in exponential time, by visiting all the partial assignments (or nodes on a depth-n binary tree).

NP-Completeness and Density[a]

Theorem 79 *If a unary language $U \subseteq \{0\}^*$ is NP-complete, then P = NP.*

- Suppose there is a reduction R from SAT to U.
- We shall use R to guide us in finding the truth assignment that satisfies a given boolean expression ϕ with n variables if it is satisfiable.
- Specifically, we use R to prune the exponential-time exhaustive search on p. 606.
- The trick is to keep the already discovered results $\phi[t]$ in a table H.

[a]Berman (1978).


```
if |t| = n then
  return φ[t];
else
  if (R(φ[t]), v) is in table H then
    return v;
  else
    if φ[t0] = "satisfiable" or φ[t1] = "satisfiable" then
      Insert (R(φ[t]), 1) into H;
      return "satisfiable";
    else
      Insert (R(φ[t]), 0) into H;
      return "unsatisfiable";
    end if
  end if
end if
```


The Proof (continued)

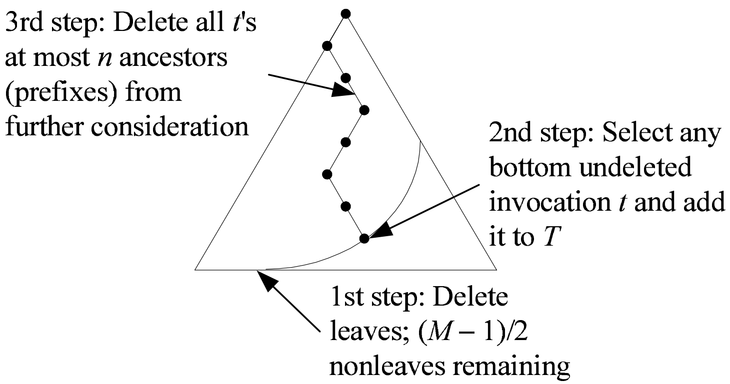
- Since R is a reduction, $R(\phi[t]) = R(\phi[t'])$ implies that $\phi[t]$ and $\phi[t']$ must be both satisfiable or unsatisfiable.
- $R(\phi[t])$ has polynomial length $\leq p(n)$ because R runs in log space.
- As R maps to unary numbers, there are only polynomially many $p(n)$ values of $R(\phi[t])$.
- How many nodes of the complete binary tree (of invocations/truth assignments) need to be visited?
- If that number is a polynomial, the overall algorithm runs in polynomial time and we are done.

The Proof (continued)

- A search of the table takes time $O(p(n))$ in the random access memory model.
- The running time is $O(Mp(n))$, where M is the total number of invocations of the algorithm.
- The invocations of the algorithm form a binary tree of depth at most n.

The Proof (continued)

- There is a set $T = \{t_1, t_2, \ldots\}$ of invocations (partial truth assignments, i.e.) such that:
  - $|T| \geq (M-1)/(2n)$.
  - All invocations in T are **recursive** (nonleaves).
  - None of the elements of T is a prefix of another.



The Proof (continued)

- All invocations $t \in T$ have different $R(\phi[\,t\,])$ values.
  - None of $s, t \in T$ is a prefix of another.
  - The invocation of one started after the invocation of the other had terminated.
  - If they had the same value, the one that was invoked second would have looked it up, and therefore would not be recursive, a contradiction.
- The existence of T implies that there are at least $(M-1)/(2n)$ different $R(\phi[\,t\,])$ values in the table.

The Proof (concluded)

- We already know that there are at most $p(n)$ such values.
- Hence $(M-1)/(2n) \leq p(n)$.
- Thus $M \leq 2np(n) + 1$.
- The running time is therefore $O(Mp(n)) = O(np^2(n))$.
- We comment that this theorem holds for any sparse language, not just unary ones.[a]

[a] Mahaney (1980).