Exponential time vs probabilistic polynomial time

Sylvain Perifel (LIAFA, Paris)

Dagstuhl – January 10, 2012

Introduction

Probabilistic algorithms:

- can toss a coin
- polynomial time (worst case)
- probability of error $< \epsilon$

$\rightarrow$ class BPP.

Introduction

Probabilistic algorithms:

- can toss a coin
- polynomial time (worst case)
- probability of error $< \epsilon$

$\rightarrow$ class BPP.

Common belief:
they can be *derandomized*

(true if some
circuit lower bounds hold)

However…

Even if it is believed that

$$\mathsf{BPP} = \mathsf{P}$$

it is open whether

$$\mathsf{NEXP} \neq \mathsf{BPP} \quad (!)$$

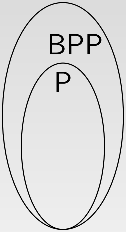
However...

Even if it is believed that

$$\mathsf{BPP} = \mathsf{P}$$

it is open whether

$$\mathsf{NEXP} \neq \mathsf{BPP} \quad (!)$$



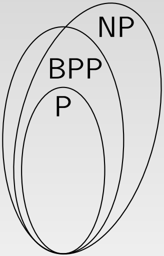
However…

Even if it is believed that

$$\mathsf{BPP} = \mathsf{P}$$

it is open whether

$$\mathsf{NEXP} \neq \mathsf{BPP} \quad (!)$$



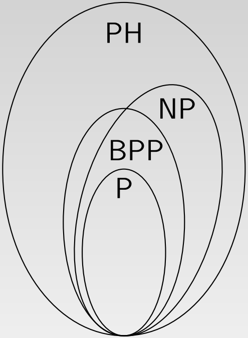
However…

Even if it is believed that

$$\mathsf{BPP} = \mathsf{P}$$

it is open whether

$$\mathsf{NEXP} \neq \mathsf{BPP} \quad (!)$$



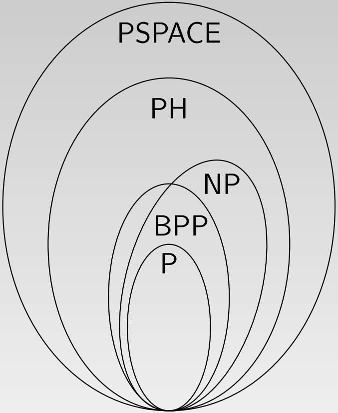
However…

Even if it is believed that

$$\mathsf{BPP} = \mathsf{P}$$

it is open whether

$$\mathsf{NEXP} \neq \mathsf{BPP} \quad (!)$$



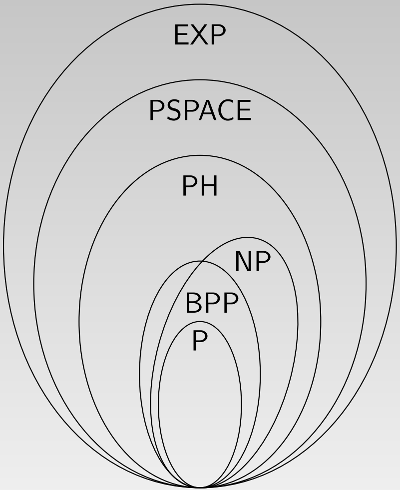
However…

Even if it is believed that

$$\mathsf{BPP} = \mathsf{P}$$

it is open whether

$$\mathsf{NEXP} \neq \mathsf{BPP} \quad (!)$$



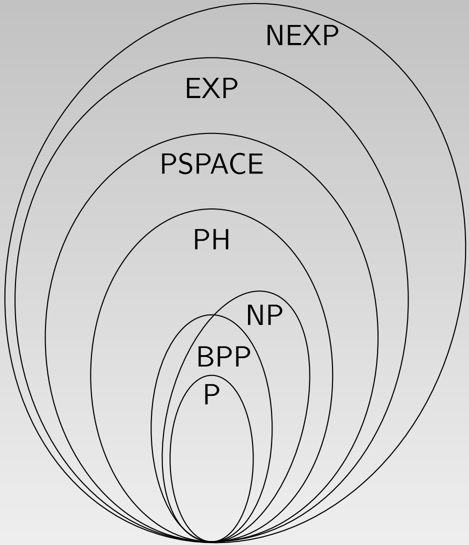
However…

Even if it is believed that

$$\mathsf{BPP} = \mathsf{P}$$

it is open whether

$$\mathsf{NEXP} \neq \mathsf{BPP} \quad (!)$$



Outline

1. Problem
2. Resource-bounded Kolmogorov complexity
3. Interactive protocols

Outline

1. Problem

2. Resource-bounded Kolmogorov complexity

3. Interactive protocols

Probabilistic algorithms

- Turing machine that can toss a coin at each step

Probabilistic algorithms

- Turing machine that can toss a coin at each step
- Another view: a deterministic TM M which takes as input x together with a string of random bits $r \rightarrow M(x, r)$

Probabilistic algorithms

- Turing machine that can toss a coin at each step
- Another view: a deterministic TM M which takes as input x together with a string of random bits $r \to M(x, r)$

BPP: class of languages A such that
there is a polynomial-time Turing machine M satisfying

- if $x \in A$ then $\Pr_{r \in \{0,1\}^{p(n)}}(M(x, r) = 1) \geq 2/3$
- if $x \notin A$ then $\Pr_{r \in \{0,1\}^{p(n)}}(M(x, r) = 0) \geq 2/3$

$\to M$ gives the correct answer with probability $\geq 2/3$.

Remarks on BPP

Remarks:

- polynomial time whatever the random bits

Remarks on BPP

Remarks:

- polynomial time whatever the random bits
- the probability of error $1/3$ can be reduced by repeting the algorithm

Remarks on BPP

Remarks:

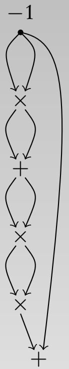
- polynomial time whatever the random bits
- the probability of error $1/3$ can be reduced
  by repeting the algorithm
- BPP has circuits of polynomial size (Adleman 1978)

Example

Integer testing:

Input: an arithmetic circuit C computing an integer N

Question: $N = 0$?



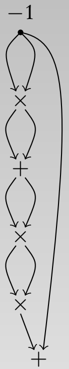
Example

Integer testing:

Input: an arithmetic circuit C computing an integer N

Question: $N = 0$?

Deterministic: no known polynomial algorithm…



Example

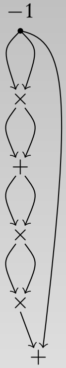
Integer testing:

Input: an arithmetic circuit C
computing an integer N

Question: $N = 0$?

Deterministic: no known
polynomial algorithm…

Probabilistic: C is evaluated modulo
random integers m_i

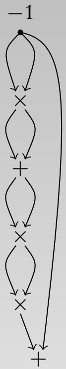


Example

Integer testing:

Input: an arithmetic circuit C computing an integer N

Question: $N = 0$?



Deterministic: no known polynomial algorithm…

Probabilistic: C is evaluated modulo random integers m_i

If we can compute a polynomial number of integers m_i st

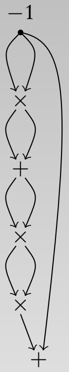
$$C() = 0 \iff \forall i, C() \equiv 0 \mod m_i, \qquad (|C| = n)$$

Example

Integer testing:

Input: an arithmetic circuit C computing an integer N

Question: $N = 0$?



Deterministic: no known polynomial algorithm...

Probabilistic: C is evaluated modulo random integers m_i

If we can compute a polynomial number of integers m_i st

$$C() = 0 \iff \forall i, C() \equiv 0 \mod m_i, \qquad (|C| = n)$$

then $\prod_i m_i$ doesn't have circuits of size n: lower bound!

Nondeterministic exponential time

NEXP:

- languages recognized by a nondeterministic TM working in exponential time

Nondeterministic exponential time

NEXP:

- languages recognized by a nondeterministic TM working in exponential time
- very large class
- complete problems: Succinct 3SAT, …

Nondeterministic exponential time

NEXP:

- languages recognized by a nondeterministic TM working in exponential time
- very large class
- complete problems: Succinct 3SAT, …

Best circuit lower bound known (Ryan Williams 2010):

$$\mathsf{NEXP} \not\subset \mathsf{ACC}^0$$

(polynomial size, constant depth circuits with modulo gates)

Nondeterministic exponential time

NEXP:

- languages recognized by a nondeterministic TM working in exponential time
- very large class
- complete problems: Succinct 3SAT, ...

Best circuit lower bound known (Ryan Williams 2010):

$$\mathsf{NEXP} \not\subset \mathsf{ACC}^0$$

(polynomial size, constant depth circuits with modulo gates)

Open:

- $\mathsf{NEXP} \not\subset \mathsf{P/poly}$ (polynomial size circuits)?
- $\boxed{\mathsf{NEXP} \neq \mathsf{BPP}?}$

Why doesn't simple diagonalization work?

In order to simulate deterministically
a probabilistic TM with $|r|$ random bits:

- run $M(x, r)$ for all r
- take the majority answer

$\rightarrow$ time $\geq 2^{|r|}$

Why doesn't simple diagonalization work?

In order to simulate deterministically
a probabilistic TM with $|r|$ random bits:

- run $M(x, r)$ for all r
- take the majority answer

$\rightarrow$ time $\geq 2^{|r|}$

One strategy for $\mathsf{NEXP} \neq \mathsf{BPP}$:
diagonalize over probabilistic machines working in time $n^{\log n}$
$\rightarrow$ time $2^{n^{\log n}}$, outside NEXP.

Outline

1. Problem
2. Resource-bounded Kolmogorov complexity
3. Interactive protocols

Definition

Kolmogorov complexity:

$C(x) =$ minimal size of a program printing x

Definition

Resource bounded Kolmogorov complexity:
$C^t(x)$ = minimal size of a program printing x in time t

Definition

Resource bounded Kolmogorov complexity:
$C^t(x)$ = minimal size of a program printing x in time t

- Other variants studied:
  Buhrman, Fortnow, Laplante, Lee, van Melkebeek, Romashchenko, …
- Results on the complexity of computing the resource bounded Kolmogorov complexity:
  Allender, Mayordomo, Ronneburger, …

Definition

Resource bounded Kolmogorov complexity:
$C^t(x)$ = minimal size of a program printing x in time t

- Other variants studied:
  Buhrman, Fortnow, Laplante, Lee, van Melkebeek, Romashchenko, ...
- Results on the complexity of computing the resource bounded Kolmogorov complexity:
  Allender, Mayordomo, Ronneburger, ...

In this talk: more basic stuffs!

Example of a link with our problem

Usually $C(x, y) \gtrsim C(x) + C(y|x)$ (symmetry of information)

Example of a link with our problem

Usually $C(x, y) \gtrsim C(x) + C(y|x)$ (symmetry of information)

The proof requires enumerating a set of exponential size
$\rightarrow$ open with polynomial time bounds.

Example of a link with our problem

Usually $C(x,y) \gtrsim C(x) + C(y|x)$ (symmetry of information)

The proof requires enumerating a set of exponential size
$\rightarrow$ open with polynomial time bounds.

Theorem (Lee and Romashchenko 2005, P. 2007)

If (SI) holds with polynomial-time bounds, then $\mathsf{EXP} \neq \mathsf{BPP}$ and even $\mathsf{EXP} \not\subset \mathsf{P/poly}$.

A classical observation

Let M be a BPP machine for a language A
working in time t with error $2^{-\epsilon t}$.

Suppose $x \in A$.

- Then there are $2^{(1-\epsilon)t}$ words r such that $M(x, r) = 0$.
- Hence if $M(x, r) = 0$ then $C^{t2^t}(r|x) \leq (1 - \epsilon)t$.

In other words,
if $C^{t2^t}(r|x) > (1 - \epsilon)t$ then $M(x, r)$ gives the correct result.

The converse?

[Cheating…]

If $M(x,r) = 0$ then $C^{t2^t}(r|x) \leq (1-\epsilon)t$.

Since there are:

- $2^{(1-\epsilon)t}$ words r such that $M(x,r) = 0$
- $\leq 2^{(1-\epsilon)t}$ words r such that $C^{t2^t}(r|x) \leq (1-\epsilon)t$

we have

$$C^{t2^t}(r|x) \leq (1-\epsilon)t \iff M(x,r) = 0 \quad (!)$$

Going outside BPP

Then for an enumeration (M_n) of probabilistic machines on input x, in NEXP:

- guess r of size $n^{\log n}$,
- check it has high complexity,
- simulate $M_n(x, r)$ for $n^{\log n}$ steps and take the opposite result.

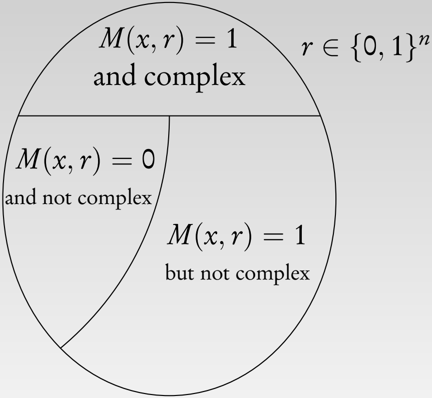
Hence $\mathsf{NEXP} \neq \mathsf{BPP}$!

The converse?

If $M(x,r) = 0$ then $C^{t2^t}(r|x) \leq (1-\epsilon)t + \alpha$.

Hence:

- if $C^{t2^t}(r|x) \geq (1-\epsilon)t + \alpha$ then $M(x,r) = 1$;
- for a fraction $2^{-\alpha}$ of the remaining words r, $M(x,r) = 0$.

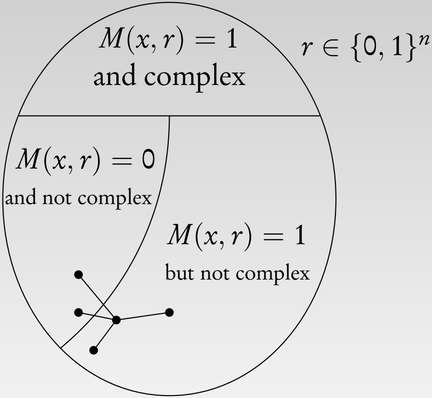


The converse?

If $M(x,r) = 0$ then $C^{t2^t}(r|x) \leq (1-\epsilon)t + \alpha$.

Hence:

- if $C^{t2^t}(r|x) \geq (1-\epsilon)t + \alpha$ then $M(x,r) = 1$;
- for a fraction $2^{-\alpha}$ of the remaining words r, $M(x,r) = 0$.



Outline

1. Problem

2. Resource-bounded Kolmogorov complexity

3. Interactive protocols

Quantifier alternation

THEOREM (Kannan 1982)

$\mathsf{NEXP}^{\mathsf{NP}}$ does not have circuits of polynomial size

Idea (in fact more subtle): in $\mathsf{NEXP}^{\mathsf{NP}}$ express

“there is a circuit A_0 of size $n^{2\log n}$ such that:
for all circuit A of size $n^{\log n}$, $A(x) \neq A_0(x)$ for some x”

Quantifier alternation

THEOREM (Kannan 1982)

$\mathsf{NEXP}^{\mathsf{NP}}$ does not have circuits of polynomial size

Idea (in fact more subtle): in $\mathsf{NEXP}^{\mathsf{NP}}$ express

"there is a circuit A_0 of size $n^{2\log n}$ such that:
for all circuit A of size $n^{\log n}$, $A(x) \neq A_0(x)$ for some x"

(even better: $\mathsf{MA}_{\mathsf{EXP}} \not\subset \mathsf{P}/\mathsf{poly}$
Buhrman, Fortnow, Thierauf 1998)

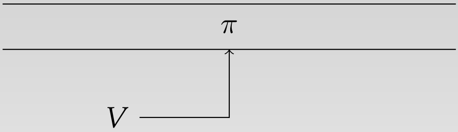
Interactive protocol for NEXP

$\mathsf{NEXP} = \mathsf{MIP}$ (Babai, Fortnow, Lund 1991) or
$\mathsf{NEXP} = \mathsf{PCP}(\mathsf{poly}, \mathsf{poly})$.

Interactive protocol for NEXP

NEXP = MIP (Babai, Fortnow, Lund 1991) or
NEXP = PCP(poly, poly).

Exponentially long proof π, BPP-style verifier V that can read a polynomial number of bits from the proof.



- if $x \in A$ then $\exists \pi$ st $V^{\pi}(x)$ accepts with proba 1;
- if $x \in A$ then $\forall \pi$, $V^{\pi}(x)$ rejects with proba $\geq 2/3$.

Protocols with powerful verifier

If $\mathsf{NEXP} = \mathsf{BPP}$ then the verifier can be $\mathsf{BPP}^{\mathsf{NEXP}}$!

Protocols with powerful verifier

If $\mathsf{NEXP} = \mathsf{BPP}$ then the verifier can be $\mathsf{BPP}^{\mathsf{NEXP}}$!

- With such a powerful verifier, we have "two quantifiers" $\rightarrow$ hope for going outside from BPP?
- Problem: only a polynomial number of bits of the proof can be read

Protocols with powerful verifier

If $\mathsf{NEXP} = \mathsf{BPP}$ then the verifier can be $\mathsf{BPP}^{\mathsf{NEXP}}$!

- With such a powerful verifier, we have "two quantifiers" $\rightarrow$ hope for going outside from BPP?
- Problem: only a polynomial number of bits of the proof can be read

Question: what can be done in $\mathsf{PCP}(\mathsf{poly}, \mathsf{poly})^{\mathsf{NEXP}}$?

Conclusion

- hard-to-believe but still open
- (resource-bounded) Kolmogorov complexity might help
- combined with interactive protocols?

Conclusion

- hard-to-believe but still open
- (resource-bounded) Kolmogorov complexity might help
- combined with interactive protocols?

Last question:

$$\text{if } M \text{ is a BPP-machine and } C^{2^n}(r) = |r|,$$
$$\text{does } M(x, r) \text{ give the correct answer?}$$

(time bound 2^n instead of $t2^t$)