**IITM-CS6840: Advanced Complexity Theory** March 31, 2012

# Lecture 59 : Instance Compression and Succinct PCP's for NP

*Lecturer: Sivaramakrishnan N.R.* *Scribe: Prashant Vasudevan*

## 1 Introduction

Classical Complexity Theory is concerned with studying the time needed for computing functions as a funciton of the input size. Parametrised Complexity studies the same running time as a function of both the input size and of a parameter of the problem instance.

**Definition 1.** A parameterization of a decision problem is a function that assigns a parameter $k$ to each input instance $x$. Each input $x$ is thus transformed to $(x, 1^k)$.

This formalism is useful in cases where a problem that is difficult to solve in general becomes tractable for inputs in which some parameter is constrained. The following definition formalises this notion.

**Definition 2.** A parameterized problem is *fixed parameter tractable*(FPT) if it has an $f(k)n^{O(1)}$ time algorithm, where $n$ is the size of the original input and $k$ is the attached parameter.

Thus, when the parameter is small, we have efficient algorithms for an FPT problem.

As an example, we look at the Vertex Cover problem. Here, the input is an undirected graph $G(V, E)$ on $n$ vertices and the objective is, given $k$, to find or report the absence of a subset $S$ of $V$ of size $k$ such that for any edge $\{u, v\} \in E$, atleast one of $u$ and $v$ is in $S$ (the vertex cover).

A straightforward, though inefficient, solution is the following recursive procedure:

1. Pick an edge $u, v \in E$.
2. Pick $u$ and delete all edges adjacent to it.
3. If $k$ vertices have been picked:
   i) If all edges have been deleted, report success and return.
   ii) If some edge remains, try again by picking $v$. If then too edges remain, report failure and return.

4. If less than $k$ vertices have been picked, repeat process on the remaining graph.
5. If procedure reports success on remaining graph, report success and return.
6. If procedure reports failure on remaining graph, try steps 2 to 4 by picking $v$ instead. If this too fails, report failure and return.

The search tree has depth at most $k$, as only $k$ vertices are picked, and hence at most $2^k$ nodes. Processing at each node involves deleting edges, of which there are at most $n^2$. Hence this procedure runs in $O(2^k n^2)$ time.

This might be bettered using the observation that if we require a vertex cover of size $k$, then any vertex of degree greater than $k$ is necessarily part of the cover. If this were not so, then all other neighbours of the vertex would have to be in the cover, but as there are more than $k$ neighbours, this cannot be the case. Hence, as part of preprocessing, if there is a vertex $v$ of degree greater than $k$, we add it to the cover, remove all edges adjacent to it, and look for a cover of size $k-1$ in the remaining graph, performing the preprocessing again if applicable.

Note that, if the initial graph had a vertex cover of size $k$, the resulting graph after preprocessing has at most $k^2$ edges. This is because the maximum degree of any vertex in this graph is $k$, and it has a cover of at most $k$ vertices which can cover no more than $k^2$ edges. It also has at most $k^2 + k$ vertices by the same argument. This way, if $k$ is constant, this preprocessing reduces the problem instance to one of constant size in $O(n^2)$ time (if a vertex cover exists, that is).

What the above preprocessing essentially does is to, in steps, reduce a given problem instance to a smaller one. We try now to generalise this notion. Consider a language $L$ . We ask whether there exists some polynomial-time computable function $f$ such that for any $x$, $f$ maps $x$ to a smaller string whose membership $L$ is the same as that of $x$. If this, a notion related to *Kernelisation* in Parametrised Complexity Theory, can be done, we may attempt to apply it iteratively until the instance size reduces to a constant, similar to what we did in the Vertex Cover problem.

## 2 Parametric Problems

If $L$ is NP-Complete, we go a bit further and ask for such an $f$ that compresses the problem to size polynomial in the witness size (which could be logarithmically smaller than the input size). To this end, we make the following definitions. (Throughout this section, $m$ is the input size and $n$ is the parameter.)

**Definition 3.** Let $L$ be a parametric problem and $A \subseteq \{0,1\}^*$. $L$ is said to be compressible within $A$ if there is a polynomial $p$ and a polynomial-time computable function $f$ such that for each $x \in \{0,1\}^*$ and $n \in \mathbb{N}$:

1. $|f(x, 1^n)| \leq p(n)$
2. $(x, 1^n) \in L \Leftrightarrow f(x, 1^n) \in A$

Further, $L$ is *compressible* if there is some $A$ for which $L$ is compressible within $A$. $L$ is *self-compressible* if $L$ is compressible within $L$.

**Definition 4.** A parametric problem $L$ is said to be *compressible with advice* $s$ if the compression function is computable in deterministic polynomial time when given access to an advice string of size $s(|x|, n)$. $L$ is *non-uniformly compressible* if $s$ is polynomially bounded in $m$ and $n$.

We now define a few parametric problems.

1. $\mathsf{SAT} = \{(\phi, 1^n) \mid \phi$ is a satisfiable formula, and $n$ is atleast the number of variables in $\phi\}$
2. $VC = \{(G, 1^{k \log m}) \mid G$ has a vertex cover of size at most $k\}$
3. $OR - SAT = \{(\{\phi_i\}, 1^n) \mid$ at least one $\phi_i$ is satisfiable and each $\phi_i$ has size at most $n\}$

Also is pertinent the following notion of reductions between parametric problems.

**Definition 5.** Given parametric problems $L_1$ and $L_2$ , $L_1$ *$W$-reduces* to $L_2$ (denoted by $L_1 \leq_W L2$) if there is a polynomial-time computable function $f$ and polynomials $p_1$ and $p_2$ such that:

1. $f(x, n_1)$ is of the form $(y, n_2)$ where $|y| \leq p_1(n1 + |x|)$ and $n_2 \leq p_2(n_1)$.
2. $f(x, n_1) \in L_2 \Leftrightarrow (x, n_1) \in L_1$.

This is useful because if $L_1 \leq_W L_2$ and $L_2$ is compressible, then $L_1$ is also compresible.

## 3 Deterministic Compression

We now prove a theorem that indicates the infeasibility of deterministic compression.

**Theorem 6.** *If $OR - SAT$ is compressible, then* $\mathsf{coNP} \subseteq \mathsf{NP/poly}$*, and so* $\mathsf{PH}$ *collapses.*

*Proof.* Consider $OR - SAT$ instances of size $m$ with each sub-formula of size at most $n$. According to the hypothesis, $\exists$ a set $A$ and function $f$ poly-time in $m$ such that:

1. $|f(\phi, 1^n)| \leq O(poly(n, \log m))$.
2. $\phi$ is satisfiable iff $f(\varphi, 1^n) \in A$.

The size of the compressed instance is $k = (n + \log m)^{O(1)}$.

Let $S$ be the set of unsatisfiable formulae of of size at most $n$, and $T$ be the set of strings in $\bar{A}$ of length at most $k$. $f$ is now found to induce a map $g : S^{m/n} \to T$.

Now, if we can find a $C \subseteq T$, of size polynomial in $n$, such that any formula in $S$ is contained in at least one tuple that maps to a string in $C$ under $g$, then we can do the following to decide the membership of $\psi$ in $S\bar{A}T$:

1. Guess a tuple of $m/n$ formulae of size $n$ with $\psi$ as one of them.
2. Check if this tuple maps to a string in $C$ under $g$.

This gives us a non-deterministic algorithm, with advice $C$, running in time polynomial in $m$ to decide $S\bar{A}T$ which, if $m$ is polynomial in $n$, is also polynomial in $n$ too, giving $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.

We now show the existence of such a set $C$ by constructing one. We start with an empty set and add strings to it iteratively.

After the $i^{th}$ iteration, let $S_i$ be the set of formulae in $S$ that are yet to be covered, and $C_i$ be the set of strings picket yet. Let $X_i \subseteq S^{m/n}$ be the set of tuples that are not in the pre-image set of $C_i$.

At the $(i+1)^{th}$ iteration, we pick a string in $T$ with the maximum number of pre-images in $X_i$ and add it to $C_i$. Do this till all tuples in $S^{m/n}$ are covered. If this procedure terminates in polynomial (in $n$) number of steps, we would have our set $C$.

We have the following observations which may be arrived at by counting arguments:

1. $|X_{i-1}| - |X_i| \geq \frac{|X_{i-1}|}{2^k}$
2. $|S_{i-1}| - |S_i| \geq \frac{|X_{i-1}|^{\frac{n}{m}}}{2^{\frac{kn}{m}}}$
3. $|X_{i-1}|^{\frac{n}{m}} \geq |S_{i-1}|$

These give us $|S_{i-1}| - |S_i| \geq \frac{|S_{i-1}|}{2^{\frac{kn}{m}}}$.

As $k = (n + \log m)^{O(1)}$, we may pick $m = n^c$ for a $c$ such that $kn < m$. This gives us $|S_i| \leq \frac{|S_{i-1}|}{2}$. This way, the number of uncovered formulae reduces by a factor of 2 in each iteration. As there are initially $O(2^n)$ of these, the procedure halts in polynomial (in $n$) number of steps. As each step adds only one string to $C$, $C$ is also of size polynomial in $n$, giving us the required advice string.

□

## 4 Probabilistic Compression

We now relax the notion of compression a bit to allow for probabilistic algorithms with errors.

**Definition 7.** Let $L$ be a parametric problem and $A \subseteq \{0,1\}$. $L$ is said to be probabilistically compressible with error $\varepsilon(n)$ within $A$ if there is a polynomial $p$ and a probabilistic polynomial-time computable function $f$ such that for each $x \in \{0,1\}$ and $n \in \mathbb{N}$, with probability atleast $1 - \varepsilon(|x|)$ we have :

1. $|f(x, 1^n)| \leq p(n)$
2. $f(x, 1^n) \in A \Leftrightarrow x \in L$

Such a probabilistic compression function is said to have randomness complexity $R$ if it uses atmost $R$ random bits.

But even so, compression with considerably less error is unlikely, as indicated by the following theorem, akin to the one proven earlier.

**Theorem 8.** *If $OR - SAT$ is compressible with $\varepsilon(m) = 2^{-m}$, then too* $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.

This may be proven by noting that, following the same counting argument used to show $\mathsf{BPP} \subseteq \mathsf{P/poly}$, an error of only $2^{-m}$ implies the existenct of a single random string that works without error for all instances of a given length. This string may be taken as part of the advice in the previous proof, yielding the same result.

## 5 Succinct PCPs

The following is the traditional statement of the PCP Theorem:

**Theorem 9** (PCP Theorem). $\mathsf{NP} = PCP(O(\log n), O(1))$.

In words, any language in $\mathsf{NP}$ has a proof polynomial in size of input $n$ that can be verified with $O(\log n)$ random bits and $O(1)$ queries such that a positive answer to membership can always be certainly verified and any negative answer is verified with mistakes at most half the time. (And vice versa.)

We now ask whether this proof, instead of being polynomial in the input size, may indeed be made as small as polynomial in the size of the witness for the $\mathsf{NP}$ problem (which, as noted earlier, may be logarithmically smaller than the input). The following definition formalises that which we seek.

**Definition 10.** Let $L$ be a parametric problem. $L$ is said to have a succint $PCP$ with completeness $c$, soundness $s$, proof size $S$ and query complexity $q$ if there is a probabilistic polynomial-time oracle machine $V$ such that the following holds for any instance $(x, 1^n)$:

1. If $(x, 1^n) \in L$, then there is a proof $y$ of size $S(n)$ such that on input $(x, 1^n)$, $V$ makes atmost $q$ queries to $y$ and accepts with probability at least $c$.
2. If $(x, 1^n) \notin L$, then for any string $y$ of size $S(n)$, on input $(x, 1^n)$, $V$ makes atmost $q$ queries to $y$ and accepts with probability at most $s$.

$L$ is said to have a *succint PCP* if it has a succint $PCP$ with completeness 1, soundness $\frac{1}{2}$, proof size polynomial in $n$ and constant query complexity.

We end with the statement of the following theorem which ties in with what we have proven earlier, indicating again the improbability of the existence of succinct $PCP$s for all $\mathsf{NP}$ problems:

**Theorem 11.** *If* $\mathsf{SAT}$ *has a succinct PCP, then* $\mathsf{SAT}$ *is self-compressible with error less than* $2^{-m}$*.*