Quelle est la puissance des algorithmes probabilistes polynomiaux ?

Sylvain Perifel (LIAFA)

Marne-la-Vallée, le 23 novembre 2010

Introduction

Entrée : un mot x contenant autant de a que de b

Question : trouver i tel que $x_i = a$.

Introduction

Entrée : un mot x contenant autant de a que de b

Question : trouver i tel que $x_i = a$.

Déterministe – pire cas $n/2$

Introduction

Entrée : un mot x contenant autant de a que de b

Question : trouver i tel que $x_i = a$.

Déterministe – pire cas $n/2$

Probabiliste – pire cas : espérance $\sum_{i \geq 1} i 2^{-i} = O(1)$.

Introduction

Entrée : un ensemble convexe $S \subset \mathbb{R}^n$ donné par boîte noire

Question : calculer le volume de S.

Introduction

Entrée : un ensemble convexe $S \subset \mathbb{R}^n$ donné par boîte noire

Question : calculer le volume de S.

Déterministe – en temps polynomial, pas mieux que $\frac{\sup}{\inf} \simeq 2^n$
(Bárány et Füredi, 1987).

Introduction

Entrée : un ensemble convexe $S \subset \mathbb{R}^n$ donné par boîte noire

Question : calculer le volume de S.

Déterministe – en temps polynomial, pas mieux que $\frac{\sup}{\inf} \simeq 2^n$
(Bárány et Füredi, 1987).

Probabiliste – précision polynomiale.

Introduction

Quicksort :

Déterministe – pire cas n^2

Introduction

Quicksort :

Déterministe – pire cas n^2

Probabiliste – pire cas : espérance $O(n \log n)$.

Introduction

Quicksort :

Déterministe – pire cas n^2

Probabiliste – pire cas : espérance $O(n \log n)$.

→ Peut-on accélérer les calculs
grâce à des algorithmes probabilistes ?

Plan

1. Algorithmes probabilistes

2. Puissance des algorithmes probabilistes

3. Quelques pistes

Plan

1. Algorithmes probabilistes

2. Puissance des algorithmes probabilistes

3. Quelques pistes

Algorithmes probabilistes

- Machine de Turing qui, à chaque étape,
  peut tirer un bit aléatoire

Algorithmes probabilistes

- Machine de Turing qui, à chaque étape,
  peut tirer un bit aléatoire

- Algorithme probabiliste polynomial :
  - temps polynomial quels que soient les tirages aléatoires
  - une certaine probabilité d'erreur (selon les tirages)

Algorithmes probabilistes

- Machine de Turing qui, à chaque étape,
  peut tirer un bit aléatoire
- Algorithme probabiliste polynomial :
  - temps polynomial quels que soient les tirages aléatoires
  - une certaine probabilité d'erreur (selon les tirages)
- BPP : problèmes possédant des algorithmes probabilistes polynomiaux avec erreur $\leq 1/3$.

Algorithmes probabilistes

- Machine de Turing qui, à chaque étape,
  peut tirer un bit aléatoire
- Algorithme probabiliste polynomial :
  - temps polynomial quels que soient les tirages aléatoires
  - une certaine probabilité d'erreur (selon les tirages)
- BPP : problèmes possédant des algorithmes probabilistes polynomiaux avec erreur $\leq 1/3$.
- Diminution de l'erreur : répéter k fois l'algorithme et prendre la réponse majoritaire $\rightarrow$ erreur $2^{-\Omega(k)}$.

Dérandomisation

Pour rendre déterministe un algorithme probabiliste :

- il suffit de simuler tous les tirages possibles
  et de prendre la réponse majoritaire

Dérandomisation

Pour rendre déterministe un algorithme probabiliste :

- il suffit de simuler tous les tirages possibles
  et de prendre la réponse majoritaire

  $\rightarrow$ temps exponentiel 2^{n^k} (si n^k bits aléatoires)

Dérandomisation

Pour rendre déterministe un algorithme probabiliste :

- il suffit de simuler tous les tirages possibles
  et de prendre la réponse majoritaire

  → temps exponentiel 2^{n^k} (si n^k bits aléatoires)

Peut-on le faire en temps polynomial ? = dérandomisation

« BPP = P ? »

Dérandomisation

Pour rendre déterministe un algorithme probabiliste :

- il suffit de simuler tous les tirages possibles
  et de prendre la réponse majoritaire

  $\rightarrow$ temps exponentiel 2^{n^k} (si n^k bits aléatoires)

Peut-on le faire en temps polynomial ? = dérandomisation

« BPP = P ? »

Exemples :
Primalité en temps polynomial :

- jusqu'en 2002, algorithmes probabilistes seulement
  (Miller-Rabin)
- en 2002 : Primes is in P (Agrawal, Kayal, Saxena)

Dérandomisation

Pour rendre déterministe un algorithme probabiliste :

- il suffit de simuler tous les tirages possibles
  et de prendre la réponse majoritaire

  → temps exponentiel 2^{n^k} (si n^k bits aléatoires)

Peut-on le faire en temps polynomial ? = dérandomisation

« BPP = P ? »

Exemples :

Méthodes génériques

(espérances conditionnelles, indépendance d à d, …)

Dérandomisation

Pour rendre déterministe un algorithme probabiliste :

- il suffit de simuler tous les tirages possibles
  et de prendre la réponse majoritaire

  → temps exponentiel 2^{n^k} (si n^k bits aléatoires)

Peut-on le faire en temps polynomial ? = dérandomisation

« BPP = P ? »

Tous les algorithmes pour des problèmes « naturels »
ont été dérandomisés…

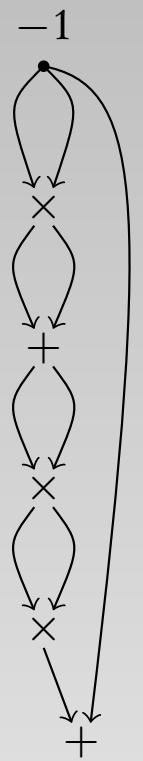
Tous ? Non ! Car un problème résiste encore et toujours…

Circuix le Gaulois

Entier Nul :

Entrée : un circuit arithmétique C calculant un entier N

Question : $N = 0$?



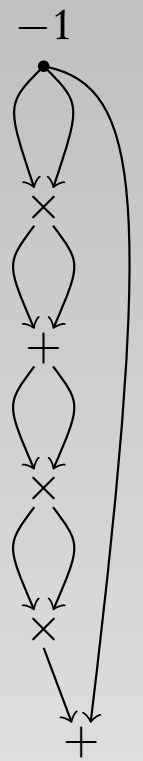
Circuix le Gaulois

Entier Nul :

Entrée : un circuit arithmétique C calculant un entier N

Question : $N = 0$?

Déterministe : pas d'algorithme polynomial connu...



Circuix le Gaulois

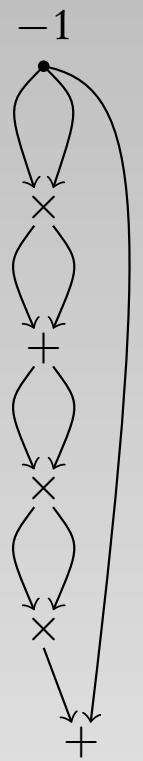
Entier Nul :

Entrée : un circuit arithmétique C calculant un entier N

Question : $N = 0$?

Déterministe : pas d'algorithme polynomial connu...

Probabiliste : on évalue modulo des entiers aléatoires m_i



Circuix le Gaulois

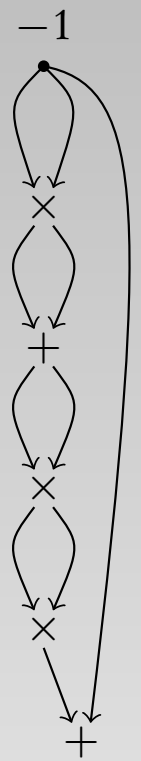
Entier Nul :

Entrée : un circuit arithmétique C calculant un entier N

Question : $N = 0$?

Déterministe : pas d'algorithme polynomial connu...

Probabiliste : on évalue modulo des entiers aléatoires m_i



Si l'on sait calculer un nombre polynomial d'entiers m_i tq

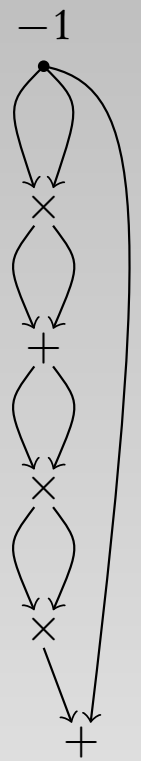
$$C() = 0 \iff \forall i, C() \equiv 0 \mod m_i,$$

Circuix le Gaulois

Entier Nul :

Entrée : un circuit arithmétique C calculant un entier N

Question : $N = 0$?



Déterministe : pas d'algorithme polynomial connu...

Probabiliste : on évalue modulo des entiers aléatoires m_i

Si l'on sait calculer un nombre polynomial d'entiers m_i tq

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

alors $\prod_i m_i$ n'a pas de circuit de taille n : borne inférieure !

Une approche pour dérandomiser

Pour dérandomiser un algorithme :

- on peut tenter de construire une suite de tirages qui convient pour toute entrée.

- La vaste majorité fonctionne ! ($p > 1 - 2^{-n}$)

Une approche pour dérandomiser

Pour dérandomiser un algorithme :

- on peut tenter de construire une suite de tirages qui convient pour toute entrée.
- La vaste majorité fonctionne ! ($p > 1 - 2^{-n}$)
- « How difficult could it be to find hay in a haystack ? »
  (Howard Karloff)

Plan

1. Algorithmes probabilistes

2. Puissance des algorithmes probabilistes

3. Quelques pistes

Dérandomisation sous hypothèse

Classe EXP : problèmes résolubles
en temps (déterministe) exponentiel $2^{\mathrm{poly}(n)}$.

Conjecture : EXP n'a pas de circuits de taille polynomiale.

Dérandomisation sous hypothèse

Classe EXP : problèmes résolubles
en temps (déterministe) exponentiel $2^{\mathrm{poly}(n)}$.

Conjecture : EXP n'a pas de circuits de taille polynomiale.

THÉORÈME (NW1994, IW2001, ...)

Si EXP n'a pas de circuits de taille polynomiale, alors tout algorithme probabiliste polynomial peut être simulé en temps déterministe 2^{n^ϵ}.

Dérandomisation sous hypothèse

Classe EXP : problèmes résolubles
en temps (déterministe) exponentiel $2^{\mathrm{poly}(n)}$.

Conjecture : EXP n'a pas de circuits de taille polynomiale.

THÉORÈME (NW1994, IW2001, ...)

$$\mathsf{EXP} \not\subset \mathsf{P/poly} \Longrightarrow \mathsf{BPP} \subseteq \bigcap_{\epsilon>0} \mathsf{DTIME}(2^{n^\epsilon})$$

Dérandomisation sous hypothèse

Classe EXP : problèmes résolubles
en temps (déterministe) exponentiel $2^{\mathrm{poly}(n)}$.

Conjecture : EXP n'a pas de circuits de taille $2^{\epsilon n}$.

THÉORÈME (NW1994, IW2001, ...)

Si EXP n'a pas de circuits de taille $2^{\epsilon n}$, alors tout algorithme probabiliste polynomial peut être simulé en temps déterministe polynomial.

Dérandomisation sous hypothèse

Classe EXP : problèmes résolubles
en temps (déterministe) exponentiel $2^{\mathrm{poly}(n)}$.

Conjecture : EXP n'a pas de circuits de taille $2^{\epsilon n}$.

THÉORÈME (NW1994, IW2001, ...)

$$[\exists \epsilon > 0 \text{ tq } \mathsf{EXP} \not\subset \mathsf{SIZE}(2^{\epsilon n})] \Longrightarrow \mathsf{BPP} = \mathsf{P}$$

Dérandomisation sous hypothèse

Classe EXP : problèmes résolubles
en temps (déterministe) exponentiel $2^{\mathrm{poly}(n)}$.

Conjecture : EXP n'a pas de circuits de taille $2^{\epsilon n}$.

Théorème (NW1994, IW2001, …)

$$[\exists \epsilon > 0 \text{ tq } \mathsf{EXP} \not\subset \mathsf{SIZE}(2^{\epsilon n})] \Longrightarrow \mathsf{BPP} = \mathsf{P}$$

Démonstration (idée)

- Du point de vue d'un algorithme polynomial,
  une fonction difficile est comme « aléatoire ».
- L'hypothèse fournit une fonction de EXP
  qui n'a pas de petits circuits.

→ Générateur pseudo-aléatoire : on remplace les bits aléatoires par les résultats de cette fonction.

Une question plus modeste (?)

- Question : $\mathsf{BPP} = \mathsf{EXP}$?

Une question plus modeste (?)

- Question : BPP = EXP ?
- Réponse : « clairement » non ! (?)

Une question plus modeste (?)

- Question : $\mathsf{BPP} = \mathsf{EXP}$?
- Réponse : « clairement » non ! (?)
- Problème : cette question ne relativise pas
  - il existe un oracle A tel que $\mathsf{EXP}^A = \mathsf{BPP}^A$
  - il existe un oracle B tel que $\mathsf{EXP}^B \neq \mathsf{BPP}^B$

  $\rightarrow$ la diagonalisation échoue

Diagonalisation

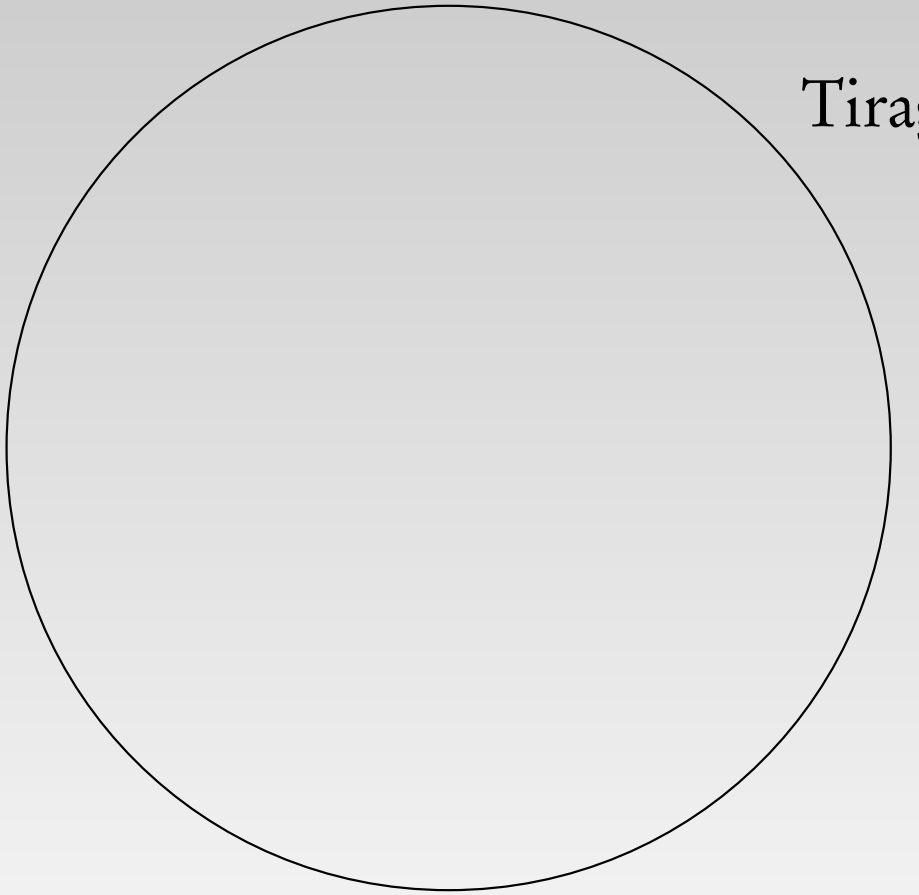
Proposition

$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.

Diagonalisation

Proposition

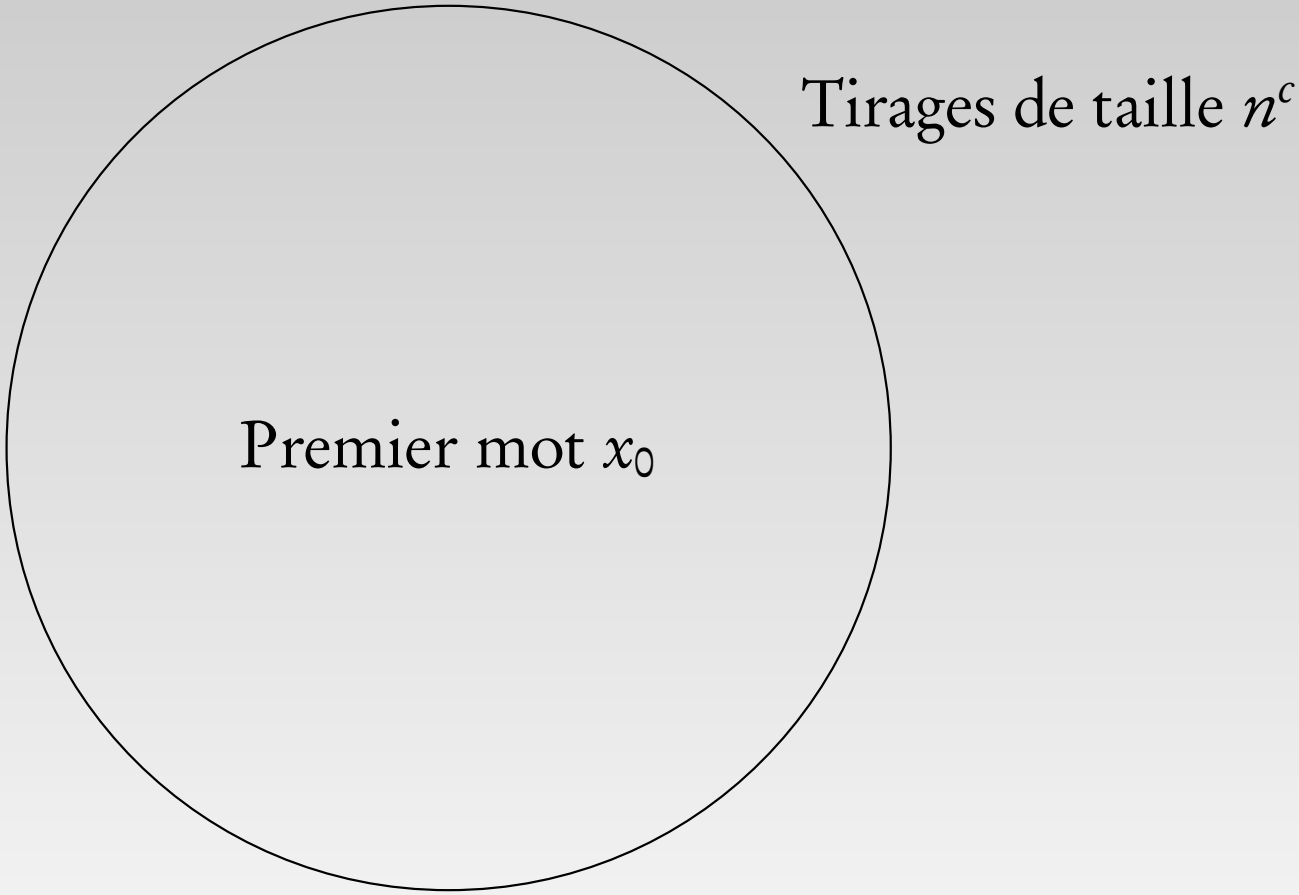
$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Diagonalisation

Proposition

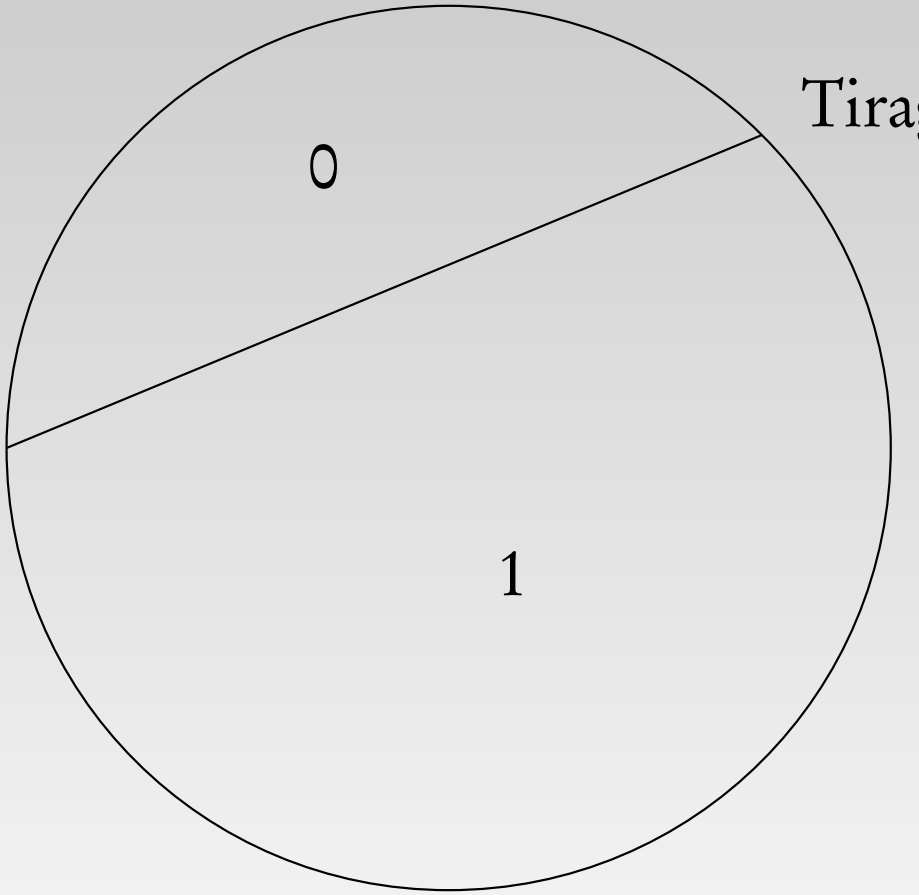
$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Diagonalisation

Proposition

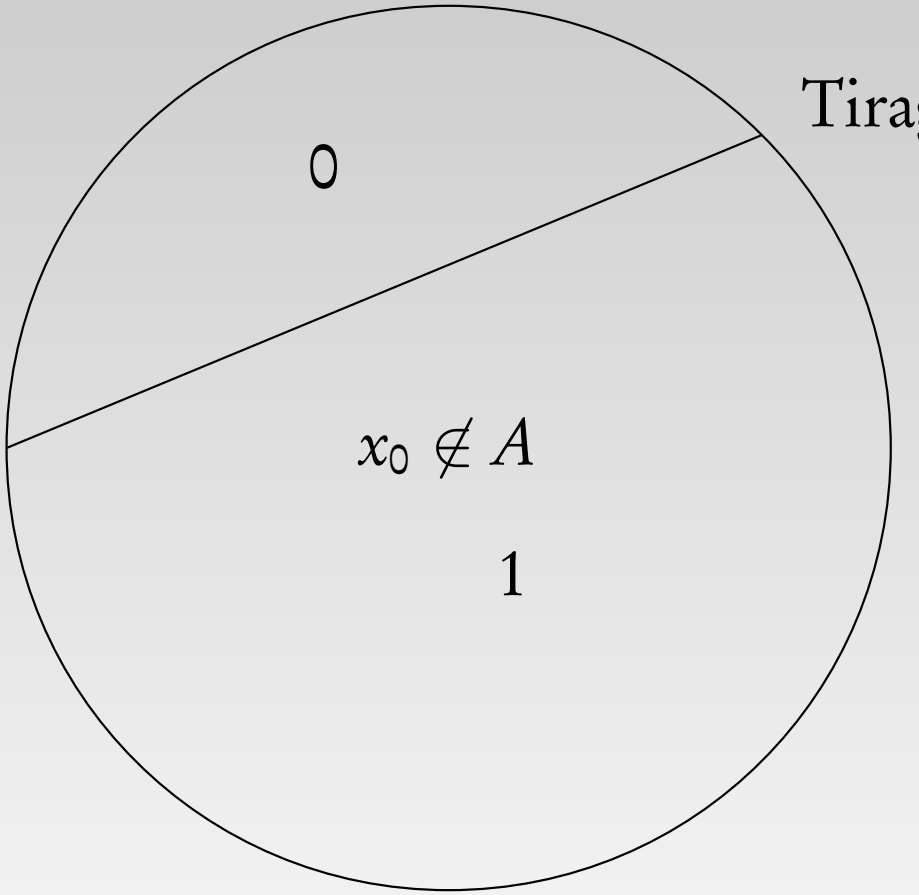
$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Diagonalisation

Proposition

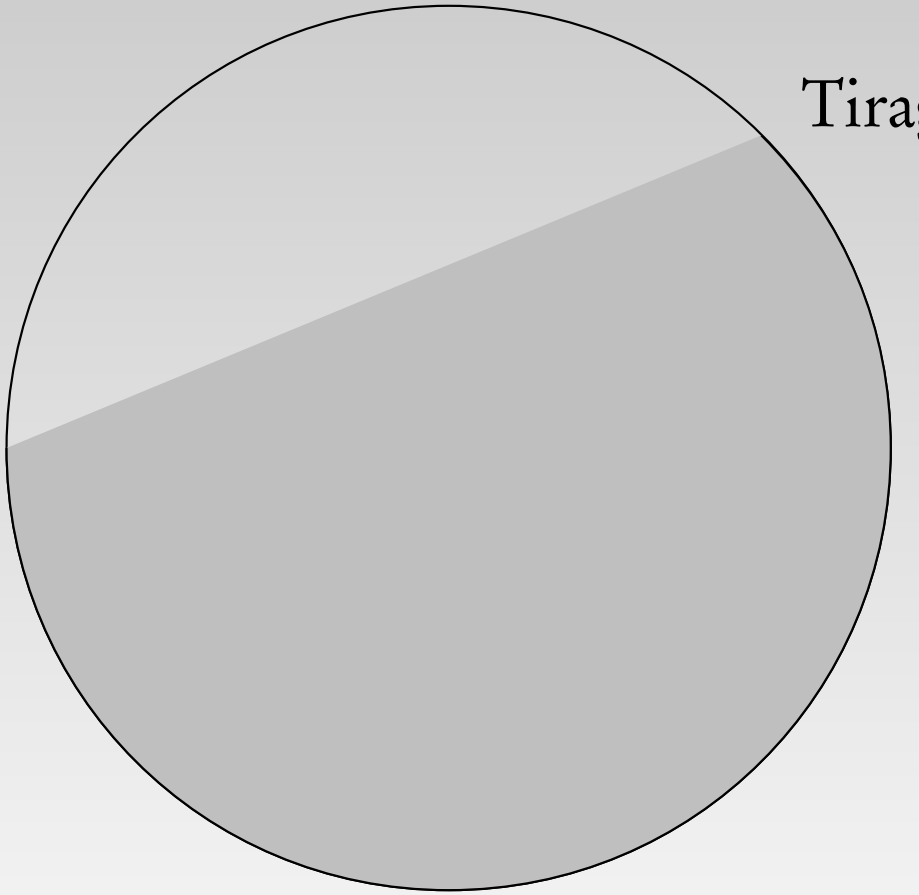
$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Diagonalisation

Proposition

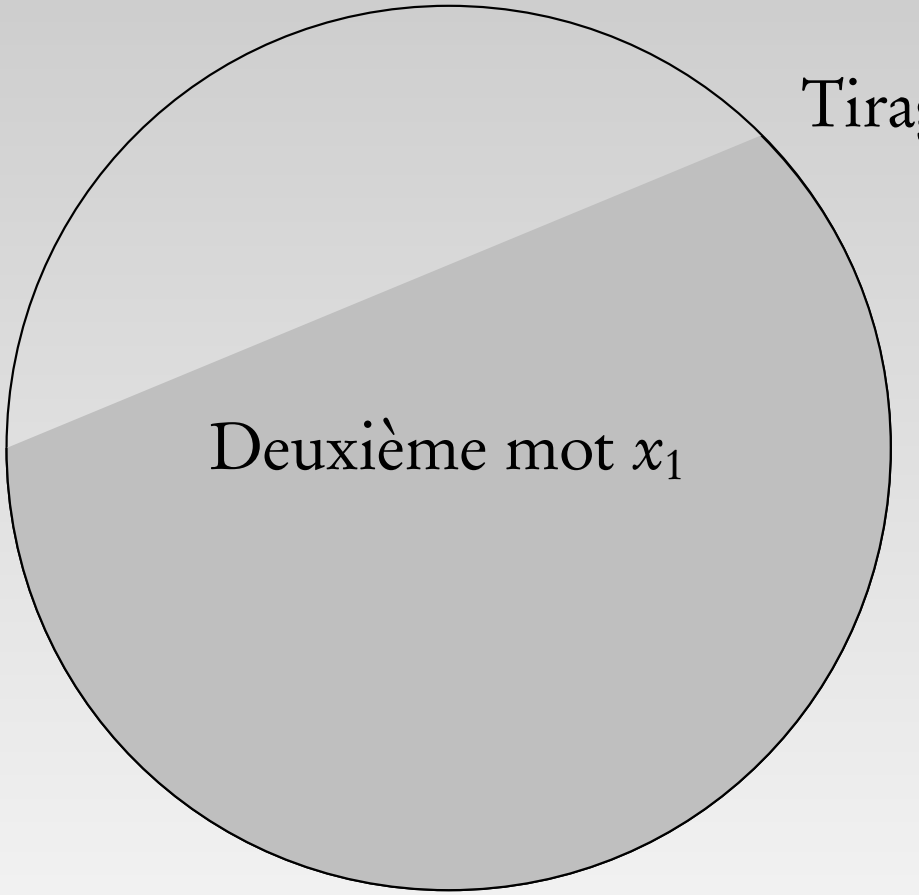
$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Diagonalisation

Proposition

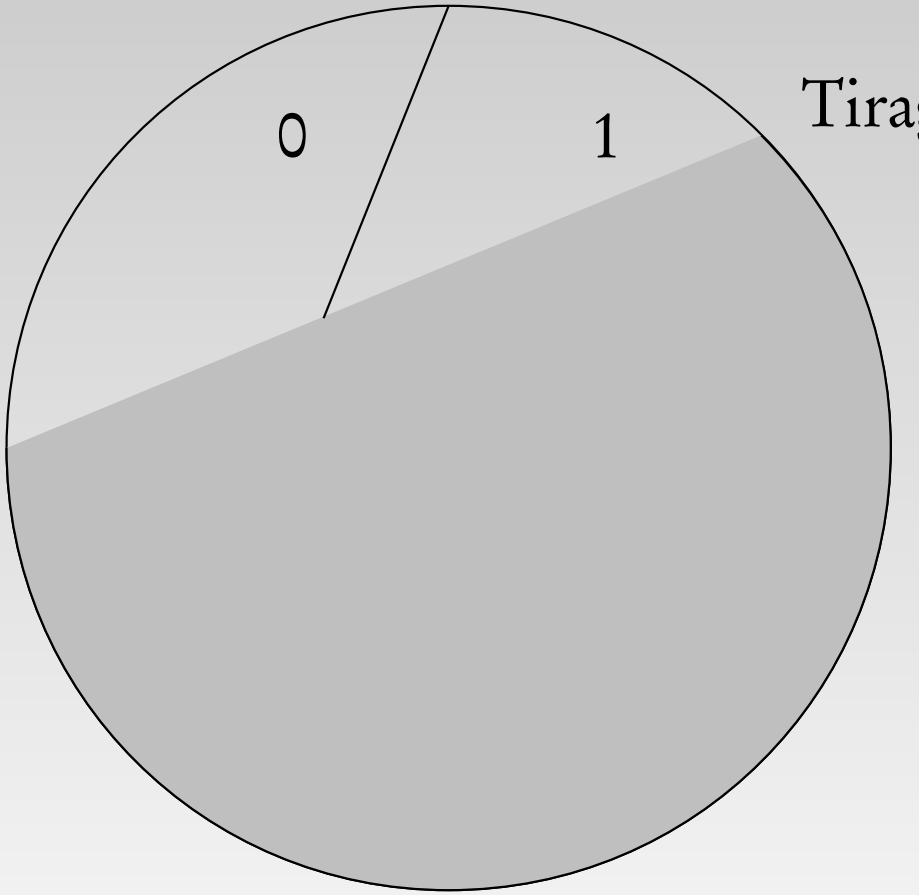
$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Diagonalisation

Proposition

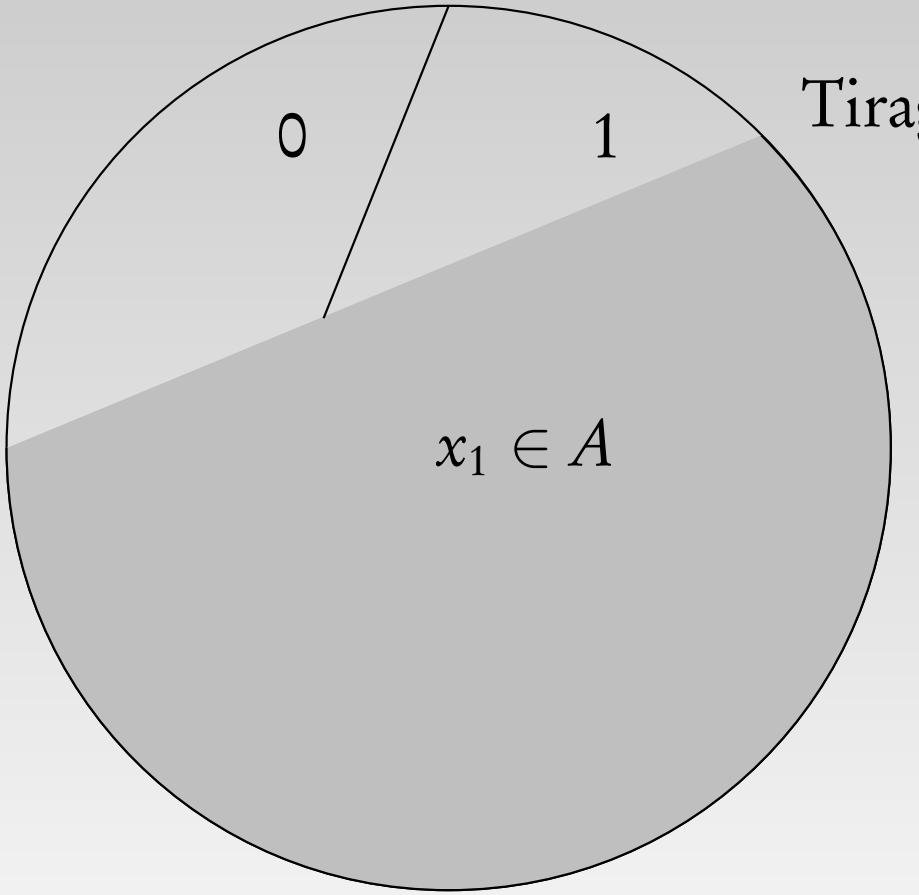
$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Diagonalisation

Proposition

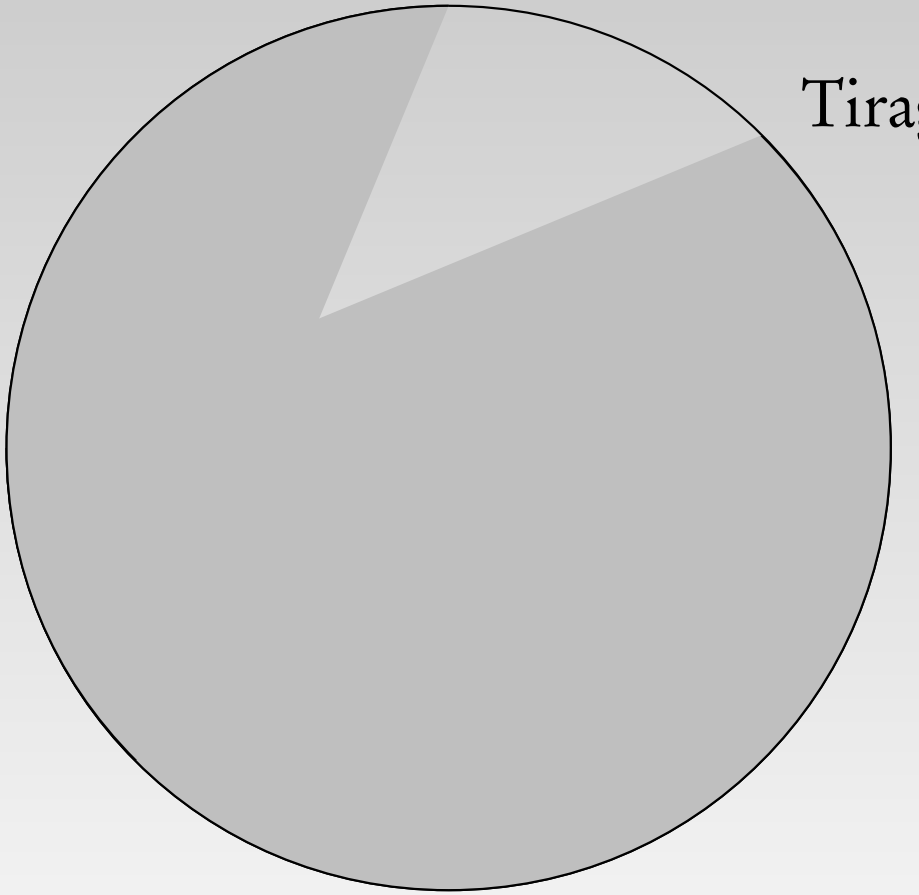
$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Diagonalisation

Proposition

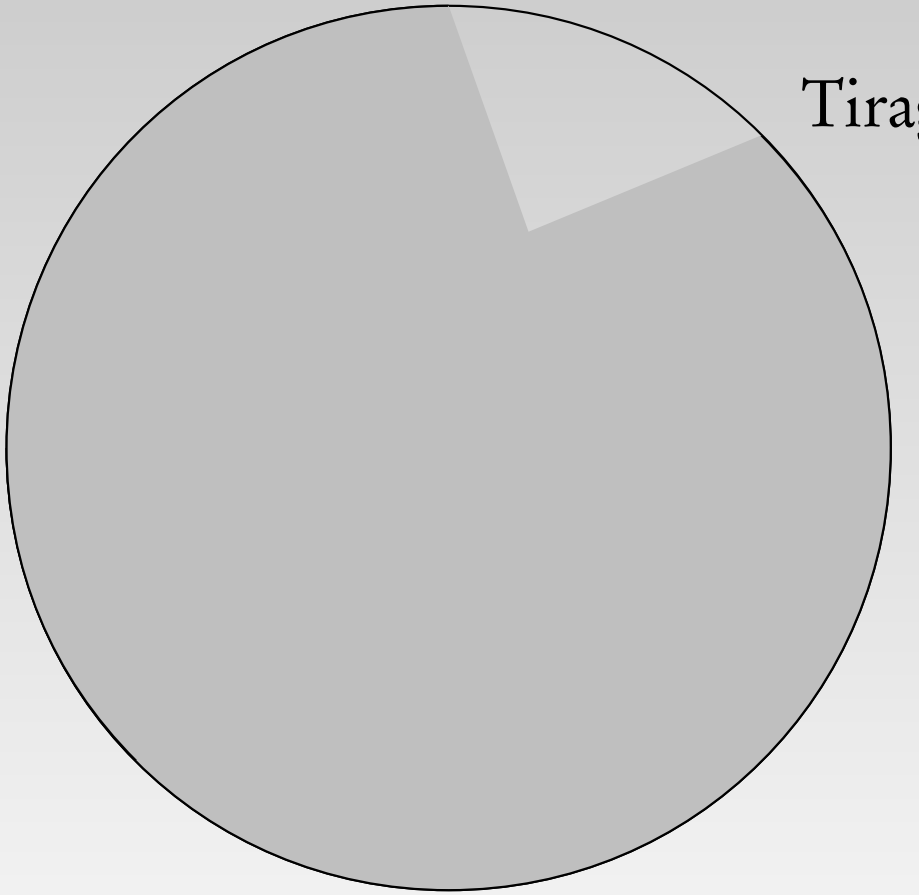
$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Diagonalisation

Proposition

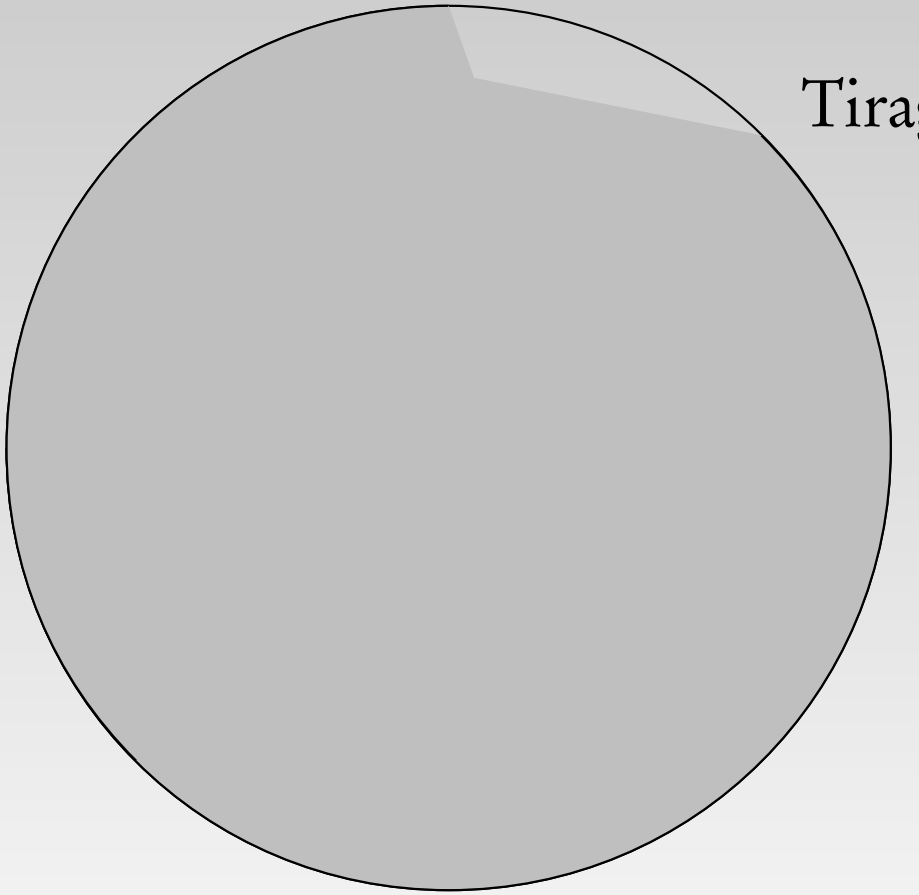
$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Diagonalisation

Proposition

$\forall c > 0$, il existe un problème A résoluble en temps déterministe 2^{n^c} mais pas en temps probabiliste n^c.



Plan

1. Algorithmes probabilistes

2. Puissance des algorithmes probabilistes

3. Quelques pistes

Symétrie de l'information

- Pour $x \in \{0,1\}^\star$, $K(x)$ est la « quantité d'information » contenue dans x
- $K(x) =$ taille du plus petit programme qui génère x

Symétrie de l'information

- Pour $x \in \{0,1\}^\star$, $K(x)$ est la « quantité d'information » contenue dans x
- $K(x)$ = taille du plus petit programme qui génère x

THÉORÈME (Levin, Kolmogorov)

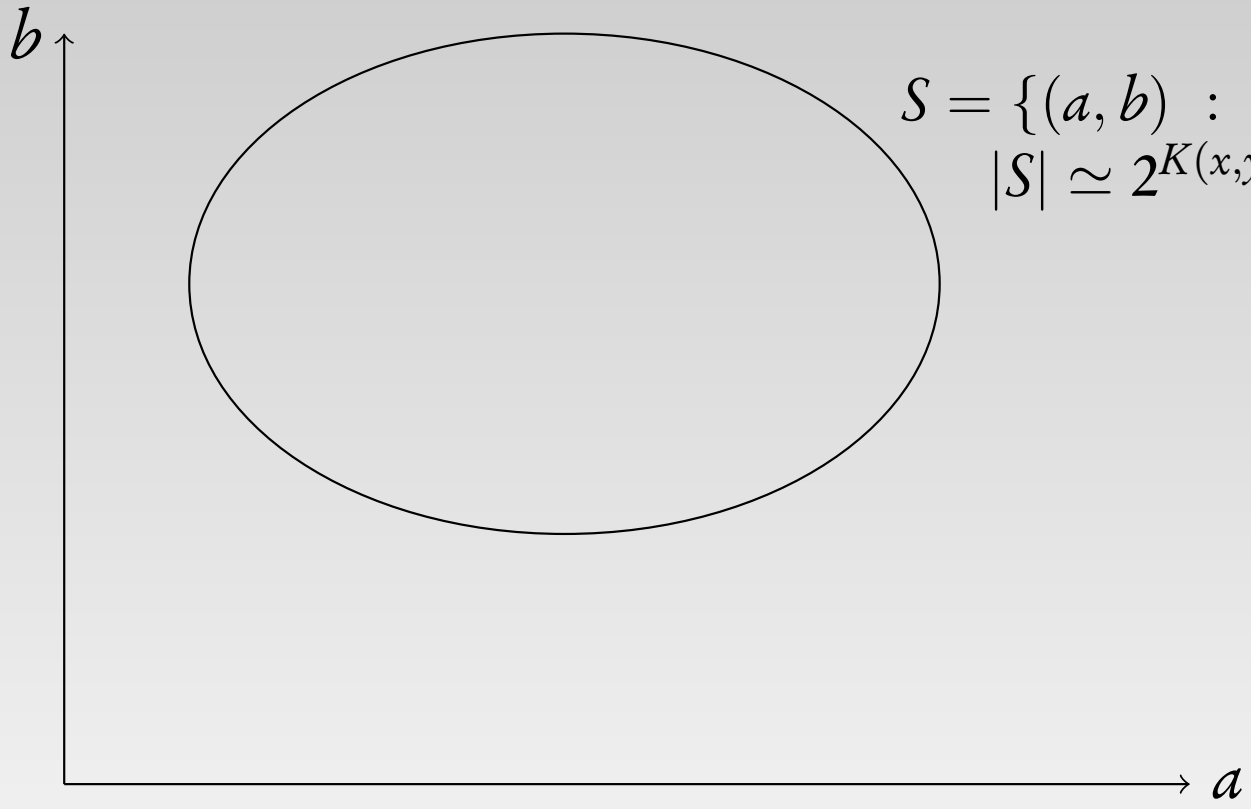
$$K(x,y) = K(x) + K(y|x).$$

Symétrie de l'information

THÉORÈME (Levin, Kolmogorov)

$$K(x,y) \geq K(x) + K(y|x).$$

Démonstration



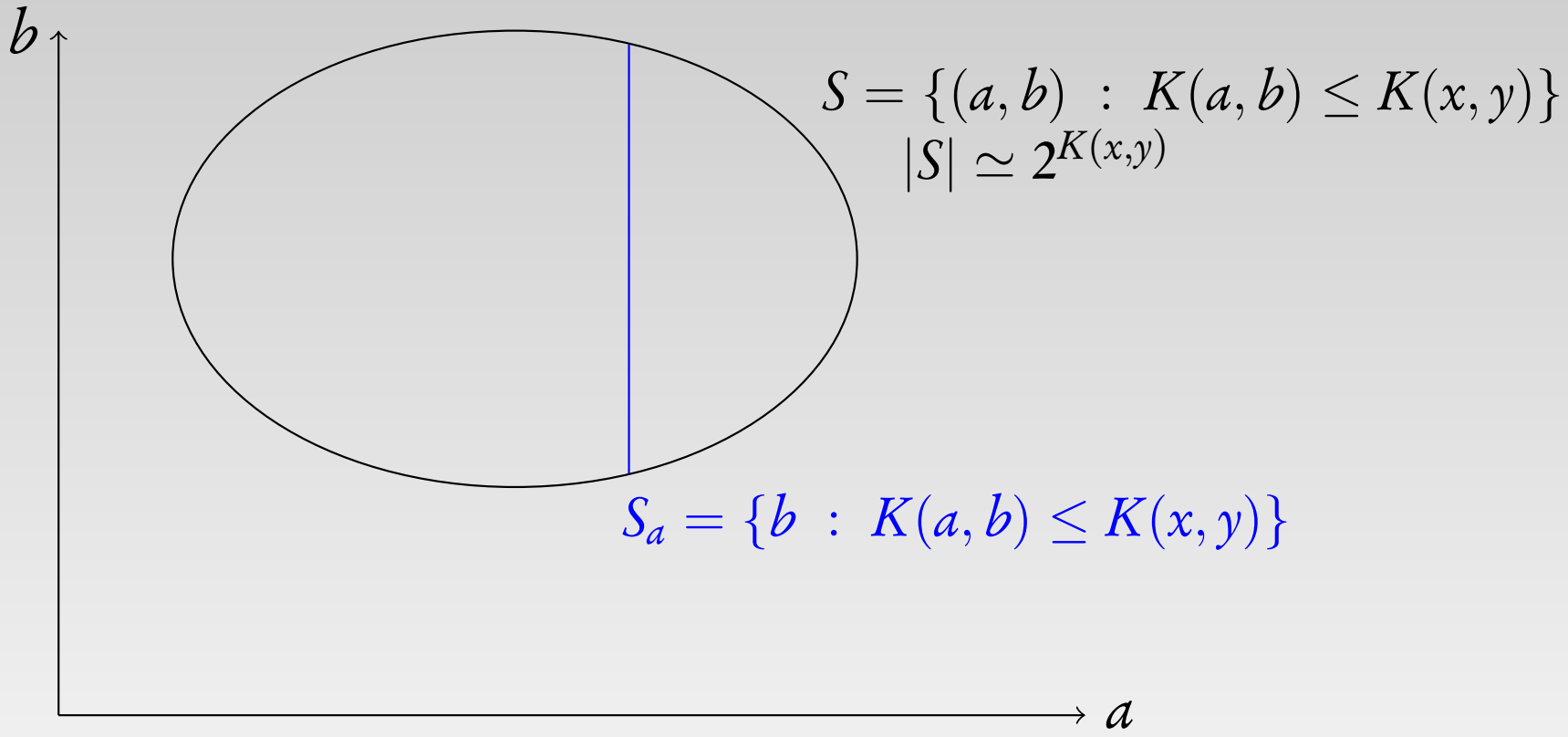
$$S = \{(a,b) \ : \ K(a,b) \leq K(x,y)\}$$
$$|S| \simeq 2^{K(x,y)}$$

Symétrie de l'information

THÉORÈME (Levin, Kolmogorov)

$$K(x,y) \geq K(x) + K(y|x).$$

Démonstration

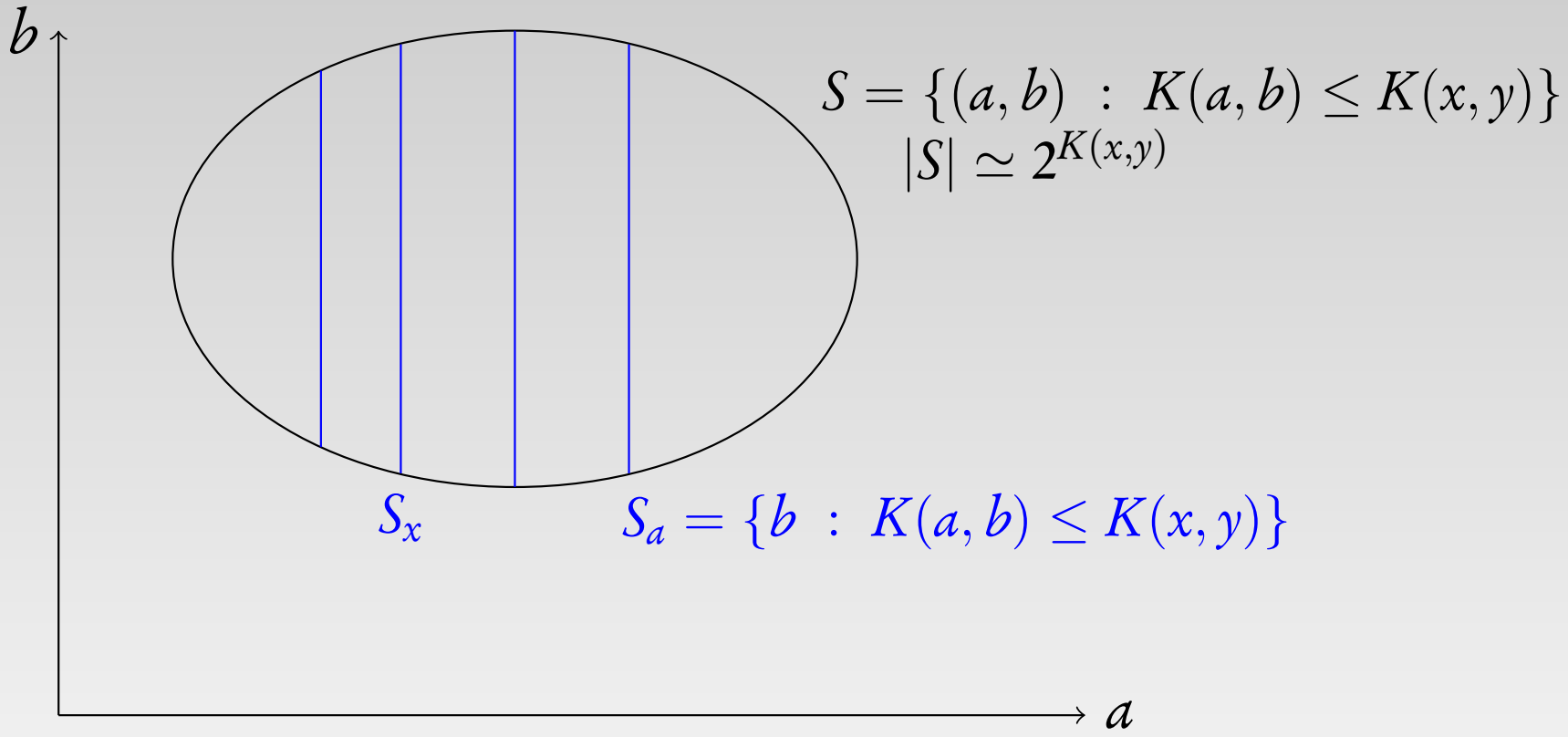


Symétrie de l'information

THÉORÈME (Levin, Kolmogorov)

$$K(x,y) \geq K(x) + K(y|x).$$

Démonstration

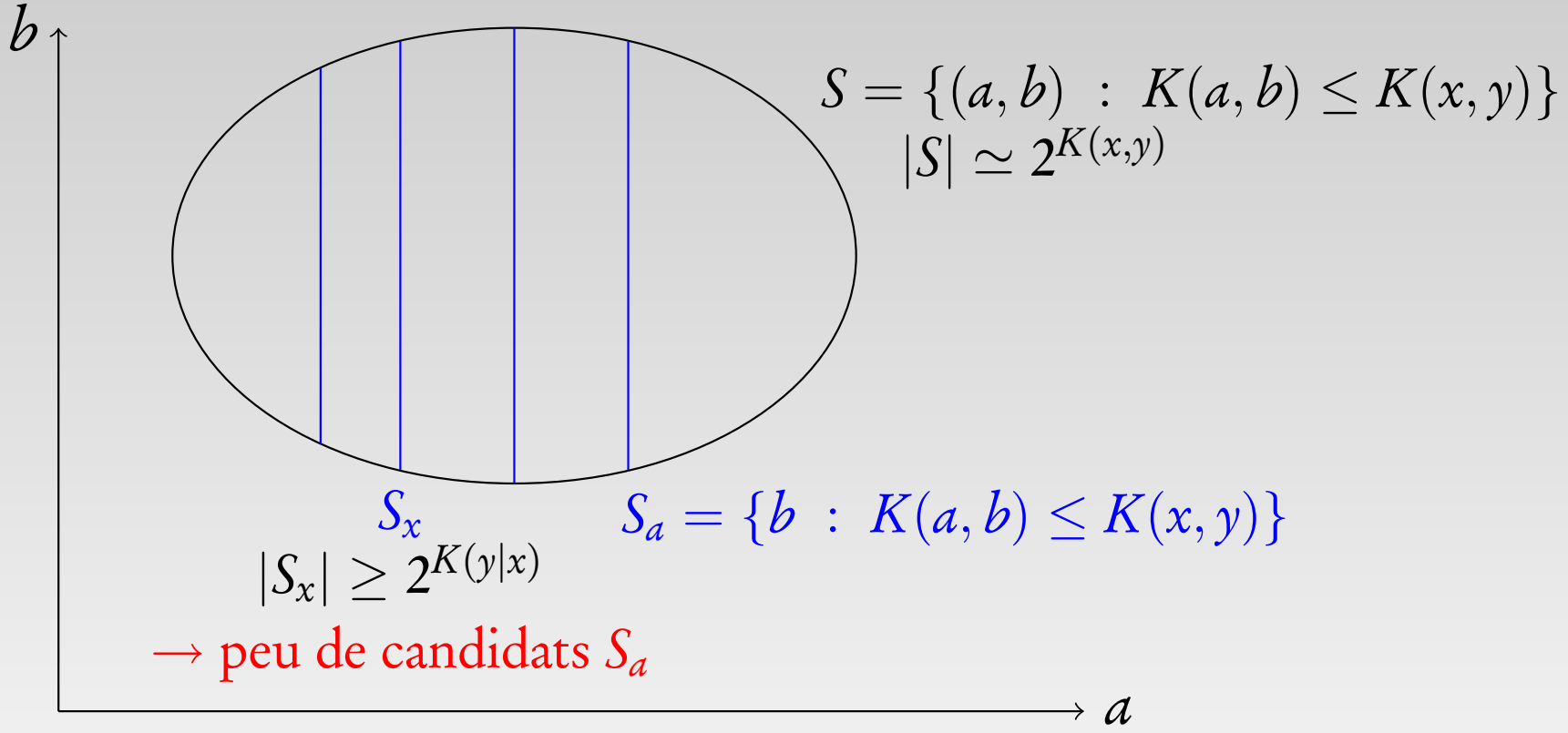


Symétrie de l'information

THÉORÈME (Levin, Kolmogorov)

$$K(x,y) \geq K(x) + K(y|x).$$

Démonstration



Lien avec BPP

- $K^p(x) =$ taille du plus petit programme qui génère x
  en temps polynomial
- Question : $K^p(x, y) \geq K^p(x) + K^p(y|x)$? (SI_p)

Lien avec BPP

- $K^p(x) =$ taille du plus petit programme qui génère x
  en temps polynomial

- Question : $K^p(x,y) \geq K^p(x) + K^p(y|x)$? $(\mathsf{SI_p})$

Théorème (Lee, Romashchenko 2004)

$(\mathsf{SI_p}) \Rightarrow \mathsf{EXP} \neq \mathsf{BPP}$.

Lien avec BPP

- $K^p(x) =$ taille du plus petit programme qui génère x en temps polynomial
- Question : $K^p(x,y) \geq K^p(x) + K^p(y|x)$? $(\mathsf{SI_p})$

Théorème (Lee, Romashchenko 2004)

$(\mathsf{SI_p}) \Rightarrow \mathsf{EXP} \neq \mathsf{BPP}$.

- Espoir : montrer que $\mathsf{EXP} = \mathsf{BPP} \Rightarrow (\mathsf{SI_p})$
- Il faudrait pouvoir décider dans BPP avec tirage a le langage $\{(x,\alpha) \ : \ K(x|a) \leq \alpha\}$.

Générateur pseudo-aléatoire

- Même approche que NW94 et IW01 :
  utiliser une fonction difficile comme
  générateur pseudo-aléatoire.

Générateur pseudo-aléatoire

- Même approche que NW94 et IW01 :
  utiliser une fonction difficile comme
  générateur pseudo-aléatoire.

- On ne veut plus $\mathsf{BPP} = \mathsf{P}$
  mais « seulement » $\mathsf{EXP} \neq \mathsf{BPP}$.

Générateur pseudo-aléatoire

- Même approche que NW94 et IW01 :
  utiliser une fonction difficile comme
  générateur pseudo-aléatoire.

- On ne veut plus $\mathsf{BPP} = \mathsf{P}$
  mais « seulement » $\mathsf{EXP} \neq \mathsf{BPP}$.

- On cherche un GPA qui fonctionne
  - sur au moins une entrée x ;
  - en temps exponentiel ;
  - à partir d'une borne inférieure existante
    (p.ex. $\mathsf{EXP} \neq \mathsf{P}$ ou $\mathsf{NEXP} \neq \mathsf{ACC}$ (Ryan 2010)).

Conclusion

- Apparemment, « it's hard to find hay in a haystack »…

Conclusion

- Apparemment, « it's hard to find hay in a haystack »…
- Liens dérandomisation et
  bornes inférieures sur les circuits

Conclusion

- Apparemment, « it's hard to find hay in a haystack »...
- Liens dérandomisation et
  bornes inférieures sur les circuits
- Même la question « NEXP $\neq$ BPP? » est difficile (!)

Conclusion

- Apparemment, « it's hard to find hay in a haystack »...
- Liens dérandomisation et
  bornes inférieures sur les circuits
- Même la question « NEXP $\neq$ BPP? » est difficile (!)
- Nouvelles idées ?
  Nouvelles bornes inférieures (Ryan 2010) ?