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. 517, 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 *length* polynomial in the input length.
- Corollary 42 (p. 299) 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 $|\, x \,|$.
  - M is an ϵ-approximation algorithm for every $\epsilon > 0$.
- 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. 516).

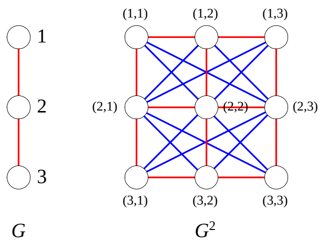
PTAS and Approximation Threshold

- If a problem has a PTAS, then its approximation threshold is 0.
- If the approximation threshold of a problem is greater than 0, then it does not have a PTAS.
- From p. 513, NODE COVER, MAXSAT, TSP, and INDEPENDENT SET do not have a PTAS.

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

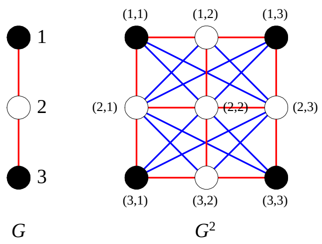
$$\{[(u, u'), (v, v')] : (u = v \wedge [u', v'] \in E) \vee [u, v] \in E\}.$$



Independent Sets of G and G^2

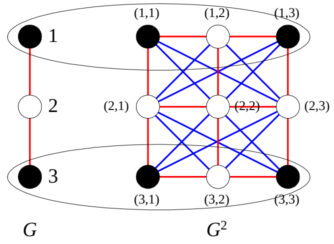
Lemma 74 *$G(V, E)$ has an independent set of size k if and only if G^2 has an independent set of size k^2.*

- 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 (concluded)

- Suppose G^2 has an independent set I^2 of size k^2.
- $\{u : \exists v \in V\ (u, v) \in I^2\}$ is an independent set of G.
- $\{v : \exists u \in V\ (u, v) \in I^2\}$ is an independent set of G.
- One of them has size $\geq k$ by the pigeonhole principle.



Approximability of INDEPENDENT SET

The approximation threshold of the maximum independent set is either zero or one.[a]

Theorem 75 *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.

[a] It is in fact one!

The Proof (continued)

- By Lemma 74 (p. 525), 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 74 (p. 525), 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 40, p. 281).
- NODE COVER has an approximation threshold at most 0.5 (p. 500).
- But INDEPENDENT SET is unapproximable.
- INDEPENDENT SET limited to graphs with degree $\leq k$ is called k-DEGREE INDEPENDENT SET.
- k-DEGREE INDEPENDENT SET is approximable.

A $k/(1+k)$-Approximation Algorithm

```
I := ∅;
while V ≠ ∅ do
    Delete an arbitrary node v from V;
    Delete nodes incident with v from E;
    Add v to I;
end while
return I;
```

Analysis

- I is an independent set.
- At most $k+1$ nodes are deleted in Step 4.
- So $|I| \geq |V|/(k+1)$.
- The maximum independent set has at most $|V|$ nodes.
- The approximation ratio is at least

$$\begin{aligned} \frac{|V|/(k+1)}{|V|} &= \frac{1}{k+1} \\ &= 1 - \frac{k}{k+1}. \end{aligned}$$

- So the approximation threshold is $\leq k/(k+1)$.

Density[a]

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

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

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

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

  - 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 $\phi[\,t0\,] \lor \phi[\,t1\,]$;
end if

The above algorithm runs in exponential time.

NP-Completeness and Density[a]

Theorem 76 *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. 536.
- The trick is to keep the already discovered results $\phi[\,t\,]$ in a table H.

[a] Berman (1978).

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

The Proof (continued)

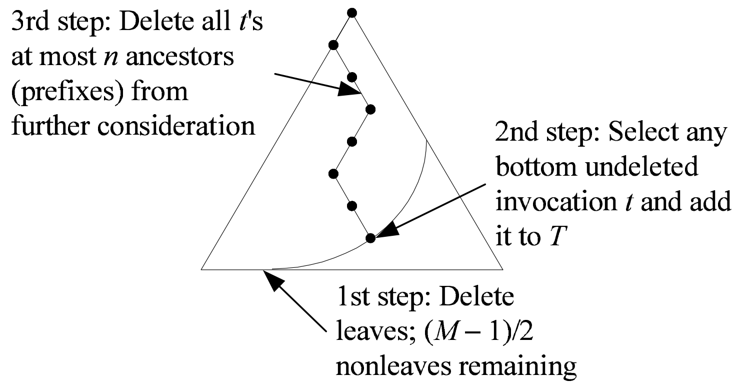
- Since R is a reduction, $R(\phi[\,t\,]) = R(\phi[\,t'\,])$ implies that $\phi[\,t\,]$ and $\phi[\,t'\,]$ must be both satisfiable or unsatisfiable.
- $R(\phi[\,t\,])$ has polynomial length $\leq p(n)$ because R runs in log space.
- As R maps to unary numbers, there are only polynomially many $p(n)$ 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).

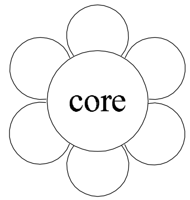
NP-Completeness and Density

Theorem 77 (Fortung (1979)) *If a unary language $U \subseteq \{0\}^*$ is coNP-complete, then P = NP.*

- Suppose there is a reduction R from SAT COMPLEMENT to U.
- The rest of the proof is basically identical except that, now, we want to make sure a formula is unsatisfiable.

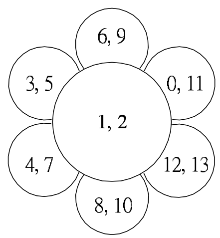
Sunflowers

- Fix $p \in \mathbb{Z}^+$ and $\ell \in \mathbb{Z}^+$.
- A **sunflower** is a family of p sets $\{P_1, P_2, \ldots, P_p\}$, called **petals**, each of cardinality at most ℓ.
- All pairs of sets in the family must have the same intersection (called the **core** of the sunflower).



A Sample Sunflower

$$\{\{1,2,3,5\},\{1,2,6,9\},\{0,1,2,11\},$$
$$\{1,2,12,13\},\{1,2,8,10\},\{1,2,4,7\}\}$$



The Erdős-Rado Lemma

Lemma 78 *Let $\mathcal{Z}$ be a family of more than $M = (p-1)^\ell \ell!$ nonempty sets, each of cardinality ℓ or less. Then $\mathcal{Z}$ must contain a sunflower.*

- Induction on ℓ.
- For $\ell = 1$, p different singletons form a sunflower (with an empty core).
- Suppose $\ell > 1$.
- Consider a *maximal* subset $\mathcal{D} \subseteq \mathcal{Z}$ of *disjoint* sets.
  - Every set in $\mathcal{Z} - \mathcal{D}$ intersects some set in $\mathcal{D}$.

The Proof of the Erdős-Rado Lemma (continued)

- Suppose $\mathcal{D}$ contains at least p sets.
  - $\mathcal{D}$ constitutes a sunflower with an empty core.
- Suppose $\mathcal{D}$ contains fewer than p sets.
  - Let D be the union of all sets in $\mathcal{D}$.
  - $|D| \leq (p-1)\ell$ and D intersects every set in $\mathcal{Z}$.
  - There is a $d \in D$ that intersects more than $\frac{M}{(p-1)\ell} = (p-1)^{\ell-1}(\ell-1)!$ sets in $\mathcal{Z}$.
  - Consider $\mathcal{Z}' = \{Z - \{d\} : Z \in \mathcal{Z}, d \in Z\}$.
  - $\mathcal{Z}'$ has more than $M' = (p-1)^{\ell-1}(\ell-1)!$ sets.
  - M' is just M with ℓ decreased by one.

The Proof of the Erdős-Rado Lemma (concluded)

- (continued)
  - $\mathcal{Z}'$ contains a sunflower by induction, say
    $$\{P_1, P_2, \ldots, P_p\}.$$
  - Now,
    $$\{P_1 \cup \{d\}, P_2 \cup \{d\}, \ldots, P_p \cup \{d\}\}$$
    is a sunflower in $\mathcal{Z}$.

Comments on the Erdős-Rado Lemma

- A family of more than M sets must contain a sunflower.
- **Plucking** a sunflower entails replacing the sets in the sunflower by its core.
- By repeatedly finding a sunflower and plucking it, we can reduce a family with more than M sets to a family with at most M sets.
- If $\mathcal{Z}$ is a family of sets, the above result is denoted by pluck($\mathcal{Z}$).

An Example of Plucking

- Recall the sunflower on p. 547:

$$\begin{aligned}\mathcal{Z} &= \{\{1,2,3,5\},\{1,2,6,9\},\{0,1,2,11\},\\ &\quad \{1,2,12,13\},\{1,2,8,10\},\{1,2,4,7\}\}\end{aligned}$$

- Then

$$\text{pluck}(\mathcal{Z}) = \{\{1,2\}\}.$$