Derandomizing Algorithms on Product Distributions

Ariel Gabizon

Avinatan Hassidim

(randomized)

A_R

We say a randomized algorithm A_R computes a function $f:\{0,1\}^n \rightarrow \{0,1\}$ if for every $x \in \{0,1\}^n$,

$A_R(x,y) = f(x)$ w.h.p over y

`Ideal' Derandomization:

(randomized)

A_R

→

(deterministic)

A_D

- running time $t(n)$
- computing $f:\{0,1\}^n \rightarrow \{0,1\}$

- running time $\sim t(n)$
- computing f correctly ***on all*** $x \in \{0,1\}^n$

seems hard.. achieving A_D with poly(t(n)) running time implies unknown circuit lower bounds [Imp-Kab-Wig, Kab-Imp],..so may take a few milllion years

Arrr.. Derandomize something now or off with your heads!

..We better relax the problem to get some results!

First `relaxation':

(Randomized)

A_R

$\longrightarrow$

(deterministic)

A_D

- running time $t(n)$
- computing $f:\{0,1\}^n \to \{0,1\}$

- running time $\sim t(n)$
- computing f correctly ***w.h.p on any distribution of inputs***

…This is no relaxation at all! as need to succeed on distribution that gives probability 1 to any $x \in \{0,1\}^n$

Real relaxation: Samplable distributions

[Impagliazzo-Wigderson]

(Randomized) A_R → (deterministic) A_D

- running time $t(n)$
- computing $f:\{0,1\}^n \rightarrow \{0,1\}$

- running time $\sim t(n)$
- computing f correctly **w.h.p on any** ***efficiently samplable*** **distribution of inputs**

conditional results by [Impagiliazzo-Wigderson, Trevisan-Vadhan] , partial unconditional results by [Kabanets]

Our relaxation: Product Distributions

Fix large enough k..

- Adversary fixes ***arbitrary*** distribution D on $\{0,1\}^n$.
- A_D gets k ***independent*** samples $x_1,\ldots,x_k$ from D.

A_D needs to compute $f(x_1),\ldots,f(x_k)$ correctly w.h.p.

Needs to do this in time $\sim k \cdot t(n)$

(recall $t(n)$ is running time of A_R)

Dfn: A product distribution X on $(\{0,1\}^n)^k$ is made of k independent copies $(X_1,\ldots,X_k)$ of an arbitrary distribution D on $\{0,1\}^n$

General Result - Algorithms

Thm: $f:\{0,1\}^n \rightarrow \{0,1\}$

- A_R - rand. alg for f running in time t_r , using r random bits, with error ϵ.
- A_D - det. alg for f running in time t_d.

For $k > 8 \cdot r \cdot t_d / t_r$,

$\exists$ **det. alg A running in time $k^* t_r + \tilde{O}(nk)$**

s.t $A(x_1,\ldots,x_k) = f(x_1),\ldots,f(x_k)$

w.p $\sim 1-\epsilon \cdot k$ over any product distribution.

- *[GolWig] get this result for uniform dist.*

Randomness Extraction – Brief review

$\mathcal{C}$ – class of distributions that `contain a lot of entropy'

E – extractor for $\mathcal{C}$: For every distribution **X** in $\mathcal{C}$, **E(X)** is uniform.`

- Classic example: Von-Neumann trick for biased coin:

  01→0 10→1 00,11→try again

Proof Sketch

Given sequence $x_1,\ldots,x_k$ let $\{z_1,\ldots,z_s\}$ be the *distinct* elements of the sequence.

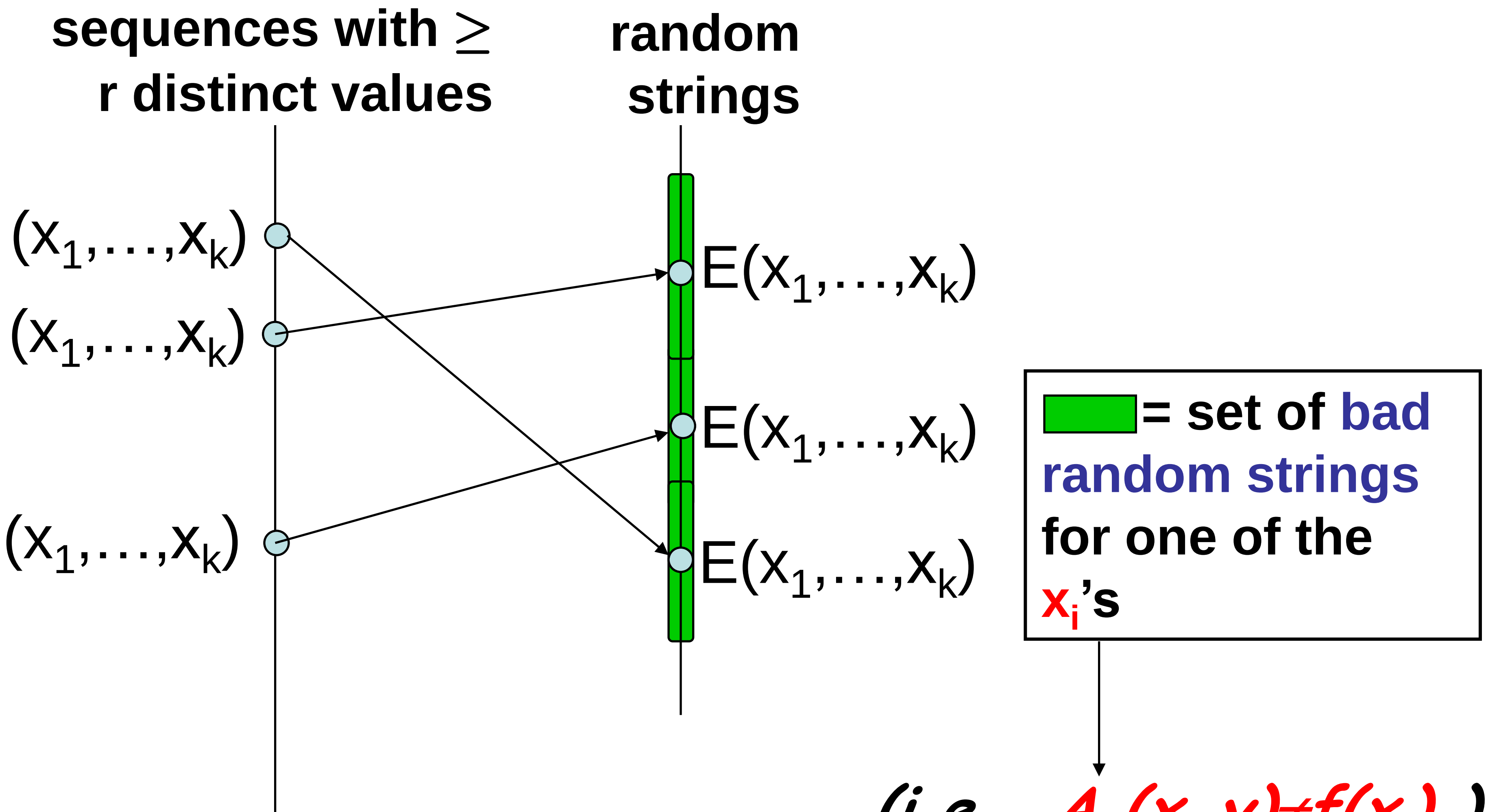
Case 1: $s<r$- run A_D on all elements. Takes time $s{*}t_d < k{*}t_r$

Case 2: $s\geq r$- sequence `contains a lot of randomness'. Extract randomness from sequence and use it run A_R!

i.e. $\forall i$ return $A_R(x_i, E(x_1,\ldots,x_k))$, where E is an extractor for product distributions

Recall $k > r \cdot t_d / t_r$

Potential Problem: Randomness correlated with input may be bad w.h.p.



(i.e., $A_R(x_i, y) \neq f(x_i)$)

Potential Problem: Randomness correlated with input may be bad w.h.p.

Solution: Extract randomness only from *order* in which elements appear

- independent from actual input values
- As we get *independent* samples adversary has no control over this.
- extraction scheme will be a

The `multinomial extractor'

- Given $x_1,\ldots,x_k$
- $\{z_1,\ldots,z_s\}$ - the distinct values among $x_1,\ldots,x_k$
  - z_i appears a_i times
- Num. of orderings is $$\binom{k}{a_1,\ldots,a_s} = \frac{k!}{a_1!\cdot\ldots\cdot a_s!}$$
- E outputs index of $(x_1,\ldots,x_k)$ in orderings. Under prod. distribution all orderings have same prob→ E is uniform!

Gives at least $\Omega(s\cdot\log(k))$ bits ($\geq r$ when $s\geq r$)

(generalization of *[Von-Neumann, Elias]*)

Correctness proof for case $s \geq r$

Want to show: $A_R(x_i, E(x_1,\ldots,x_k))$ *usually correct for all* $1 \leq i \leq k$.

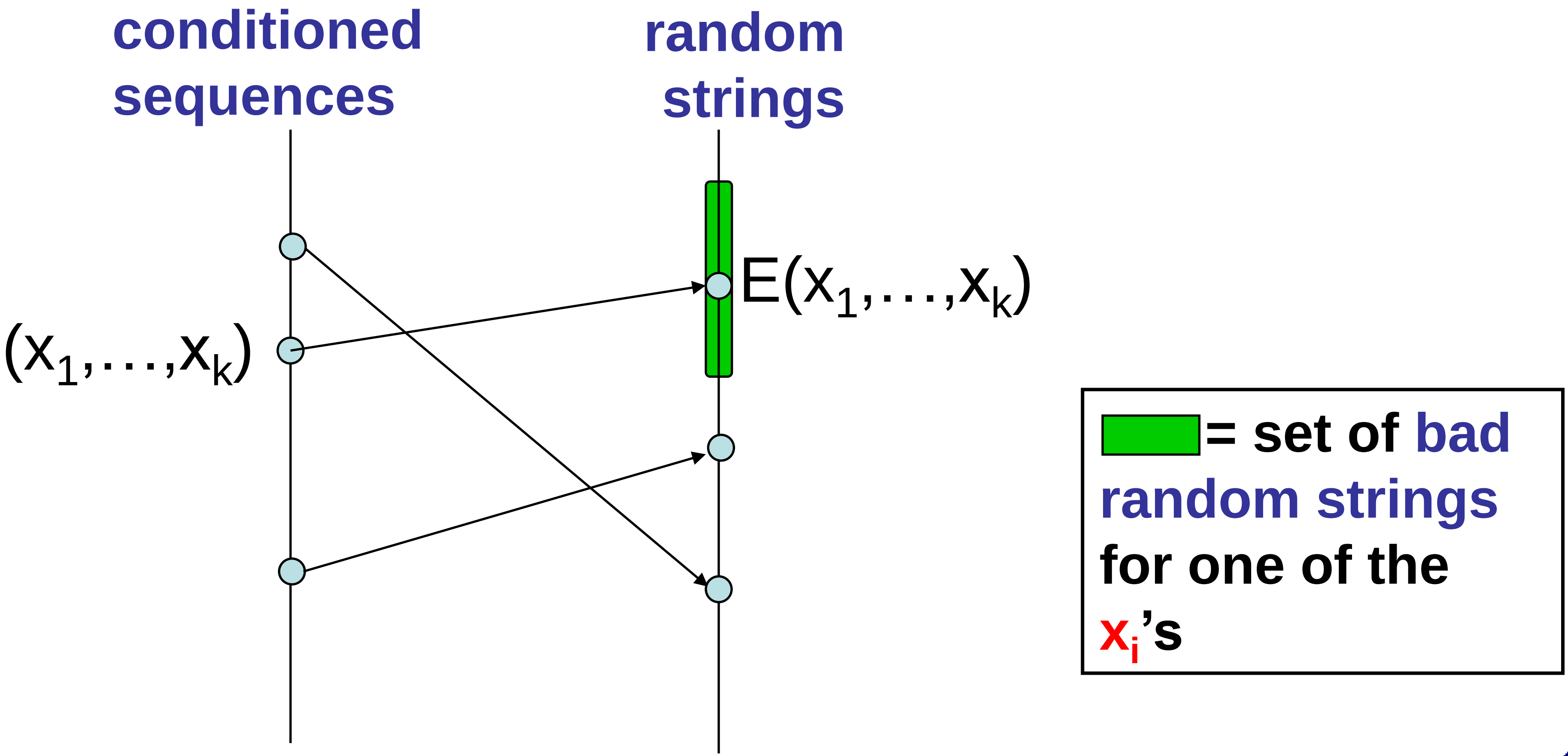
Look at product distribution conditioned on seeing $z_1,\ldots,z_s$ with freq. $a_1,\ldots,a_s$.

→Get uniform distribution on orderings.

Set of bad random strings for $\{z_1,\ldots,z_s\}$ has mass at most ϵk.

→$1-\epsilon k$ frac. of orderings correspond to random string that is good for whole sequence.

Prf by picture: Condition product dist. on seeing $z_1,\ldots,z_s$, $a_1,\ldots,a_s$ times. E is random, and set of bad random strings is fixed (depends on distinct values, not order).



Reminder - Algorithm

1. $x_1, \ldots, x_k$ consist of s distinct elements
2. $s < r$
   1. Run A_D on each instance.
3. $s \geq r$
   1. Extract r bits of randomness using multinomial extractor E
   2. Run A_R on each instance with the same random string $E(x_1, \ldots, x_k)$

- *Will require* $k > rt_d/t_r$

Toy example using VN

- $k=2$, A_R uses one random bit.

Given (x_1,x_2):

- $x_1=x_2$: solve with A_D
- $x_1 \neq x_2$: run A_R on (x_1,x_2) with $r=0$ if $x_1<x_2$ and $r=1$ otherwise.

We run A_D at most once.

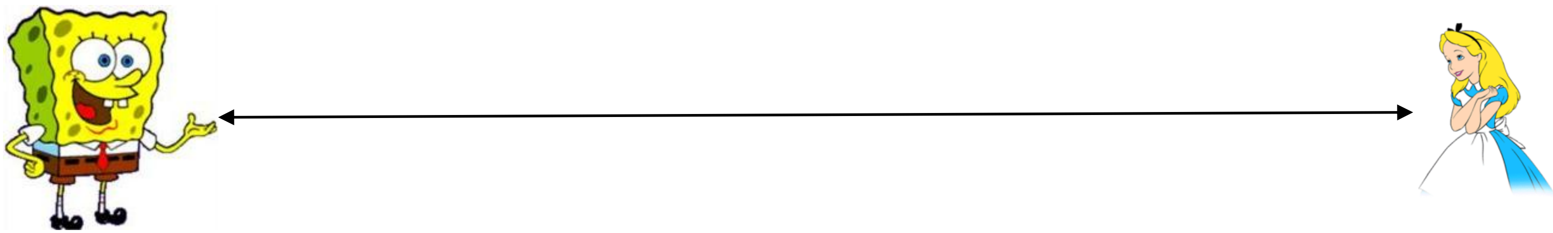
General Result- Communication Protocols.

Thm: $f:\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$

- P_R - rand. protocol for f using c_r communication bits, r random bits with error ϵ
- P_D - det. protocol for f using c_d com. bits.

$\to$ **For $k > \sim r \cdot c_d / c_r$, $\exists$ det protocol P using $O(k^* c_r)$ com. bits s.t.**
for any product distribution X,
P answers correctly w.p. $\sim 1-\epsilon \cdot k$ on $(x_1,y_1),\ldots,(x_k,y_k) \leftarrow X$,

An example of our results- Communication complexity of equality

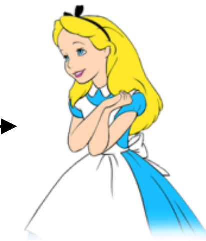


$y_1, \ldots, y_n$ $x_1, \ldots, x_n$

Equality: $f(x,y) = 1$ iff $\forall i,\ x_i = y_i$

Com. Complexity	Deterministic	Randomized
One Shot	O(n)	O(1)

k instances



y^1 x^1

y^2 x^2

⋮

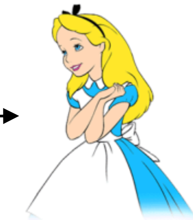
y^k x^k

$\forall\, j$, Equality: $f(x^j, y^j) = 1$ iff $\forall i,\ x^j_i = y^j_i$

Com. Complexity	Deterministic	Randomized
One Shot	O(n)	O(1)
k correct answers	O(kn)	O(k log k)

D - arbitrary unknown distribution

(x^j, y^j) sampled independently from D



y^1

y^2

y^k

x^1

x^2

x^k

Com. Complexity	Deterministic	Randomized
One Shot	O(n)	O(1)
k correct answers	O(kn)	O(k log k)
k>n log n **Succeed w.h.p on all instances**	**O(k log k)**	

Other Results

1. **Multiple Distributions-**

- We assumed $\forall i, \; x_i \sim D$
- What happens if there are multiple distributions?
- Fix unknown $D_1, \ldots D_m$
  - $\forall i, \; x_i \sim D_j$ *(we do not know which j)*

we get similar results for these `m-part product distributions'.

2. **Improved implicit o(logn) probe search –** [Yao, Fiat-Naor]
3. **Derandomizing Streaming algorithms in the random-order model.**

谢谢你!

Proof Sketch

Given sequence $x_1,...,x_k$ let $\{z_1,...,z_s\}$ be the *distinct* elements of the sequence.

Case 1: $s<r$- run A_D on all elements. Takes time $s*t_d < k*t_r$

Case 2: $s\geq r$- sequence `contains a lot of randomness'. Extract randomness from sequence and use it run A_R!

i.e. $\forall i$ return $A_R(x_i, E(x_1,\ldots,x_k))$,

where E is an extractor for *product dist. conditioned on seeing $\geq r$ distinct values*)

Recall $k> r\cdot t_d/t_r$