# Complex Systems Theory and Practice

Lecture 15

Piotr Syga

2022-01-28

# Simulating NTM with a DTM

**Theorem**

Let NTM $N$ decide $L$ in time $f(n)$. Then, there exists a DTM $M$ with 3 tapes that decides $L$ in time $O\left(c^{f(n)}\right)$, for some $c > 1$ that depends only on $N$.

**Caveat**

$$\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n)) \subseteq \bigcup_{c \in \mathbb{N}_+} \mathrm{TIME}(c^{f(n)})$$

# Not all complexity classes are equal

## Theorem on time hierarchy

If $f(n) \geq n$ is a proper complexity function then $\mathrm{TIME}(f(n))$ is a proper subset of $\mathrm{TIME}(f(n)\log(f(n)))$.

## Caveat

$\mathbb{P}$ is a proper subset of $\mathbb{EXP}$.

## Theorem on space hierarchy

If $f(n) \geq n$ is a proper complexity function then the class $\mathrm{SPACE}(f(n))$ is a proper subset of $\mathrm{SPACE}(f(n)\log f(n))$.

# More on the hierarchy

**Theorem**

Let $f(n)$ be a proper complexity function, then:

- $\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$, $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$
- $\mathrm{NTIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))$
- $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{TIME}(c^{f(n)+\log n})$.

**Caveat**

$\mathbb{L} \subseteq \mathbb{NL} \subseteq \mathbb{P} \subseteq \mathbb{NP} \subseteq \mathbb{PSPACE}$
Moreover, $\mathbb{L} \subset \mathbb{PSPACE}$.

# The relations between problems

## Reduction (operation)

Having two problems $\mathcal{A}$ and $\mathcal{B}$, we say that $\mathcal{A}$ may be reduced to $\mathcal{B}$, when the solution of $\mathcal{B}$ can be easily translated to the solution of $\mathcal{A}$. Additionally, we say that $\mathcal{A}$ is not harder than $\mathcal{B}$.

## Reduction (algorithm)

A function that may be calculated on a deterministic TM in a logarithmic space (or a polynomial time for classes $\mathbb{P}$ and above) that transforms an instance of a problem $\mathcal{A}$ into an instance of $\mathcal{B}$ is called a reduction algorithm.

## Composition

If $R$ is a reduction of a problem $L_1$ to $L_2$ and $R'$ is a reduction of $L_2$ to $L_3$ then the composition $RR'$ is a reduction of $L_1$ to $L_3$.

# 3-Coloring $\xrightarrow{R}$ 3-SAT

## 3-Coloring

Having $G = (V, E)$, we decide if we can color the graph using only 3 colors so that no two connected vertices have the same color.

## 3-SAT

Having a logical formula $\phi$, we decide if there is a value assignment $\bar{x}$ for $\phi$'s variables so that $\phi(\bar{x}) = 1$.

# Reachability $\xrightarrow{R}$ Circuit Value

## Reachability

Having a graph $G = (V, E)$. Is there a path connecting two vertices $u, v \in V$?

## Circuit Value

Having a logical circuit (with gates `AND`, `OR`, `NOT`) we calculate the value (0 or 1) at the end of the circuit.

# Problems that are hard and complete problems

## Definition

Let $\mathcal{C}$ be a complexity class and $L$ be a language. We say that $L$ is $\mathcal{C} - \text{hard}$ if any language $L' \in \mathcal{C}$ can be reduced to $L$. Additionally, if $L \in \mathcal{C}$, then $L$ is $\mathcal{C} - \text{complete}$.

## Reduction

Note that the classes $\mathbb{L}$, $\mathbb{NL}$, $\mathbb{P}$, $\mathbb{NP}$, $\mathbb{coNP}$, $\mathbb{PSPACE}$, $\mathbb{EXP}$ are closed in terms of reduction.

## Caveat

If two complexity classes $\mathcal{C}$, $\mathcal{C}'$ are closed under reduction and there is a language $L$ that is complete for both of the classes then $\mathcal{C} = \mathcal{C}'$.

# Completeness in $\mathbb{P}$ and $\mathbb{NP}$

**Theorem**

Circuit Value is $\mathbb{P}-$complete.

**Theorem**

3-SAT is $\mathbb{NP}-$complete.

# Example of pseudo-(deterministic)polynomial problem

## Knapsack Problem is $\mathbb{NP}-\text{complete}$

Let us assume that we have $n$ objects. The $i$-th object weights $w_i \in \mathbb{N}_+$ and has a value of $v_i \in \mathbb{N}_+$. We also have an upper bound on weight $W$ and a target value $V$. We are looking for a subset of objects $K$, s.t. $\sum_{i \in K} w_i \leq W$ and $\sum_{i \in K} v_i \geq V$.
We may assume that for every $i$ we have $w_i = v_i$ and $W = V$ (the difficulty does not change).
*Idea*: 3SAT$\rightarrow$Tripartite Matching$\rightarrow$Exact Cover by 3-Sets$\rightarrow$Knapsack

# Knapsack – sample solution

## Solving Knapsack Problem in $O(nW)$ time

$V(w, i)$ – denotes the highest value we can achieve by selecting only from $i$ first objects and the weight $w$

$V(w, i+1) = \max\{V(w, i), v_{i+1} + V(w - w_{i+1}, i)\}$

$V(W, n) = ?$

# Sample code

```
def knapsack(W, wi, val, n):
    K = [[0 for x in range(W + 1)] for x in range(n + 1)]

    for i in range(n + 1):
        for w in range(W + 1):
            if i == 0 or w == 0:
                K[i][w] = 0
            elif wi[i-1] <= w:
                K[i][w] = max(val[i-1]
                              + K[i-1][w-wi[i-1]],
                                  K[i-1][w])
            else:
                K[i][w] = K[i-1][w]

    return K[n][W]
```

# Monte Carlo

## Monte Carlo algorithms

For a decisional problem $A$, a polynomial algorithm $\mathcal{A}$ is a Monte Carlo-type algorithm if:

F-biased for each positive instance, it returns `yes`, for a negative instance it returns `no` with probability greater than $\frac{1}{2}$

T-biased for each negative instance, it returns `no`, for a positive instance it returns `yes` with probability greater than $\frac{1}{2}$

2-sided we have bounded probability of error on both sides.

# Las Vegas

## Las Vegas algorithms

A randomized alogrithm is a Las Vegas-type if for a decisional problem $A$ always returns the correct answer in a finite time and the expected (mean) termination time is polynomial.

# Randomness and TM

## Normalized NTM

We say that NTM $N$ is normalized if all its calculation paths are of equal length and in each step we decide between exactly two nondeterministic paths to take.

# RP and coRP

## RP class

If for some $L$, a NTM $N$ stops in a polynomial time $p(|x|)$ on each input $x$ and at least half of the paths end with `yes` for $x \in L$ and for each $x \notin L$ $N$ returns `no`, then $L$ is in RP.

## coRP class

If for some $L$, a NTM $N$ stops in a polynomial time $p(|x|)$ on each input $x$ and at least half of the paths end with `no` for $x \notin L$ and for each $x \notin L$ $N$ returns `no`, then $L$ is in RP.

# ZPP

## ZPP class

ZPP complexity class is defined as the set intersection of RP∩coRP. Problems from ZPP can be solved using Las Vegas algorithms.

- Let $L$ recognized by RP algorithm $A$ and coRP algorithm $B$.
- For any $x$ run $A$ for one step. If it returns `yes`, return `YES`. Otherwise, run $B$ on $x$ for one step. If it returns `no`, return `NO`. If neither occurs, repeat this step.
- Only one of $A$ and $B$ can give a wrong answer (with probability at most $\frac{1}{2}$ in each step). Hence, reaching the $k$-th round shrinks exponentially in $k$ giving polynomial expected running time. This shows that RP intersect co-RP is contained in ZPP
- Let us assume that we have a LV-type algorithm $V$.
- Run $V$ for at least double its expected running time. If it returns the answer it, return it for TM. Otherwise, return `no` for RP (or `yes` for coRP).

# BPP

## BPP class

All languages $L$, for which a normalized NTM has at least $\frac{3}{4}$ accepting paths for all $x \in L$ and at least $\frac{3}{4}$ rejecting paths for $x \notin L$ create BPP class.

# PP

## PP class

A language $L$ belongs to PP class if there exists a normalized NTM such that $x \in L$ **iff** over half of paths for $x$ is accepting.

## Example

A problem, that for a given logic formula $\phi$ with $n$ variables decides if most of the possible value assignments returns `true` is called MajSAT and it belongs to PP.

# PP

## PP class

A language $L$ belongs to PP class if there exists a normalized NTM such that $x \in L$ **iff** over half of paths for $x$ is accepting.

## Example

A problem, that for a given logic formula $\phi$ with $n$ variables decides if most of the possible value assignments returns `true` is called MajSAT and it belongs to PP.

## Caveat

NP $\subseteq$PP

# Decision and optimization problems

## Relation

Let us assume that we have an optimization problem $\mathbb{P}$, we can easily determine a decision problem $\mathbb{D}$ related with it – „Is there an acceptable solution for a given instance $I$ of $\mathbb{P}$ and set value $V$ that has the value of objective function lower than $V$?" (e.g. Can a graph $G$ be 3-colored). Knowing the optimal answer for $\mathbb{P}(I)$ we can solve $\mathbb{D}$, having an algorithm solving $\mathbb{D}$ we can find the optimal solution of $\mathbb{P}$.

# Basic definitions

## Cost function

Let $A$ be an optimization problem that for each instance $x$ has a set of solutions $S(x)$. For each $s \in S(x)$ we determine a *cost* $c(s) \in \mathbb{N}_+$ as function determined by the value of parameters in $s$.

## Optimal solution

A solution $s^*$ to an optimization problem which minimizes the cost function (i.e. $\forall_{s \in S(x)} c(s^*) \leq c(s)$) is called optimal.

## $\varepsilon$–approximation

Let $\mathrm{Opt}(x) = \min_{s \in S(x)} c(s)$. For algorithm $M$, that solves optimization problem s.t. $M(x) \in S(x)$, we say that it is an $\varepsilon$–approximation for $\varepsilon \geq 0$, if for every $x$ we have

$$\frac{|c(M(X)) - \mathrm{Opt}(x)|}{\max\{\mathrm{Opt}(x), c(M(x))\}} \leq \varepsilon \ .$$

## Approximation threshold

The lower bound of all $\varepsilon > 0$ for which there is a polynomial $\varepsilon$–approximating algorithm $A$ for a problem $P$ is called the approximation threshold for $P$.

# Example

## Knapsack

The approximation threshold for Knapsach is 0.
A dual algorithm for the pseudopolynomial solution gives $\varepsilon = \frac{n2^b}{V}$–approximation in time $O(\frac{n^2 V}{2^b})$, for some $b \in \mathbb{N}_+$.

# Approximation schemes

## Polynomial approximation scheme

A polynomial approximation scheme for a problem $A$ is an algorithm that for each $\varepsilon > 0$ and instance $x$ returns a solution with relative error of at most $\varepsilon$ and in time bounded by $\mathrm{poly}(|x|)$.