CS 830: Randomness and Complexity 03/31/05

Lecture 20: Saks-Zhou Algorithm

instructor: Jin-Yi Cai Scribe: Anand Kumar Sinha, Vinay Choudhary

1 Introduction

In the last lecture we finished Nisan's construction of a pseudorandom generator that fools space bounded computation. As a consequence we obtained BPL $\subseteq$ DTISP$(n^{O(1)}, O(\log^2 n))$. In this lecture we'll see how Saks ans Zhou improve upon Nisan's idea to obtain BPL $\subseteq$ DTISP$(2^{\log^{3/2} n}, O(\log^{3/2} n))$. We'll give an informal presentation of the idea and will define Saks-Zhou rounding. The proofs will come in the next lecture.

2 Saks-Zhou Idea

As we saw in the last lecture that the following property is what enabled Nisan to save on time.

Lemma 2.1. *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 $\alpha \in \{0,1\}^m$,*

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

We also noted that we can save on work space too, which is only $O(\log n)$ but everything is spoiled because we need to store $O(\log n)$ hash functions each occupying $O(\log n)$ bits. So a natural approach to save space is to use same hash functions multiple number of times. Ideally we hope that everything else remains the same and since we need to store less hash functions, we gain on space. We needed the following lemma in our proof.

Lemma 2.2. *Let Q be a (d, m)-automaton, and $\epsilon > 0$. Let $h_1, h_2, \ldots, h_r$ be hash functions over $\{0,1\}^m$ such that, for each $1 \le i \le r$, h_i is ϵ-pseudorandom for $Q_{h_1,h_2,\ldots,h_{i-1}}$. Then the sequence $h_1, h_2, \ldots, h_r$ is $2^r\epsilon$-pseudorandom for Q.*

Since we were choosing $h_1, \ldots, h_r$ independently, we were able to simply use the union bound. The probability that h_i is good for $Q_{h_1,h_2,\ldots,h_{i-1}}$ is high because $Q_{h_1,h_2,\ldots,h_{i-1}}$ is independent of h_i. This probability is same that of a random hash function h being good for a fixed Q. But if we want to use the same hash function, say h_1, again; we need that the probability of h_1 being good for $Q_{h_1,h_2,\ldots,h_{i-1}}$ is large. But we cannot claim that as $Q_{h_1,h_2,\ldots,h_{i-1}}$ depends on h_1 and there is no reason for this probability to be large. It could well be that for most h_1, they are not good for $Q_{h_1,h_2,\ldots,h_{i-1}}$.

One way to tackle this problem is to use a fixed automaton at each level. Suppose we have hash functions $h_1, h_2, \ldots, h_i$ which are good for Q. Now we would like to use h_1 again. I.e. we would like to have an h_1 which is good for $Q_{h_1,h_2,\ldots,h_i}$. If we were guaranteed the existence of a fixed automaton Q^* such that $Q_h = Q^*$ for most choices of h, then we could use $Q_{i+1} = Q_i^*$, and safely

use the same hash function h_1 again. Unfortunately, we only know that Q_h is "close" to some fixed Q^* for most h and not that it coincides with Q^* for most h.
Saks and Zhou devised a randomized rounding scheme to make the automata at every level of recursion independent of the hash functions at the previous levels (for most choices of these hash functions), namely by randomly perturbing and then truncating the entries of the transition matrices. We will now discuss the Saks-Zhou rounding scheme.

3 Saks-Zhou Rounding

The motivation for rounding is that for most h, Q^2 is close to Q_h. So it is reasonable to expect that if we round Q_h to sufficiently small number of places, we would get the same value for most h.

DEFINITION 3.1. *Let t, D be positive integers, and $\delta \in \{0,1\}^D$. We define the rounding operator $\mathcal{R}_{\delta,t}$ as the function mapping $z \in [0,1]$ to $\lfloor \max(z - \underline{\delta} 2^{-t}, 0) \rfloor_t$, where $\underline{\delta}$ denotes the number with binary representation $.\delta$, and $\lfloor z \rfloor_t = \lfloor 2^t z \rfloor 2^{-t}$. The operator extends to real-valued matrices with each entry in $[0,1]$ by entry-wise application.*

The effect of $\mathcal{R}_{\delta,t}$ is to perturb the binary expansion of every number after the t-th bit by δ (obtaining a smaller number but never going negative), and then truncate it after the t-th bit. For detailed explanation and figures, please refer to the handout.
The storage space consists of two components - $O(m)$ bits for the hash function h, and $O(r \log d)$ bits for all the perturbations. In order to balance the two components of the space complexity, Saks and Zhou considered a hybrid algorithm that uses r_1 hash functions $\vec{h} = (h_1, h_2, \ldots, h_{r_1})$ cyclically r_2 times, where $r_1 r_2 = r$, and applies the rounding between cycles.
But there is one unfortunate side-effect of the rounding scheme - there is no analogue of lemma (**??**). We can no longer circumvent the recursion because we need to act on the matrices, and therefore have to transform between automata and matrices at every level of recursion. As a consequence, the processing space increases from $O(m + r + \log d)$ to $O(r(m + \log d))$ again. In our new notation, the processing space becomes $O(r_2(m + \log d))$ and the storage space $O(r_1 m + r_2 \log d)$. By setting $r_1 = r_2 = \sqrt{r}$, Saks and Zhou obtained their result that $\mathrm{BPL} \subseteq \mathrm{DTISP}[n^{O(\log^{0.5} n)}, O(\log^{1.5} n)]$.
We'll present the proof in the next lecture.