

Lecture 15: Simulating BPP - High Min-Entropy Sources

November 4, 2004

Scribe: Ehsan Amiri

1 Simulating BPP using SV-sources

Recall the definition of SV-Sources from lecture 12:

Definition 1 A random variable X over the binary strings of length n is a sample of a **Santha-Vazirani source** if for a fixed constant¹ $0 < \delta \leq \frac{1}{2}$ and any $1 \leq i \leq n$ the i -th bit of X , denoted by X_i , satisfies the following property:

$$\delta \leq \Pr[X_i = 1 | X_1 = b_1, \dots, X_{i-1} = b_{i-1}] \leq 1 - \delta$$

It's easy to check that any specific string of length n has probability at most $(1 - \delta)^n$, hence SV-sources are $n \log \frac{1}{1-\delta}$ -sources. Also recall that SV-sources are (k, ℓ) block sources where $k = \ell \log \frac{1}{1-\delta}$. Suppose $\delta' = \log \frac{1}{1-\delta}$, then $k = \delta' \ell$.

Take a hash-based extractor² $\mathcal{E} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^{d+m}$. Here, our weak-source is a SV-source over the strings of length n and suppose $k = n \log \frac{1}{1-\delta}$ is the lower bound for min-entropy of the source. According to construction of hash-based extractors (Lecture 13) $d = O(n)$, $d + m = k + d - 2 \log \frac{1}{\epsilon}$, hence $m = k - 2 \log \frac{1}{\epsilon}$. (ϵ is, as usual, statistical distance of output and uniform distribution).

Recall that we are going to simulate a **BPP** algorithm. Assume that input length of our **BPP** algorithm is n' , so it may use up to $\text{poly}(n')$ random bits. If we want to use extractor \mathcal{E} described above then we should have $d + m = \text{poly}(n')$. So, clearly $n = O(\text{poly}(n'))$ which means $d = O(\text{poly}(n'))$ which is quite useless. Our main purpose of this simulation is making a significant decrease in the length of truly random bit required to run a **BPP** algorithm, for example we would like to have $d = O(\text{polylog}(n'))$. Now we are going to improve our construction to meet this requirement:

Now set $\ell = \frac{10 \log n}{\delta'}$. So X would be a (k, ℓ) -source for $k = 10 \log n$. We still want to use hash-based extractors. So set $\epsilon = \frac{1}{n^2}$, the parameters of extractor would be

$$d = O(\ell) = O(\log n) \tag{1}$$

$$m \geq 6 \log n \tag{2}$$

Using the SV-source as a (k, ℓ) block source and the above base extractor, as we saw in the previous lecture one can build extractor \mathcal{E}' with the following parameters:

$$\mathcal{E}' : \{0, 1\}^{\frac{n}{\ell} \cdot \ell} \times \{0, 1\}^d \rightarrow \{0, 1\}^{d + \frac{n}{\ell} \cdot 6 \log n}$$

¹In general δ may not be a constant, but in this section we work with constant δ .

²Here, our notation for output length is different from usual one to make coming calculations simpler.

And we would have $\epsilon' \leq \frac{1}{n}$. The amount of entropy in the input source is $n\delta'$. It's easy to check that

$$\frac{n\delta'\ell}{\ell} \leq \frac{n}{\ell} \times 6 \log n$$

We conclude that the above constructor extracts at least half of the randomness of the weak-source.

2 Extraction From high min-entropy sources

The purpose of this section is constructing an "explicit" optimal extractor for high min-entropy sources. Our construction is somewhat complicated and can be compared to zigzag product in graphs. Explicitness of the construction can be disputed, since it includes exhaustive search for some components instead of an actual description, yet we call it "explicit" because the exhaustive search takes reasonable time. (Anyway, that's why word 'explicit' is double-quoted). This is still much better than a probabilistic argument to prove the existence of an optimal extractor.

In this section, instead of our usual notion of min-entropy we will use the notion of *entropy deficiency* as defined below:

Definition 2 An $n - \Delta$ source $X \in \{0, 1\}^n$ is said to have **entropy deficiency** Δ .

Also, we'll use the notion of *entropy loss*, defined below, to measure the quality of an extractor.

Definition 3 Assume that $\mathcal{E} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ is a (k, ϵ) -extractor. **Entropy loss** of \mathcal{E} is defined to be $\Lambda = k + d - m$

We have already mentioned a tight lower bound for Λ , namely $\Lambda \geq 2 \log \frac{1}{\epsilon}$ and we know that non-constructive optimal extractor achieves this bound. Also the optimal value of d is $\log(n - k) + 2 \log \frac{1}{\epsilon} + O(1)$ which in our case that $k = n - \Delta$ can be written as $d = \log \Delta + 2 \log \frac{1}{\epsilon} + O(1)$.

Assume that Δ and ϵ are constants. We want to construct an "explicit" extractor that achieves $d = O(\log \Delta)$ and $\Lambda = O(1)$ ³.

As our starting point consider the expander based extractor. For this extractor we have $d = O(\Delta)$ and $\Lambda = \Theta(\Delta)$. However the seed length of extractor is relatively good, it's entropy loss is too much.

In the first step we try to improve the seed length. We need to decrease the seed length of expander-based extractor from $O(\Delta)$ to $O(\log \Delta)$. The idea is to use a "small" extractor with seed length $O(\log \Delta)$ and output length $O(\Delta)$. Then the output of this extractor can be used as the seed of the expander-based one. However the "small" extractor we need is essentially the optimal extractor, because of our choice of parameters it can be constructed efficiently by exhaustive search over all possible candidates. See Figure 1 for a pictorial representation of the extractors.

The weird thing in Figure 1 is string z in the output of *Ext*. Recall that we want to keep entropy deficiency as small as possible. *Ext* throws away some entropy. If we choose *Ext* to be a permutation (when the output is the whole string $d'z$) it would actually keep all the entropy in the input string,

³In general Δ could be $\omega(1)$ as long as it grows slowly enough, say like $O(\log \log n)$, so it makes sense to distinguish between $O(\log \Delta)$ and $O(1)$

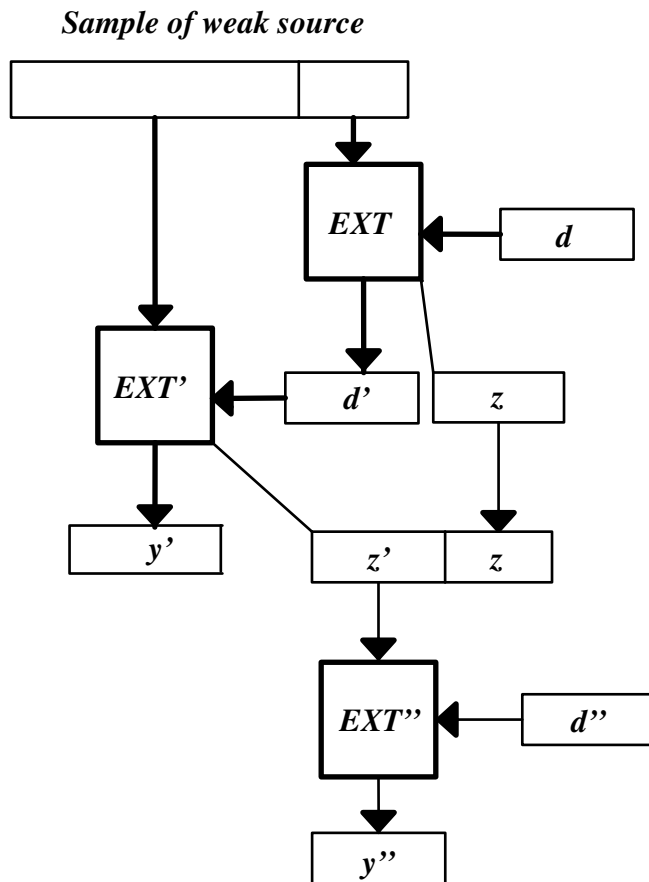


Figure 1: Extractor for high min-entropy sources

since permutations do not decrease (neither increase) the entropy. So, z has some entropy that will be recycled later by Ext'' .

We will fix parameters later, but before that the main unanswered question is where the input of the "small" extractor comes from? The following proposition helps us to use a small part of our high-min-entropy source for this purpose:

Proposition 4 *If $X = (X_1, X_2)$ is a Δ -deficient source then*

1. X_1 has deficiency at most Δ .
2. For any $\epsilon \geq 0$, with probability at least $1 - \epsilon$ over the choice of x_1 the conditional distribution $X_2|_{X_1=x_1}$ has deficiency at most $\Delta + \log \frac{1}{\epsilon}$

The proposition says that any randomness source is, loosely speaking, a block source.

Ext' is the expander-based extractor discussed above. Ext is the "small" extractor discussed above that provides Ext' with seed. There is another extractor Ext'' whose role would be more clear when we consider the analogy of this construction and zigzag product.

Assume that we have two "small" and "large" graphs and consider the zigzag product of them. (Recall the asymmetry in the definition of zigzag product). One step of the random walk on the zigzag product of the two graphs consists of three steps in the two initial graphs. The first step (zig) is in the small graph. What actually happens is that we walk on the cloud around one of the nodes of the "large" graph. Then we take one deterministic step in the "large" graph and actually we move to a new cloud in the zigzag product. This step is deterministic and fixed by the previous step taken in the small graph. Then we take another step in the "small" graph (zag).

Now consider bipartite graphs corresponding to the small and large extractors in the Figure 1. The first step that chooses the seed for the next step, is like the first step in the zigzag product that actually fixes the step that is going to be taken in the large graph. The operation of big extractor is now deterministic. The variable z' remembers the edge label of the "large" graph. Unlike the zig-zag product, here we also used the variable z to remember the edge label of the "small" graph (where we took the "zig" step). The third step is like the "zag" step, except we apply the extractor to the concatenation of the strings $z'z$ and the result is a shorter, but almost uniform, string y'' .

Slightly more formally, let's denote the source distribution as X_1X_2 , and let R be the uniform distribution on the seeds to the first "permuting extractor" Ext . Let $Ext(X_2R) = SZ$, where S will be used as the seed for Ext' . By Proposition 4, one can show that $X_1S \stackrel{2\epsilon}{\approx} X_1U$ (to simplify the notation, we will just write U to mean the uniform distribution on strings of appropriate length). The idea is that the statistical distance will be the average of distances between the distributions S and U , conditioned on a value of X_1 . For all but ϵ fraction of the values x_1 of X_1 , the conditional distribution $X_2|_{X_1=x_1}$ will have "high" min-entropy, and hence, S will be ϵ -close to U ; the remaining ϵ fraction of x_1 's, we get the distance at most 1. The total distance will be $(1 - \epsilon) * \epsilon + \epsilon * 1 \leq 2\epsilon$.

Let $Ext'(X_1S) = Y'Z'$. Since X_1S is 2ϵ -close to X_1U , and since applying the deterministic function Ext' cannot increase the statistical distance, we get Y' is 2ϵ -close to $Ext'(X_1U)$. The latter is ϵ -close to U , since by Proposition 4, X_1 has "high" min-entropy. Thus, we get Y' is 3ϵ -close to U .

Finally, since $Y'Z'Z$ is the result of a permutation applied to X_1X_2R , we know that $Y'Z'Z$ has all the min-entropy of the input distribution. Let $Ext''(Z'Z, U) = Y''$. By Proposition 4, $Y'Y''$ is 2ϵ -close to $Y'U$. On the other hand, since Y' is 3ϵ -close to U , we get that $Y'U$ is also 3ϵ -close to U (of the same length as $Y'U$). Hence, the output $Y'Y''$ of our combined extractor is 5ϵ -close to U .

For the parameters, note that the seed lengths of Ext and Ext'' are $O(\log \Delta)$. The seed length of Ext' is $c\Delta$ for some constant c , so we choose the input of Ext to be the last $(c + 1)\Delta$ bits of the high min-entropy source.

3 Summary of extractor constructions

The following is a summary of parameters of different constructions of extractors. For the sake of simplicity we have assumed that ϵ is a constant.

(k, ϵ) -extractor	seed length d	output length n
non-constructive optimal	$\log(n - k) + O(1)$	$k + d - O(1)$
best known explicit-I	$O(\log n)$	$(1 - \gamma)k, \forall \gamma > 0$
best known explicit-II	$O(\log^2 n)$	$k + d - O(1)$
expander based	$O(n - k)$	n
hash-based	$O(n)$	$k + d - O(1)$
almost-universal hash based	$O(\log n + k)$	$k + d - O(1)$

Table 1: Parameters of different constructions of extractors

We conclude with the big open problem of area:

Open Problem 5 *Explicitly construct an optimal extractor, even with $d = O(\log n)$ but with optimal entropy loss.*