LOWER BOUNDS ON THE TIME OF PROBABILISTIC ON-LINE SIMULATIONS

(preliminary version)

Ramamohan Paturi and Janos Simon

Department of Computer Science
The Pennsylvania State University

Abstract

We study probabilistic on-line simulators for several machine models (or memory structures). The simulators have a more constrained access to data than the virtual machines, but are allowed to use probabilistic means to improve average access time. We show that in many cases coin tosses can not make up for inadequate access.

1. Introduction

While it seems impossible at present to derive non-trivial lower bounds that would show the differences that we believe exist between machine models, several such bounds were obtained for the special case of on-line computations [H], [G], [PSS], [Pi], [P], [DGPR]. In particular, it is widely believed that nothing can make up for inadequate access to storage: more heads, tapes, or higher dimensional medium should yield better computers.

The problem has been studied with the following model: there is an input tape (one-way, read only) of commands, an (one-way, write only) output tape, a finite control, and one or more storage modules. Access to the storage module is by means of one or more read/write heads, that can be shifted according to certain functions. An example is two-dimensional tape, where the shift functions consist of moving left, right, up and down (with the usual relations: the shifts commute, up and down are inverses of each other, etc.). A command is of the form \((h,d,a)\) and its effect is to shift the head \(h\) in the direction \(d\) and perform the action \(a\) where \(a \in \{\text{print 0}, \text{print 1}, \text{NOP(do nothing)}\}\). Before the shift, the character stored at the position of the head to be shifted is written on the output tape. Given such a machine \(V\) (that we will call the virtual machine), a different model \(S\) is said to simulate \(V\) on line, if, given any sequence of commands on the input tape, \(S\) produces the same sequence of characters on the output tape as \(V\) does. Note that both machines are symbol by symbol transducers. The simulation time \(T(n)\) of \(S\) is the number of moves of \(S\) sufficient to process any sequence of \(n\) commands to \(V\). If the simulating machine is probabilistic (in which case, we denote it by \(PS\)), it flips an unbiased coin to determine its next move and we define \(T(n)\) to be the worst (over inputs of length \(n\)) expected (over coin tosses) time.

The following is known for deterministic on-line simulations:

- to simulate a \(d\)-dimensional storage unit by a \(d'\)-dimensional one \((d>d')\) a lower bound of \(\Omega(n^{\frac{1}{d'-d}})\) holds ('Hennie's theorem' [H], [Gr], [PSS]). The bound is fairly tight.

- to simulate \(h\) heads by \(h'\) heads \((h>h')\), on \(d\)-dimensional tapes, \(d>1\), \(\Omega(n^e)\) moves are necessary, where \(e\) is positive, and depends on \(d\) ('more heads are better'). This is true even if the simulator has tapes of higher dimension than \(d\), as long as the difference in dimensions is not too big [PSS]. This bound is also fairly tight.

- to simulate \(k+1\) (1-dimensional) tapes by \(k\) tapes, \(\Omega(n\log\frac{k+1}{k})\) steps are necessary ('Aanderaa's theorem' [A], [PSS], [Pi]). This is true even if one wants to simulate \(k\) pushdowns [DGPR].

Probabilistic simulations were studied by Pippenger [Pi], who showed that Hennie's theorem could be extended to probabilistic simulators.

Our results include

Theorem 1: On-line simulation of \(h\)-head \(d\)-dimensional Turing machines by probabilistic \(h'\)-head \(d\)-dimensional ones takes time \(\Omega(n^{\frac{1}{d'-d}})\), where \(e\) is positive and depends on \(h, h', d \geq 2\). This extends the result of Paul, Seifer and Simon on insufficient number of heads [PSS] to probabilistic simulators.

Theorem 2: On-line simulation of \(k+1\) tape Turing machines by probabilistic \(k\) tape Turing machines takes \(\frac{1}{k}\) time \(\Omega(n\log\frac{k+1}{k})\). This is the generalization of Aanderaa's theorem, as strengthened by Paul [P].

Theorem 3: On-line simulation of \(d\)-dimensional iterative arrays with central control \([S]\) by \(d'\)-dimensional probabilistic ones (distributed coin-tossing) requires \(\Omega(n^{\frac{1}{d'-d}})\) for \(d>d'\). This extends previous results of Hennie [H], [Gr], [PSS] and Pippenger [Pi].

and confirm the intuition that nothing can compensate for inadequate access to information in memory units. Because of space limitations, we include only the proofs for the first two theorems.

Our proofs use Kolmogorov complexity arguments, adapted to the stochastic setting, and two techniques to handle probabilistic computations. First, we prove a lemma (lemma 1) that lets us talk about incompressible

\(^1\)Research partially supported by US ARO Contract DAAG 28-82-K-0110 and by NSF Grant MCS81-04676.
strings with respect to the majority of computations of a probabilistic machine.

To explain our techniques, we need to review the outline of the previous proofs. The main idea was to record in the virtual memory unit a long incompressible string, that has the property that certain substrings of it are also incompressible. At the end of this writing phase, the virtual machine may retrieve, efficiently, some random substrings. Since these substrings are random, relatively long strings are needed to specify them. If a simulator is to be efficient, the specification for the string to be output must be located in the vicinity of the simulator's heads. The lower bounds result from arguments showing that, because of the relative access deficiency, not all such specifications can be near the simulator's heads if the simulator did not have the opportunity to do extensive preprocessing. So, given a simulator, the virtual machine can force it to be inefficient either by retrieving incompressible data located far from the simulator's heads (such data must exist if the simulator did not do preprocessing), or the simulator is already inefficient, since it performed time-consuming preprocessing. The strategy is repeated again and again, retrieving eventually enough of the original string so that the number of steps that had to be simulated inefficiently by the simulator constitute at least a constant fraction of the total number of instructions.

There are two difficulties in extending these techniques to probabilistic simulators. The first one is that the probabilistic machine may use stochastic methods to decrease the Kolmogorov complexity of some strings. A simple argument (lemma 1) shows that this will occur very seldom. The second, more important difficulty is that, if we try to mimic the strategy above, instead of having to fool a single simulator in a given configuration (for which we can find a hard query, as a function of this configuration), we will have to deal with an ensemble of configurations, since the simulator may have moved its heads in different patterns, used different encoding schemes etc., depending on the outcome of the coin tosses during the previous steps.

We use two different techniques to overcome the second difficulty. In some of the deterministic results, the counting argument that shows the existence of hard queries is very strong - in fact an overwhelming fraction of all queries is hard (for example in the case of Hennie's theorem). But, it may be difficult to find a single hard input so that a sufficient number of computations of PS will spend a lot of time to make the average high. Since most queries are hard to simulate, it may be easier to find a hard distribution on inputs. First, based on an idea of Pippenger [P], we use an argument equivalent to the easy half of von Neumann's minimax theorem: the worst (over inputs) case time complexity of a probabilistic simulator is bounded below by the average (with respect to arbitrary input distributions) complexity of the same probabilistic simulator. We select an uniform distribution on queries and a query selected at random will be difficult with a high probability, since most queries are difficult for the simulator. In addition, we apply lemma 1 to show that most guess strings are not helpful in reducing the query time. This is the idea underlying the proof of theorem 1 (simulation with fewer heads). The technique also yields Pippenger's probabilistic result - we believe with a somewhat simpler proof than the one using entropy [P].

Our second technique, used to prove the probabilistic version of Aanderaa's theorem is more delicate. The counting argument is more complex and does not show that a random query is difficult. The deterministic proof only shows that if \( V \) reads different incompressible strings on its tapes at very different rates, the simulator will 'neglect' a tape, except if the simulator spent enough time preprocessing (i.e. the computation up to this point has high overlap). A potential query about the neglected tape will be hard. Our proof uses the same basic strategy. Again, \( V \) will read incompressible strings at different rates and stores them on its tapes. After a certain initial segment of the input is consumed by the probabilistic simulator \( PS \), the adversary considers the collection of all computations of \( PS \). An input interval \( I \) will be selected and if, for a large enough set of these computations the overlap in \( I \) is low, then we show that a fixed tape must also be neglected by a sufficiently large fraction of the computations of \( PS \). We can now make inquiries about (the contents of) this tape to degrade the performance of the simulator and we show that we succeed in doing so. Here, we use our lemma 1 to show that sufficiently incompressible strings remain sufficiently incompressible even in the presence of a large enough fraction of the guess strings. In the other case, when a large fraction of the computations of \( PS \) has high enough overlap in the input interval \( I \), the adversary makes no inquiries. A new input interval is selected next and the adversary repeats this process until the collection of input intervals is exhausted. It can now be shown that, if few queries are made, the average time used by the simulator is already high (because of the high overlap in many computations), and otherwise each query will be time consuming in a significant fraction of the computations and thus the average time will again be high.

Section 2 contains the necessary definitions, facts and the proof of the lemma 1. Sections 3, and 4 contain the proofs of theorems 1, and 2 respectively.

2. Definitions and Facts

This section contains some definitions of and facts about machines, Kolmogorov complexity, and probabilistic simulations. A proof of lemma 1 is also given here.

Machines: We assume the reader is familiar with multi-tape, multihead Turing machines with \( d \)-dimensional tapes. Our machines have a one-way read-only input tape, and a one-way write-only output tape. The machines operate on-line: the \( n \)-th output symbol must be written before the \( n+1 \)-st input symbol can be read. A probabilistic machine has an unbiased coin, and the next move function also depends on the outcome of the last coin toss. The concatenation of the outcome of the coin tosses during a computation, representing heads by 1's and tails by 0's is called a guess string. The probability of a computation with a guess string of length \( p \) is \( 2^{-p} \). We will use the notation \( T(u,r) \) to denote the time required by the simulator to process \( r \) input commands after an initial command sequence \( u \). \( T(n) \) is the time necessary to process the first \( n \) input symbols. For a probabilistic simulator, time means the worst (over inputs of length \( n \)) expected time, given the probabilities associated with the guess strings.

Kolmogorov complexity: Given strings \( x,y \in \{0,1\}^* \) the Kolmogorov complexity, \( K(x|y) \) of \( x \) given \( y \) is the length of the shortest string \( z \) such that for a certain fixed universal Turing machine \( U \), \( U(z\#y)(U \) given \( z \) on
the input tape and \( y \) on a work tape) outputs \( x \) and halts. \( K(x) = K(x| y) \) with \( y \) = empty string.

Given a string \( w \), let \(|w|\) denote the length of \( w \), \( \overline{w} \) the string obtained by replacing each letter \( a \) of \( w \) by \( \overline{a} \), and let \( w' \), the self-delimiting version of \( w \) be \( bin(|w|)0\overline{w} \), where \( bin(n) \) denotes the binary representation of the integer \( n \). The Kolmogorov complexity of a sequence of strings is defined as the complexity of the concatenation of their self-delimiting versions.

A string \( w \) is random or incompressible if \( K(w) \geq |w| \). Sometimes, we abuse the concept by referring informally to 'almost incompressible' strings. By a simple counting argument one can show that random strings of any length exist, and that sufficiently long substrings of a random string are almost random. More precisely, if \( u \) is a substring of a random string \( w \), \( K(u) \geq |u| - O(\log|w|) \).

**Probabilistic Simulations:** Let \( z \) be an input string of length \( n \). Let \( PS(g)[z] \) denote the computation of the probabilistic simulator \( PS \) on input \( z \) with \( g \) as its guess string and, \( T_{z,g} \) denote the number of steps in this computation. Then, by definition,

\[
T(n) = \max_{z,g} \text{ave} T_{z,g}
\]

where \( g \) is the usual probability distribution associated with the guess strings, and the maximum is over all input strings of length \( n \).

Let \( p \) be an arbitrary probability distribution on inputs of length \( n \). Then

\[
T(n) = \max_{z,g} \text{ave} T_{z,g}
\]

\[
\geq \text{ave} \text{ave} T_{z,g}
\]

\[
= \text{ave} \text{ave} T_{z,g}
\]

This inequality is helpful when it is difficult to find a single hard input, since it lets us work instead with some input distribution. This is the easy part of von Neumann's minimax theorem.

**Strategy used in Proofs:**

In all our theorems, the basic strategy used in proving a lower bound \( T'(n) \) on the time \( T(n) \) of a probabilistic simulator \( PS \) is as follows:

We restrict ourselves to the set \( G \) of all guess strings of length \( c T'(n) \) for a sufficiently large constant \( c > 0 \). This does not create any problem since we are dealing with lower bounds.

We then prove that only a vanishingly small fraction of guess strings in \( G \) give enough information about a sufficiently incompressible string (lemma 1).

Finally, we apply one of the techniques described in section 1 to yield the desired lower bound.

**Counting Lemma:** We now present the lemma that counts the number of guess strings of a given length that give 'enough' information about an almost random string. It is the basis for adapting the tools used to obtain the lower bounds on the running time of a deterministic simulator to lower bounds on a significant fraction of the total number of computations of a probabilistic simulator.

Let \( G \) be the set of guess strings of length \( t \) (\( G = \{0,1\}^t \}). Let \( x,y \) be strings. Define

\[
G(x:z) = \{ g \in G | K(x|g|y) < \lambda \}
\]

Remember that \( g' \) is the self-delimiting version of \( g \). \( G(x:z) \) is the set of guess strings that allow descriptions of \( x \) of length smaller than \( \lambda \) given \( y \).

**Lemma:** \( \text{ave ave} T_{z,g} \geq \frac{L}{2} + O(\log|x|) \), where

\[
L = t + \lambda - \frac{K(x|y)}{2} + O(\log|x|)
\]

provided \( \log t = O(\log|x|) \).

**Proof:** We argue by contradiction, showing that if \( |G(x:z)| > 2^L \), then it is possible to give a description of \( x \) given \( y \) that is shorter than \( K(x|y) \).

Let \( G(x:z) > 2^L \) and let \( W \) be the set of strings of length at most \( \lambda \). For each \( v \in \{0,1\}^\lambda \) define \( G_v = \{ wg | w \in W, g \in G \} \) and \( U(w|gg|y) \) outputs \( v \) and halts.

Note that for distinct \( v \)'s, the corresponding \( G_v \)'s are disjoint and \( |G_v| > 2^t \), since \( |G(x:z)| > L \). Let \( H \) be the set of all \( v \in \{0,1\}^\lambda \) such that \( |G_v| > 2^L \), and let \( h = |H| \). It is possible to determine \( H \), given \( h,t,\lambda,|x| \), by simulating \( U \) on \( w|gg|y \) for each \( w \in W \) and \( g \in G \). (\( h \) is necessary since some of the computations may not halt.)

Since the \( G_v \)'s are pairwise disjoint, \( h \geq |G|/2^L \).

Given \( H, x \) can be specified by giving its rank in the standard enumeration of \( H \), using an additional \( \log h \) bits - i.e. \( \log(\frac{|G|}{2^L}) \) bits. Thus we have

\[
K(x|y) < \log t + O(\log\log t) + \log \lambda
\]

\[
+ O(\log\log\lambda) + \log x + O(\log\log x)
\]

\[
/ \text{to specify } t, \lambda \text{ and } x
\]

\[
+ \log |G| + \log |W| - L + O(\log(\frac{|G||W|}{2^L}))
\]

\[
/ \text{to specify } h
\]

\[
+ \log |G| + \log |W| - L + O(\log(\frac{|G||W|}{2^L}))
\]

\[
/ \text{to specify the rank of } x \text{ in } H
\]

\[
+ O(1) / \text{ *explanation *}
\]

Therefore,

\[
L < t + \lambda - \frac{K(x|y)}{2} + O(\log|x|)
\]

since \( \log t = O(\log|x|) \) and this is the desired contradiction.

3. Probabilistic Simulation of Multidimensional Tape Turing Machines

Let \( PS \) be a probabilistic simulator (Turing machine) with \( h \) head \( d \)-dimensional tape unit which simulates an \( h \)-head (\( h > h' \)) \( d \)-dimensional storage unit.
Let $T(u;r)$ (respectively, $T_q(u;r)$) be the number of steps required by $PS$ (respectively, $PS'(g)$) in the worst case to handle a sequence of $r$ commands following the initial command sequence $u$. aue $T_q(u;r)$ is the average time over input command sequences of length $r$ with some probability distribution $p$. We say that two command sequences $u$, and $u'$ are equivalent if, for any command sequence $w$, inputs $uw$, and $uw'$ produce the same output corresponding to the input portion $w$.

We prove the following theorem in this section.

**Theorem 1:** For $h' < h$, suppose an $h'$ head $d$-dimensional probabilistic Turing machine can simulate an $h'$ head $d$-dimensional storage unit $M$ in time $T(n)$. For each sufficiently large $n$,

$$T(n) = \Omega(n^{1+\epsilon}), \quad \epsilon \leq \frac{(d-1)(h^2-h')}{dh+h-h'}$$

The basic ideas in the proof of theorem 1 are as follows.

We define a *ball of radius $r$* in a $d$-dimensional tape unit as the set of tape cells reachable by a head within $r$ steps from a given tape cell. The volume $V(r)$ of such a ball is the number of tape cells in it. We use an initial command sequence $u_0$ which will make the virtual machine write a random string $x$ in a ball of radius $r$ and send its heads to the center. We partition this ball into subballs such that each subball contains a 'sufficiently random' substring. We, then, select an $h$-tuple of subballs and send all the $h$ heads of $M$ to the centers of subballs. We now make queries about the contents of the subballs. We show that there exist some $h$-tuples of subballs such that the corresponding queries will be difficult for the simulator. Otherwise, we could obtain a contradiction by giving a succinct description of $x$ in terms of an $h'$-tuple of subballs surrounding the simulator heads. The intuition is that the simulator has access deficiency because it has fewer heads.

For a probabilistic simulator $PS$, careful counting shows that a majority of the queries are hard for a sufficiently large fraction of the computations of $PS$ (lemmas 4, and 3) which will imply that a random query will be hard with a high probability for an appropriate probability distribution on queries. Then, the easy half of von Neumann's minimax theorem gives the desired lower bound.

First, we prove the following fact about the Kolmogorov complexity of the substrings of a random string.

**Fact 1:** Let $X$ be a set of $k$ strings $x(1), \ldots, x(k)$ each of length $m$; and let $z = x(1) \ldots x(k)$. Let $X^b$ be the set of $b$-tuples of $X$ with distinct components for some $b$ (to be selected later) less than $k$. Let $k_b$ be the minimum of $K(x_i, \ldots, x_k)$, for $(x_i, \ldots, x_k) \in X^b$. Then, if $x$ is incompressible,

$$k_b \geq bm - O(b \log r)$$

provided $\log k = O(\log r)$ and $\log m = O(\log r)$.

**Proof:** Let $x'$ be the string obtained from $x$ by deleting the substrings $x_{i_1}, \ldots, x_{i_j}$ for some distinct $i_j$, $1 \leq j \leq b$. $x$ can be specified by giving $x_{i_1}, \ldots, x_{i_b}$ positions of $x_i$ in $x$ for $1 \leq j \leq b$, $m$, $k$, and the bits of $x'$.

$$|z| = km \leq K(x_{i_1}, \ldots, x_{i_b}) + O(b \log k) + O(\log m) + |x'|$$

Hence,

$$K(x_{i_1}, \ldots, x_{i_b}) \geq bm - O(b \log r)$$

The following lemma is an application of lemma 1 to $b$-tuples of $X$, which shows that a large fraction of the guess strings does not give sufficient information about any $b$-tuple in $X^b$.

**Lemma 2:** Let $G(\lambda;r)$ be the set of guess strings of length $r$. Let $X, x, k, \lambda, b, k_b$ and $X^b \in \lambda$ be as in fact 1. If

$$G(X^b;\lambda) = \{ g \in G | \exists x_{i_1}, \ldots, x_{i_b} \in X^b \}$$

such that

$$K(x_{i_1}, \ldots, x_{i_b}) \geq bm - O(b \log r)$$

then $|G(X^b;\lambda)| \leq 2^b$ where $L = t + \lambda - k_b/2 + O(b \log r)$, provided $\log t, \log b, \log m$ and $\log k$ are $O(\log r)$.

**Proof:** Given a string in $G(X^b;\lambda)$, less than $\lambda$ bits determine at least one $b$-tuple in $X^b$. There are at most $|b|$ $b$-tuples in $X^b$. If $G(X^b;\lambda) > 2^b$, there exists a $(x_{i_1}, \ldots, x_{i_b})$ such that at least $2^b / |b|$ strings belonging to $G$ require less than $\lambda$ bits to determine $(x_{i_1}, \ldots, x_{i_b})$.

By a similar argument to that of lemma 1, it can be shown now that

$$K(x_{i_1}, \ldots, x_{i_b}) \leq 2t + 2\lambda - 2L + 2 \log |b| + O(\log r)$$

Therefore,

$$L < t + \lambda - k_b/2 + O(b \log r)$$

Note that the complement $\bar{G}(X^b;\lambda)$ of $G(X^b;\lambda)$ contains almost all of the guess strings of length $t$ large enough $r$ with an appropriate value of $\lambda$, say $bm/3$, provided $m = O(r^2)$ for some $\delta > 0$. Also note that we are only interested in guess strings whose length is polynomial in $r$, if the input command sequence is polynomial in $r$.

The following lemma shows the existence of input command sequences which the simulator finds hard to simulate.

**Lemma 3:** For each sufficiently large $r$, there is a command sequence $u_0$ of length $r$ and a probability distribution $p$ on input command sequences of length $r$ such that for $u = u_0$ and every longer command sequence $u$ equivalent to $u_0$, the following holds

$$\text{ave}_{p} T_q(u;r) = \Omega(r^{1+\epsilon})$$

provided $g \in \bar{G}(X^b;bm/3)$.

**Proof of Theorem 1:** Since $T(u;r) \preceq \text{ave}_{p} T_q(u;r)$, it is clear that repeated application of lemma 3 yields theorem 1.

The rest of the section contains a proof of the lemma 3. The combinatorial lemma (lemma 4) and its corollary, similar to those in [PSS] are necessary to prove lemma 3. The $h'$-tuples represent the information available at the simulator heads and the $h$-tuples correspond to the output produced by $M$. Lemma 4 deals with coding $h$-tuples of balls in $d$-dimensional tape unit by $h'$-tuples of balls, and shows that for a large fraction of these $h'$-tuples, the corresponding $h$-tuples must have large radius.

Let $g$ be a guess string such that $g \in \bar{G}(X^b;bm/3)$ i.e. for any $(x_{i_1}, \ldots, x_{i_b}) \in X^b$, $K(x_{i_1}, \ldots, x_{i_b}) \geq bm/3$. Let $g$ remain fixed throughout the rest of this section.
Lemma 4: Let $Y$ be a set of $k$ strings each of length $m$. Then, there exist at least $k^h/2$ $h$-tuples of $X$ such that $K(x_1 \ldots x_k | g') > m/8$ for every $h$-tuple $(y_1 \ldots y_h)$ of $Y$, provided $m' \leq bm/(8h')$ and $k / (2^{h'h}k^{-h'}) \geq b$.

Proof: Suppose, to the contrary, that for more than $k^h/2$ $h$-tuples, there is an $h'$-tuple such that $K(x_1 \ldots x_k | g') \leq m/8$. A contradiction can be obtained by showing that $K(x_1 \ldots x_k | g') < bm/3$ for some $(x_1 \ldots x_k) \in X$.

The number of $h$-tuples from $X$ is $k^h$, while the number of $h'$-tuples from $Y$ is only $k^{h'}$; hence, there will be some $h$-tuple $(y_1 \ldots y_h)$ which works for at least $p = k^h/2k^{h'}$ distinct $h'$-tuples. The number of distinct components of these $h$-tuples must be at least $q = \lceil p/2 \rceil \geq k/2^{h'h'}k^{-h'}$; let $x_1, \ldots, x_q$ be these $q$ components. Let $x_1, \ldots, x_q$ be a $b$-tuple with $b$ distinct components out of these $q$ components. (Such a $b$-tuple exists since $q \geq b$.)

$x_1, \ldots, x_q$ can now be described in terms of $y_1, \ldots, y_h$. For each $j (1 \leq j \leq h)$, $x_j$ appears, say as $x_{p_j}$, in some $h$-tuple $(x_1 \ldots x_q)$ for which $K(x_1 \ldots x_q | g') \geq m/8$, say via shortest description $d_j$.

$y_1' \ldots y_h'd'\prod' \cdots d'_k \prod$ describes $(x_1, \ldots, x_q)$ given $g$ with $O(1)$ bits of explanation.

Therefore, $K(x_1 \ldots x_k | g') < m'h' + bm/8 + O(b \log m)$ or

$$\frac{bm}{3} < \frac{bm'h'}{8h'} + \frac{bm}{8} + O(b \log m) \quad (\text{since } m' \leq \frac{bm}{8h'})$$

which is a contradiction.

Corollary: In addition, let the set $Z$ of $Y$ be the set of all strings $x$ for which $K(x | g)$ is a small enough fraction of $m_0$, for some $y$ in the set $X$. Let $\lambda_0$ be the set of all $h$-tuples of $X$ such that $K(x_1 \ldots x_k | g') > m/8$ for every $h$-tuple $(y_1 \ldots y_h)$ of $Y$. Note that, $|\lambda_0| \geq k^h/2$ from lemma 4. For each $h$-tuple $(x_1 \ldots x_k)$ in $\lambda_0$, we still get $K(x_1 \ldots x_k | g') > m/9$ for every $h$-tuple $(x_1 \ldots x_k)$ of strings from $Z$.

Proof of Lemma 2: We will use the initial command sequence $u_0$ to write a sufficiently incompressible ball $B$ of radius $\tau/2$ and to send all the virtual heads to its centre. More precisely, cover the ball of radius $\tau/2$ with $k = \Theta(V(\tau)/V(\sigma))$ disjoint subballs, each of radius $s(1 \leq s \leq r)$. $(x$ will be selected later.) In each of these subballs, we store in some canonical manner a string of length $m = V(\sigma)$, chosen so that some concatenation of these strings is incompressible. Let $X$ be the set of these strings.

Consider the following probability distribution over input command sequences of length $P$

$\tau/2$ commands: Send the virtual heads to the centers of the subballs where $x_1, \ldots, x_9$ are stored, with each $h$-tuple of $X$ being equally likely.

$\tau/2$ commands: Repeatedly, in $O(s)$ commands, make an inquiry and then return each virtual head to the center of its corresponding subball, with each of the $hm$ inquiries being equally likely.

Let $u_0$ be any longer command sequence equivalent to $u_0$. For $t$, $1 \leq t \leq \Theta(\text{avg } T_\gamma(u_\gamma))$, cover each ball of radius $T$ centered around simulator heads with $k' = \Theta(V(\tau)/V(\sigma))$ balls of radius $\Theta(t)$ such that each subball of radius $t$ lies entirely within a member of the cover. Select a listing of the contents of each cover member and let $Y$ be the set of these $k'$ strings, each unambiguously padded out to length $m = \Theta(V(\tau))$. As in the corollary to lemma 4, let $Z$ be the set of strings $x$ for which, for some $y$ in $Y$, $K(x | g)$ is a sufficiently small fraction of $m_0$. Note that this set includes description of the contents of each member of the cover at any possible time within the next $\tau$ steps, provided $T$ is a small fraction of $m_0$.

We now show that $T = \Omega(\tau/\epsilon)$, provided $V(\sigma)$ is a large enough multiple of $T, k' / 2^{h'h'h''} \geq b$ and $m' \leq bm/8h'$ (in the next $\tau$ steps, provided $T$ is a small fraction of $m_0$).

Let $w$ be a random input command sequence drawn from the set of input command sequences of length $\tau$ with probability distribution $\gamma$ as given above. With very high probability (depends on the exact choice of $T$), the simulator takes no more than $\tau$ steps to handle $w$. The $h$-tuple in $w$ belongs to $\lambda_0$ (defined in corollary to lemma 4) with probability at least $1/2$. If the $h$-tuple of $w$ is in $\lambda_0$, then each inquiry independently requires at least $\tau$ simulator steps with at least a constant probability (which depends only on $h$). Otherwise, we can construct $(x_1, \ldots, x_9)$ from the simulator, its radius $\tau$ instantaneous description at that time, the length of the guess string so far consumed and the missing bits in the same order i.e. the results of inquiries which require more than $\tau$ simulator steps. For some $h'$-tuple $(x_1, \ldots, x_9)$ of strings from $Z$, then an upper bound for $K(x_1 \ldots x_9 | g')$ can be made smaller than $m_0/9$ which results in a contradiction, provided $m_0$ is sufficiently large. It follows that $T = \Omega(\epsilon/\tau)$.

Now, we will choose the parameters $s, b$ and $T$. We select $s$ such that $s = \Theta(\tau^2/\epsilon^2)$. Then $T = \Omega(\tau^2/\epsilon^4)$. An optimal value for $b$ is $\tau^2/\epsilon$ with $\epsilon \leq (d-1)(\tau^2/h')$. Hence, $\text{avg } T_\gamma(u_\gamma) = \Omega(\epsilon^2/\tau^2)$. As in [PSS], a similar result holds under certain conditions even when then the simulator has a higher dimensional tape.

4. Probabilistic Simulation of $k+1$ Tapes by $k$ Tapes

In this section, we prove the following theorem.

Theorem 2. For every $k$, there is a $k+1$-tape deterministic machine $M_{k+1}$ that works in real time such that every $k$-tape probabilistic simulator $PS$ of $M_{k+1}$ that works online is $O(n \log^2 (k+1)n)$ time bounded.

The strategy used in proving this theorem is similar to that of [A] and [P]. For every $k$, we define a $k+1$-tape deterministic machine $M_{k+1}$ that works in real time. We store incompressible information on the $k+1$-tales of $M_{k+1}$ at very different rates. This can be accomplished by storing random strings of different lengths on the $k+1$-tales of $M_{k+1}$. The idea is that any $k+1$-tape probabilistic simulator $PS(g)$ (for sufficiently many guess strings $g$) will have difficulty in accessing some of this information
(since it has only $k$-heads) unless it does extensive preprocessing. This intuition suggests the need to measure the amount of preprocessing time. Overlap - roughly, the number of tape cells revisited in a time interval - is a convenient measure of this preprocessing time. (A precise definition of overlap will be given later.)

The input command sequence which stores the random strings can be divided into a set of intervals such that the overlaps of these intervals are disjoint. If the overlap for sufficiently many of these intervals is large, we can use the overlap lemma from [A] and [PSS] which states that "every computation where large overlap is frequent is long" and obtain our bound.

If the overlap for some input interval $I$ is low, we show that some head $h$ of $M_{k+1}$ is neglected by $PS(g)$, in the sense of [A, PSS]. We will now make queries about tape $h$ in the form of an $t$-loop for head $h$ at the end of interval $I$, where an $t$-loop for head $h$ is a command sequence of the form:

$(h, \text{left}, \text{NOP}) \cdots t$-times $(h, \text{right}, \text{NOP}) \cdots t$-times.

Note that, if $u$ is the inscription of consecutive cells which are at most $l$ cells to the left of the head $h$ of $M_{k+1}$, at the end of interval $I$, an $t$-loop for $M_{k+1}$ at the end of interval $I$ will output a string which contains $u$ and its suffix $u_l$. If $PS(g)$ can simulate this $t$-loop in $t$ steps, we will be able to determine $u$ given $U_t$, the contents of the $t$ tape cells to the left and right of head $h_1$ for $i \in \{1, \ldots, k\}$, the specification of $PS$, the position of $u$ in the output string that is produced during the simulation of $t$-loop, $l$, and $h$. We, therefore, have

$$K(u | U_1 \cdots U_{k-1} U_k) = O(\log l) \tag{1}$$

The fact that the overlap in the interval $I$ is low, and the head $h$ of $M_{k+1}$ is neglected by $PS(g)$ will make it possible to determine the $U_t$s with relatively small information, if $\tau$ is small. But, a long enough $u$ is an almost random string since it is a substring of a random string. This enables us to obtain a contradiction.

Thus, in either case, we are assured that the simulating machine spends enough time to make up for access deficiency.

Note that, this strategy can not, in general, guarantee that the performance of $PS$ will be degraded for a sufficiently large number of its computations. To obtain the desired average time bound for the probabilistic simulator $PS$, we will base our choice of $t$-loops on the ensemble of computations of $PS$. If the overlap in the interval $I$ is low for a large set of the computations of $PS$, we show that a fixed tape must be neglected by a sufficiently large fraction of the computations of $PS$ and perform an $t$-loop at the end of the interval $I$ to degrade the performance of the simulator. Here, we need lemma 1 to ensure that a sufficiently incompressible string remains sufficiently incompressible, even in the presence of a large enough fraction of the guess strings. In the other case, when a large fraction of computations of $PS$ has high enough overlap in the input interval $I$, no $t$-loop is performed for interval $I$. It can now be shown that, if few loops are performed, the average time used by the simulator is already high (because of the high overlap in many computations), and otherwise, each loop will be time consuming in a significant fraction of the computations and thus the average time will again be high.

The rest of this section contains a proof of theorem 2.

Overlapping: We give here a precise definition of overlap and state a lemma which counts the total overlap of a set of input intervals.

We can associate a computation multigraph with each computation $PS(g)$ of a $k$-tape probabilistic Turing machine $PS$ with $g$ as its guess string. Let $T$ be the number of steps in this computation. The nodes of this graph are the steps $1, \ldots, T$ and there are $s$ edges from $i$ to $j$ if there are $s$ tape cells that are visited in steps $i$ and $j$ but not in between. The indegree of this graph is bounded by $k$ and hence

$$T \geq \# \text{edges} / k$$

If $I = \{f, f+1, \ldots, l\}$ is a time interval of the computation and $\epsilon \in I$, then the number of edges going from $\{f, f+1, \ldots, l\}$ to $\{f+1, \ldots, l\}$ is denoted by $\omega(I, \epsilon)$. The number $\omega(I) = \max \{\omega(I, \epsilon) \mid \epsilon \in I\}$ is called the interval overlap of time interval $I$. The number of edges of a computation graph can be estimated in terms of the numbers $\omega(I, \epsilon)$ with the help of a set of intervals $I = \{I_a\}$ and a set of steps $\epsilon = \{\epsilon_a\}$ such that

$$I_a \subset I_a \quad \text{for all } a$$

$$I_a \cap I_b = \emptyset \quad \text{or } I_a \cap I_b = \phi \quad \text{for all } a, b$$

One can easily verify that each edge is counted in at most one $\{I_a, \epsilon_a\}$ and thus

$$\# \text{edges} \geq \sum_a \omega(I_a, \epsilon_a)$$

Let $m$ and $s$ be natural numbers and suppose we partition the computation into $2^s$ time intervals $I_{2^s-1}$, $I_{2^s-2}$, $\ldots$, $I_1$, $I_0$. For $0 \leq s < s'$, let $I_{s-s'} = I_{s-1} \cup I_{s-2} \cup \cdots \cup I_{2^s-1}$. The weight $w(I_{s-s})$ of $I_{s-s}$ is defined as $2^{s-s}$ and for a set $A$ of such intervals the weight $w(A)$ of $A$ is defined as the sum of the weights of the intervals in $A$.

Lemma 5: If $w(A) \geq 3(s+1)2^s / 4$ and $\omega(I) = \omega(V(I))$ for all $I \in A$, then $T \leq m(s+1)^2 / 4k$.

For a proof of this lemma, see [P].

Design of input: We will give here a command sequence which will store incompressible strings on the $k$-tapes of $M_{k+1}$ at different rates, and which include $t$-loops for certain input intervals so that the simulator $PS$ will have difficulty in simulating this command sequence.

Let $h$ be a natural number, $s = \lceil \log n \rceil / 2$ and $d = \lceil \log n \rceil / (k+1)$. For convenience, assume $16k(k+1)$ divides $n$. Let $w_0 = \ldots w_d \in [0,1]^n$ be a random string where $|w_0| = n2^{d+1}$. For $h \in \{0, \ldots, k\}$ let $C_h$ be an input sequence that makes head $h$ of $M_{k+1}$ print $w_h$ from left to right. Let $D_0 = C_0$ and for $h \in \{1, \ldots, k\}$ obtain $D_h$ from $D_{h-1}$ by inserting $d$ commands from $C_h$ after every command from $D_{h-1}$ that is already inserted in $D_{h-1}$. The sequence $E = D_h$ has length

$$|E| = n2^d(1+d+\ldots+d^k) = \Theta(n2^d \log^{k/k+1} n) \tag{2}$$

and any subsequence $E'$ of $E$ of length $N(1+d+\ldots+d^k)$ that begins with a command from $C_h$ makes $M_{k+1}$ write $N^d$ symbols from $w_h$ on tape $h$ for $h = 0, \ldots, k$.

We are now ready to specify the actual input sequence for $M_{k+1}$. We will choose below command sequences $F_t$, each consisting of 0 to $s+1$ $t$-loops. Divide $E$ into $2^s$ parts $E_t$ of length $n(1+d+d^2\ldots+d^k)$. The input will be $E_0 F_0 E_1 F_1 \cdots$. To choose $F_t$, assume
$F_0, \ldots, F_{t-1}$ have already been chosen.

Partition the computations of the simulator $PS$ given $E_0, E_1, F_1, \ldots$ into intervals $I_{0,0}, I_{0,1}, \ldots$ where interval $I_{a,b}$ lasts from the last step of the simulation of $E_a$ to the last step of the simulation of $F_b$. For all $i$, let $A_i = \{(a,b) : |a|, |b| \leq n \}$ be the intervals in which each of these computations terminates. Let $A_i \circ E_j$ be ordered in the increasing order of the weights of the intervals in $A_i$. For each $(a,b) \in A_i$, there will be a part $F_{a,b}$ in $F_j$ which will be either an $l$-loop or empty.

Let $(a,b) \in A_j$ be the next interval such that for all $(a',b') \in A_j$, whose weights are less than that of $(a,b)$, $F_{a',b'}$ has already been defined and simulated by $PS$. Let $t_j$ be the last step performed by $PS$ with guess string $g$. Let $t$ be the maximum of $t_j$ over all $j$. Let $G$ be $\{0,1\}^t$. For each $g \in G$, let $I_{a,b}(t_j)$ be the time interval in the simulation of $PS$ with guess string $g$, that begins with the first step $I_{a,b}$ and lasts until $t_j$. Let

$$m = \frac{n}{16(k+1)}$$

and let

$$G_i = \{ g \in G : |I_{a,b}(t_j)| < nw(I_{a,b}) \}.$$

If $|G_i| \leq |G|/2$, then part $F_{a,b}$ is empty and the next interval in $A_j$ is considered. Otherwise an $l$-loop will be defined for $(a,b)$. Let $g \in G_i$. Consider the computation during the interval $I_{a,b}(t_j)$. Let $I_g$ denote $I_{a,b}(t_j)$ and $w(I)$ denote $w(I_{a,b})$.

Let $E'$ be the portion of $E$ that is simulated during $I_g$. $E'$ is the same portion for all $g \in G_i$. Partition $E'$ into $k+1$ parts $E'_1, \ldots, E'_{k+1}$ of equal length. This induces a partition of $I_g$ into parts $I_{g,1}, \ldots, I_{g,k+1}$, where each interval begins with the first step of the simulation of $E_i$. Let $N = nw(I)$ be the number of commands from $C_0$ in $E'$ and $N' = N/\log(16(k+1))$. We say, $PS(g)$ neglects tape $h$ of $M_{k+1}$ during $I_g$ if every head of $PS(g)$ that visits at least $N'd^k$ tape cells during interval $I_{g,1} \cdots I_{g,k+1}$ visits at least $N'd^k$ tape cells during $I_{g,1} \cdots I_{g,k+1}$.

If $PS(g)$ would neglect no tape during $I_g$, then for all $h \in \{0, \ldots, k\}$ some head $j(h)$ of $PS(g)$ visits at least $N'd^k$ cells during $I_{g,1}$ and at most $N'd^k+1$ cells during $I_{g,1} \cdots I_{g,k+1}$. Because $PS$ has only $k$ heads, we have $j(h) = j(h')$ for some $h$ and $h' \geq h + 1$. Head $h$ would have to visit during $I_{g,1}$ both at least $N'd^k$ and at most $N'd^k+1$ tape cells.

Therefore, for each $g \in G_i$, a head will be neglected by $PS(g)$ during $I_g$. Hence, there exists a head $h$ and a set of guess strings $G_h \subseteq G_i$ with $|G_h| \geq |G|/2(k+1)$ such that for all $g \in G_h$, head $h$ is neglected by $PS(g)$ during $I_g$. Let $w_h$ be the substring of $w_h$ that is written on tape $h$ of machine during interval $I_{g,1}$, and let $x$ and $y$ be such that $w = w_2y$. Then

$$|u| = Nd^k/(k+1)$$

and at the end of $I_{a,b}$, the first symbol of $u$ is exactly $l = Nd^k(k-h+1)/(k+1)$ tape cells to the left of head $h$ of $M_{k+1}$. Part $F_{a,b}$ is an $l$-loop for head $h$ with $l$ as above. Notice

$$|F_{a,b}| = \frac{nw(l)d^k}{k+1}$$

$$= \frac{nw(I)d^k}{k+1}$$

$$\geq \frac{nw(I)}{k+1}$$

The effect of $F_{a,b}$: We will consider the effect of $F_{a,b}$ on $PS(g)$ with $g \in G_h$. Since $w$ is a random string, we know that

$$K(u | x'y) \geq e^{-\log|u|}$$

$$|u| = O(\log|w|)$$

By lemma 1, we also know that most of the strings in $G$ do not give enough information about $u$. With $\lambda = |u|/3$ we have from lemma 1

$$G(u : |u|/3) \leq 2 \cdot |u|/\log(16(k+1)).$$

Therefore, we have

$$|G(u : |u|/3) \cap G_h| \leq 2 \cdot |u|/\log(16(k+1))$$

and

$$|G(u : |u|/3) \cap G_h| \leq 2 \cdot |u|/\log(16(k+1))$$

for large $n$.

Let $G_h = G(u : |u|/3) \cap G_h$.

**Lemma 6:** Suppose $F_{a,b}$ is an $l$-loop for head $h$ and $PS(g)$ with $g \in G_h$ simulates $F_{a,b}$ in $\tau$ steps. Then

$$\tau \geq Nd^h/(17(k+1)).$$

**Proof:** For head number $i$ of $PS(g)$, let $T_i = t_1 t_2 \ldots t_{k+1}$ be the portion of tape $i$ of $PS(g)$ visited until the last step of time interval $I_g$ where $t_i$ is the portion of $T_i$ visited during $I_{g,i}$. Let $V_i$ be the portion of $W_i$ that is revisited during $I_{g,i} \cdots I_{g,k+1}$ and $U_i$ be the portion of $T_i$ consisting of $\tau$ tape cells left and right of head $i$ after time interval $I_g$. Let $H$ be the set of head numbers $i$ such that head $i$ of $PS(g)$ visited at most $Nd^h$ tape cells during $I_{g,i}$. Then

$$|u|/3 \leq K(u | x'y)$$

for $g \in G_h$ and

$$\leq K(u | U_1 \cdots U_{k+1} \ldots U_k$$

+ $K(U_1 \cdots U_{k+1} \ldots U_k$ and $I_g | g | x'y)$

+ $O(\log n).$ (4)

By (1), the first term is $O(\log n)$. The contents of $T_1, \ldots, T_h$ before $I_{g,h}$ can be determined from $x, y$ and $F_{0,0} F_{1,1} \cdots F_{h-1,h-1} \cdots F_{a,b}$ by simulation of $PS(g)$. From this, $W_i$ after $I_{g,h}$ and the positions of all $W_i$'s within the $T_i$'s, one finds $T_i$ after $I_{g,h}$ and $I_{g,h} \ldots I_{g,h}$. If, moreover, $V_i$ after $I_{g,h}$ and $I_{g,h} \ldots I_{g,h}$, the positions of the $U_i$'s as well as state and head positions of $PS(g)$ after $I_{g,h}$ are specified, then one can determine $T_i$ after $I_{g,h}$ and $I_{g,h} \ldots I_{g,h}$ by simulation of $PS(g)$. We, therefore, have

$$K(T_i | I_{g,h} \ldots I_{g,h} | t \in H)$$

$$\leq K(F_{0,0} \cdots F_{a,b})$$

$$+ K(W_i | I_{g,h} \ldots I_{g,h} | t \in H)$$

$$+ K(V_i | I_{g,h} \ldots I_{g,h} | t \in H) + O(\log n).$$
$V_1, \ldots, V_k$ is bounded by the internal overlap $\omega(I_f)$. Thus, the third term is at most

$$2\omega(I) \leq N_2 + (8(k+1)) \leq N_2^h/(8(k+1)) \text{ and}$$

$$K(\{T_i | i \in H\}, I_0, R_1 \text{ after } I_f | i \notin H | g'x'y) \leq |u|/4 + o(n)$$

(5)

Because $PS(g)$ neglected tape $h$ of $M_{k+1}$ during interval $I_f$, every head $i$ of $PS$ with $i \notin H$ visits at least $N_2^h/(k+1)$ cells away from $W_i$. Thus, if lemma 6 is false, then for $i \notin H$, the $\tau$-neighbourhood $U_i$ of head $i$ of $PS(g)$ lies completely in $L_i$ or $R_i$ and

$$K(U_i \text{ after } I_f | L_i, R_i \text{ after } I_f) = O(\log n).$$ (6)

(4), (5), and (6) imply that

$$|u|/3 \leq |u|/4 + o(n)$$

which is a contradiction. □

**Running Time:** Because of the way loops $F_{a,b}$ were chosen, for each loop $F_{a,b}$ of $F$ at least $\frac{1}{4(k+1)}$ of the total number of computations of $PS$ will spend $O(|F|\log^k(k+1)n)$ time. If $|F| \geq n^{2^k \log^k(k+1)n}$, then the running time of $PS$ is $\Omega((|E| + |F|) \log^l(k+1)n)$.

If $|F| < n^{2^k \log^k(k+1)n}$, let $C$ be the set of intervals such that no $i$-loop is performed. For $I_{a,b} \in C$, we have $|F_{a,b}| \leq \frac{|W|}{2n}$ by (3), and hence

$$w(C) \leq \frac{|F|}{2n} \leq 2^{s-1} \log^k(k+1)n$$

$$= \frac{s \cdot 2^s}{\log^l(k+1)n}$$

$$\leq (s+1)2^s/4$$

which implies $w(C) \geq \frac{3}{4}(s+1)2^s$.

By the definition of the parts $F_{a,b}$, we have $\omega(I_f) \geq mw(I)$ for $I \in C$ for at least half of the set of all possible guess strings. Therefore, from lemma 5 the average time over all guess strings is $\Omega(m2^s(k+1)/4k)$

$$= \Omega(s4k(k+1))$$

which again gives the desired bound since $|E| = \theta(n2^s \log^k(k+1)n)$.

5. References.


24th Annual
Symposium on Foundations of Computer Science
(Formerly called the Annual Symposium on Switching and Automata Theory)

NOVEMBER 7-9, 1983
IEEE 83CH1938-0

sponsored by
the IEEE Computer Society's Technical Committee on
Mathematical Foundations of Computing
Table of Contents

Monday, November 7, 1983

**Session 1: Oscar Ibarra, Chairman**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solving Low Density Subset Sum Problems</td>
<td>1</td>
</tr>
<tr>
<td>J.C. Lagarias and A.M. Odlyzko</td>
<td></td>
</tr>
<tr>
<td>How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin</td>
<td>11</td>
</tr>
<tr>
<td>M. Luby, S. Micali, and C. Rackoff</td>
<td></td>
</tr>
<tr>
<td>Trapdoor Pseudo-Random Number Generators, with Applications to Protocol Design</td>
<td>23</td>
</tr>
<tr>
<td>U.V. Vazirani and V.V. Vazirani</td>
<td></td>
</tr>
<tr>
<td>A Topological Approach to Easiveness</td>
<td>31</td>
</tr>
<tr>
<td>J. Kahn, M. Saks, and D. Sturtevant</td>
<td></td>
</tr>
<tr>
<td>On the Security of Multi-Party Ping-Pong Protocols</td>
<td>34</td>
</tr>
<tr>
<td>S. Even and O. Goldreich</td>
<td></td>
</tr>
<tr>
<td>The Program Complexity of Searching a Table</td>
<td>40</td>
</tr>
<tr>
<td>H.G. Mairson</td>
<td></td>
</tr>
<tr>
<td>Improved Upper Bounds on Shellsort</td>
<td>48</td>
</tr>
<tr>
<td>J. Incerpi and R. Sedgewick</td>
<td></td>
</tr>
<tr>
<td>Monte-Carlo Algorithms for Enumeration and Reliability Problems</td>
<td>56</td>
</tr>
<tr>
<td>R. Karp and M. Luby</td>
<td></td>
</tr>
<tr>
<td>Optimum Algorithms for Two Random Sampling Problems</td>
<td>65</td>
</tr>
<tr>
<td>J.S. Vitter</td>
<td></td>
</tr>
<tr>
<td>Probabilistic Counting</td>
<td>76</td>
</tr>
<tr>
<td>P. Flajolet and N.G. Martin</td>
<td></td>
</tr>
</tbody>
</table>

**Session 2: Dexter Kozen, Chairman**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructing Arrangements of Lines and Hyperplanes with Applications</td>
<td>83</td>
</tr>
<tr>
<td>H. Edelsbrunner, J. O'Rourke, and R. Seidel</td>
<td></td>
</tr>
<tr>
<td>Dynamic Computational Geometry</td>
<td>92</td>
</tr>
<tr>
<td>M.J. Atallah</td>
<td></td>
</tr>
<tr>
<td>A Kinetic Framework for Computational Geometry</td>
<td>100</td>
</tr>
<tr>
<td>L. Guibas, L. Ramshaw, and J. Stolfi</td>
<td></td>
</tr>
<tr>
<td>Geometric Retrieval Problems</td>
<td>112</td>
</tr>
<tr>
<td>R. Cole and C.K. Yap</td>
<td></td>
</tr>
<tr>
<td>Filtering Search: A New Approach to Query-Answering</td>
<td>122</td>
</tr>
<tr>
<td>B. Chazelle</td>
<td></td>
</tr>
</tbody>
</table>
Tuesday, November 8, 1983

Session 3: Richard Statman, Chairman

Some Relationships between Logics of Programs and Complexity Theory ...................................................... 180

J. Tiuryn and P. Urzyczyn

Reasoning about Infinite Computation Paths .................................................. 185

P. Wolper, M.Y. Vardi, and A.P. Sistla

Propositional Game Logic ................................................................. 195

R. Parikh

Reasoning about Functional Programs and Complexity Classes Associated with Type Disciplines .................. 201

D. Leivant

Decision Procedures for Time and Chance ................................................. 202

S. Kraus and D. Lehmann

Algebras of Feasible Functions ............................................................. 210

Y. Gurevich

On Context-Free Generators ...................................................................... 215

J. Beauquier and F. Gire

Legal Coloring of Graphs ........................................................................... 216

N. Linial

The Power of Geometric Duality ................................................................. 217

B. Chazelle, L.J. Guibas, and D.T. Lee

Fast Algorithms for the All Nearest Neighbors Problem ...................... 226

K.L. Clarkson

Minimum Partition of Polygonal Regions into Trapezoids ..................... 233

Tetsuo Asano and Takao Asano

Session 4: Zvi Galil, Chairman

Shortest Path Problems in Planar Graphs .................................................. 242

G.N. Frederickson

Scaling Algorithms for Network Problems ................................................ 248

H.N. Gabow

Partition of Planar Flow Networks ........................................................... 259

D.B. Johnson and S.M. Venkatesan

Approximation Algorithms for NP-Complete Problems on Planar Graphs ......................................................... 265

B.S. Baker
A Polynomial Algorithm for the Min Cut Linear
Arrangement of Trees .............................................. 274
  M. Yannakakis
Tree Structures for Partial Match Retrieval .......................... 282
  P. Flajolet and C. Puech
Bin Packing with Items Uniformly Distributed over Intervals [a, b] .... 289
  G.S. Lueker
Information Bounds Are Good for Search Problems on Ordered Data Structures .................................................. 298
  N. Linial and M.E. Saks
Hash Functions for Priority Queues .................................. 299
  M. Ajtai, M. Fredman, and J. Komlós

Wednesday, November 9, 1983

Session 5: Gary Miller, Chairman

Lower Bounds on Graph Threading by Probabilistic Machines ......... 304
  P. Berman and J. Simon
On the Computational Complexity of the Permanent .................. 312
  J. Ja'Ja'
Multiplication Is the Easiest Nontrivial Arithmetic Function .......... 320
  H. Alt
On Depth-Reduction and Grates ....................................... 323
  G. Schnitger
Relativized Circuit Complexity ....................................... 329
  C.B. Wilson
Randomness and the Density of Hard Problems .......................... 335
  R.E. Wilber
Lower Bounds on the Time of Probabilistic On-Line Simulations ......... 343
  R. Paturi and J. Simon
Techniques for Solving Graph Problems in Parallel Environments ....... 351
  P.H. Hochschild, E.W. Mayr, and A.R. Siegel
An Algorithm for the Optimal Placement and Routing of a Circuit within a Ring of Pads ........................................... 360
  B.S. Baker and R.Y. Pinter
Global Wire Routing in Two-Dimensional Arrays ........................ 371
Period-Time Tradeoffs for VLSI Models with Delay ..................... 372
  A. Aggarwal

Session 6: Robert Tarjan, Chairman

Estimating the Multiplicities of Conflicts in Multiple Access Channels .................................................. 383
  A.G. Greenberg and R.E. Ladner
On the Minimal Synchronism Needed for Distributed Consensus ........ 393
  D. Dolev, C. Dwork, and L. Stockmeyer
Foreword

The papers in this volume were presented at the 24th Annual Symposium on Foundations of Computer Science, held on November 7-9, 1983, in Tucson, Arizona. The symposium was sponsored by the IEEE Computer Society Technical Committee for Mathematical Foundations of Computing.

These 60 papers were selected on June 30-July 1, 1983, at a meeting of the full program committee, from 160 extended abstracts submitted in response to the call for papers. The selection was based on perceived originality, quality, and relevance to theoretical computer science. The submissions were not refereed, and many of them represent preliminary reports on continuing research. It is anticipated that most of these papers will appear, in more polished and complete form, in scientific journals.

The program committee wishes to thank all who submitted papers for consideration.

Manuel Blum
Zvi Galil
Oscar H. Ibarra
Dexter Kozen
Gary L. Miller
J. Ian Munro
W.L. Ruzzo
Lawrence Snyder, Chairman
Richard Statman
Robert E. Tarjan
Macht ey Award

"The Program Complexity of Searching a Table" by Harry G. Mairson is the 1983 winner of the Macht ey Award for the most outstanding paper written by a student or students, as judged by the program committee.