Agenda

- Inapproximability
  - Max Ind. Set
  - Survey
- Algebraic Complexity
  - Model
  - Complexity classes
  - Completeness of Det and Perm

1 Recall

- There exists a poly-time 2-approximation algorithm for MIN-VC
- There exists $\varepsilon > 0$ such that $\text{Gap}_{\frac{2}{3}, \frac{2}{3} + \varepsilon} \text{MIN-VC}$ is $\text{NP}$-hard.

This implies that there does not exists a poly-time $(1 + \varepsilon')$ approximation algorithm for MIN-VC with $\varepsilon' < \varepsilon/2$ unless $\text{P} = \text{NP}$. So MIN-VC has a poly-time approximation, but not an arbitrarily good one.

2 Inapproximability

2.1 MAX-IS

Definition 1 MAX-IS: Given a graph $G$, find an independent set of maximum size.

Theorem 2 (Assuming PCP theorem) For every constant $\rho > 0$, there exists no $\rho$-approximation for MAX-IS unless $\text{P} = \text{NP}$.

Proof:

Lemma 3 $\text{Gap}_{1-a,1-b} \text{MIN-VC} \leq_l \text{Gap}_{a,b} \text{MAX-IS}$.

Here $\text{Gap}_{a,b} \text{MAX-IS}$, for $a > b$ is the promise problem where YES instances are graphs with an independent set of size at least $an$ and NO instances are graphs in which every independent set has size at most $bn$. 
Proof: The reduction is the identity mapping, and the correctness follows from the fact that a set $S$ is a vertex cover iff its complement is an independent set.

**Lemma 4** $\text{GAP}_{a,b} \text{MAX-IS} \leq \text{GAP}_{a^k,b^k} \text{MAX-IS}$. 

Note: if $a > b$, then $b^k/a^k \to 0$ as $n \to \infty$

**Proof:** Given $G = (V,E)$, map $(V,E) \mapsto G_k = (V^k,E_k)$, where 

$$E_k = \{(u_1, \ldots, u_k), (v_1, \ldots, v_k) : \exists i(u_i, v_i) \in E\}.$$ 

If $S$ is an i.s. in $G$, then $S^k$ is an i.s. in $G_k$, so $\text{MAX-IS}(G_k) \geq \text{MAX-IS}(G)^k$. We want to show that the converse holds:

Let $T \subseteq V^k$ be an i.s. in $G_k$. Then the coordinate-wise projections $\pi_1(T), \ldots, \pi_k(T)$ are all ind. sets in $G$ (if you have 2 coordinates which are connected in $G$, there is an edge between them in $G_k$). Then 

$$T \subseteq \pi_1(T) \times \ldots \times \pi_k(T),$$

which implies that 

$$|T| \leq \text{MAX-IS}(G^k).$$

Hence, $\text{MAX-IS}(G_k) = \text{MAX-IS}(G)^k$.

Note: This says nothing about VC. In VC we would have $\frac{1-b^k}{1-a^k} \to 1$, so this method of “amplification” gets worse and worse in VC. Moral of the story: switching from maximization to minimization is *not* equivalent for approximation.

### 2.2 Survey

Below $\varepsilon > 0$ denotes an arbitrarily small constant.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Best known approximation algorithm</th>
<th>NP-hard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EUCLIDEAN TSP</strong></td>
<td>$(1 + \varepsilon)$-approx</td>
<td></td>
</tr>
<tr>
<td><strong>MAX-3SAT</strong></td>
<td>$7/8$-approx</td>
<td>$(7/8 + \varepsilon)$-approx</td>
</tr>
<tr>
<td><strong>MIN-SETCOVER</strong></td>
<td>$\ln n$-approx</td>
<td>$(1 - \varepsilon) \ln n$-approx</td>
</tr>
<tr>
<td><strong>MAX-IS</strong></td>
<td>$(\text{polylog}(n)/n)$-approx</td>
<td>$(1/\sqrt{n^{1-\varepsilon}})$-approx</td>
</tr>
</tbody>
</table>

Note: The NP-hardness of the above problems is proven using PCP optimized for each problem. You begin with a PCP problem and do clever amplifications, compositions.

Here are some problems where we don’t have tight NP-hardness results:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Algorithm</th>
<th>NP-hard</th>
<th>UG-hard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MIN-VC</strong></td>
<td>$2$-approx</td>
<td>$\approx 1.36\ldots$-approx</td>
<td>$(2 - \varepsilon)$-approx</td>
</tr>
<tr>
<td><strong>MAX-CUT</strong></td>
<td>$0.878\ldots$-approx (semi-definite programming)</td>
<td>$17/16 - 3 \approx 0.94\ldots$-approx</td>
<td>$0.878\ldots$-approx (semi-definite programming)</td>
</tr>
<tr>
<td><strong>SHORTEST VECTOR</strong></td>
<td>$\approx 2^{n/\log n}$-approx (looks bad, but really useful)</td>
<td>$2^{\log^{1+\varepsilon} n}$-approx</td>
<td></td>
</tr>
</tbody>
</table>

Where MAX-CUT is the problem of partitioning a graph so that a single cut will sever as many
edges as possible, and **Shortest Vector** is the problem of finding the approximate length of the shortest vector in a lattice graph.

**UG = “unique games”:** this is an approximation problem/PCP variant that is conjectured to be hard. An equivalent problem: given an (inhomogeneous) system of linear equations mod \( q \), with 2 variables per equation, where \( q = q(\varepsilon) \) is a large constant, distinguish \((1 - \varepsilon)\)-satisfiable from \(\varepsilon\)-satisfiable. If this is hard, then our understanding of lots of approximation problems gets resolved (as illustrated by the examples of Min-VC and Max-Cut above). As a result, this conjecture is currently the subject of intense study. This study has also uncovered interesting connections with mathematical questions in metric geometry and discrete Fourier analysis.

### 3 Algebraic Complexity

**Question:** How many arithmetic operations are needed to compute various polynomials of interest?

#### 3.1 Model

Look at algebraic circuits (and formulas), \( C(x_1, \ldots, x_n) \) over a fixed field \( F \)

- inputs: \( x_1, \ldots, x_n \) and constants from \( F \).
- gates: \(+, \cdot\).

(Fact: \(\div\) doesn’t help much.) We view algebraic circuits as computing *formal* polynomials over the field \( F \). These can be evaluated at points in \( F \) (by substituting for the variables \( x_i \)), but are not necessarily determined only by the function they compute. (For example, \( x^2 \) and \( x \) are different polynomials over \( \mathbb{Z}_2 \), even though they compute the same function.)

#### 3.2 Complexity Measures

- **size** = \#gates (including inputs)
- **non-scalar complexity/#“essential” operations** = \# multiplications *not* by a constant.

Motivation: this measure seeks to count the most expensive operations

- **depths** (as in boolean circuits): longest path from input to output, measures parallelism.

#### 3.3 Examples

- **Matrix Multiplication**:

\[
(X_{ij})(Y_{ij}) \rightarrow (Z_{ij} = \sum_k X_{ik}Y_{kj})
\]

This sends \(2n^2\) variables to \(n^2\) output polynomials.

Naive algorithm: \(O(n^3)\)

Best known algorithm: \(O(n^{2.37})\)

Best lower bound: \(\approx 3n^2\)
Discrete Fourier Transform: $\mathbb{F} = \mathbb{C}$

\[
\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \mapsto \begin{pmatrix} \omega^{ij} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}.
\]

(1)

Where $\omega$ = the primitive $n$th root of unity.

Naive algorithm: $O(n^2)$
Fast Fourier Transform (FFT): $O(n \log n)$
Best lower bound: no super linear lower bounds are known.

**Determinant:**

Naive algorithm: $O(n \cdot n!)$
Gaussian elimination: $O(n^3)$
Best known: $O(n^{2.37})$

**Permanent:**

Naive algorithm: $O(n \cdot n!)$
Best known algorithm: $O(2^n)$

**Definition 5** Fix a field $\mathbb{F}$. A sequence $(p_n(x_1, \ldots, x_n))_{n \in \mathbb{N}}$ of polynomials over $\mathbb{F}$ is in $\text{AlgP/poly}$ (also called $\text{VP}$ for “Valiant’s $\text{P}$”) if there exist polynomials $d(n), s(n)$ such that for all $n$

1. $\deg(p_n) \leq d(n)$
2. $p_n$ is computable by an arithmetic circuit of size at most $s(n)$.

Why bound degree?
- $\deg \leq 2^{\text{size}}$ in any arithmetic circuit
- most functions of interest have low polynomially bounded degree
- it is useful in results

**Definition 6** $(p_n(x_1, \ldots, x_n))_{n \in \mathbb{N}}$ is in $\text{AlgNP/poly}$ (a.k.a. $\text{VNP}$) if there exists a sequence of polynomials $(q_n(x_1, \ldots, x_n))_{n \in \mathbb{N}}$ in $\text{AlgP/poly}$ and a polynomial $t(n)$ such that for all $n$,

\[
p_n(x_1, \ldots, x_n) = \sum_{e_{n+1}, \ldots, e_{t(n)} \in \{0,1\}} q_{t(n)}(x_1, \ldots, x_n, e_{n+1}, \ldots, e_{t(n)})
\]

3.4 Next time

**Theorem 7**

- Det is complete for $\text{AlgP/poly}$.
- Perm is complete for $\text{AlgNP/poly}$.

Hence, $\text{AlgP/poly} = \text{AlgNP/poly} \iff$ Perm is a “projection” of the determinant.