Complexity Theory of Polynomial-Time Problems

Lecture 13: Recap, Further Directions, Open Problems

Karl Bringmann

I. Recap

II. Further Directions

III. Open Problems

I. Recap

Hard problems

SAT: given a formula in conj. normal form on n variables

is it satisfiable?

conjecture: no $O(2^{(1-\varepsilon)n})$ algorithm (SETH)

OV: given n vectors in $\{0,1\}^d$ (for small d)

are any two orthogonal?

conjecture: no $O(n^{2-\varepsilon})$ algorithm

APSP: given a weighted graph with n vertices

compute the distance between any pair of vertices

conjecture: no $O(n^{3-\varepsilon})$ algorithm

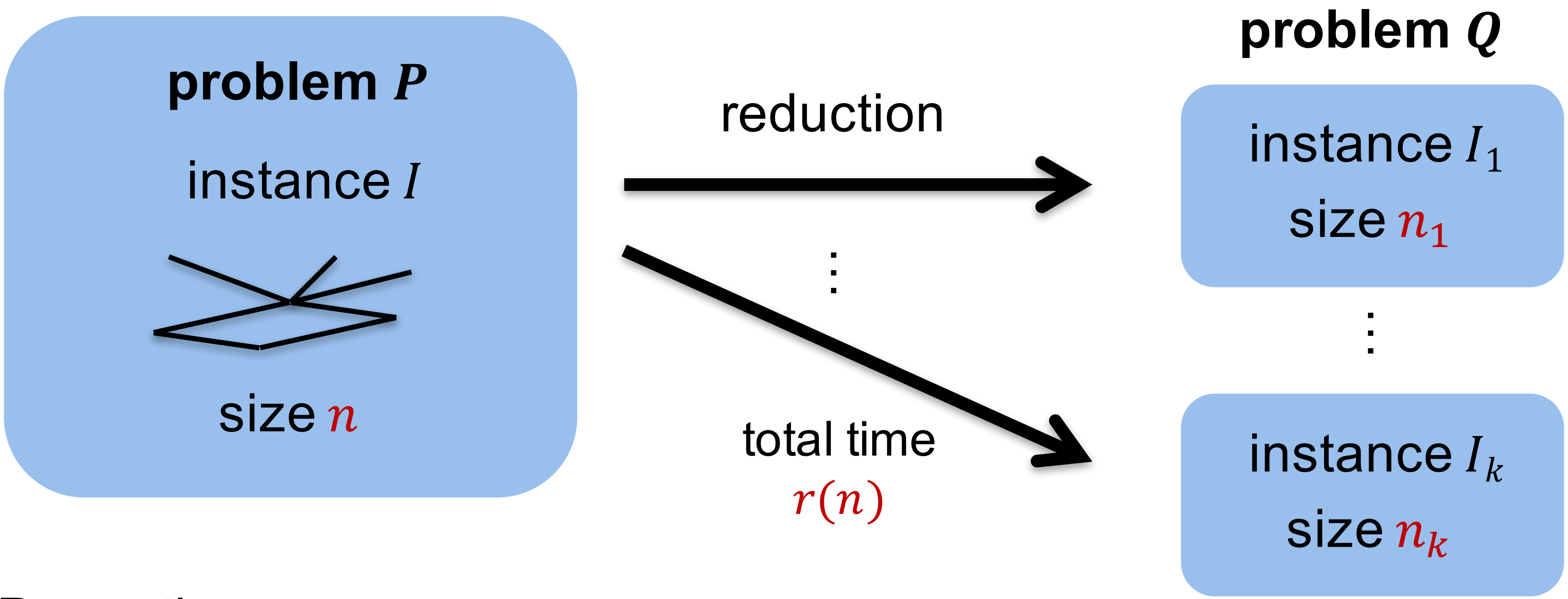
3SUM: given n integers

do any three sum to 0?

conjecture: no $O(n^{2-\varepsilon})$ algorithm

Fine-Grained Reductions

A **fine-grained reduction** from (P,T) to (Q,T') is an algorithm A for P with **oracle** access to Q s.t.:



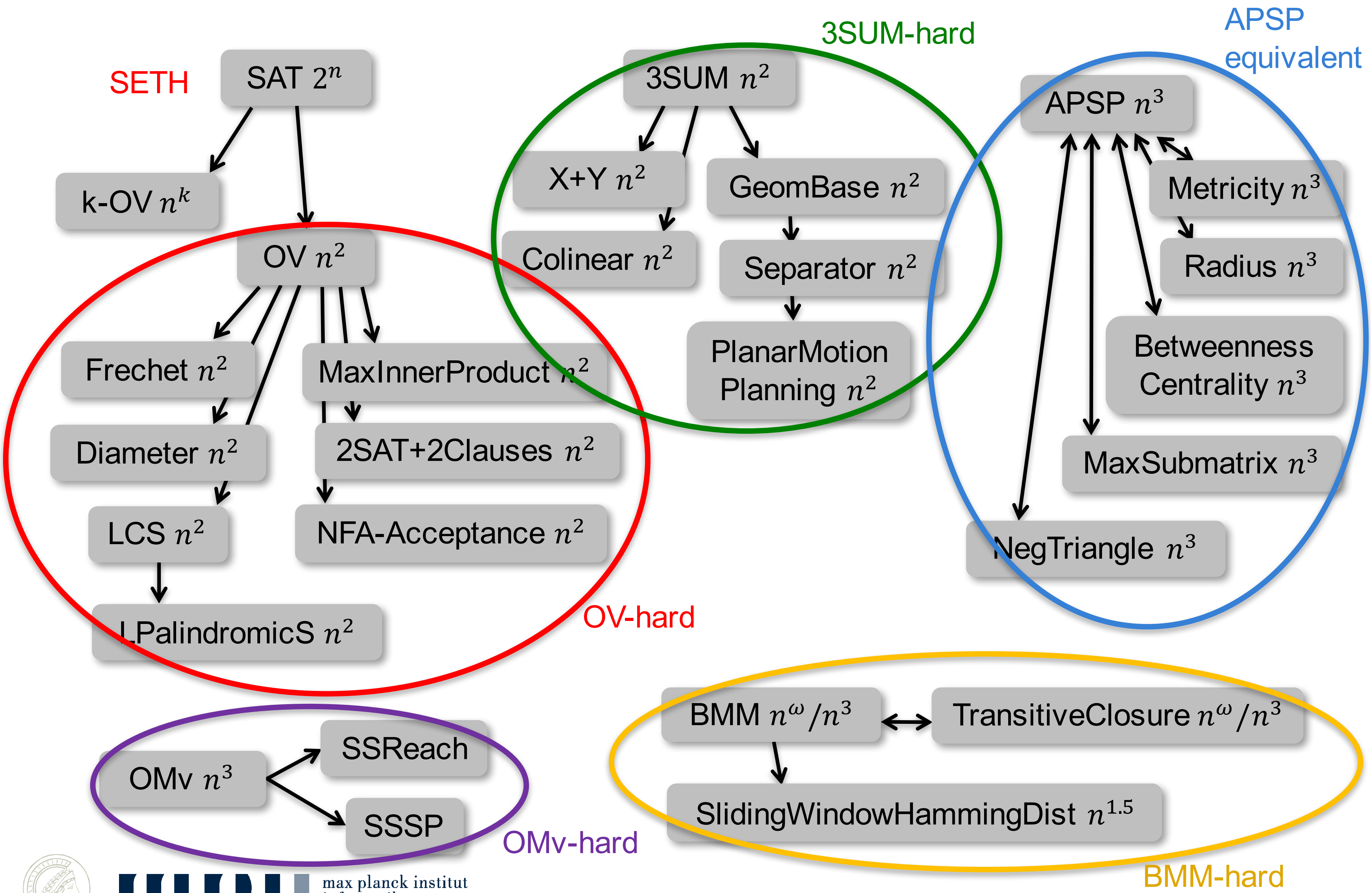
Properties:

for any instance I, algorithm $A(I)$ correctly solves problem P on I

A runs in time $r(n) = O(T(n)^{1-\gamma})$ for some $\gamma > 0$

for any $\varepsilon > 0$ there is a $\delta > 0$ s.t. $\sum_{i=1}^{k} T'(n_i)^{1-\varepsilon} \leq T(n)^{1-\delta}$

Complexity Inside P



max planck institut
informatik

Conditional Lower Bounds …

… allow to classify polynomial time problems

… are an analogue of NP-hardness

yield good reasons to stop searching for faster algorithms

should belong to the basic toolbox of theoretical computer scientists

… allow to search for new algorithms with better focus

improve SAT before longest common subsequence

non-matching lower bounds suggest better algorithms

… motivate new algorithms

relax the problem and study approximation algorithms,
parameterized running time, …

Algorithms

BMM:

fast matrix multiplication: $\omega < 3$

NodeWeightedNegativeTriangle $O(n^{\omega})$

k-Clique $O(n^{\omega k/3})$

MaxCut $O(2^{\omega n/3}\mathrm{poly}(n))$

lower order improvements:

4 Russians trick yields log-factor improvements

polynomial method:

OV in $O(n^{2-1/O(\log(d/\log n))})$ or $O(n^2/2^{\Omega(\sqrt{\log n})})$

APSP and OMv in $O(n^3/2^{\Omega(\sqrt{\log n})})$

dynamic graph algorithms:

incremental/decremental/fully dynamic, amortized/worst-case, update/query time for shortest path and reachability problems

More (Algorithmic) Concepts

Decision Trees:

$\tilde{O}(n^{3/2})$ for 3SUM

(Co-)Nondeterministic Algorithms:

$\tilde{O}(n^{3/2})$ for 3SUM

Nondeterministic SETH

Polynomials and Circuits:

Over the integers, or $\mathbb{Z}_p$, or AND-OR

polynomial multiplication (FFT!), division, interpolation

multipoint evaluation

Modern Algorithms Philosophy

“Fast Matrix Multiplication and Fast Fourier Transform are the only non-trivial algorithmic tools that we have.”

“Whenever you design an algorithm, try to prove a matching conditional lower bound to show optimality.”

II. Further Directions

1. Hardness Under Multiple Hypotheses

Hardness Under Multiple Hypotheses

so far we identified **a reason** for the hardness of some problems

what about **multiple reasons**?

if a problem Q is SETH-hard **and** 3SUM-hard **and** APSP-hard

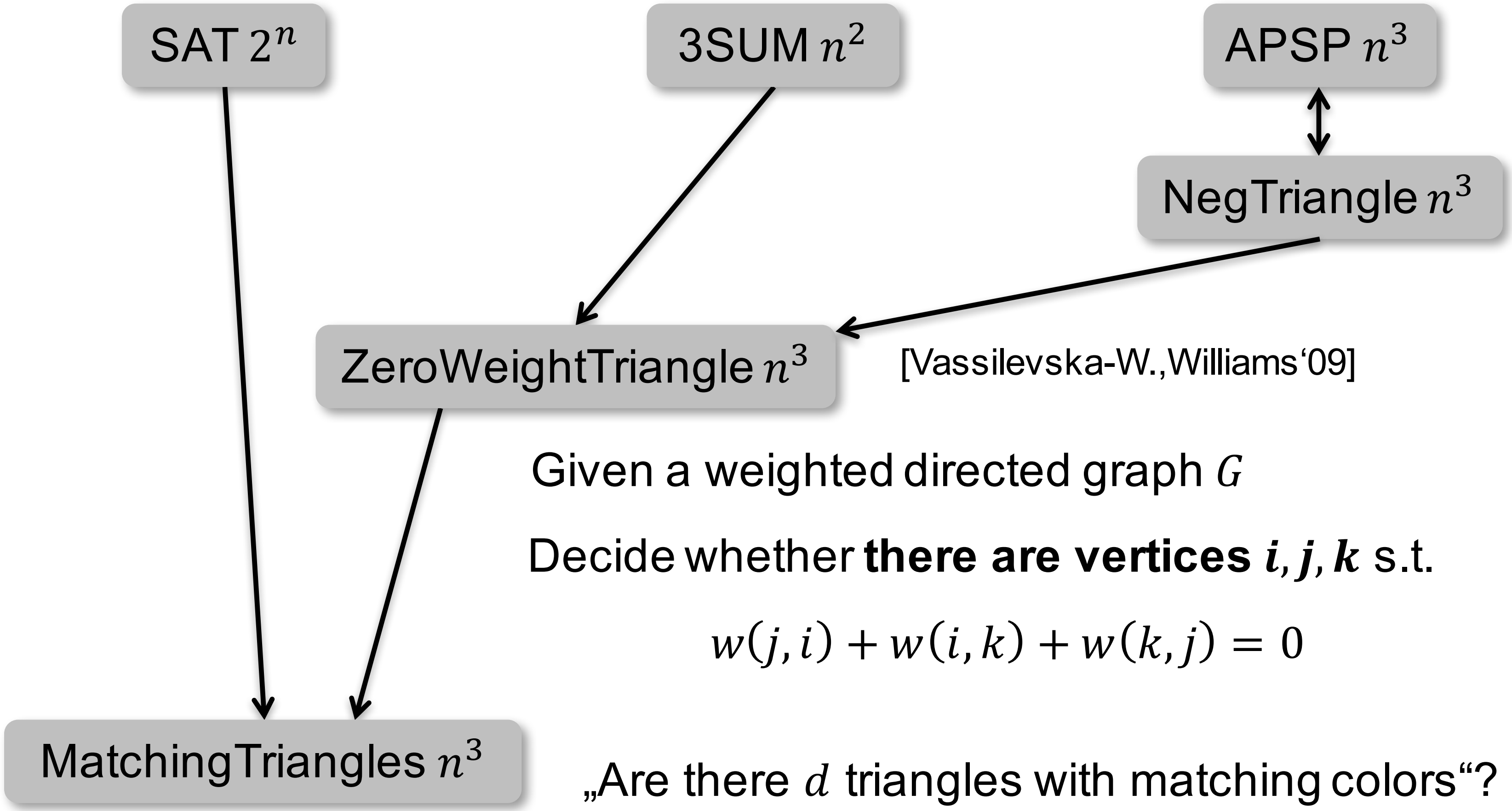
then it is as hard as it gets

we can forget about improved algorithms any time soon

Q is hard under the following weak conjecture:

At least one of SETH or 3SUM-H or APSP-H holds

Complexity Inside P



[Vassilevska-W.,Williams‘09]

Given a weighted directed graph G

Decide whether **there are vertices** $\boldsymbol{i, j, k}$ s.t.

$$w(j,i) + w(i,k) + w(k,j) = 0$$

„Are there d triangles with matching colors“?

Given a directed graph G with colored nodes, and number d

Are there colors a, b, c s.t. G contains at least d triangles (i, j, k) where i has color a, j has color b, and k has color c?

[Abboud,Vassilevska-W.,Yu‘15]

II. Further Directions

2. Ruling out Superpolylogarithmic Improvements

Ruling out Superpolylog Improvements

we have seen for Longest Common Subsequence (LCS):

4 Russians trick: $O(n^2/\log^2 n)$

OV-hardness: lower bound $\Omega(n^{2-\varepsilon})$

does the polynomial method apply?

is LCS in time $n^2/2^{\Omega(\sqrt{\log n})}$? or $n^2/\log^{\omega(1)} n$?

how to rule out superpolylogarithmic improvements?

SETH/OVH are too coarse!

OV has superpolylogarithmic improvements!

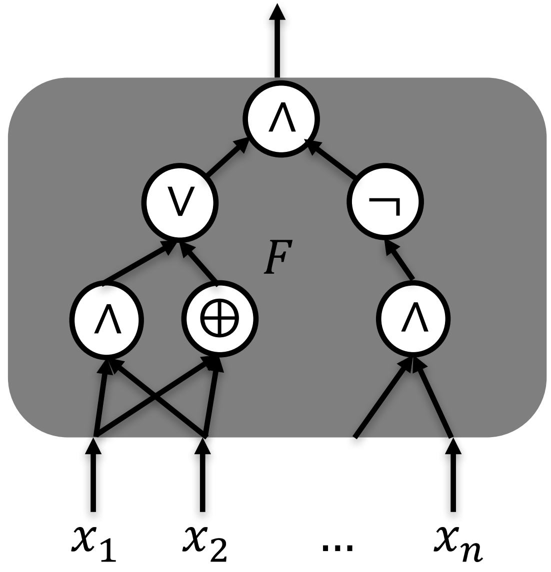
SAT

$F\colon \{0,1\}^n \to \{0,1\}$

Is F satisfiable?

Need $\Omega(2^n)$ queries, unless we analyze F

Hardness of F-SAT depends on our ability to analyze F



formula = circuit over AND/OR/NOT (and maybe XOR/...), where every gate has fanout 1, except for inputs

F given by a...

DNF: polytime

CNF: SETH

Turing machine: extremely hard

formula: harder than SETH, but still easy to work with

no $2^n/n^{\omega(1)}$-algorithm known, even for $|F| = n^{O(1)}$

Formula-SAT Hardness

Formula-SAT

formula F
on n variables
size $|F| = n^{O(1)}$

reduction

time $2^{n/2} n^{O(1)}$

LCS

strings x, y

of length $2^{n/2} \cdot n^{O(1)}$

$2^{(1-\varepsilon/2)n} \cdot n^{O(1)}$ algorithm	$\Longleftarrow$	$O(n^{2-\varepsilon})$ algorithm
$2^n / n^{\omega(1)}$ algorithm	$\Longleftarrow$	$n^2 / \log^{\omega(1)} n$ algorithm

no $2^n / n^{\omega(1)}$ algorithm known,
even for $|F| = n^{O(1)}$

Thm: [Abboud,Hansen,Vassilevska-W,Williams'16]
LCS has no $n^2 / \log^{\omega(1)} n$ algorithm
unless Formula-SAT has an $2^n / n^{\omega(1)}$ algorithm for $|F| = n^{O(1)}$.

max planck institut
informatik

this only works for few problems!

II. Further Directions

3. Hardness Classes

Hardness Classes

the field of „fine-grained complexity“ is very **problem-centric**

everything evolves around hard problems

what about **classes of problems**, as in classic complexity?

can we prove hardness/completeness of a problem for a class
w.r.t. fine-grained reductions?

Hardness Classes

class of **first-order graph properties**:

given a graph G with m edges

given a first-order formula ϕ with $k+1$ quantifiers

check whether ϕ evaluates to true on G

is in time $O(m^k)$
[Gao,Impagliazzo'16]

e.g. $\exists u \exists v \exists w\colon E(u,v) \wedge E(v,w) \wedge E(u,w)$
„does G contain a triangle?"

$\forall u \forall v \exists w\colon E(u,w) \wedge E(w,v)$
„does G have diameter=2?"

Thm: The following are equivalent: [Gao,Impagliazzo'16]

- $\exists \varepsilon, \delta > 0$: OV in dimension $d \leq n^{\delta}$ has an $O(n^{2-\varepsilon})$ algorithm (¬OVH)
- $\forall k \geq 2$: for any first-order formula ϕ with $k+1$ quantifiers there exists $\varepsilon' > 0$ s.t. ϕ can be decided in time $O(m^{k-\varepsilon'})$ on any given graph with m edges

II. Further Directions

4. Multivariate Analysis

Multivariate Analysis

LCS is OV-hard: lower bound $\Omega(n^{2-\varepsilon})$

what if $|y| = m \ll n = |x|$? How fast can we compute $\mathrm{LCS}(x,y)$?

dynamic programming: $O(nm)$

Parameter Setting $LCS(\beta)$**:** $(0 \leq \beta \leq 1)$

.. is the LCS problem restricted to strings with $m = \Theta(n^\beta)$

what is the best running time $n^{f(\beta)+o(1)}$ for $LCS(\beta)$ for any β?

Cor: unless OVH fails, for any β any algorithm for $LCS(\beta)$ takes time

$$\Omega(n) + n^{2\beta - o(1)} = \Omega(n) + m^{2-o(1)}$$

Proof: for any n, $m = \lceil n^\beta \rceil$, for any (binary) strings with $|x'|, |y'| = m - 1$:

$x := x'\ 2^{n-|x'|}$ with length $|x| = n$ where 2 is a fresh symbol

$y := y'\ 2^{m-|y'|}$ $|y| = m$

$$LCS(x,y) = LCS(x',y') + \min\{n - |x'|, m - |y'|\}$$

Multivariate Analysis

LCS is OV-hard: lower bound $\Omega(n^{2-\varepsilon})$

what if $|y| = m \ll n = |x|$? How fast can we compute $\mathrm{LCS}(x, y)$?

dynamic programming: $O(nm)$

Parameter Setting $LCS(\beta)$**:** $(0 \leq \beta \leq 1)$

.. is the LCS problem restricted to strings with $m = \Theta(n^{\beta})$

what is the best running time $n^{f(\beta)+o(1)}$ for $LCS(\beta)$ for any β?

Cor: unless OVH fails, for any β any algorithm for $LCS(\beta)$ takes time

$$\Omega(n) + n^{2\beta - o(1)} = \Omega(n) + m^{2-o(1)}$$

Algorithm with time $\tilde{O}(n + m^2)$ exists! [Hirschberg'77]

LCS differs from other similarity measures that take time $(nm)^{1-o(1)}$

[B.'14, B.,Künnemann'15]

Multivariate Analysis

Multivariate fine-grained complexity

Parameter Setting $LCS(\beta)$**:** $(0 \leq \beta \leq 1)$

.. is the LCS problem restricted to strings with $m = \Theta(n^\beta)$

what is the best running time $n^{f(\beta)+o(1)}$ for $LCS(\beta)$ for any β?

Cor: unless OVH fails, for any β any algorithm for $LCS(\beta)$ takes time

$$\Omega(n) + n^{2\beta - o(1)} = \Omega(n) + m^{2-o(1)}$$

Algorithm with time $\tilde{O}(n + m^2)$ exists!

More Parameters for LCS

more parameters have been studied for LCS since the 70s:

$n = |x| = \max\{|x|, |y|\}$ .. length of longer string

$m = |y| = \min\{|x|, |y|\}$ .. length of shorter string

$L = \mathrm{LCS}(x, y)$ .. length of LCS

$|\Sigma|$ .. size of alphabet Σ

$\Delta = n - L$ .. number of deletions in x

$\delta = m - L$ .. number of deletions in y

$M = \{(i, j) \mid x[i] = y[j]\}$ .. number of *matching pairs*

	a	b	b	c	a	d
a	1	1	1	1	1	1
c	1	1	1	2	2	2
d	1	1	1	2	2	3
a	1	1	1	2	3	3
a	1	1	1	2	3	3
b	1	2	2	2	3	3
d	1	2	2	2	3	4

More Parameters for LCS

more parameters have been studied for LCS since the 70s:

$n = |x| = \max\{|x|, |y|\}$ .. length of longer string

$m = |y| = \min\{|x|, |y|\}$ .. length of shorter string

$L = \mathrm{LCS}(x, y)$ .. length of LCS

$|\Sigma|$ .. size of alphabet Σ

$\Delta = n - L$ .. number of deletions in x

$\delta = m - L$ .. number of deletions in y

$M = \{(i, j) \mid x[i] = y[j]\}$ .. number of *matching pairs*

d .. number of *dominant pairs*

= “entry (i, j) is dominant if all entries to the top left of it are strictly smaller”

	a	b	b	c	a	d
a	1	1	1	1	1	1
c	1	1	1	2	2	2
d	1	1	1	2	2	3
a	1	1	1	2	3	3
a	1	1	1	2	3	3
b	1	2	2	2	3	3
d	1	2	2	2	3	4

Known Algorithms

$n = \max\{|x|, |y|\}$ $m = \min\{|x|, |y|\}$
$L = \mathrm{LCS}(x, y)$ $|\Sigma|$ alphabet size
$\Delta = n - L$ M matching pairs
$\delta = m - L$ d dominating pairs

$O(nm)$	[Wagner,Fischer'74]
$\tilde{O}(n+M)$	[Hunt,Szymanski'77]
$\tilde{O}(n+\delta m)$	[Hirschberg'77]
$\tilde{O}(n+Lm)$	[Hirschberg'77]
$\tilde{O}(n+d)$	[Apostolico'86]
$\tilde{O}(n+\delta\Delta)$	[Wu,Manber,Myers,Miller'90]

logfactor improvements:

[Masek,Paterson'80],
[Apostolico,Guerra'87],
[Eppstein,Galil,Giancarlo,
Italiano'92],
[Bille,Farach-Colton'08],
[Iliopoulos,Rahman'09]

What is the best possible algorithm for any "parameter setting"?

Parameter Settings

$n = \max\{\lvert x\rvert, \lvert y\rvert\}$	$m = \min\{\lvert x\rvert, \lvert y\rvert\}$
$L = \mathrm{LCS}(x, y)$	$\lvert\Sigma\rvert$ alphabet size
$\Delta = n - L$	M matching pairs
$\delta = m - L$	d dominating pairs

let $\alpha = (\alpha_m, \alpha_L, \alpha_\Sigma, \alpha_\Delta, \alpha_\delta, \alpha_M, \alpha_d) \in \mathbb{R}^7_{\geq 0}$

parameter setting LCS(α): is the LCS problem restricted to strings x, y with

$$(n = |x|) \quad m = \Theta(n^{\alpha_m}) \quad L = \Theta(n^{\alpha_L}) \quad |\Sigma| = \Theta(n^{\alpha_\Sigma}) \quad \text{etc.}$$

we always have $L \leq m$

so $\alpha_L > \alpha_m$ is contradictory

in this case LCS(α) has only finitely many instances = LCS(α) is **trivial**

We have to understand the interdependencies of parameters first!

Parameter Relations

$n = \max\{\lvert x\rvert, \lvert y\rvert\}$	$m = \min\{\lvert x\rvert, \lvert y\rvert\}$
$L = \mathrm{LCS}(x, y)$	$\lvert\Sigma\rvert$ alphabet size
$\Delta = n - L$	M matching pairs
$\delta = m - L$	d dominating pairs

For any strings x, y we have:

$$\left.\begin{array}{l} L \leq m \leq n \\ m \geq \delta \leq \Delta \leq n \\ d \leq M \end{array}\right\} \text{trivial}$$

$$\left.\begin{array}{l} |\Sigma| \leq m \\ M \geq n \end{array}\right\} \text{w.l.o.g. every symbol in } \Sigma \text{ appears in } x \text{ and in } y$$

$$L \leq d \leq Lm$$

$$|\Sigma| \leq d \leq L^2|\Sigma|$$

$$d \leq 2L(\Delta + 1)$$

$$L^2/|\Sigma| \leq M \leq 2Ln$$

complex dependencies of the parameters!

max planck institut informatik

up to constants, this is a complete list of relations

Known Algorithms

$n = \max\{\lvert x\rvert, \lvert y\rvert\}$	$m = \min\{\lvert x\rvert, \lvert y\rvert\}$
$L = \mathrm{LCS}(x, y)$	$\lvert\Sigma\rvert$ alphabet size
$\Delta = n - L$	M matching pairs
$\delta = m - L$	d dominating pairs

$\cancel{O(nm)}$	[Wagner,Fischer'74]
$\cancel{\tilde{O}(n + M)}$	[Hunt,Szymanski'77]
$\tilde{O}(n + \delta m)$	[Hirschberg'77]
$\cancel{\tilde{O}(n + Lm)}$	[Hirschberg'77]
$\tilde{O}(n + d)$	[Apostolico'86]
$\tilde{O}(n + \delta\Delta)$	[Wu,Manber,Myers,Miller'90]

parameter relations:

$$\delta \le m \le n$$
$$d \le M$$
$$d \le Lm$$
$$d \le L^2 \lvert\Sigma\rvert$$
$$\dots$$

Best algorithm: $\tilde{O}(n + \min\{d, \delta m, \delta\Delta\})$

Parameter Settings

$n = \max\{|x|,|y|\}$	$m = \min\{|x|,|y|\}$
$L = \mathrm{LCS}(x,y)$	$|\Sigma|$ alphabet size
$\Delta = n - L$	M matching pairs
$\delta = m - L$	d dominating pairs

let $\alpha = (\alpha_m, \alpha_L, \alpha_\Sigma, \alpha_\Delta, \alpha_\delta, \alpha_M, \alpha_d) \in \mathbb{R}^7_{\geq 0}$

parameter setting LCS(α): is the LCS problem restricted to strings x, y with

$(n = |x|)$ $m = \Theta(n^{\alpha_m})$ $L = \Theta(n^{\alpha_L})$ $|\Sigma| = \Theta(n^{\alpha_\Sigma})$ etc.

a parameter setting is **nontrivial** if it contains infinitely many instances

iff the target values $(n, n^{\alpha_m}, n^{\alpha_L}, \ldots)$ satisfy our parameter relations (for $n \to \infty$)

What is the best possible running time $n^{f(\alpha)+o(1)}$ for any nontrivial LCS(α)?

Matching Lower Bound

$n = \max\{|x|,|y|\}$	$m = \min\{|x|,|y|\}$
$L = \mathrm{LCS}(x,y)$	$|\Sigma|$ alphabet size
$\Delta = n - L$	M matching pairs
$\delta = m - L$	d dominating pairs

Best algorithm: $\tilde{O}(n + \min\{d, \delta m, \delta\Delta\})$

What is the best possible running time $n^{f(\alpha)+o(1)}$ for any nontrivial LCS(α)?

[B.,Künnemann'16+]

Thm:

Unless OVH fails, for any non-trivial parameter setting LCS(α) any algorithm takes time at least

$$\Omega(n) + \min\{d, \delta m, \delta\Delta\}^{1-o(1)}$$

III. Open Problems

Major Open Problems

1) prove conditional lower bounds for **more types of problems**

2) relate **SAT**, **3SUM**, **APSP** or show (more) barriers for such relations

3) advance **subquadratic approximation algorithms** and
develop tools for **hardness of approximation**

4) explain gap between **deterministic / randomized** algorithms

5) **average case** hardness? **distributed** algorithm? other settings?

… this is a young field of research!

k-Longest Common Subsequence (k-LCS)

given strings $x_1, \ldots, x_k$, each of length at most n,
compute longest string z that is a subsequence of all x_i

natural dynamic program $O(n^k)$

[Abboud,Backurs,V-Williams'15]

reduction SAT → k-OV → k-LCS yields lower bound of $\Omega(n^{k-\varepsilon})$

but only for strings over alphabets of size $\Omega(k)$

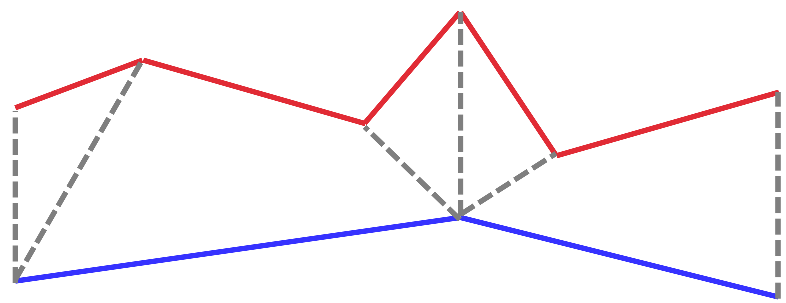
Open Problem: prove conditional lower bound $\Omega(n^{k-\varepsilon})$ for strings over alphabet size $O(1)$, or even 2

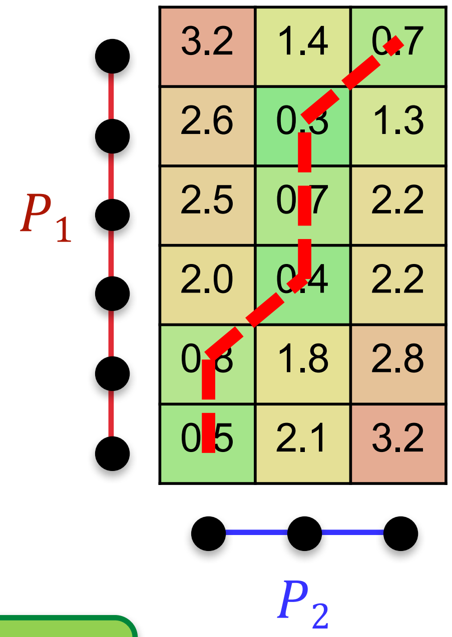
Open Problem: which log-factor improvements are possible?

Dynamic Time Warping

same setting as for Frechet distance: DTW is a similarity measure

for curves P_1, P_2 = sequences over $\mathbb{R}^2$





natural dynamic programming algorithm: $O(n^2)$

$$T[i,j] = \mathrm{DTW}(P_1[1..i], P_2[1..j])$$

$$T[i,j] = \|P_1[i] - P_2[j]\| + \min\{T[i-1,j], T[i,j-1], T[i-1,j-1]\}$$

= current distance

last step in P_1

last step in P_2

last step in both

Dynamic Time Warping

same setting as for Frechet distance: DTW is a similarity measure

for curves P_1, P_2 = sequences over $\mathbb{R}^2$

natural dynamic programming algorithm: $O(n^2)$

OV-hardness: lower bound $\Omega(n^{2-\varepsilon})$ [B.,Künnemann'15+, Abboud,Backurs,V-Williams'15]

slight improvement: $O(n^2 \log\log\log n \,/\, \log\log n)$ [Gold,Sharir'16]

Open Problem: log-factor improvement $n^2/(\log n)^{\Omega(1)}$?

Open Problem: $n^{O(1)}$ –approximation in time $O(n^{2-\varepsilon})$?

3SUM

given sets A, B, C of n integers

are there $a \in A, b \in B, c \in C$ such that $a + b + c = 0$?

log-factor improvement: $O(n^2 \cdot \frac{(\log \log n)^2}{\log n})$ [Gronlund,Pettie'14]

we showed a simplified version: $O(n^2 \cdot \frac{\text{poly} \log \log n}{\sqrt{\log n}})$

Open Problem: $n^2 / 2^{\Omega(\sqrt{\log n})}$ **algorithm?**

Dynamic Single Source Reachability

we have seen:

under OMv, no fully dynamic SSR with update $O(n^{1-\varepsilon})$, query $O(n^{2-\varepsilon})$

incremental SSR in total update $O(m)$, query $O(1)$

decremental SSR in DAGs in total update $O(m)$, query $O(1)$

decremental SSR in total update $O(mn)$, query $O(1)$

fastest known:

decremental SSR in total update $\tilde{O}(m\sqrt{n})$, query $\tilde{O}(1)$

Open Problem: faster decremental SSR? or lower bound?

IV. Outro

Oral Exam

on a day in September

(up to) 30 minutes

covers whole lecture and all exercises

please mark possible dates for you in this doodle:

http://doodle.com/poll/v9bktxdrktv5w98e

End of Course

“Fast Matrix Multiplication and Fast Fourier Transform are the only non-trivial algorithmic tools that we have.”

“Whenever you design an algorithm, try to prove a matching conditional lower bound to show optimality.”