CS 830: Randomness and Complexity 3/29/05

Lecture 19: Time-Space Tradeoff in Derandomizing BPL

Instructor: Jin-Yi Cai Scribe: Aparna Das, Giordano Fusco

# 1 Overview

In the present lecture and in the following two we are going to present these three results:

- Nisan in [Nis92] proved that $BPL \subseteq DTISP[n^{O(1)}, O(\log^2 n)]$.
- Saks and Zhou in [SZ99] proved that $BPL \subseteq DTISP[n^{O(\log^{0.5} n)}, O(\log^{1.5} n)]$.
- Cai, Chakaravarthy and van Melkebeek in [CCM04] gave a time-space tradeoff between the previous two: $BPL \subseteq DTISP[n^{O(\log^{0.5-\alpha} n)}, O(\log^{1.5+\alpha} n)]$, where $0 \leq \alpha \leq 0.5$.

We are going to start by giving some definitions and describing the instruments we will need. In the current lecture we will concentrate on restating Nissan's result in terms of these new definitions. We will show the other results in the next lectures.

# 2 Basic Definitions

We start by defining the $(d, m)$-automaton which will be a key tool in our simulation of BPL.

Definition 2.1. *For positive integers $d$ and $m$, a $(d, m)$-automaton $Q$ is a mapping*

$$Q : \{0, 1, \dots, d\} \times \{0, 1\}^m \to \{0, 1, \dots, d\},$$

*such that $Q(0, \alpha) = 0$, $\forall \alpha \in \{0, 1\}^m$.*
*$Q$ has $d + 1$ states numbered $0, 1, 2, \dots, d$. The state $0$ has a special purpose and it is called* dead state.

Later on we will clarify why we need to define a dead state.

We can extend the mapping $Q$ to a mapping $\widehat{Q}$ in the following way

$$\widehat{Q} : \{0, 1, \dots, d\} \times \{0, 1\}^m \to \{0, 1, \dots, d\}$$

where

$$\widehat{Q}(i, \epsilon) = i, \quad \text{and } \epsilon \text{ is the empty string}$$

$$\widehat{Q}(i, \alpha_1 \dots \alpha_t) = Q(\widehat{Q}(i, \alpha_1 \dots \alpha_{t-1}), \alpha_t)$$

Definition 2.2. *For a $d \times d$ matrix $M$ over the real numbers, we define the norm of $M$ as*

$$||M|| = \max_{1 \le i \le d} \left[ \sum_{j=1}^{d} |m_{ij}| \right]$$

*We say that $M$ is* sub-stochastic *if $m_{ij} \ge 0$, $\forall i, j \in \{1, 2, \dots, d\}$, and $||M|| \le 1$. Such a matrix is called a $(d, m)$*-matrix *if $2^m m_{ij}$ is an integer, $\forall i, j \in \{1, 2, \dots, d\}$.*

Definition 2.3. *Let $Q$ be a $(d, m)$-automaton. The transition matrix of $Q$, denoted by $\mathcal{M}(Q)$, is the $(d, m)$-matrix $M$ s.t., $\forall i, j \in \{1, 2, \dots, d\}$,*

$$m_{ij} = \Pr_{\alpha \in \{0,1\}^m} [\widehat{Q}(i, \alpha) = j] \tag{1}$$

We can interpret $M^k$ as "moving" $k$ steps. The intuitive meaning of the previous definition is that given an automaton we can construct a sub-stochastic matrix corresponding to it. We are now going to show the opposite: we start from a sub-stochastic matrix and we show that there is an automaton corresponding to it. In general this operation is not unique, but we are going to define a canonical way of doing it proceeding in lexicographical order.

Definition 2.4. *Let $M$ be a $(d, m)$-matrix. The canonical automaton corresponding to $M$, denoted by $\mathcal{Q}(M)$, is a $(d, m)$-automaton $Q$ defined as follows. Let $i \in \{1, 2, \dots, d\}$ and $\alpha \in \{0, 1\}^m$. Let $\underline{\alpha}$ denote the number with binary representation $.\alpha$. If $\underline{\alpha} > \sum_{j=1}^{d} m_{ij}$, then $Q(i, \alpha) = 0$; otherwise, let $k \in \{1, 2, \dots, d\}$ be the least integer such that $\underline{\alpha} \le \sum_{j=1}^{k} m_{ij}$, then we let $Q(i, \alpha) = k$. An automaton $Q$ is called canonical if it is the canonical automaton corresponding to some $M$.*

Note that $\forall$ $(d, m)$-matrix $M$, $\mathcal{M}(Q(M)) = M$. Moreover if we start from a canonical automaton $Q$, then we have $\mathcal{Q}(\mathcal{M}(Q)) = Q$.

# 3 Estimating the probability of accepting

We now describe Nissan's main idea. Suppose we are given the description of a finite automaton. Given an input $w$ we would like to determine both the function $Q$ and the transition matrix $M = \mathcal{M}(Q)$.
We associate the following meanings to the states: 1 is the starting state and $d$ is the unique (absorbing) state. Suppose that the machine runs for $p(n)$ steps. By equation (1) we know that the acceptance probability of the machine on input $w$ is given by $(M^{p(n)})_{1,d}$ (i.e. the $(1, d)$ entry of the matrix $(M^{p(n)})$). We also know that this entry is not close to $1/2$, so it is either $> 2/3$, or $< 1/3$. We would like to approximate $(M^{p(n)})_{1,d}$, and it is enough to do it within $1/6$. Since $d$ is an absorbing state $(M^{p(n)})_{1,d} = (M^r)_{1,d}$, for all $r \ge p(n)$. In particular we choose $r = O(\log d)$.
The standard way to compute this is via repeated squaring

$$M \to M^2 \to M^4 \to M^8 \to \dots$$

until we find the approximation of $(M^{2^r})_{id}$ which is equal to $(M^{p(n)})_{id}$. This process requires only $\log n$ space.
A recursive implementation of doing this as in Savitch's theorem, would require $O(\log^2 d)$ space, since we would need $O(\log n)$ recursive levels and use $O(\log d)$ space per level. Nissan's simulation of $BPL$ can be viewed as performing repeated approximate squaring without the need to store intermediary results. Below we restate Nissan's results in terms of automaton $Q$.

## 4 Alternative view of Nissan's Pseudorandom Generator

Before giving a formal definition of Nissan's generator in the context of $Q$ and $M$ defined above, we consider the following example:
Suppose we want to compute $M^2_{ik}$, which is $\Pr[Q$ moves from $i$ to $k$ in 2 steps$]$. Using the definitions from above we have that:

$$\begin{aligned} \Pr[Q \text{ moves from } i \to k \text{ in 2 steps}] &= \sum_j \Pr_{\alpha,\beta \in \{0,1\}^m}[Q(i,\alpha) = j \wedge Q(j,\beta) = k] \quad (2) \\ &= \sum_j M_{ij} M_{jk} \end{aligned}$$

Using Nissan's generator, the $\Pr[Q$ moves from $i$ to $k$ in 2 steps$]$ would be approximated by

$$\Pr_{\alpha \in \{0,1\}^m}[Q(i,\alpha) = j \wedge Q(j, h(\alpha)) = k] \quad (3)$$

for some "good" hash function $h$. Now we formally define Nissan's generator on $Q$ as follows:

Definition 4.1. *Let $Q$ be a $(d,m)$-automaton and $h : \{0,1\}^m \to \{0,1\}^m$ a hash function. Define $Q_h$ (also denoted $N_h(Q)$) as the function, $Q_h : \{0,1,\ldots,d\} \times \{0,1\}^m \to \{0,1,\ldots d\}$ where:*

$$Q_h(i,\alpha) = \widehat{Q}(i, \alpha \circ h(\alpha))$$

Thus $\mathcal{M}(Q)^2$ corresponds to running $Q$ on randomly picked $\alpha, \beta \in \{0,1\}^n$, where as $\mathcal{M}(Q_h)$ corresponds to running $Q$ on $\alpha \circ h(\alpha)$. We say that:

Definition 4.2. *$h$ is $\epsilon$-random, where $\epsilon > 0$, for $Q$ if*

$$||\mathcal{M}(Q_h) - \mathcal{M}(Q)^2|| < \epsilon$$

We have proved a general version of the following key lemma in an earlier lecture.

Lemma 4.3. *If $Q$ is as above, $H_m$ universal hash family and $\epsilon > 0$,*

$$\Pr_{h \in \mathcal{H}_m}[h \textit{ is not } \epsilon\textit{-random for } Q] \leq \frac{d^5}{\epsilon^2 \cdot 2^n}$$

Note that we can perform an exhaustive search to find $\epsilon$-random $h$ using space $O(m + \log d)$. This can be done by iterating through all $\alpha, \beta$ and each pair of states $i, k$, and checking the sums from equations (2) and (3) to ensure that $||\mathcal{M}(Q_h) - \mathcal{M}(Q)^2|| < \epsilon$. The space bound of $O(m + \log d)$ follows since there are $2^m$ hash functions to try, and $\alpha, \beta \in \{0,1\}^m$ and there are $d$ states.
The lemma above gives us a way the following way to approximate $M^2$:

1. Construct the canonical $(d,m)$-automaton $Q = \mathcal{Q}(M)$,
2. find a hash $h \in \mathcal{H}_m$ that is $\epsilon$-random for $Q$,
3. construct $Q_h$ and transform it into its transition matrix $\mathcal{M}(Q_h)$.

We would like to apply this procedure $r$ times successively to approximate $M^{2^r}$. The following defines notation for intermediary automata.

Definition 4.4. *Let $Q$ be a $(d,m)$-automaton. Let $h_1,\dots,h_r$ be hash functions from $\{0,1\}^m \to \{0,1\}^m$. Then,*

$$Q_{h_1,\dots h_r} = (Q_{h_1,\dots h_{r-1}})_{h_r}$$

We would also like describe how the error accumulates from applying the procedure $r$ times.

Definition 4.5. *Let $Q$ be a $(d,m)$-automaton and $h_1,h_2,\dots,h_r$ be hash functions over $\{0,1\}^m$. The sequence $h_1,h_2,\dots,h_r$ is $\epsilon$-random for $Q$, if*

$$||M(Q_{h_1,h_2,\dots,h_r}) - (M(Q))^{2^r}|| < \epsilon$$

Lemma 4.6. *Let $Q$ be a $(d,m)$-automaton. If $\forall i, 1 \le i \le r$, $h_i$ is $\epsilon$-random for $Q_{h_1,\dots,h_{i-1}}$ then $h_1,h_2,\dots,h_r$ is $2^r \cdot \epsilon$ for $Q$.*

*Proof.*

The proof is by induction. The proof is easier if $2^r$ is replaced by $2^r - 1$, so we will show that:

$$||\mathcal{M}(Q_{h_1,\dots,h_{r-1}}) - \mathcal{M}(Q)^{2^r}|| < (2^r - 1)\epsilon$$

The base case ($r = 1$) is given by lemma 4.3.

By the inductive hypothesis we have that $||\mathcal{M}(Q_{h_1,\dots,h_{r-1}}) - \mathcal{M}(Q)^{2^{r-1}}|| < (2^{r-1} - 1)\epsilon$. For ease of notation let:

$$\begin{aligned} M &= \mathcal{M}(Q_{h_1,\dots,h_r}) \\ N &= \mathcal{M}(Q)^{2^r} \\ L &= \mathcal{M}(Q_{h_1,\dots,h_{r-1}}) \\ K &= \mathcal{M}(Q)^{2^{r-1}} \end{aligned}$$

Restating our task in terms of this notation: given $||L - K|| < (2^{r-1} - 1)\epsilon$ we want to show that $||M - N|| < (2^r - 1)\epsilon$

Substituting $N = K^2$ we have that $M - N = M - L^2 + L^2 - K^2$. We have that:

$$\begin{aligned} ||M - N|| &= ||M - L^2 + L^2 - K^2|| \\ &\le ||M - L^2|| + ||L^2 - K^2|| && \text{by triangle inequality} \\ &< \epsilon + ||L^2 - LK + LK - K^2|| \\ &\le \epsilon + ||L^2 - LK|| + ||LK - K^2|| && \text{by triangle inequality} \\ &\le \epsilon + ||L|| \cdot ||L - K|| + ||L - K|| \cdot ||K|| && \text{by the definition (2.2)} \\ &\le \epsilon + 2(2^{r-1} - 1)\epsilon && \text{since } L \text{ and } K \text{ are sub-stochastic} \\ &= (2^r - 1)\epsilon \end{aligned}$$

To obtain the third inequality note that $h_r$ is $\epsilon$-random for $Q_{h_1},\dots,h_{r-1}$ (by assumption of this lemma), so by applying definition (4.2) we get that $||M - L^2|| < \epsilon$. ☐

Recall that Nissan's generator was defined recursively as:

Definition 4.7. *Let $h_1,h_2,\dots,h_r$ be hash function over $\{0,1\}^m$. Recursively define $G_{h_1,h_2,\dots,h_r}$ as a mapping from $\{0,1\}^m$ to $(\{0,1\}^m)^{2^r}$ by*

$$G(\alpha) = \alpha$$

$$G_{h_1,h_2,\dots,h_i}(\alpha) = G_{h_1,h_2,\dots,h_{i-1}}(h_i(\alpha))$$

A recursive implementation of Nissan's algorithm would use roughly $O(\log^2 d)$ space. The crucial property below allows Nissan to do better.

PROPOSITION 4.8. *Let $Q$ be a $(d, m)$-automaton and $h_1, h_2, \ldots, h_r$ be hash functions over $\{0,1\}^m$. Then for any $i \in \{0, 1, \ldots, d\}$ and $\alpha \in \{0,1\}^m$,*

$$Q_{h_1,h_2,\ldots,h_r}(i, \alpha) = \widehat{Q}(i, G_{h_1,h_2,\ldots,h_r(\alpha)})$$

Although the proposition above looks innocuous, it provides the key shortcut for storing the intermediate results in the recursive computation of $Q_{h_1,h_2,\ldots,h_r}$. Given $h_1, h_2, \ldots h_r$, $i \in \{0, 1, \ldots d\}$ and $\alpha \in \{0,1\}^m$, it allows us to compute $Q_{h_1,\ldots h_r}(i, \alpha)$ in space $O(m + r + \log d)$. This follows since we need $O(m)$ storage for the intermediary computations of $h_i(\alpha)$, $r$ bits to indicate which $h_i$ functions to apply, and $\log d$ bits to store state $i$. Since all computations can be done using $O(m + r + \log d)$ workspace, the time for this computation is at most $2^{O(m+r \log d)}$. The only additional space needed is $O(rm)$ bits to store the hash functions. Since $m, r = O(\log d)$, this establishes Nissan's result that $BPL \subseteq DTISP[n^{O(1)}, O(\log^n)]$.

## References


[CCM04] Jin-Yi Cai, Venkatesan T. Chakaravarthy, Dieter van Melkebeek, *Time-Space Tradeoff in Derandomizing Probabilistic Logspace*, STACS 2004, LNCS 2996, pp. 571–583, 2004.

[Nis92] Noam Nisan, *RL* $\subseteq$ *SC*, STOC 1992, pp. 619–623, 1992.

[SZ99] Michael E. Saks, Shiyu Zhou, *$BP_H SPACE(S) \subseteq DSPACE(S^{3/2})$*, J. Comput. Syst. Sci., 58(2):376–403, 1999.