Karlsruhe Institute of Technology

Probability and Computing – Randomised Complexity Classes

Stefan Walzer | WS 2024/2025

Lecture Notes

Lecture notes by Thomas Worsch available:

Today: Decision Problems Only

Karlsruhe Institute of Technology

- ~~approximation algorithms~~
- ~~average case analysis~~
- ~~data structures~~
- ~~optimisation problems~~
- **decision problems**
  - for some language L such as $L = \text{PRIMES}$
  - decide for input x the question “is $x \in L$?”
  - can you do it in polynomial time?
  - does randomisation help?

Turing machines

KIT Karlsruhe Institute of Technology

(Non-) deterministic Turing machine

- S: finite state set
- B: finite tape alphabet including blank symbol $\square$
- $A \subseteq B - \{\square\}$: input alphabet
- one tape, one head
- transition functions
  - *deterministic:* one
    $\delta : S \times B \to (S \cup \{\text{YES}, \text{NO}\}) \times B \times \{-1, 0, 1\}$
  - *non-deterministic* two (or more)
    $\delta_0, \delta_1 : S \times B \to (S \cup \{\text{YES}, \text{NO}\}) \times B \times \{-1, 0, 1\}$
    (alternatively: general transition *relation*)
  - in states YES and NO: “T halts”
- accepted language
  $L(T) = \{w \in A^+ \mid \exists\text{YES-computation for } w\}$

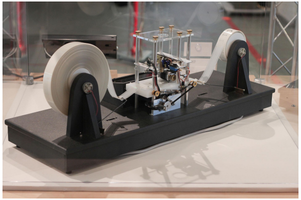


Photo: Rocky Acosta

Probabilistic Turing machine

- definition like non-deterministic TM
- uses δ_0 or δ_1 with probability $1/2$ in each step
- output $T(w)$ is random variable
- difference to NTM:
  - *quantified* non-determinism
  - can study e.g. *probability* of acceptance

When is a PTM polynomial time?

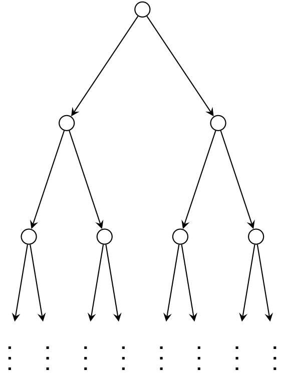
Karlsruhe Institute of Technology

Annoying

Running time for input x is random variable $T(x) \in \mathbb{N} \cup \{\infty\}$.

Simplification for Today: PTM in normal form

- For all inputs of length n, the PTM *halts* and does so after the *same number of steps* $t(n)$.
  $\hookrightarrow$ this is without loss of generality under weak conditions
- computation tree of PTM in normal form is complete binary tree of depth $t(n)$.
- call $t(n)$ the *running time*
- PTM runs in *polynomial time*, if $t(n) \leq p(n)$ for a polynomial $p(n)$.
- acceptance probability is the $\frac{\text{number of accepting computations}}{2^{t(n)}}$.



“Classic” Complexity Classes

class $\mathcal{C}$	requirement for $L \in \mathcal{C}$
$\mathbf{P}$	polynomial time DTM can decide L
$\mathbf{NP}$	polynomial time NTM can decide L
$\mathbf{PSPACE}$	polynomial space TM can decide L

Complement Classes

For class $\mathcal{C}$ let $\mathrm{co}-\mathcal{C} = \{L \mid \bar{L} \in \mathcal{C}\} = \{\bar{L} \mid L \in \mathcal{C}\}$, e.g.

- $\mathbf{P} = \mathrm{co}-\mathbf{P}$
- $\mathbf{P} \subseteq \mathbf{NP} \cap \mathrm{co}-\mathbf{NP}$
- relationship between $\mathbf{NP}$ and $\mathrm{co}-\mathbf{NP}$ unknown
- $\mathbf{NP} \cup \mathrm{co}-\mathbf{NP} \subseteq \mathbf{PSPACE}$

Polynomial time reduction from L_1 to L_2

- in polynomial time computable function $f : A^+ \to A^+$, such that
- $\forall w \in A^+ \colon w \in L_1 \iff f(w) \in L_2$.

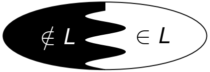
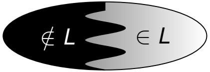
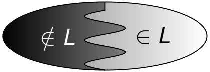
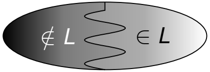
$\hookrightarrow$ then e.g. $L_2 \in \mathbf{NP}$ implies $L_1 \in \mathbf{NP}$.

Hardness

- A language H is $\mathcal{C}$*-hard*, if every language $L \in \mathcal{C}$ can be reduced to H in polynomial time.
- A language is $\mathcal{C}$*-complete*, if it is $\mathcal{C}$-hard and in $\mathcal{C}$.

Probabilistic Complexity Classes

A language L is in class $\mathbf{P}/\mathbf{RP}/\mathbf{BPP}/\mathbf{PP}$, if there exists a probabilistic polynomial time turing machine T such that...

class	name	requirement	visualisation	
$\mathbf{P}$	polynomial time	$\forall w \notin L : \Pr[T(w) = \text{YES}] = 0$ $\forall w \in L : \Pr[T(w) = \text{YES}] = 1$		no error
$\mathbf{RP}$	randomised polynomial time	$\forall w \notin L : \Pr[T(w) = \text{YES}] = 0$ $\forall w \in L : \Pr[T(w) = \text{YES}] \geq 1/2$		one-sided error
$\mathbf{BPP}$	bounded-error probabilistic polynomial time	$\forall w \notin L : \Pr[T(w) = \text{YES}] < 1/4$ $\forall w \in L : \Pr[T(w) = \text{YES}] > 3/4$		two-sided error
$\mathbf{PP}$	probabilistic polynomial time	$\forall w \notin L : \Pr[T(w) = \text{YES}] \leq 1/2$ $\forall w \in L : \Pr[T(w) = \text{YES}] > 1/2$		two-sided error

$\mathbf{ZPP} := \mathbf{RP} \cap \text{co}-\mathbf{RP}$. zero error probabilistic polynomial time

$\hookrightarrow$ requires *two* Turing machines, one for $\mathbf{RP}$, one for $\text{co}-\mathbf{RP}$.

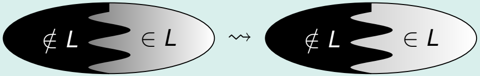
We say a polynomial time PTM is an $\mathbf{RP}$-PTM, $\mathbf{BPP}$-PTM or $\mathbf{PP}$-PTM if it is of the corresponding form.

Probability Amplification

Karlsruhe Institute of Technology

Theorem

Instead of "$1/2$" we can use "$1 - 2^{-q(n)}$" in the definition of $\mathbf{RP}$ without affecting the class.



Proof.

Let T be the Turing machine witnessing $L \in \mathbf{RP}$.
By running T independently $q(n)$ times the error probability is $2^{-q(n)}$.
Running time increases by polynomial factor $q(n)$.

for $i = 1$ *to* $q(n)$ **do**
  if $T(w) =$ YES **then**
    return YES
return NO

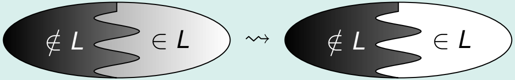
□

Probability Amplification (2)

Karlsruhe Institute of Technology

Theorem

Instead of "$1/4$" and "$3/4$" we can use "$2^{-q(n)}$" and "$1 - 2^{-q(n)}$" in the definition of $\mathbf{BPP}$ without affecting the class.



Proof.

Repeat $\mathcal{O}(q(n))$ times and take the majority answer.
See exercise sheet on probability amplification. $\square$

ZPP: Zero-Error-Probabilistic Polynomial Time

Karlsruhe Institute of Technology

Theorem: $L \in \mathbf{ZPP} \Rightarrow$ Las-Vegas Algorithm for L

If $L \in \mathbf{ZPP} := \mathbf{RP} \cap \text{co}-\mathbf{RP}$ then there exists a PTM that

- decides L with no error
- has *expected* polynomial running time

  $\hookrightarrow$ this PTM is not in normal form

Las Vegas Algorithm

Randomised Algorithm that never outputs an incorrect result.

Some definitions allow the algorithm to “give-up”, reporting failure.

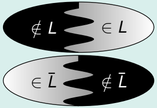
Proof

Let T be an $\mathbf{RP}$-PTM for L with running time $p(n)$.

$\hookrightarrow$ never errs for $x \notin L$

Let $\bar{T}$ be an $\mathbf{RP}$-PTM for $\bar{L}$ with running time $p(n)$.

$\hookrightarrow$ never errs for $x \notin \bar{L}$



repeat
  $r_1 \leftarrow T(w)$
  $r_2 \leftarrow$ **not** $\bar{T}(w)$
until $r_1 = r_2$
return r_1

- T and $\bar{T}$ never *both* answer incorrectly $\Rightarrow$ we always answer correctly.
- Every round gives $r_1 = r_2$ with probability $\geq 1/2$.

$$\mathbb{E}[\text{running time}] \leq 2p(|w|) \cdot \mathbb{E}[\#\text{rounds}] \overset{\text{TSF}}{=} 2p(|w|) \cdot \sum_{i \geq 1} \Pr[\#\text{rounds} \geq i] \leq 2p(n) \cdot \sum_{i \geq 1} 2^{-(i-1)} = 2p(n) \cdot \sum_{i \geq 0} 2^{-i} = 4p(n). \quad \square$$

Complete Problems?

Remark

The classes $\mathbf{RP}$, $\mathrm{co}{-}\mathbf{RP}$ and $\mathbf{BPP}$ are not believed to have complete problems unless, e.g. $\mathbf{BPP} = \mathbf{P}$.

A complete problem for $\mathbf{NP}$

$$L = \{(T, x) \mid T \text{ is an } \mathbf{NP}\text{-NTM in normal form and } T \text{ accepts } x\}$$

- L is $\mathbf{NP}$-hard ✓

  Assume $L' \in \mathbf{NP}$
  $\Rightarrow$ there exists $\mathbf{NP}$-NTM T for L' in normal form
  $\Rightarrow$ reduction: $x \in L' \Leftrightarrow (T, x) \in L$

- $L \in \mathbf{NP}$ ✓
  - check if T is $\mathbf{NP}$-NTM in normal form // $\in \mathbf{P}$
  - check if T accepts x // simulate

A complete problem for $\mathbf{RP}$?

$$L = \{(T, x) \mid T \text{ is an } \mathbf{RP}\text{-PTM in normal form and } \Pr[T \text{ accepts } x] \geq 1/2\}$$

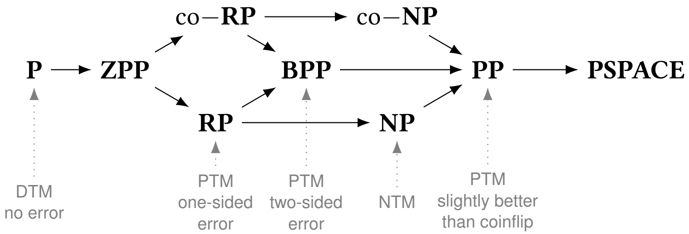
- L is $\mathbf{RP}$-hard ✓

  Assume $L' \in \mathbf{RP}$
  $\Rightarrow$ there exists $\mathbf{RP}$-PTM T for L' in normal form
  $\Rightarrow$ reduction: $x \in L' \Leftrightarrow (T, x) \in L$

- $L \in \mathbf{RP}$ ✗
  - check if T is $\mathbf{RP}$-PTM in normal form ✗
    ⚠ **undecidable**!
  - check if T accepts x // simulate

Content

Beziehungen zwischen Komplexitätsklassen



Exercise

- $\mathbf{P} \subseteq \mathbf{ZPP}$
- $\mathbf{ZPP} \subseteq \mathbf{RP}$ and $\mathbf{ZPP} \subseteq \text{co}-\mathbf{RP}$
- $\mathbf{RP} \subseteq \mathbf{NP}$ and $\text{co}-\mathbf{RP} \subseteq \text{co}-\mathbf{NP}$
- $\mathbf{RP} \subseteq \mathbf{BPP}$ and $\text{co}-\mathbf{RP} \subseteq \mathbf{BPP}$
- $\mathbf{BPP} \subseteq \mathbf{PP}$

Following Slides

- $\mathbf{NP} \subseteq \mathbf{PP}$ and $\text{co}-\mathbf{NP} \subseteq \mathbf{PP}$
- $\mathbf{PP} \subseteq \mathbf{PSPACE}$

“Typecasting” Turing Machines

DTM as NTM

Given DTM T with transition function δ, consider NTM T' with transition functions $\delta_0 = \delta_1 = \delta$.
$\hookrightarrow$ No change in behaviour: $T(w) = \text{YES} \Leftrightarrow T'(w) = \text{YES}$.

NTM as PTM

Given NTM T, we can reinterpret it as a PTM T':

$$T(w) = \text{YES} :\Leftrightarrow \exists\text{YES-computation for } T \text{ and } w \Leftrightarrow \Pr[T'(w) = \text{YES}] > 0$$
$$T(w) = \text{NO} :\Leftrightarrow \nexists\text{YES-computation for } T \text{ and } w \Leftrightarrow \Pr[T'(w) = \text{YES}] = 0$$

PTM as DTM

Given PTM T, we can view it as DTM T' with random bitstring $b = b_1 b_2 \dots$ as additional input.
In step i transition function δ_{b_i} is used.

$$\Pr[T(w) = \text{YES}] = \Pr_{b_1, b_2, \dots \sim Ber(1/2)}[T'(w, b) = \text{YES}].$$

Theorem: $\mathbf{NP} \subseteq \mathbf{PP}$ (analogously $\text{co}-\mathbf{NP} \subseteq \mathbf{PP}$)

i.e. show that each $L \in \mathbf{NP}$ satisfies $L \in \mathbf{PP}$

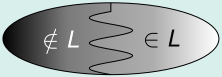
Have: NTM T certifying that $L \in \mathbf{NP}$

$w \in L \Leftrightarrow \exists$YES-computation for T and w

Use the NTM T as a PTM T':

$\forall w \notin L : \Pr[T'(w) = \text{YES}] = 0$
$\forall w \in L : \Pr[T'(w) = \text{YES}] > 0$

Want: PTM T'' certifying that $L \in \mathbf{PP}$



$\forall w \notin L : \Pr[T''(w) = \text{YES}] \leq 1/2$
$\forall w \in L : \Pr[T''(w) = \text{YES}] > 1/2$

T'' achieves this shift with a simple trick

$r \leftarrow T'(w)$ // T' is T as PTM
if $r = \text{YES}$ **then**
  return YES
else
  sample $b \sim \mathcal{U}(\{\text{YES}, \text{NO}\})$ // coinflip
  return b

Theorem: $\mathbf{PP} \subseteq \mathbf{PSPACE}$

i.e. show that each $L \in \mathbf{PP}$ satisfies $L \in \mathbf{PSPACE}$

Karlsruhe Institute of Technology

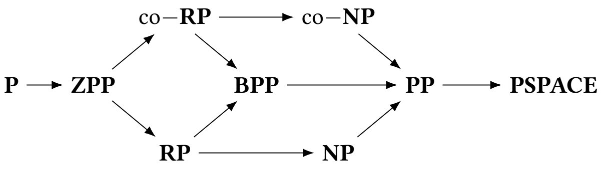
Proof

- Let T a $\mathbf{PP}$-PTM for L with running time $p(n)$.
- Consider DTM T' that simulates T for given w and random choices $b_1 b_2 \dots b_{p(n)}$.
- Consider DTM T'' that for input w runs $T'(w, b_1 b_2 \dots b_{p(n)})$ for all $2^{p(n)}$ possible $b_1 b_2 \dots b_{p(n)}$. Return YES if T' returns YES in majority of cases.
- space complexity:
  - $p(n)$ bits for counter a
  - $p(n)$ bits for $b_1, \dots, b_k$
  - $\mathcal{O}(p(n))$ space for simulating T (can only use $p(n)$ space in its $p(n)$ steps)

$\hookrightarrow$ T'' decides L in space $\mathcal{O}(p(n))$ (and time $\Omega(2^{p(n)})$). $\square$

```
n ← |w|
k ← p(n)
a ← 0 // k-bit counter
for b_1 ... b_k ← 00...0 to 11...1 do
    r ← T'(w, b_1 ... b_k)
    if r = YES then
        a ← a + 1
if a > 2^(k−1) then
    return YES
else
    return NO
```

Conclusion



What we learned – not much

- Only “obvious” inclusions known
  ↪ e.g. one-sided error vs. two-sided error
- since $\mathbf{P} \stackrel{?}{=} \mathbf{PSPACE}$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
  - obviously: in $\text{co}-\mathbf{NP}$.
  - 1976: in $\text{co}-\mathbf{RP}$ (Rabin).
  - 1987: in $\mathbf{RP}$, hence in $\mathbf{ZPP}$ (Adleman, Huang).
  - 2002: in $\mathbf{P}$ (Agrawal, Kayal, Saxena).

A boring topic?

- People believe $\mathbf{BPP} = \mathbf{P}$
  ↪ “each $\mathbf{BPP}$ algorithm can be fully derandomised”
- $\mathbf{PP}$ is somewhat esoteric
  ↪ no interesting randomised classes remain?
- quantum computing may change the story. People suspect $\mathbf{NP} \not\subseteq \mathbf{BQP} \not\subseteq \mathbf{NP}$
  ↪ https://en.wikipedia.org/wiki/BQP

Anhang: Mögliche Prüfungsfragen

- Definiere: Was ist eine PTM? Was ist der Unterschied zu einer NTM?
- Definiere die Komplexitätsklassen $\mathbf{RP}$, $\mathrm{co}-\mathbf{RP}$ $\mathbf{BPP}$, $\mathbf{PP}$, $\mathbf{ZPP}$.
- Inwiefern spielen die Konstanten von $\frac{1}{2}$, $\frac{1}{4}$, $\frac{3}{4}$, die in den Definitionen vorkommen, einen Rolle? Inwiefern sind sie egal?
- Inwiefern steht die Klasse $\mathbf{ZPP}$ mit dem Konzept eines Las-Vegas Algorithmus in Verbindung? Wie sehen die Umwandlungen in die eine Richtung (Vorlesung) und in die andere Richtung (Übung) aus?
- Welche Inklusionsbeziehungen zwischen diesen Komplexitätsklassen sind bekannt?
- Begründe jede dieser Inklusionsbeziehungen. (In der tatsächlichen Prüfung würde man sich aus Zeitgründen nur eine oder zwei herausgreifen.)
- Gibt es Inklusionsbeziehungen von denen man weiß, dass sie strikt sind? Gibt es Klassen, von denen Experten vermuten, dass sie in Wirklichkeit identisch sind?