Randomized algorithms in computability theory

Laurent Bienvenu (CNRS & University of Paris 7)

Logic Colloquium, Leeds, UK

August 1st, 2016

1. How useful is randomness?

How useful is randomness? (1)

Whether having access to a ‘random source’ can help use achieve more than what we could do without is perhaps one of the most fundamental questions in theoretical computer science.

P: Class of languages which can be decided in (deterministic) polynomial time.

BPP: Class of languages which can be decided in polynomial time if given access to a random source, with probability, say, 0.99.

Open question: Does P = BPP?

How useful is randomness? (2)

The analogue question in computability theory has been resolved a long time ago.

How useful is randomness? (2)

The analogue question in computability theory has been resolved a long time ago.

Theorem (De Leeuw-Moore-Shannon-Shapiro, 1956)

Let X be an infinite binary sequence (or language). If there is an algorithm (machine) Φ with access to a random source R (also a sequence of bits) such that

$$Pr[\Phi(R) = X] > 0,$$

then X is computable.

How useful is randomness? (2)

The analogue question in computability theory has been resolved a long time ago.

Theorem (De Leeuw-Moore-Shannon-Shapiro, 1956)
Let X be an infinite binary sequence (or language). If there is an algorithm (machine) Φ with access to a random source R (also a sequence of bits) such that

$$Pr[\Phi(R) = X] > 0,$$

then X is computable.

Proof. If the probability is $> 1/2$, find the value of each bit of X by a 'majority vote'. If not, apply the Lebesgue density theorem to get a relative probability $> 1/2$ and do the same.

Mass problems (1)

But the story does not end here if we consider **mass problems** (computability analogue of search problems in complexity theory).

Definition

A *mass problem* is a set $\mathcal{A}$ of infinite binary sequences. Its members can be viewed as its ‘solutions’.

Mass problems (1)

But the story does not end here if we consider **mass problems** (computability analogue of search problems in complexity theory).

Definition

A *mass problem* is a set $\mathcal{A}$ of infinite binary sequences. Its members can be viewed as its 'solutions'.

Definition

$\mathcal{A}$ is easier (not harder) to solve than $\mathcal{B}$ if we can computably get *some* solution of $\mathcal{A}$ from *any* solution of $\mathcal{B}$.

Non-uniform version, noted $\mathcal{A} \leq_w \mathcal{B}$:

$$\text{for every } X \in \mathcal{B} \text{ there is } \Phi \text{ such that } \Phi(X) \in \mathcal{A}$$

Uniform version, noted $\mathcal{A} \leq_s \mathcal{B}$:

$$\text{there is } \Phi \text{ such that for every } X \in \mathcal{B},\ \Phi(X) \in \mathcal{A}$$

Mass problems (2)

A mass problem is trivial (easier than any other one) iff it contains a computable member.

Mass problems (2)

A mass problem is trivial (easier than any other one) iff it contains a computable member. Examples of non-trivial problems:

$\mathbf{nCOMP}$: non-computable sequences(!).

Mass problems (2)

A mass problem is trivial (easier than any other one) iff it contains a computable member. Examples of non-trivial problems:

$\mathbf{nCOMP}$: non-computable sequences(!).

$\mathbf{PA}$: complete, coherent extensions of Peano Arithmetic.

Mass problems (2)

A mass problem is trivial (easier than any other one) iff it contains a computable member. Examples of non-trivial problems:

$\mathbf{nCOMP}$: non-computable sequences(!).

$\mathbf{PA}$: complete, coherent extensions of Peano Arithmetic.

$\mathbf{HIGH}$: functions $f : \mathbb{N} \to \mathbb{N}$ which dominate all computable functions (for every g computable, $g(n) \leq f(n)$ for all but finitely many n).

Mass problems (2)

A mass problem is trivial (easier than any other one) iff it contains a computable member. Examples of non-trivial problems:

$\mathbf{nCOMP}$: non-computable sequences(!).

$\mathbf{PA}$: complete, coherent extensions of Peano Arithmetic.

$\mathbf{HIGH}$: functions $f : \mathbb{N} \to \mathbb{N}$ which dominate all computable functions (for every g computable, $g(n) \leq f(n)$ for all but finitely many n).

$\mathbf{HI}$: functions $f : \mathbb{N} \to \mathbb{N}$ which are dominated by no computable function (for every g computable, $g(n) \leq f(n)$ for infinitely many n).

Mass problems (2)

A mass problem is trivial (easier than any other one) iff it contains a computable member. Examples of non-trivial problems:

$\mathbf{nCOMP}$: non-computable sequences(!).

$\mathbf{PA}$: complete, coherent extensions of Peano Arithmetic.

$\mathbf{HIGH}$: functions $f : \mathbb{N} \to \mathbb{N}$ which dominate all computable functions (for every g computable, $g(n) \leq f(n)$ for all but finitely many n).

$\mathbf{HI}$: functions $f : \mathbb{N} \to \mathbb{N}$ which are dominated by no computable function (for every g computable, $g(n) \leq f(n)$ for infinitely many n).

$\mathbf{DNC}$: functions $f : \mathbb{N} \to \mathbb{N}$ such that $f(e) \neq \phi_e(e)$ whenever $\phi_e(e)$ is defined.

Mass problems (2)

A mass problem is trivial (easier than any other one) iff it contains a computable member. Examples of non-trivial problems:

$\mathbf{nCOMP}$: non-computable sequences(!).

$\mathbf{PA}$: complete, coherent extensions of Peano Arithmetic.

$\mathbf{HIGH}$: functions $f : \mathbb{N} \to \mathbb{N}$ which dominate all computable functions (for every g computable, $g(n) \leq f(n)$ for all but finitely many n).

$\mathbf{HI}$: functions $f : \mathbb{N} \to \mathbb{N}$ which are dominated by no computable function (for every g computable, $g(n) \leq f(n)$ for infinitely many n).

$\mathbf{DNC}$: functions $f : \mathbb{N} \to \mathbb{N}$ such that $f(e) \neq \phi_e(e)$ whenever $\phi_e(e)$ is defined.

$\mathbf{DNC}_{\mathrm{bis}}$: functions $f : \mathbb{N} \to \mathbb{N}$ such that $\mathrm{K}(f(n)) > n$.

Mass problems (3)

If we now allow probabilistic computations, then some non-trivial mass problems become easily 'solvable'.

Mass problems (3)

If we now allow probabilistic computations, then some non-trivial mass problems become easily ‘solvable’.

- The most obvious is $\mathbf{nCOMP}$: just output your random source!

Mass problems (3)

If we now allow probabilistic computations, then some non-trivial mass problems become easily ‘solvable’.

• The most obvious is $\mathbf{nCOMP}$: just output your random source!

• $\mathbf{DNC}$ is also pretty easy: for all e we are trying to avoid a single value: $\phi_e(e)$ (if it is even defined). Thus it suffices to pick the value of $f(e)$ at random between 0 and $q(e)$, for a function q such that $\prod_e (1 - 1/q(e)) > 0$.

Mass problems (3)

If we now allow probabilistic computations, then some non-trivial mass problems become easily 'solvable'.

• The most obvious is $\mathbf{nCOMP}$: just output your random source!

• $\mathbf{DNC}$ is also pretty easy: for all e we are trying to avoid a single value: $\phi_e(e)$ (if it is even defined). Thus it suffices to pick the value of $f(e)$ at random between 0 and $q(e)$, for a function q such that $\prod_e(1 - 1/q(e)) > 0$.

• Much more interesting is the case of $\mathbf{HI}$ (functions $f : \mathbb{N} \to \mathbb{N}$ not dominated by any computable one)... We will come back to it later.

Mass problems (4)

Of course a lot of mass problems that are unsolvable by deterministic means remain unsolvable if we allow randomized algorithms. For example:

Mass problems (4)

Of course a lot of mass problems that are unsolvable by deterministic means remain unsolvable if we allow randomized algorithms. For example:

$\mathbf{HIGH}$: functions $f : \mathbb{N} \to \mathbb{N}$ which dominate all computable functions (for every g computable, $g(n) \leq f(n)$ for all but finitely many n).

Mass problems (4)

Of course a lot of mass problems that are unsolvable by deterministic means remain unsolvable if we allow randomized algorithms. For example:

$\mathbf{HIGH}$: functions $f : \mathbb{N} \to \mathbb{N}$ which dominate all computable functions (for every g computable, $g(n) \leq f(n)$ for all but finitely many n).

Indeed, if some Φ generates a member of $\mathbf{HIGH}$ with positive probability, then we can again use Lebesgue's density theorem to get a Ψ such that Ψ generates a member of $\mathbf{HIGH}$ with probability $> 2/3$.

Mass problems (4)

Of course a lot of mass problems that are unsolvable by deterministic means remain unsolvable if we allow randomized algorithms. For example:

$\mathbf{HIGH}$: functions $f : \mathbb{N} \to \mathbb{N}$ which dominate all computable functions (for every g computable, $g(n) \leq f(n)$ for all but finitely many n).

Indeed, if some Φ generates a member of $\mathbf{HIGH}$ with positive probability, then we can again use Lebesgue's density theorem to get a Ψ such that Ψ generates a member of $\mathbf{HIGH}$ with probability $> 2/3$.

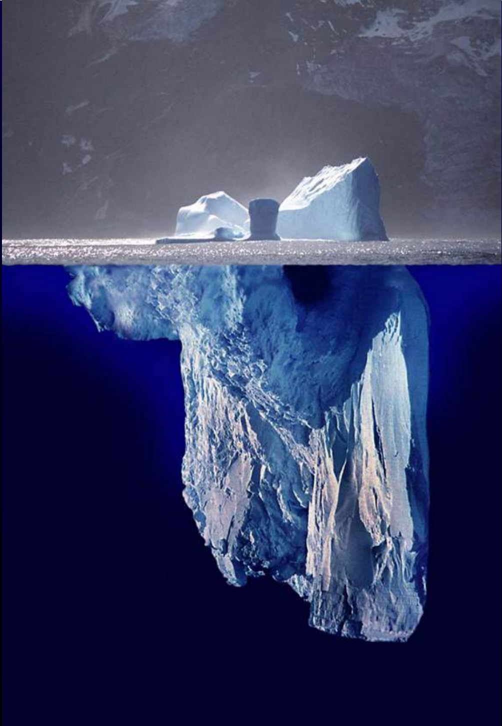
Then we can apply a version of 'majority vote': for each n, wait until $\Psi(R)(n)$ returns a value for $2/3$ of R's. Take $g(n)$ = the maximum value seen over all those R's. Then g is computable and for every n:

$$Pr[\Psi(R)(n) > g(n)] \leq 1/3$$

and by Fatou's lemma,

$$Pr[\Psi(R) \text{ dominates } g] \leq 1/3$$

2. Randomness vs depth



Completing PA ‘at random’

We now turn our attention to

$\mathbf{PA}$: complete, coherent extensions of Peano Arithmetic.

Completing PA 'at random'

We now turn our attention to

$\mathbf{PA}$: complete, coherent extensions of Peano Arithmetic.

As argued by Levin, imagine that members of $\mathbf{PA}$ could be generated via a randomized algorithm... this would be a hard blow to Gödel's incompleteness theorem!

Completing PA ‘at random’

We now turn our attention to

$\mathbf{PA}$: complete, coherent extensions of Peano Arithmetic.

As argued by Levin, imagine that members of $\mathbf{PA}$ could be generated via a randomized algorithm... this would be a hard blow to Gödel’s incompleteness theorem!

(Un)fortunately,

Theorem (Jockusch-Soare 1972)

For any probabilistic algorithm Φ,

$$Pr[\Phi(R) \in \mathbf{PA}] = 0$$

Completing PA ‘at random’

We now turn our attention to

$\mathbf{PA}$: complete, coherent extensions of Peano Arithmetic.

As argued by Levin, imagine that members of $\mathbf{PA}$ could be generated via a randomized algorithm... this would be a hard blow to Gödel’s incompleteness theorem!

(Un)fortunately,

Theorem (Jockusch-Soare 1972)

For any probabilistic algorithm Φ,

$$Pr[\Phi(R) \in \mathbf{PA}] = 0$$

(A later improvement by F. Stephan: if R is Martin-Löf random and computes a member of $\mathbf{PA}$, then R computes the halting problem $\emptyset'$).

Measuring the difficulty

Levin (2013): even generating a large but finite initial segment of a completion of $\mathbf{PA}$ would be great!

Measuring the difficulty

Levin (2013): even generating a large but finite initial segment of a completion of $\mathbf{PA}$ would be great! Can we quantify how unlikely this is?

Measuring the difficulty

Levin (2013): even generating a large but finite initial segment of a completion of $\mathbf{PA}$ would be great! Can we quantify how unlikely this is?

The right tool:

Definition

There is a **universal probabilistic algorithm** Ξ, that is, for any probabilistic algorithm Φ, for some constant c all every class $\mathcal{C}$ of finite and infinite objects:

$$\text{Pr}[\Phi \text{ generates a member of } \mathcal{C}] \leq c \cdot \text{Pr}[\Xi \text{ generates a member of } \mathcal{C}]$$

Measuring the difficulty

Levin (2013): even generating a large but finite initial segment of a completion of $\mathbf{PA}$ would be great! Can we quantify how unlikely this is?

The right tool:

Definition

There is a **universal probabilistic algorithm** Ξ, that is, for any probabilistic algorithm Φ, for some constant c all every class $\mathcal{C}$ of finite and infinite objects:

$$\text{Pr}[\Phi \text{ generates a member of } \mathcal{C}] \leq c \cdot \text{Pr}[\Xi \text{ generates a member of } \mathcal{C}]$$

Thus we can set $\mathbf{M}(\mathcal{C}) = \text{Pr}[\Xi \text{ generates a member of } \mathcal{C}]$, which is independent of the choice of Ξ up to a multiplicative constant.

Measuring the difficulty

Levin (2013): even generating a large but finite initial segment of a completion of $\mathbf{PA}$ would be great! Can we quantify how unlikely this is?

The right tool:

Definition

There is a **universal probabilistic algorithm** Ξ, that is, for any probabilistic algorithm Φ, for some constant c all every class $\mathcal{C}$ of finite and infinite objects:

$$\Pr[\Phi \text{ generates a member of } \mathcal{C}] \leq c \cdot \Pr[\Xi \text{ generates a member of } \mathcal{C}]$$

Thus we can set $\mathbf{M}(\mathcal{C}) = \Pr[\Xi \text{ generates a member of } \mathcal{C}]$, which is independent of the choice of Ξ up to a multiplicative constant.

Levin's coding theorem: $\mathbf{M}(\{x\}) = 2^{-\mathrm{K}(x)} \cdot O(1)$

Measuring the difficulty (2)

Let $\mathbf{PA}_n$ be the set of coherent finite theories of arithmetical formulas such that for every formula ψ encodable in n bits, ψ or $\neg\psi$ is in the theory (think of $\mathbf{PA}_n$ as the set of strings of length 2^n which can be extended into a member of $\mathbf{PA}$).

Measuring the difficulty (2)

Let $\mathbf{PA}_n$ be the set of coherent finite theories of arithmetical formulas such that for every formula ψ encodable in n bits, ψ or $\neg\psi$ is in the theory (think of $\mathbf{PA}_n$ as the set of strings of length 2^n which can be extended into a member of $\mathbf{PA}$).

Theorem (Levin 2013)
$\mathbf{M}(\mathbf{PA}_n) \leq 2^{-n+O(1)}$

Measuring the difficulty (2)

Let $\mathbf{PA}_n$ be the set of coherent finite theories of arithmetical formulas such that for every formula ψ encodable in n bits, ψ or $\neg\psi$ is in the theory (think of $\mathbf{PA}_n$ as the set of strings of length 2^n which can be extended into a member of $\mathbf{PA}$).

Theorem (Levin 2013)
$\mathbf{M}(\mathbf{PA}_n) \leq 2^{-n+O(1)}$

In fact, he proved more:

Theorem (Levin 2013)
For any $x \in \mathbf{PA}_n$, $\mathbf{I}(x : \emptyset') \gtrsim n - O(1)$

Measuring the difficulty (2)

Let $\mathbf{PA}_n$ be the set of coherent finite theories of arithmetical formulas such that for every formula ψ encodable in n bits, ψ or $\neg\psi$ is in the theory (think of $\mathbf{PA}_n$ as the set of strings of length 2^n which can be extended into a member of $\mathbf{PA}$).

Theorem (Levin 2013)
$\mathbf{M}(\mathbf{PA}_n) \leq 2^{-n+O(1)}$

In fact, he proved more:

Theorem (Levin 2013)
For any $x \in \mathbf{PA}_n, \quad \mathbf{I}(x : \emptyset') \gtrsim n - O(1)$

This is better since, for Φ a randomized algorithm and Z a finite or infinite object,

$$\mathbb{E}\left(2^{\mathbf{I}(\Phi(R):Z)}\right) = O(1)$$

(Zvonkin-Levin's information conservation theorem)

A philosophical interlude

The reason Levin cared more about this second result is of philosophical nature.

A philosophical interlude

The reason Levin cared more about this second result is of philosophical nature.

He wanted to show that there is **no physical mean** that allows us to get a completion of Peano arithmetic (or even a large finite initial segment thereof), thus strengthening Gödel’s theorem.

A philosophical interlude

The reason Levin cared more about this second result is of philosophical nature.

He wanted to show that there is **no physical mean** that allows us to get a completion of Peano arithmetic (or even a large finite initial segment thereof), thus strengthening Gödel’s theorem.

To do this, he proposes the following thesis:

A philosophical interlude

The reason Levin cared more about this second result is of philosophical nature.

He wanted to show that there is **no physical mean** that allows us to get a completion of Peano arithmetic (or even a large finite initial segment thereof), thus strengthening Gödel's theorem.

To do this, he proposes the following thesis:

Independence Postulate (IP): if *x* is an object obtainable in the physical world, and *y* is a mathematically definable sequence, then $\mathbf{I}(x : y)$ ought to be small.

A philosophical interlude

The reason Levin cared more about this second result is of philosophical nature.

He wanted to show that there is **no physical mean** that allows us to get a completion of Peano arithmetic (or even a large finite initial segment thereof), thus strengthening Gödel's theorem.

To do this, he proposes the following thesis:

Independence Postulate (IP): if x is an object obtainable in the physical world, and y is a mathematically definable sequence, then $\mathbf{I}(x : y)$ ought to be small.

In particular, for x physically obtainable, $\mathbf{I}(x : \emptyset')$ is small. On the other hand completions of PA have high common information with $\emptyset'$ (Levin's theorem). Thus they cannot be physically obtainable!

Deep classes (1)

Regardless of whether one believes in the IP, one can see that Levin's argument works under more general assumptions:

Deep classes (1)

Regardless of whether one believes in the IP, one can see that Levin's argument works under more general assumptions:

Definition (B., Porter)
Let $\mathcal{P}$ be a Π^0_1 class. Let $\mathcal{P}_n$ be a set of finite strings of length n which can be extended to an element of $\mathcal{P}$. We say that $\mathcal{P}$ is **deep** if

$$\mathbf{M}(\mathcal{P}_n) \leq \frac{1}{h(n)}$$

for some computable h which tends to infinity.

Deep classes (2)

It turns out that there many Π^0_1 classes studied in the literature are deep.

Deep classes (2)

It turns out that there many Π^0_1 classes studied in the literature are deep.

Theorem (B., Porter)

The following Π^0_1 classes are deep:

- Levin complex sequences: binary sequences such that $\mathrm{K}(X_n \ldots X_{n+k}) \geq 0.9\,k$ for all n and all $k \geq c$ (**after Rumyantsev, Khan**).
- $\mathbf{DNC}_q$, with $\prod_n (1 - 1/q(n)) = 0$: functions $f : \mathbb{N} \to \mathbb{N}$ such that $f(e) \neq \phi_e(e)$ and $f(e) \leq q(e)$ (**after Miller**).
- Sequences of sets $(F_0, F_1, \ldots)$ where F_i is a finite set of strings of length i, and $card(F_i) \geq f(i)$ for some computable non-decreasing f tending to ∞.
- ...

Deep classes (3)

The interesting thing is that even when the corresponding mass problem is easier than $\mathbf{PA}$, a deep Π^0_1 class 'behaves like $\mathbf{PA}$' in its interactions with randomness. For example:

Theorem (B., Porter)

- If R is Martin-Löf random and does not compute $\emptyset'$, then R does not compute any element of a deep Π^0_1 class (Stephan for $\mathbf{PA}$).
- This remains true for $R \oplus A$, when A is K-trivial (Miller-Day for $\mathbf{PA}$).

Why ‘deep’? (1)

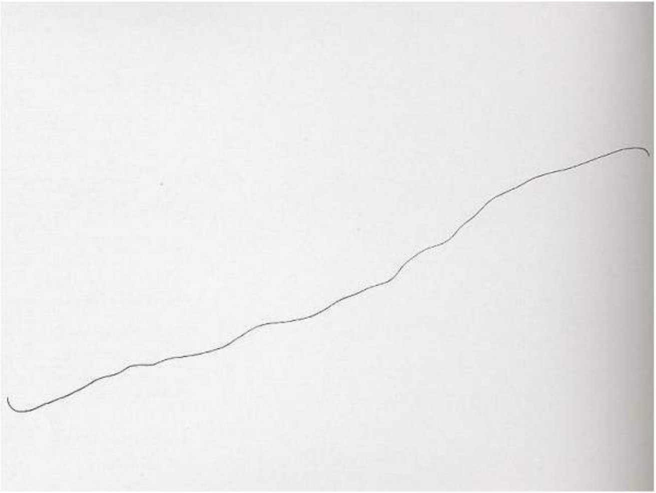
Deep classes were named this way as their elements seem highly ‘structured’, **neither too simple nor obtainable ‘by chance’.**

Why ‘deep’? (1)

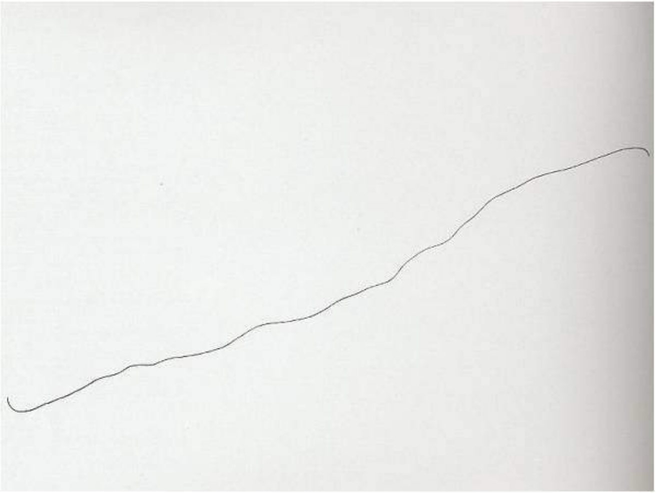
Deep classes were named this way as their elements seem highly ‘structured’, **neither too simple nor obtainable ‘by chance’.**

This relates to an old idea due to Bennett, who argued that Kolmogorov complexity captures the idea of ‘information’, but not of ‘depth’.

Why ‘deep’? (2)



Why ‘deep’? (2)



shallow: too simple!

Why ‘deep’? (3)

Why ‘deep’? (3)

shallow: too random!

Why ‘deep’? (4)

Why ‘deep’? (4)

non-random / compressible...

Why ‘deep’? (4)

non-random / compressible... but deep!

Logical depth

Bennett's idea: **what constitutes depth is not how simple an object is to describe, it is how long it takes us to build the object from its shortest description.**

Logical depth

Bennett's idea: **what constitutes depth is not how simple an object is to describe, it is how long it takes us to build the object from its shortest description.**

In other words, its Kolmogorov complexity is much smaller than its time-bounded Kolmogorov complexity.

Logical depth

Bennett's idea: **what constitutes depth is not how simple an object is to describe, it is how long it takes us to build the object from its shortest description.**

In other words, its Kolmogorov complexity is much smaller than its time-bounded Kolmogorov complexity.

Definition (after Bennett 1988)

An infinite binary sequence X is *logically deep* if for every computable time bound (function) T,

$$\mathrm{K}^T(X_0 \ldots X_n) - \mathrm{K}(X_0 \ldots X_n) \to \infty$$

Logical depth

It turned out that the connection between deep classes and logical depth is more than a mere analogy!

Logical depth

It turned out that the connection between deep classes and logical depth is more than a mere analogy!

Theorem (B., Porter)
Every member of a deep Π^0_1 class is logically deep.

Logical depth

It turned out that the connection between deep classes and logical depth is more than a mere analogy!

Theorem (B., Porter)
Every member of a deep Π^0_1 class is logically deep.

(It is not true however that a Π^0_1 class whose members are all logically deep must be deep).

3. When randomness helps

Getting a hyperimmune function ‘at random’ (1)

We come back to the mass problem

$\mathbf{HI}$: functions $f : \mathbb{N} \to \mathbb{N}$ which are dominated by no computable function (for every g computable, $g(n) \leq f(n)$ for infinitely many n).

Getting a hyperimmune function ‘at random’ (1)

We come back to the mass problem

$\mathbf{HI}$: functions $f : \mathbb{N} \to \mathbb{N}$ which are dominated by no computable function (for every g computable, $g(n) \leq f(n)$ for infinitely many n).

This one **does** admit a probabilistic algorithm, due to Kautz (1991), and clarified by Gács and Shen (2012).

Getting a hyperimmune function ‘at random’ (2)

We try to build a function $f : \mathbb{N} \rightarrow \mathbb{N}$ not dominated by any total ϕ_e for any e.

Getting a hyperimmune function 'at random' (2)

We try to build a function $f : \mathbb{N} \to \mathbb{N}$ not dominated by any total ϕ_e for any e.

For a given e, let us pick an n,

Getting a hyperimmune function 'at random' (2)

We try to build a function $f : \mathbb{N} \to \mathbb{N}$ not dominated by any total ϕ_e for any e.

For a given e, let us pick an n,

- Case 1: If $\phi_e(n)$ is undefined, we have nothing to do! (for this e)

Getting a hyperimmune function ‘at random’ (2)

We try to build a function $f : \mathbb{N} \to \mathbb{N}$ not dominated by any total ϕ_e for any e.

For a given e, let us pick an n,

- Case 1: If $\phi_e(n)$ is undefined, we have nothing to do! (for this e)
- Case 2: If $\phi_e(n)$ is defined, all we need to do is set $f(n) = \phi_e(n) + 1$

Getting a hyperimmune function 'at random' (2)

We try to build a function $f : \mathbb{N} \to \mathbb{N}$ not dominated by any total ϕ_e for any e.

For a given e, let us pick an n,

- Case 1: If $\phi_e(n)$ is undefined, we have nothing to do! (for this e)
- Case 2: If $\phi_e(n)$ is defined, all we need to do is set $f(n) = \phi_e(n) + 1$

In both cases we have a very simple way to satisfy the requirement computably.
But it is the distinction between the two cases which is not computable!

Getting a hyperimmune function ‘at random’ (2)

We try to build a function $f : \mathbb{N} \to \mathbb{N}$ not dominated by any total ϕ_e for any e.

For a given e, let us pick an n,

- Case 1: If $\phi_e(n)$ is undefined, we have nothing to do! (for this e)
- Case 2: If $\phi_e(n)$ is defined, all we need to do is set $f(n) = \phi_e(n) + 1$

In both cases we have a very simple way to satisfy the requirement computably.
But it is the distinction between the two cases which is not computable!

However, there is a probabilistic way to do this, via a **fireworks argument**.

Fireworks (1)

Suppose we walk into a fireworks shop.

- The fireworks sold there are very cheap so we are suspicious that some of them are defective.
- Since they are cheap we can ask the owner to test a few of them before buying one.
- **Our goal: either buy a good one (untested) and take it home OR get the owner to fail a test, and then sue him.**

Fireworks (2)

Clearly there is no deterministic strategy which works in all cases. There is however a good probabilistic strategy, which wins with probability at least $n/(n+1)$ in all cases, where n is the number of fireworks boxes in the shop:

Fireworks (2)

Clearly there is no deterministic strategy which works in all cases. There is however a good probabilistic strategy, which wins with probability at least $n/(n+1)$ in all cases, where n is the number of fireworks boxes in the shop:

- Pick a number k at random between 0 and n
- Test the k first fireworks
- Buy the $(k+1)$-st box (unless $k = n$)

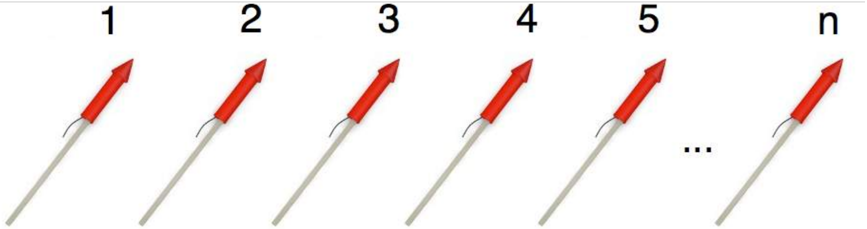
Fireworks (2)

Clearly there is no deterministic strategy which works in all cases. There is however a good probabilistic strategy, which wins with probability at least $n/(n+1)$ in all cases, where n is the number of fireworks boxes in the shop:

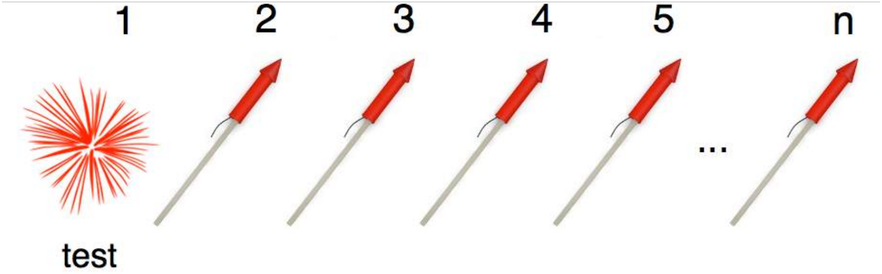
- Pick a number k at random between 0 and n
- Test the k first fireworks
- Buy the $(k+1)$-st box (unless $k = n$)

This works because the only bad case is when $k+1$ is the position of the first bad box.

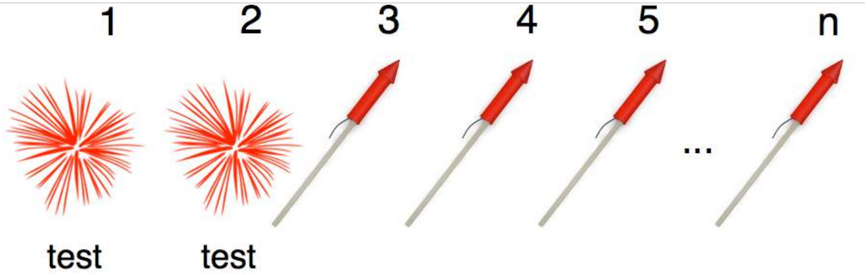
Fireworks (3)



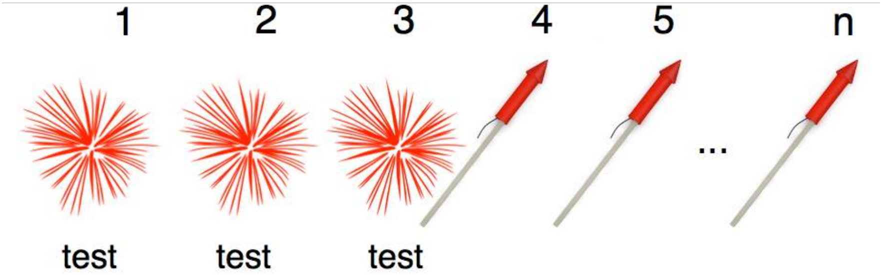
Fireworks (3)



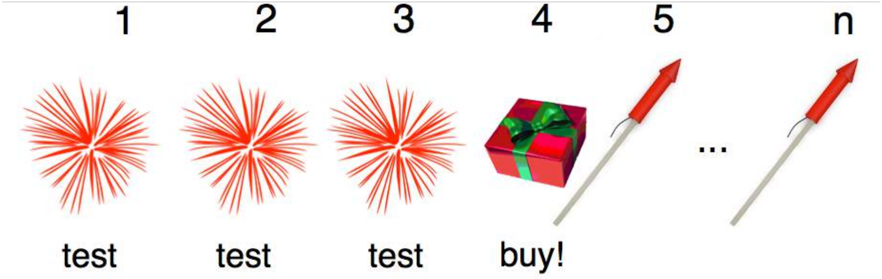
Fireworks (3)



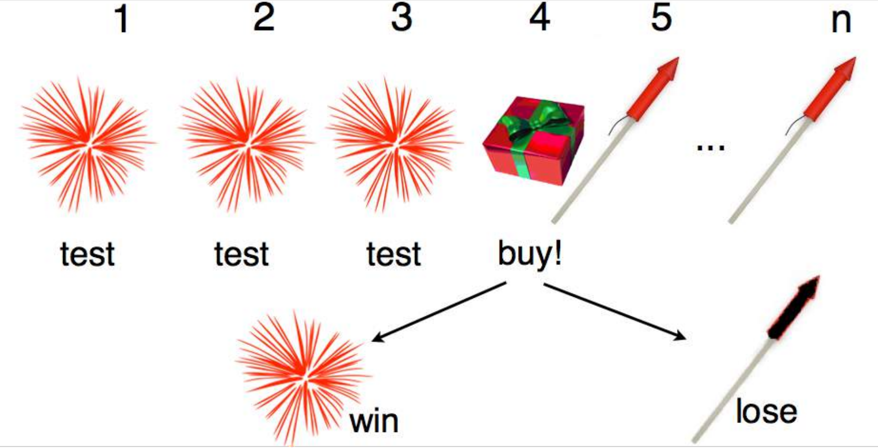
Fireworks (3)



Fireworks (3)



Fireworks (3)



Fireworks (4)

Back to our construction of $f : \mathbb{N} \to \mathbb{N}$, where we want to satisfy for all e:

Fireworks (4)

Back to our construction of $f : \mathbb{N} \to \mathbb{N}$, where we want to satisfy for all e:

$(\mathcal{R}_e)$: either ϕ_e is partial or ϕ_e does not dominate f.

Fireworks (4)

Back to our construction of $f : \mathbb{N} \to \mathbb{N}$, where we want to satisfy for all e:

$(\mathcal{R}_e)$: either ϕ_e is partial or ϕ_e does not dominate f.

The algorithm for a requirement e:

Step 1 Pick a number k_e between 1 and $q(e)$ at random, with $\prod_e (1 - 1/q(e)) > 0$. Set the 'error counter' to 0

Fireworks (4)

Back to our construction of $f : \mathbb{N} \to \mathbb{N}$, where we want to satisfy for all e:

$$(\mathcal{R}_e)\text{: either } \phi_e \text{ is partial or } \phi_e \text{ does not dominate } f.$$

The algorithm for a requirement e:

Step 1 Pick a number k_e between 1 and $q(e)$ at random, with $\prod_e(1 - 1/q(e)) > 0$. Set the 'error counter' to 0

Step 2
- Pick the smallest n on which f has not yet been defined.
- Set $f(n) = 0$ (here we are 'guessing' that $\phi_e(n)$ is undefined)
- Start handling other requirements until we see that $\phi_e(n)$ is in fact defined, then increase the error counter by 1
- If the error counter is $< k_e$, go back to the beginning of Step 2; if it is $= k_e$, go to Step 3.

Fireworks (4)

Back to our construction of $f : \mathbb{N} \to \mathbb{N}$, where we want to satisfy for all e:

$$(\mathcal{R}_e)\text{: either } \phi_e \text{ is partial or } \phi_e \text{ does not dominate } f.$$

The algorithm for a requirement e:

Step 1 Pick a number k_e between 1 and $q(e)$ at random, with $\prod_e (1 - 1/q(e)) > 0$. Set the 'error counter' to 0

Step 2
- Pick the smallest n on which f has not yet been defined.
- Set $f(n) = 0$ (here we are 'guessing' that $\phi_e(n)$ is undefined)
- Start handling other requirements until we see that $\phi_e(n)$ is in fact defined, then increase the error counter by 1
- If the error counter is $< k_e$, go back to the beginning of Step 2; if it is $= k_e$, go to Step 3.

Step 3 Pick a fresh m, and define $f(m) = \phi_e(m)$

Fireworks (5)

It is not too hard to argue that the algorithm succeeds with probability at least $\prod_e (1 - 1/q(e))$.

Fireworks (5)

It is not too hard to argue that the algorithm succeeds with probability at least $\prod_e (1 - 1/q(e))$.

Thus we have:

Theorem (Kautz)

There is Φ such that

$$Pr[\Phi(R) \in \mathbf{HI}] > 0$$

Fireworks (5)

It is not too hard to argue that the algorithm succeeds with probability at least $\prod_e (1 - 1/q(e))$.

Thus we have:

Theorem (Kautz)

There is Φ such that

$$Pr[\Phi(R) \in \mathbf{HI}] > 0$$

In fact, Kautz showed: **Every sequence which is Martin-Löf random relative to $\emptyset'$ computes a function in $\mathbf{HI}$.**

Fireworks (6)

Using fireworks arguments, we can show that the following mass problems can be solved probabilistically:

Fireworks (6)

Using fireworks arguments, we can show that the following mass problems can be solved probabilistically:

- Set of 1-generic binary sequences (Kautz)

Fireworks (6)

Using fireworks arguments, we can show that the following mass problems can be solved probabilistically:

- Set of 1-generic binary sequences (Kautz)
- Set of pairs (A, B) such that B is c.e. relative to A, and $B >_T A$ (Kautz)

Fireworks (6)

Using fireworks arguments, we can show that the following mass problems can be solved probabilistically:

- Set of 1-generic binary sequences (Kautz)
- Set of pairs (A, B) such that B is c.e. relative to A, and $B >_T A$ (Kautz)
- Set of X which are in $\mathbf{DNC}$ but compute no Martin-Löf random sequence (B.-Patey)

Fireworks (6)

Using fireworks arguments, we can show that the following mass problems can be solved probabilistically:

- Set of 1-generic binary sequences (Kautz)
- Set of pairs (A, B) such that B is c.e. relative to A, and $B >_T A$ (Kautz)
- Set of X which are in $\mathbf{DNC}$ but compute no Martin-Löf random sequence (B.-Patey)

Question: are there other types of non-trivial probabilistic algorithms which could apply to computability theory? (currently no other known type)

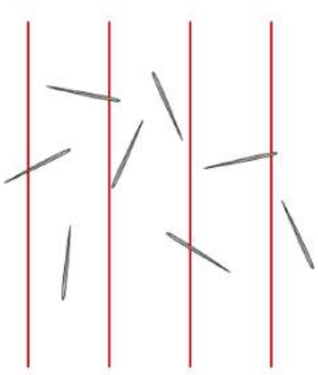
Turning De Leeuw et al's theorem around

One last interesting aspect of randomized algorithms: showing computability!

Turning De Leeuw et al's theorem around

One last interesting aspect of randomized algorithms: showing computability!

Buffon's needle shows that π is a computable number ;-)
(one can use the needle to get a probabilistic algorithm to compute π, thus by De Leeuw et al's theorem π is computable)

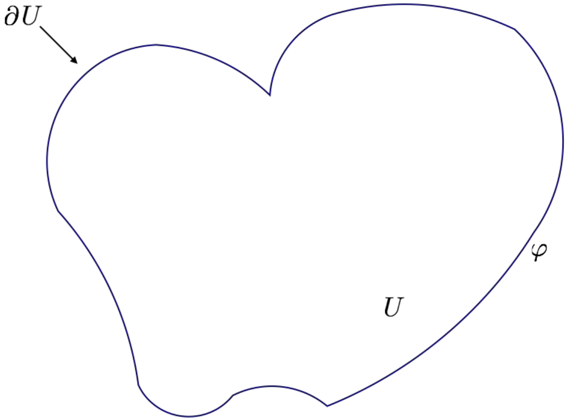


Dirichlet's problem (1)

More interestingly, consider the computable version of **Dirichlet's problem**:

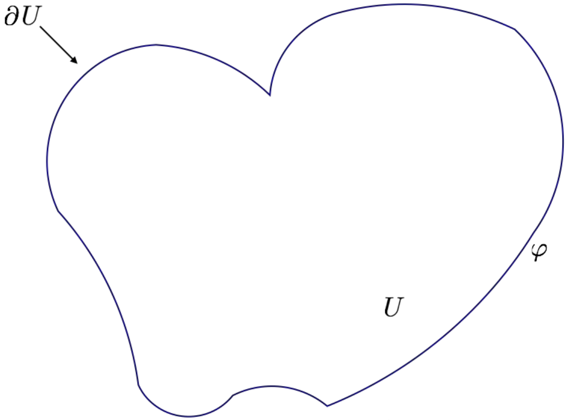
Dirichlet's problem (1)

More interestingly, consider the computable version of **Dirichlet's problem**:



Dirichlet's problem (1)

More interestingly, consider the computable version of **Dirichlet's problem**:



$$\text{Dirichlet's problem: find } f \text{ s.t.} \begin{cases} f \text{ continuous on } \bar{U} \\ \Delta f = 0 \\ f = \varphi \text{ on } \partial U \end{cases}$$

Dirichlet's problem (2)

Under reasonable assumptions (U bounded, ∂U sufficiently smooth), there is exactly one solution to Dirichlet's problem.

Dirichlet's problem (2)

Under reasonable assumptions (U bounded, ∂U sufficiently smooth), there is exactly one solution to Dirichlet's problem.

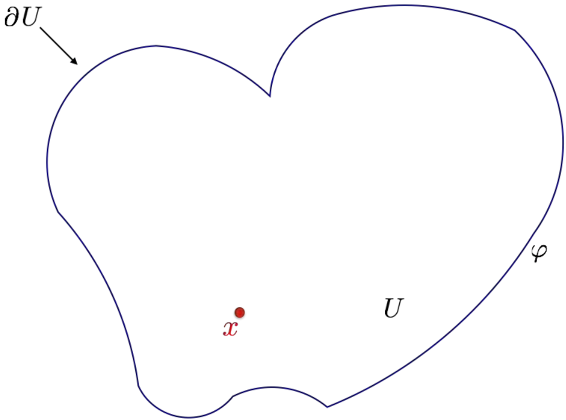
Is this solution computable? (= when ∂U, φ are computable, is f computable?)

Dirichlet's problem (3)

A fascinating result of random processes is that the unique solution can be found via **Brownian motion**.

Dirichlet's problem (3)

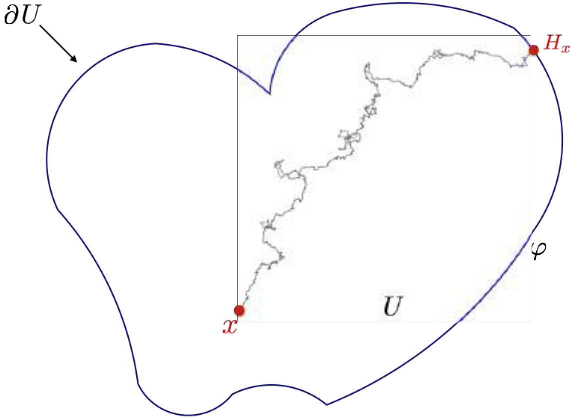
A fascinating result of random processes is that the unique solution can be found via **Brownian motion**.



$$\text{Dirichlet's problem: find } f \text{ s.t.} \begin{cases} f \text{ continuous on } \bar{U} \\ \Delta f = 0 \\ f = \varphi \text{ on } \partial U \end{cases}$$

Dirichlet's problem (3)

A fascinating result of random processes is that the unique solution can be found via **Brownian motion**.



$$\text{Dirichlet's problem: find } f \text{ s.t.} \begin{cases} f \text{ continuous on } \bar{U} \\ \Delta f = 0 \\ f = \varphi \text{ on } \partial U \end{cases}$$

Dirichlet's problem (4)

Theorem

The function $f : x \mapsto \mathbb{E}(\varphi(H_x))$ is the unique solution to Dirichlet's problem.

Dirichlet's problem (4)

Theorem

The function $f : x \mapsto \mathbb{E}(\varphi(H_x))$ is the unique solution to Dirichlet's problem.

In the computable setting:

Dirichlet's problem (4)

Theorem
The function $f : x \mapsto \mathbb{E}(\varphi(H_x))$ is the unique solution to Dirichlet's problem.

In the computable setting:

Theorem (Allen-B.-Slaman)
Given x and a random source R, one can compute a Brownian path starting from x and compute its first intersection with ∂U.

Thus, we have a probabilistic algorithm to compute $f(x)$ given x!

Thus f is computable! (by De Leeuw et al's theorem, essentially)

In conclusion...

S. Barry Cooper (1943-2015)

Thank you, Barry !