Lecture Space-Bounded Derandomization

Jonathan Katz

1 Space-Bounded Derandomization

We now discuss derandomization of space-bounded algorithms. Here non-trivial results can be shown *without making any unproven assumptions*, in contrast to what is currently known for derandomizing time-bounded algorithms. We show first that[1] $\mathcal{BPL} \subseteq \mathsf{SPACE}(\log^2 n)$ and then improve the analysis and show that[2] $\mathcal{BPL} \subseteq \textsc{TimeSpc}(\mathsf{poly}(n), \log^2 n) \subseteq \mathcal{SC}$. (Note: we already know

$$\mathcal{RL} \subseteq \mathcal{NL} \subseteq \mathsf{SPACE}(\log^2 n)$$

but this does not by itself imply $\mathcal{BPL} \subseteq \mathsf{SPACE}(\log^2 n)$.)

With regard to the first result, we actually prove something more general:

Theorem 1 *Any randomized algorithm (with two-sided error) that uses space $S = \Omega(\log n)$ and R random bits can be converted to one that uses space $\mathcal{O}(S \log R)$ and $\mathcal{O}(S \log R)$ random bits.*

Since any algorithm using space S uses time at most 2^S (by our convention regarding probabilistic machines) and hence at most this many random bits, the following is an immediate corollary:

Corollary 2 *For $S = \Omega(\log n)$ it holds that $\mathsf{BPSPACE}(S) \subseteq \mathsf{SPACE}(S^2)$.*

Proof Let $L \in \mathsf{BPSPACE}(S)$. Theorem 1 shows that L can be decided by a probabilistic machine with two-sided error using $\mathcal{O}(S^2)$ space and $\mathcal{O}(S^2)$ random bits. Enumerating over all random bits and taking majority, we obtain a deterministic algorithm that uses $\mathcal{O}(S^2)$ space. ■

2 $\mathcal{BPL} \subseteq \mathsf{SPACE}(\log^2 n)$

We now prove Theorem 1. Let M be a probabilistic machine running in space S (and time 2^S), using R random bits, and deciding a language L with two-sided error. (Note that S, R are functions of the input length n, and the theorem requires $S = \Omega(\log n)$.) We will assume without loss of generality that M always uses exactly R random bits on all inputs. Fixing an input x and letting ℓ be some parameter, we will view the computation of M_x as a random walk on a multi-graph in the following way: the nodes of the graph correspond to all $N \stackrel{\text{def}}{=} 2^{\mathcal{O}(S)}$ possible configurations of M_x, and there is an edge from a to b labeled by the string $r \in \{0,1\}^\ell$ if and only if M_x moves from configuration a to configuration b after reading r as its next ℓ random bits. Computation of M_x is then equivalent to a random walk of length R/ℓ on this graph, beginning from the node corresponding to the initial configuration of M_x. if $x \in L$ then the probability that this random

[1] $\mathcal{BPL}$ is the two-sided-error version of $\mathcal{RL}$.

[2] $\mathcal{SC}$ stands for "Steve's class", and captures computation that *simultaneously* uses polynomial time and polylogarithmic space.

walk ends up in an accepting state is at least $2/3$, while if $x \notin L$ then the probability that this random walk ends up in an accepting state is at most $1/3$.

It will be convenient to represent this process using an $N \times N$ *transition matrix* Q_x, where the entry in column i, row j is the probability that M_x moves from configuration i to configuration j after reading ℓ random bits. Vectors of length N whose entries are non-negative and sum to 1 correspond to probability distributions over the configurations of M_x in the natural way. If we let $\mathbf{s}$ denote the probability distribution that places probability 1 on the initial configuration of M_x (and 0 elsewhere), then $Q_x^{R/\ell} \cdot \mathbf{s}$ corresponds to the probability distribution over the final configuration of M_x; thus:

$$\begin{aligned} x \in L \quad &\Rightarrow \quad \sum_{i \in \mathsf{accept}} \left(Q_x^{R/\ell} \cdot \mathbf{s} \right)_i \geq 3/4 \\ x \notin L \quad &\Rightarrow \quad \sum_{i \in \mathsf{accept}} \left(Q_x^{R/\ell} \cdot \mathbf{s} \right)_i \leq 1/4. \end{aligned}$$

The statistical difference between two vectors/probability distributions $\mathbf{s}, \mathbf{s}'$ is

$$\mathsf{SD}(\mathbf{s}, \mathbf{s}') \stackrel{\text{def}}{=} \frac{1}{2} \cdot \left\| \mathbf{s} - \mathbf{s}' \right\| = \frac{1}{2} \cdot \sum_i |\mathbf{s}_i - \mathbf{s}'_i|.$$

If Q, Q' are two transition matrices — meaning that all entries are non-negative, and the entries in each column sum to 1 – then we abuse notation and define

$$\mathsf{SD}(Q, Q') \stackrel{\text{def}}{=} \max_{\mathbf{s}} \{ \mathsf{SD}(Q\mathbf{s}, Q'\mathbf{s}) \},$$

where the maximum is taken over all $\mathbf{s}$ that correspond to probability distributions. Note that if Q, Q' are $N \times N$ transition matrices and $\max_{i,j}\{|Q_{i,j} - Q'_{i,j}|\} \leq \varepsilon$, then $\mathsf{SD}(Q, Q') \leq N\varepsilon/2$.

2.1 A Useful Lemma

The pseudorandom generator we construct will use a family H of pairwise-independent functions as a building block.

Definition 1 $H = \{h_k : \{0,1\}^\ell \rightarrow \{0,1\}^\ell\}$ *is a* family of pairwise-independent functions *if for all distinct* $x_1, x_2 \in \{0,1\}^\ell$ *and any* $y_1, y_2 \in \{0,1\}^\ell$ *we have:*

$$\Pr_{h \in H} \left[h(x_1) = y_1 \wedge h(x_2) = y_2 \right] = 2^{-2\ell}.$$

It is easy to construct a pairwise-independent family H whose functions map ℓ-bit strings to ℓ-bit strings and such that (1) $|H| = 2^{2\ell}$ (and so choosing a random member of H is equivalent to choosing a random 2ℓ-bit string) and (2) functions in H can be evaluated in $\mathcal{O}(\ell)$ space.

For $S \subseteq \{0,1\}^\ell$, define $\rho(S) \stackrel{\text{def}}{=} |S|/2^\ell$. We define a useful property and then show that a function chosen from a pairwise-independent family satisfies the property with high probability.

Definition 2 *Let* $A, B \subseteq \{0,1\}^\ell$, $h : \{0,1\}^\ell \rightarrow \{0,1\}^\ell$, *and* $\varepsilon > 0$. *We say* h *is* (ε, A, B)-good *if:*

$$\left| \Pr_{x \in \{0,1\}^\ell} \left[x \in A \bigwedge h(x) \in B \right] - \rho(A) \cdot \rho(B) \right| \leq \varepsilon.$$

Note that this is equivalent to saying that h is (ε, A, B)-good if

$$\left| \Pr_{x \in A} [h(x) \in B] - \rho(B) \right| \leq \varepsilon / \rho(A).$$

Lemma 3 *Let $A, B \subseteq \{0,1\}^\ell$, H be a family of pairwise-independent functions, and $\varepsilon > 0$. Then:*

$$\Pr_{h \in H} [h \text{ is not } (\varepsilon, A, B)\text{-good}] \leq \frac{\rho(A)\rho(B)}{2^\ell \varepsilon^2}.$$

Proof The proof is fairly straightforward. Consider the quantity

$$\begin{aligned}
\mu &\stackrel{\text{def}}{=} \mathsf{Exp}_{h \in H} \left[\left(\rho(B) - \Pr_{x \in A} [h(x) \in B] \right)^2 \right] \\
&= \mathsf{Exp}_{h \in H} \left[\rho(B)^2 + \Pr_{x_1 \in A} [h(x_1) \in B] \cdot \Pr_{x_2 \in A} [h(x_2) \in B] - 2\rho(B) \cdot \Pr_{x_1 \in A} [h(x_1) \in B] \right] \\
&= \rho(B)^2 + \mathsf{Exp}_{x_1, x_2 \in A;\, h \in H} \left[\delta_{h(x_1) \in B} \cdot \delta_{h(x_2) \in B} - 2\rho(B) \cdot \delta_{h(x_1) \in B} \right],
\end{aligned}$$

where $\delta_{h(x) \in B}$ is an indicator random variable which is equal to 1 if $h(x) \in B$ and 0 otherwise. Since H is pairwise independent, it follows that:

- For any x_1 we have $\mathsf{Exp}_{h \in H}[\delta_{h(x_1) \in B}] = \Pr_{h \in H}[h(x_1) \in B] = \rho(B)$.
- For any $x_1 = x_2$ we have $\mathsf{Exp}_{h \in H}[\delta_{h(x_1) \in B} \cdot \delta_{h(x_2) \in B}] = \mathsf{Exp}_{h \in H}[\delta_{h(x_1) \in B}] = \rho(B)$.
- For any $x_1 \neq x_2$ we have $\mathsf{Exp}_{h \in H}[\delta_{h(x_1) \in B} \cdot \delta_{h(x_2) \in B}] = \Pr_{h \in H}[h(x_1) \in B \wedge h(x_2) \in B] = \rho(B)^2$.

Using the above, we obtain

$$\mu = \rho(B)^2 + \frac{\rho(B)}{|A|} + \frac{\rho(B)^2(|A| - 1)}{|A|} - 2\rho(B)^2 = \frac{\rho(B) - \rho(B)^2}{|A|} = \frac{\rho(B)(1 - \rho(B))}{|A|}.$$

Using Markov's inequality,

$$\begin{aligned}
\Pr_{h \in H} [h \text{ is not } (\varepsilon, A, B)\text{-good}] &= \Pr_{h \in H} \left[\left(\Pr_{x \in A} [h(x) \in B] - \rho(B) \right)^2 > (\varepsilon / \rho(A))^2 \right] \\
&\leq \frac{\mu \cdot \rho(A)^2}{\varepsilon^2} = \frac{\rho(B)(1 - \rho(B))\rho(A)}{2^\ell \varepsilon^2} \leq \frac{\rho(B)\rho(A)}{2^\ell \varepsilon^2}.
\end{aligned}$$

■

2.2 The Pseudorandom Generator and Its Analysis

2.2.1 The Basic Step

We first show how to reduce the number of random bits by roughly half. Let H denote a pairwise-independent family of functions, and fix an input x. Let Q denote the transition matrix corresponding to transitions in M_x after reading ℓ random bits; that is, the (i, j)th entry of Q is the probability that M_x, starting in configuration i, moves to configuration j after reading ℓ random bits. So Q^2 is a transition matrix denoting the probability that M_x, starting in configuration i, moves to configuration j after reading 2ℓ random bits. Fixing $h \in H$, let Q_h be a transition matrix where the (i,j)th entry in Q_h is the probability that M_x, starting in configuration i, moves to configuration j after reading the 2ℓ "random bits" $r \| h(r)$ (where $r \in \{0,1\}^\ell$ is chosen uniformly at random). Put differently, Q^2 corresponds to taking two uniform and independent steps of a random walk, whereas Q_h corresponds to taking two steps of a random walk where the first step (given by r) is random and the second step (namely, $h(r)$) is a deterministic function of the first. We now show that these two transition matrices are "very close". Specifically:

Definition 3 *Let Q, Q_h, ℓ be as defined above, and $\varepsilon \geq 0$. We say $h \in H$ is* ε-good for Q *if*

$$\mathsf{SD}(Q_h, Q^2) \leq \varepsilon/2\,.$$

Lemma 4 *Let H be a pairwise-independent function family, and let Q be an $N \times N$ transition matrix where transitions correspond to reading ℓ random bits. For any $\varepsilon > 0$ we have:*

$$\Pr_{h \in H}[h \textit{ is not } \varepsilon\textit{-good for } Q] \leq \frac{N^6}{\varepsilon^2 2^\ell}\,.$$

Proof For $i, j \in [N]$ (corresponding to configurations in M_x), define

$$B_{i,j} \stackrel{\text{def}}{=} \{x \in \{0,1\}^\ell \mid x \text{ takes } Q \text{ from } i \text{ to } j\}.$$

For *fixed* i, j, k, we know from Lemma 3 that the probability that h is not $(\varepsilon/N^2, B_{i,j}, B_{j,k})$-good is at most $N^4 \rho(B_{i,j})/\varepsilon^2 2^\ell$. Applying a union bound over all N^3 triples $i, j, k \in [N]$, and noting that for any i we have $\sum_j \rho(B_{i,j}) = 1$, we have that h is $(\varepsilon/N^2, B_{i,j}, B_{j,k})$-good for *all* i, j, k except with probability at most $N^6/\varepsilon^2 2^\ell$.

We show that whenever h is $(\varepsilon/N^2, B_{i,j}, B_{j,k})$-good for all i, j, k, then h is ε-good for Q. Consider the (i,k)th entry in Q_h; this is given by: $\sum_{j \in [N]} \Pr[r \in B_{i,j} \wedge h(r) \in B_{j,k}]$. On the other hand, the (i,k)th entry in Q^2 is: $\sum_{j \in [N]} \rho(B_{i,j}) \cdot \rho(B_{j,k})$. Since h is $(\varepsilon/N^2, B_{i,j}, B_{j,k})$-good for every i, j, k, the absolute value of their difference is

$$\begin{aligned}
&\left| \sum_{j \in [N]} \Big(\Pr[r \in B_{i,j} \wedge h(r) \in B_{j,k}] - \rho(B_{i,j}) \cdot \rho(B_{j,k}) \Big) \right| \\
&\quad \leq \sum_{j \in [N]} \Big| \Pr[r \in B_{i,j} \wedge h(r) \in B_{j,k}] - \rho(B_{i,j}) \cdot \rho(B_{j,k}) \Big| \\
&\quad \leq \sum_{j \in [N]} \varepsilon/N^2 \;=\; \varepsilon/N.
\end{aligned}$$

It follows that $\mathsf{SD}(Q_h, Q^2) \leq \varepsilon/2$ as desired. ■

The lemma above gives us a pseudorandom generator that reduces the required randomness by (roughly) half. Specifically, define a pseudorandom generator $G_1 : \{0,1\}^{2\ell + R/2} \to \{0,1\}^R$ via:

$$G_1(r_1, \ldots, r_{R/2\ell}; h) = r_1 \,\|\, h(r_1) \,\|\, \cdots \,\|\, r_{R/2\ell} \,\|\, h(r_{R/2\ell}), \tag{1}$$

where $h \in H$ (so $|h| = 2\ell$) and $r_i \in \{0,1\}^\ell$. Assume h is ε-good for Q. Running M_x using the output of $G_1(h, \cdots)$ as the "random tape" generates the probability distribution

$$\overbrace{Q_h \cdots Q_h}^{R/2\ell} \cdot \mathbf{s}$$

for the final configuration, where $\mathbf{s}$ denotes the initial configuration of M_x (i.e., $\mathbf{s}$ is the probability distribution that places probability 1 on the initial configuration of M_x, and 0 elsewhere). Running M_x on a truly random tape generates the probability distribution

$$\overbrace{Q^2 \cdots Q^2}^{R/2\ell} \cdot \mathbf{s}$$

for the final configuration. Letting $k = R/2\ell$ we have

$$\begin{aligned}
2 \cdot \mathsf{SD}\left(\overbrace{Q_h \cdots Q_h}^{k} \cdot \mathbf{s}, \overbrace{Q^2 \cdots Q^2}^{k} \cdot \mathbf{s} \right) &= \left\| \left(\overbrace{Q_h \cdots Q_h}^{k} - \overbrace{Q^2 \cdots Q^2}^{k} \right) \cdot \mathbf{s} \right\| \\
&= \left\| \sum_{i=0}^{k-1} \left(\overbrace{Q_h \cdots Q_h}^{k-i} \overbrace{Q^2 \cdots Q^2}^{i} - \overbrace{Q_h \cdots Q_h}^{k-i-1} \overbrace{Q^2 \cdots Q^2}^{i+1} \right) \cdot \mathbf{s} \right\| \\
&\leq \sum_{i=0}^{k-1} \left\| \left(\overbrace{Q_h \cdots Q_h}^{k-i} \overbrace{Q^2 \cdots Q^2}^{i} - \overbrace{Q_h \cdots Q_h}^{k-i-1} \overbrace{Q^2 \cdots Q^2}^{i+1} \right) \cdot \mathbf{s} \right\| \\
&= \sum_{i=0}^{k-1} \left\| \overbrace{Q_h \cdots Q_h}^{k-i-1} \cdot (Q_h - Q^2) \cdot \overbrace{Q^2 \cdots Q^2}^{i} \cdot \mathbf{s} \right\| \\
&\leq k \cdot \varepsilon.
\end{aligned}$$

This means that the behavior of M_x when run using the output of the pseudorandom generator is very close to the behavior of M_x when run using a truly random tape: in particular, if $x \notin L$ then M_x in the former case accepts with probability at most

$$\Pr[\text{accepts} \wedge h \text{ is } \varepsilon\text{-good for } Q] + \Pr[h \text{ is not } \varepsilon\text{-good for } Q] \;\leq\; (1/4 + k\varepsilon/2) + N^6/\varepsilon^2 2^\ell;$$

similarly, if $x \in L$ then M_x in the former case accepts with probability at least $3/4 - k\varepsilon/2 - N^6/\varepsilon^2 2^\ell$. Summarizing (and slightly generalizing):

Corollary 5 *Let H be a pairwise-independent function family, let Q be an $N \times N$ transition matrix where transitions correspond to reading ℓ random bits, let $k > 0$ be an integer, and let $\varepsilon > 0$. Then except with probability at most $N^6/\varepsilon^2 2^\ell$ over choice of $h \in H$ we have:*

$$\mathsf{SD}\left(\overbrace{Q_h \cdots Q_h}^{k}, \overbrace{Q^2 \cdots Q^2}^{k} \right) \leq k\varepsilon/2.$$

2.2.2 Recursing

Fixing $h_1 \in H$, note that Q_{h_1} is a transition matrix and so we can apply Corollary 5 to *it* as well. Moreover, if Q uses R random bits then Q_{h_1} uses $R/2$ random bits (treating h_1 as fixed). Continuing in this way for $I \stackrel{\text{def}}{=} \log(R/2\ell) + 1 = \log(R/\ell)$ iterations, we obtain a transition matrix $Q_{h_1,\ldots,h_I}$. Say *all h_i are ε-good* if h_1 is ε-good for Q, and for each $i > 1$ it holds that h_i is ε-good for $Q_{h_1,\ldots,h_{i-1}}$. By Corollary 5 we have:

- All h_i are ε-good except with probability at most $N^6 I/\varepsilon^2 2^\ell$.
- If all h_i are ε-good then

$$\mathsf{SD}(Q_{h_1,\ldots,h_I}, \overbrace{Q^2 \cdots Q^2}^{R/2\ell}) \;\leq\; \frac{\varepsilon}{2} \cdot \sum_{i=1}^{I} \frac{R}{2^i \ell} \;=\; \frac{\varepsilon}{2} \cdot \left(\frac{R}{\ell} - 1\right).$$

Equivalently, we obtain a pseudorandom generator

$$G_I(r; h_1, \ldots, h_I) \stackrel{\text{def}}{=} G_{I-1}(r; h_1, \ldots, h_{I-1}) \,\|\, G_{I-1}(h_I(r); h_1, \ldots, h_{I-1}),$$

where G_1 is as in Equation (1).

2.2.3 Putting it All Together

We now easily obtain the desired derandomization. Recall $N = 2^{\mathcal{O}(s)}$. Set $\varepsilon = 2^{-S}/10$, and set $\ell = \Theta(S)$ so that $\frac{N^6 S}{\varepsilon^2 2^\ell} \leq 1/20$. Then the number of random bits used (as input to G_I from the previous section) is $\mathcal{O}(\ell \cdot \log(R/\ell) + \ell) = \mathcal{O}(S \log R)$ and the space used is bounded by that as well (using the fact that each $h \in H$ can be evaluated using space $\mathcal{O}(\ell) = \mathcal{O}(S)$). All h_i are good except with probability at most $N^6 \log(R/\ell)/\varepsilon^2 2^\ell \leq N^6 S/\varepsilon^2 2^\ell \leq 1/20$; assuming all h_i are good, the statistical difference between an execution of the original algorithm and the algorithm run with a pseudorandom tape is bounded by $2^{-S}/20 \cdot R \leq 1/20$. Theorem 1 follows easily.

3 $\mathcal{BPL} \subseteq \mathcal{SC}$

A deterministic algorithm using space $\mathcal{O}(\log^2 n)$ might potentially run for $2^{\mathcal{O}(\log^2 n)}$ steps; in fact, as described, the algorithm from the proof of Corollary 2 uses this much time. For the particular pseudorandom generator we have described, however, it is possible to do better. The key observation is that instead of just choosing the $h_1, \ldots, h_I$ at random and simply hoping that they are all ε-good, we will instead deterministically search for $h_1, \ldots, h_I$ which *are* each ε-good. This can be done in polynomial time (when $S = \mathcal{O}(\log n)$) because: (1) for a given transition matrix $Q_{h_1,\ldots,h_{i-1}}$ and candidate h_i, it is possible to determine in polynomial time and polylogarithmic space whether h_i is ε-good for $Q_{h_1,\ldots,h_{i-1}}$ (this relies on the fact that the number of configurations N is polynomial in n); (2) there are only a polynomial number of possibilities for each h_i (since $\ell = \Theta(S) = \mathcal{O}(\log n)$).

Once we have found the good $\{h_i\}$, we then cycle through all possible choices of the seed $r \in \{0,1\}^\ell$ and take majority (as before). Since there are a polynomial number of possible seeds, the algorithm as a whole runs in polynomial time.

(For completeness, we discuss the case of general $S = \Omega(\log n)$ assuming $R = 2^S$. Checking whether a particular h_i is ε-good requires time $2^{\mathcal{O}(S)}$. There are $2^{\mathcal{O}(S)}$ functions to search through at each stage, and $\mathcal{O}(S)$ stages altogether. Finally, once we obtain the good $\{h_i\}$ we must then enumerate through $2^{\mathcal{O}(S)}$ seeds. The end result is that $\mathsf{BPSPACE}(S) \subseteq \textsc{TimeSpc}(2^{\mathcal{O}(S)}, S^2)$.)

Bibliographic Notes

The results described here are due to [2, 3], both of which are very readable. See also [1, Lecture 16] for a slightly different presentation.

References

[1] O. Goldreich. Introduction to Complexity Theory (July 31, 1999).

[2] N. Nisan. Pseudorandom Generators for Space-Bounded Computation. *STOC* '90.

[3] N. Nisan. $RL \subseteq SC$. *Computational Complexity* 4: 1–11, 1994. (Preliminary version in *STOC* '92.)