max planck institut
informatik

Complexity Theory of Polynomial-Time Problems

Lecture 2: SETH and OV

Karl Bringmann

Tutorial Slot

Tuesday, 16:15 - 18:00

works for everybody?

alternative time slot:

	Mo	Tue	Wed	Thu	Fri
10-12	(red)				
12-14		(red)			new
14-16				(red)	
16-18		current	(red)	(red)	

I. SETH

Satisfiability Problem

$$(x_1 \vee \neg x_2 \vee x_4) \wedge (x_3 \vee \neg x_3) \wedge$$
$$(\neg x_1 \vee x_2 \vee x_3 \vee x_4)$$

CNF-SAT: boolean *variables* $x_1, \ldots, x_N$

clauses $C_1, \ldots, C_M$ are an OR over *literals*

= variable or negated variable

decide whether an assignment of $x_1, \ldots, x_N$ *satisfies* ALL clauses

unbounded *clause width*

= number of literals per clause

k-SAT: clause width bounded by k

thus $M \leq N^k$

Satisfiability Hypotheses

$\mathrm{P} \neq \mathrm{NP}$: **k-SAT** not in time $\mathrm{poly}(N)$ $\forall k \geq 3$ or $\exists k \geq 3$

$\Uparrow$

ETH (Exponential Time Hypothesis) [Impagliazzo,Paturi,Zane'01]

k-SAT not in time $2^{o(N)}$ $\forall k \geq 3$ or $\exists k \geq 3$

$\Uparrow$

SETH (Strong Exponential Time Hypothesis)

$\forall \varepsilon > 0$: $\exists k \geq 3$: **k-SAT** not in time $O(2^{(1-\varepsilon)N})$

best-known algorithm for k-SAT: $O(2^{(1-c_k)n})$ where $c_k = \Theta(1/k)$

[Paturi,Pudlak,Saks,Zane'98]

Satisfiability Hypotheses

$P \neq NP$: **k-SAT** not in time $\text{poly}(N)$ $\quad \forall k \geq 3$ or $\exists k \geq 3$

$\Uparrow$

ETH (Exponential Time Hypothesis) [Impagliazzo,Paturi,Zane'01]

k-SAT not in time $2^{o(N)}$ $\quad \forall k \geq 3$ or $\exists k \geq 3$

$\Uparrow$

SETH (Strong Exponential Time Hypothesis)

$\forall \varepsilon > 0$: $\exists k \geq 3$: **k-SAT** not in time $O(2^{(1-\varepsilon)N})$

$\Downarrow$

"CNF-SETH"

CNF-SAT not in time $O(\text{poly}(M)\ 2^{(1-\varepsilon)N})$

best-known algorithm for CNF-SAT: [Calabro,Impagliazzo,Paturi'06]

$O(2^{(1-x)N})$ where $x = \Theta(1/\log(M/N))$

Satisfiability Hypotheses

P $\neq$ NP: **k-SAT** not in time $\mathrm{poly}(N)$ $\forall k \geq 3$ or $\exists k \geq 3$

$\Uparrow$

ETH (Exponential Time Hypothesis) [Impagliazzo,Paturi,Zane'01]

k-SAT not in time $2^{o(N)}$ $\forall k \geq 3$ or $\exists k \geq 3$

$\Uparrow$

SETH (Strong Exponential Time Hypothesis)

$\forall \varepsilon > 0$: $\exists k \geq 3$: **k-SAT** not in time $O(2^{(1-\varepsilon)N})$

$\Downarrow$

"CNF-SETH"

CNF-SAT not in time $O(\mathrm{poly}(M)\, 2^{(1-\varepsilon)N})$

$\Downarrow$

OV-Hypothesis

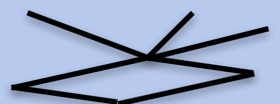
OV not in time $O(\mathrm{poly}(d)\, n^{2-\varepsilon})$

Reminder: Definition of Reductions

transfer hardness of one problem to another one by reductions

problem P

instance I



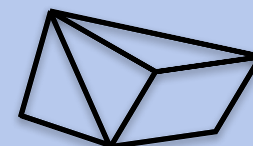
size n

reduction

$r(n)$ time

problem Q

instance J



size $s(n)$

I is a 'yes'-instance $\iff$ J is a 'yes'-instance

$t(n)$ algorithm for Q implies a $r(n) + t(s(n))$ algorithm for P

if P has no $r(n) + t(s(n))$ algorithm then Q has no $t(n)$ algorithm

SETH-Hardness for OV

CNF-SAT

N variables

M clauses

reduction

time $O(M 2^{N/2})$

OV

sets A, B

of size $n = 2^{N/2}$

in dimension $d = M$

$O(2^{(1-\varepsilon/2)N} \text{poly}(M))$ algorithm $\Longleftarrow$ $O(n^{2-\varepsilon} \text{poly}(d))$ algorithm

Thm: SETH implies OVH [Williams‘05]

$O(2^{(1-1/O(\log(M/N)))N})$ algorithm $\Longleftarrow$ $O(n^{2-1/O(\log(d/\log n))})$ algorithm

best-known algorithm for CNF-SAT! [Lecture 3]

SETH-Hardness for OV

CNF-SAT

N variables

M clauses

reduction →

time $O(M 2^{N/2})$

OV

sets A, B

of size $n = 2^{N/2}$

in dimension $d = M$

Proof:

$U :=$ assignments of $x_1, \dots, x_{N/2}$
$\cong \{1, \dots, n\}$

$V :=$ assignments of $x_{N/2+1}, \dots, x_N$
$\cong \{1, \dots, n\}$

we say that *partial assignment* u *satisfies clause* C

iff $\exists i$: x_i is set to **true** in u and x_i appears **unnegated** in C

or $\exists i$: x_i is set to **false** in u and x_i appears **negated** in C

in this case we write: $sat(u, C) = 1$ otherwise: $sat(u, C) = 0$

$unsat(u, C)$
$:= 1 - sat(u, C)$

$A = \{(unsat(u, C_1), \dots, unsat(u, C_M)) \mid u \in U\}$

$B = \{(unsat(v, C_1), \dots, unsat(v, C_M)) \mid v \in V\}$

max planck institut
informatik

SETH-Hardness for OV

CNF-SAT

N variables

M clauses

reduction →

time $O(M 2^{N/2})$

OV

sets A, B

of size $n = 2^{N/2}$

in dimension $d = M$

Proof:

U := assignments of $x_1, \dots, x_{N/2}$

$\cong$ ...

V := assignments of $x_{N/2+1}, \dots, x_N$

...

we say ...

iff $\exists$ ...

or $\exists$ ...

what if we split into k parts?

U_i := assignments of $x_{(i-1)N/k+1}, \dots, x_{iN/k}$

$A_i := \{(unsat(u, C_1), \dots, unsat(u, C_M)) \mid u \in U_i\}$

in this case we write: $sat(u,C) = 1$ otherwise: $sat(u,C) = 0$

$unsat(u, C) := 1 - sat(u, C)$

$A = \{(unsat(u, C_1), \dots, unsat(u, C_M)) \mid u \in U\}$

$B = \{(unsat(v, C_1), \dots, unsat(v, C_M)) \mid v \in V\}$

SETH-Hardness for k-OV

CNF-SAT

N variables

M clauses

reduction

time $O(M2^{N/k})$

k-OV

sets $A_1, \dots, A_k$

of size $n = 2^{N/k}$

in dimension $d = M$

k-OrthogonalVectors:

Input: Sets $A_1, \dots, A_k \subseteq \{0,1\}^d$ of size n

Task: Decide whether there are $a^{(1)} \in A_1, \dots, a^{(k)} \in A_k$

such that $\forall 1 \le i \le d$: $\prod_{j=1}^{k} a^{(j)}{}_i = 0$

$\Leftrightarrow \forall 1 \le i \le d$: $\exists j$: $a^{(j)}{}_i = 0$

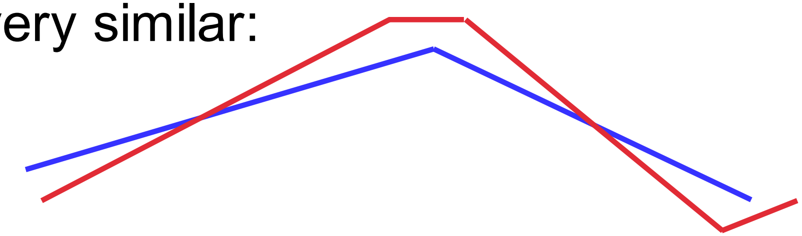
Thm: k-OV has no $O(n^{k-\varepsilon})$ algorithm unless SETH fails. [Williams,Patrascu'10]

II. Fréchet Distance

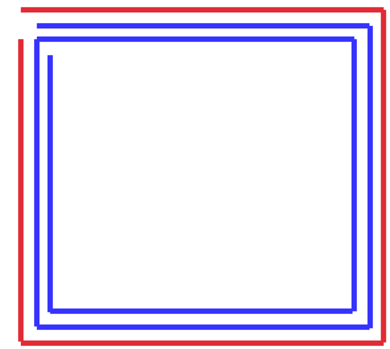
Curve Similarity

Given two polygonal curves, how similar are they?

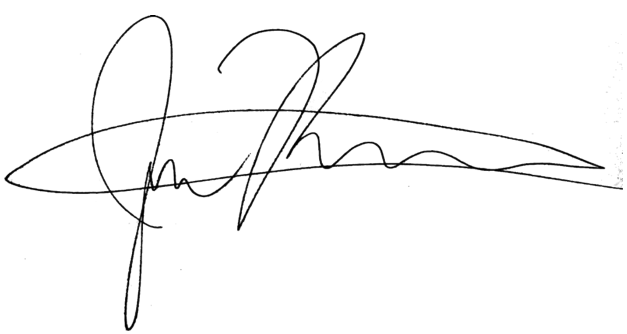
✔ very similar:

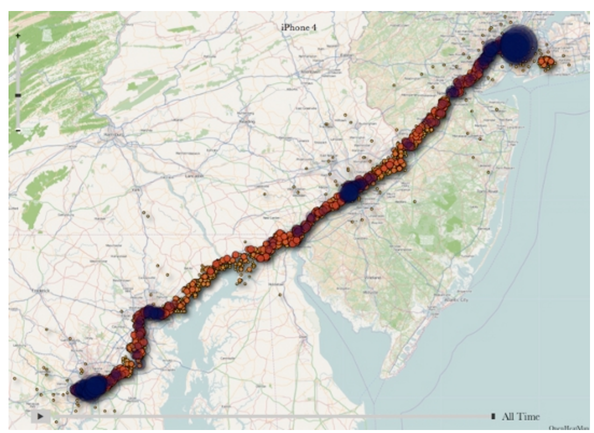


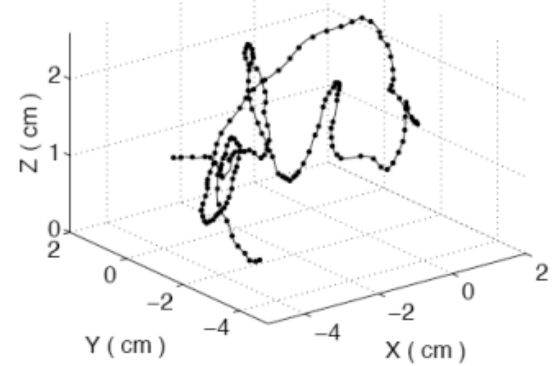
✗ less similar:



Applications in: signature recognition, analysis of moving objects







max planck institut informatik

Discrete Fréchet Distance

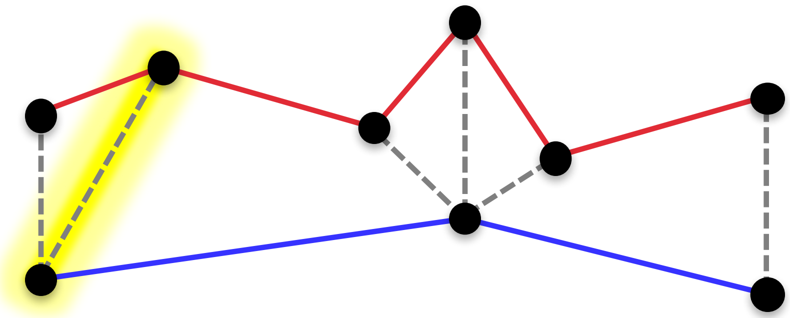
natural measure for curve similarity

rich field of research: many extensions and applications

Discrete Fréchet Distance

natural measure for curve similarity

rich field of research: many extensions and applications



man and dog walk along two curves

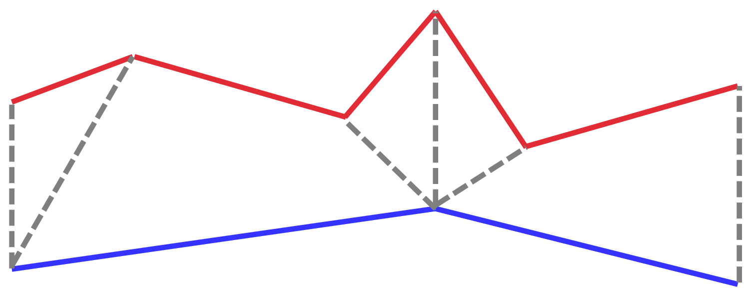
only allowed to go forward

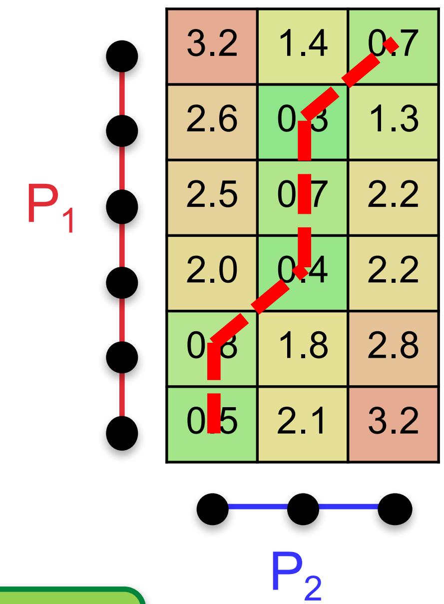
in every time step: advance in one or both curves to the next vertex

what is the minimum possible length of the **leash**?

$$d_{\mathrm{dF}}({\color{red}P_1}, {\color{blue}P_2}) = \min_{\substack{\text{all ways of} \\ \text{traversing} \\ {\color{red}P_1} \text{ and } {\color{blue}P_2}}} \; \max_{\text{time step } t} \; \text{distance at time } t$$

Dynamic Program and Known Results





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

$$T[i,j] = d_{Fr}(P_1[1..i], P_2[1..j])$$

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

= current distance

last step in P_1

last step in P_2

last step in both

logfactor improvement: $O(n^2 \frac{\log\log n}{\log n})$

[Agarwal,Avraham,Kaplan,Sharir'13]

OV-Hardness Result

OV

sets $A, B \subseteq \{0,1\}^d$

of size n

reduction

time $O(dn)$

Fréchet distance

curves P_1, P_2

$O(dn)$ vertices

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm $\Leftarrow$ $O(n^{2-\varepsilon})$ algorithm

Thm: Fréchet distance has no $O(n^{2-\varepsilon})$ algorithm unless the OV-Hypothesis fails. [B.'14]

Proof: Vector Gadgets

OV

n vectors, dimension d

reduction →

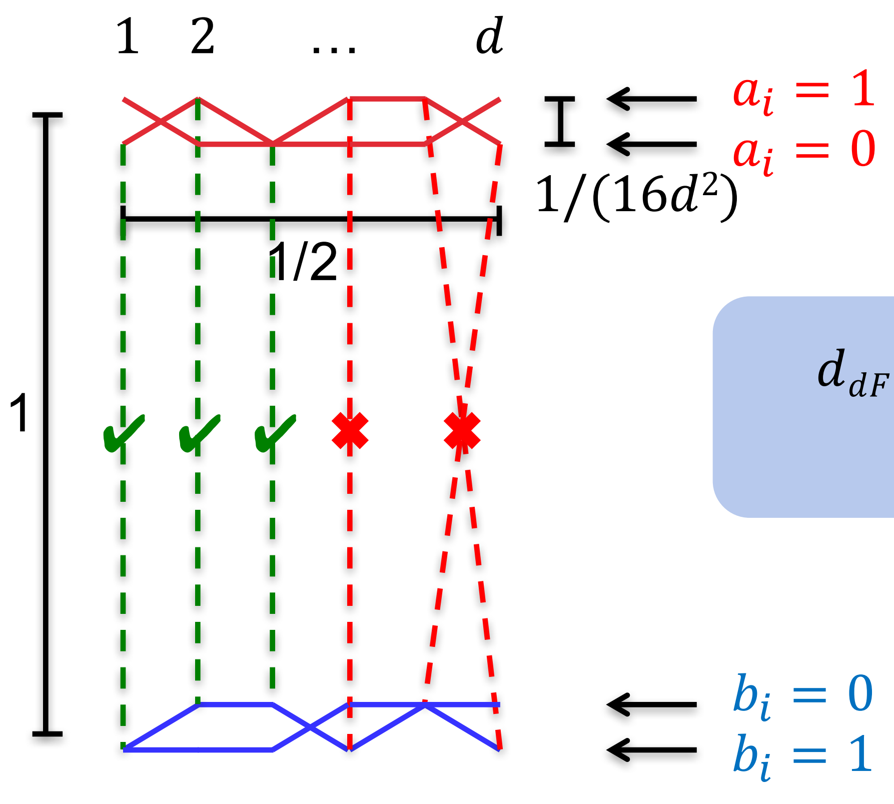
time $O(dn)$

Fréchet distance

$O(dn)$ vertices

for $a \in A$:

$\mathrm{VG}(a)$



for $b \in B$:

$\mathrm{VG}(b)$

$d_{dF}(\mathrm{VG}(a), \mathrm{VG}(b)) \leq 1$
iff $a \perp b$

cross distances:

$$\sqrt{(1 - 2/16d^2)^2 + (1/2d)^2}$$

$$= \sqrt{1 + (2/16d^2)^2} > 1$$

for $a, a' \in A$: $\mathrm{VG}(a)$ and $\mathrm{VG}(a')$ are "on top of each other"

Proof: OR-Gadget

OV

n vectors, dimension d

reduction → time $O(dn)$

Fréchet distance

$O(dn)$ vertices

we add some control points s.t.:

Fréchet distance ≤ 1 iff $\exists a, b: a \perp b$

final curves:

$P_1 = s_1 - r_1 - \mathrm{VG}(A[1]) - t_1 - \ldots$

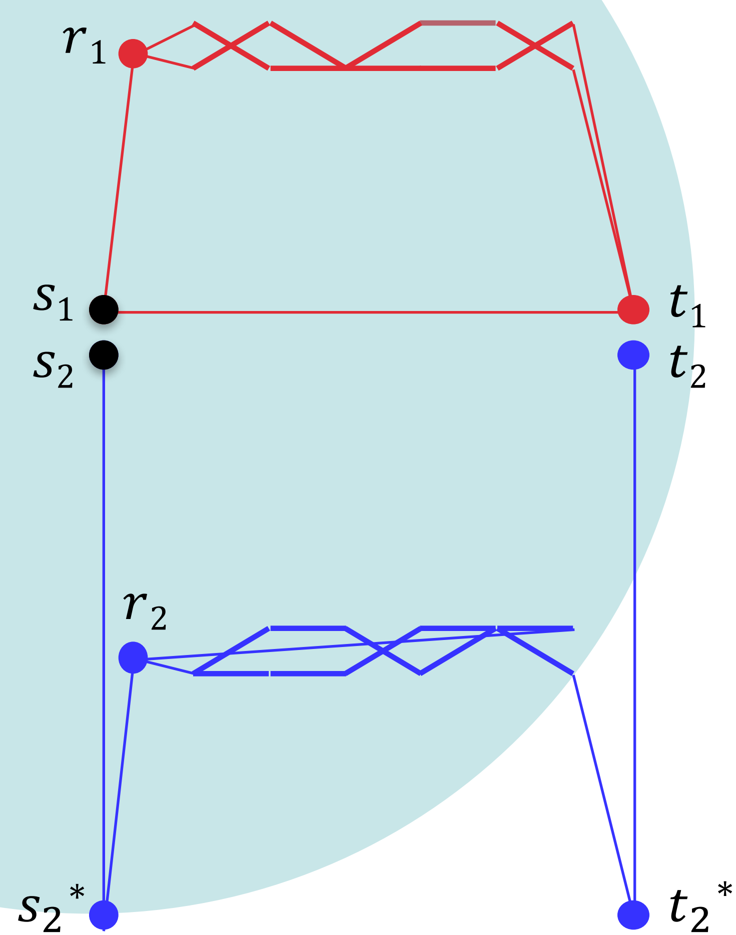
$\quad - s_1 - r_1 - \mathrm{VG}(A[n]) - t_1$

$P_2 = s_2 - s_2{}^*$

$\quad - r_2 - \mathrm{VG}(B[1]) - \ldots$

$\quad - r_2 - \mathrm{VG}(B[n])$

$\quad - t_2{}^* - t_2$



Proof: Correctness

OV

n vectors, dimension d

reduction → time $O(dn)$

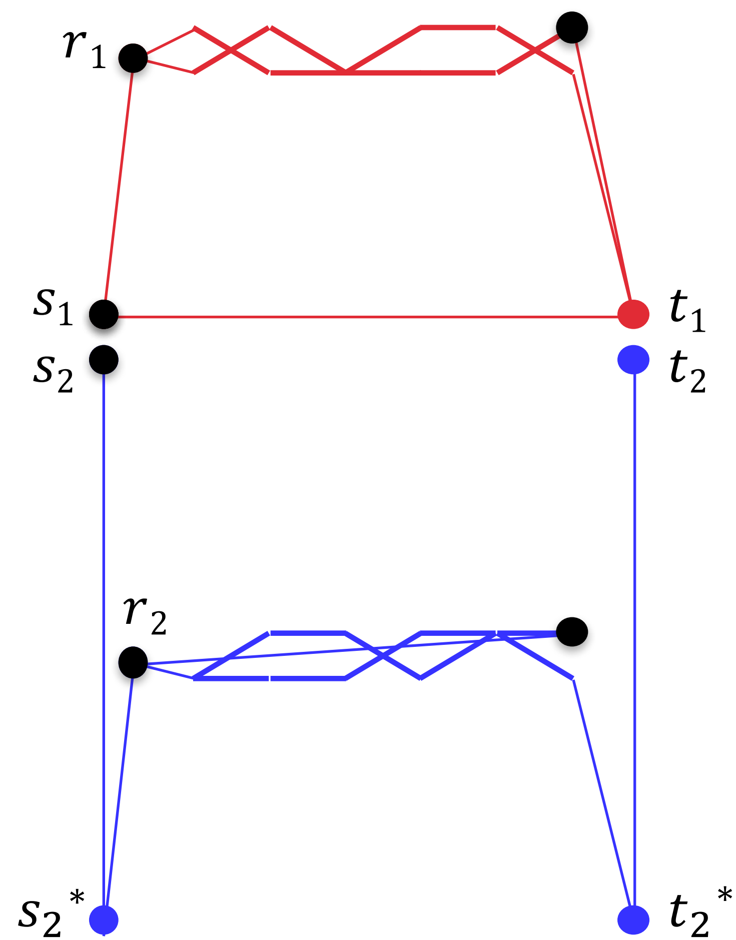
Fréchet distance

$O(dn)$ vertices

Fréchet distance ≤ 1 iff $\exists a, b: a \perp b$

"$\Leftarrow$": let $a \in A, b \in B$ with $a \perp b$

- stay at s_2 and walk to the a_1-copy of s_1
- stay at s_1 and walk to the a_2-copy of r_2
- step to r_1
- walk through $VG(a_1), VG(a_2)$ in parallel
- step to t_1
- stay at t_1 and walk to t_2
- P_2 is completely traversed, now finish traversing P_1



max planck institut
informatik

Proof: Correctness

OV

n vectors, dimension d

reduction → time $O(dn)$

Fréchet distance

$O(dn)$ vertices

Fréchet distance ≤ 1 iff $\exists a, b: a \perp b$

"$\Rightarrow$" : consider a traversal staying in distance 1

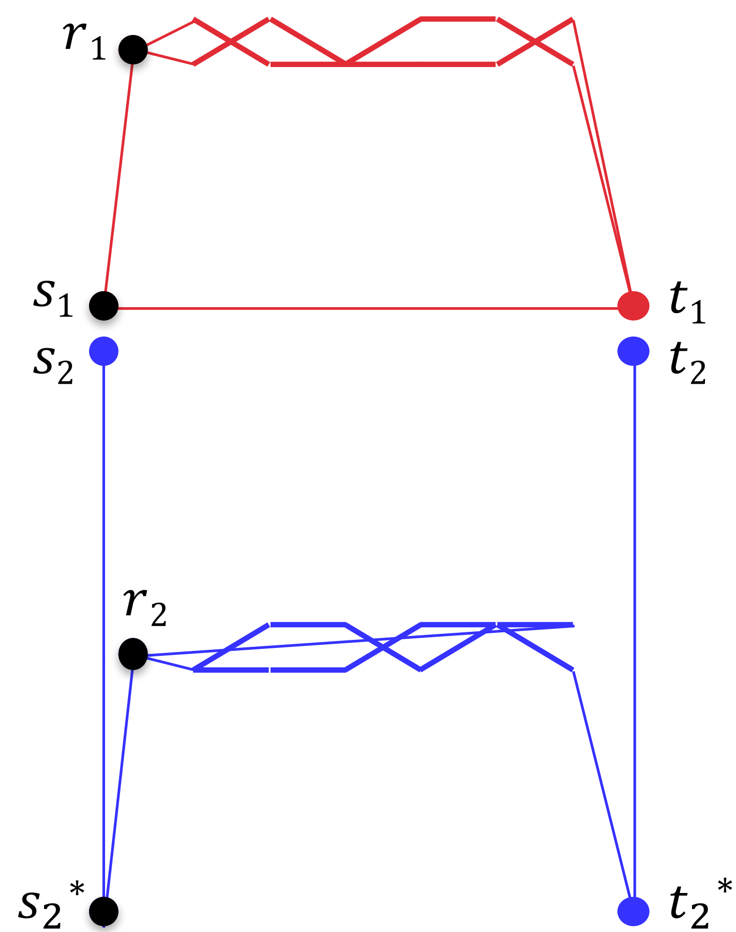
when at $s_2{}^*$: have to be at s_1 (say at a-copy)

after that, at first time we are at r_1:

- could be at s_2, t_2, or r_2
- s_2: already passed
- t_2: not reachable because of $t_2{}^*$
- so have to be at r_2 (say at b-copy)

in the following we have to traverse $\mathrm{VG}(a), \mathrm{VG}(b)$ in parallel

this is only possible if $a \perp b$



OV-Hardness Result

OV

sets $A, B \subseteq \{0,1\}^d$

of size n

reduction

time $O(dn)$

Fréchet distance

curves P_1, P_2

$O(dn)$ vertices

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm $\Longleftarrow$ $O(n^{2-\varepsilon})$ algorithm

Thm: Fréchet distance has no $O(n^{2-\varepsilon})$ algorithm unless the OV-Hypothesis fails. [B.'14]

Inapproximability

Thm: Fréchet distance has no 1.001-approximation in time $O(n^{2-\varepsilon})$ unless the OV-Hypothesis fails. [B.'14]

different construction yields 1.399-inapproximability [B.,Mulzer'15]

Q: improve constant

Thm: Fréchet distance has has an α-approximation in time $O(n^2/\alpha + n \log n)$ [B.,Mulzer'15]

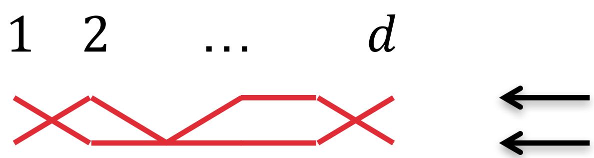
Q: close this gap

Inapproximability

Thm: Fréchet distance has no 1.001-approximation in time $O(n^{2-\varepsilon})$ unless the OV-Hypothesis fails. [B.'14]

Proof Idea:

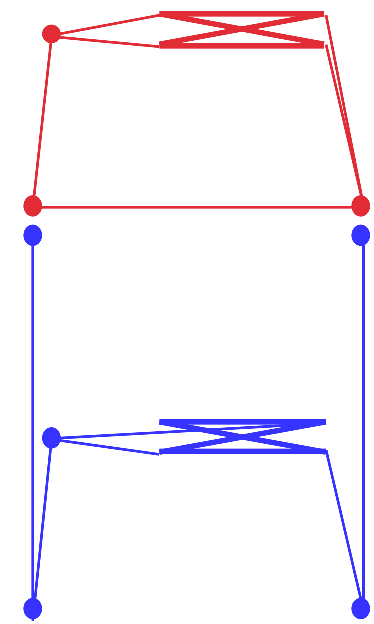
for $a \in A$:
$\mathrm{VG}(a)$



$a_i = 1$
$a_i = 0$

replace by:

we still have to walk in parallel
through vector gadgets!



Inapproximability

Thm: Fréchet distance has no 1.001-approximation in time $O(n^{2-\varepsilon})$ unless the OV-Hypothesis fails. [B.'14]

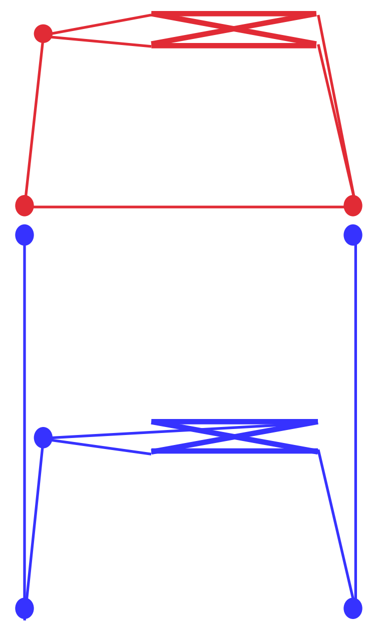
Proof Idea:

construction has a **fixed (constant) set of points**

minimal distance between any pair of points in distance > 1 in P_1 and P_2 is $\boldsymbol{C > 1.001}$

if $d_{dF}(P_1, P_2) > 1$ then $d_{dF}(P_1, P_2) > 1.001$

thus any 1.001-approximation of the Fréchet distance can decide OV



Generalizations

Fréchet distance has no $O(n^{2-\varepsilon})$ algorithm unless OVH fails even on **one-dimensional** curves [B.,Mulzer'15]

A generalization to ***k* curves** has no $O(n^{k-\varepsilon})$ algorithm unless OVH fails (for curves in the plane)

[Buchin,Buchin,Konzack,Mulzer,Schulz'16]

Q: $\Omega(n^{k-\varepsilon})$ lower bound for k one-dimensional curves

continuous Fréchet distance:

- **Q:** $\Omega(n^{2-\varepsilon})$ lower bound for one-dimensional curves
- **Q:** $\Omega(n^{k-\varepsilon})$ lower bound for k curves

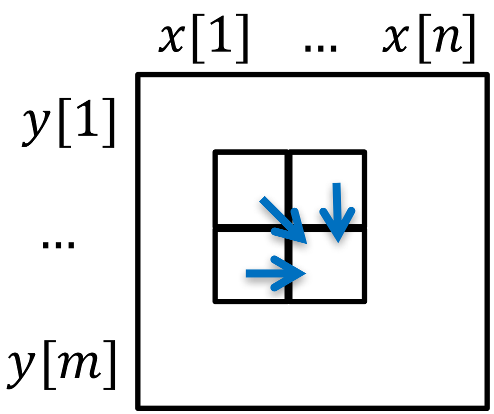
III. Longest Common Subsequence

Longest Common Subsequence (LCS)

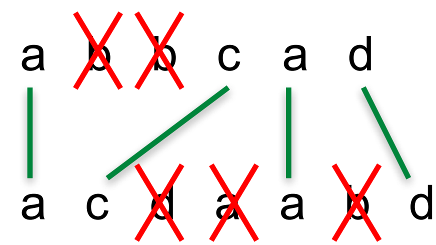
given strings x, y of length $n \geq m$, compute longest string z that is a subsequence of both x and y

write $LCS(x, y) = |z|$

natural dynamic program $O(n^2)$



$T[i, j] = LCS(x[1..i], y[1..j])$



delete in x

delete in y

$T[i, j] = \max\{T[i-1, j], T[i, j-1]\}$

if $x[i] = y[j]$:

$T[i, j] = \max\{T[i, j], T[i-1, j-1] + 1\}$

match

logfactor improvement:

$O(n^2 / \log^2 n)$

[Masek,Paterson'80]

OV-Hardness Result

OV

sets $A, B \subseteq \{0,1\}^d$

of size n

reduction

→

time $O(d^2 n)$

LCS

strings x, y

of length $O(d^2 n)$

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm $\Leftarrow$ $O(n^{2-\varepsilon})$ algorithm

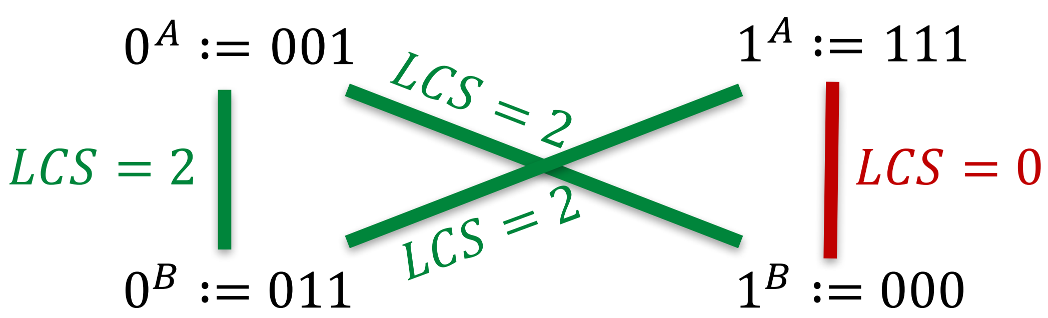
Thm: Longest Common Subsequence has no $O(n^{2-\varepsilon})$ algorithm unless the OV-Hypothesis fails.

[B.,Künnemann’15+ Abboud,Backurs,V-Williams‘15]

Proof: Coordinate Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

we want to simulate the **coordinates** $\{0,1\}$ and the behavior of $a_i \cdot b_i$



replace a_i by ${a_i}^A$ and b_i by ${b_i}^B$

$LCS({a_i}^A, {b_i}^B)$ can be written as $f(a_i \cdot b_i)$, with $f(0) > f(1)$

Proof: Vector Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

we want to simulate **orthogonality** of $a \in A, b \in B$ in the picture: $d = 4$

concatenate ${a_1}^A, \dots, {a_d}^A$, padded with a new symbol 2

length $4d$

$$VG(a) := {a_1}^A\, 2 \dots 2\, {a_2}^A\, 2 \dots 2\, {a_3}^A\, 2 \dots 2\, {a_4}^A$$

$$VG(b) := {b_1}^B\, 2 \dots 2\, {b_2}^B\, 2 \dots 2\, {b_3}^B\, 2 \dots 2\, {b_4}^B$$

- no LCS matches symbols in ${a_i}^A$ with symbols in ${b_j}^B$ where $i \neq j$

Proof: Vector Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i\colon\ a_i \cdot b_i = 0$

we want to simulate **orthogonality** of $a \in A, b \in B$

concatenate $a_1{}^A, \dots, a_d{}^A$, padded with a new symbol 2

length $4d$

$$VG(a) := a_1{}^A\ 2 \dots 2\ a_2{}^A\ 2 \dots 2\ a_3{}^A\ 2 \dots 2\ a_4{}^A$$

$$VG(b) := b_1{}^B\ 2 \dots 2\ b_2{}^B\ 2 \dots 2\ b_3{}^B\ 2 \dots 2\ b_4{}^B$$

- no LCS matches symbols in $a_i{}^A$ with symbols in $b_j{}^B$ where $i \neq j$

  assume otherwise

  then we could match $\leq (d-2)4d$ symbols 2 and $\leq 3d$ symbols $0/1$

  but $LCS\big(VG(a), VG(b)\big) \geq (d-1)4d > (d-2)4d + 3d$

Proof: Vector Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i\colon\ a_i \cdot b_i = 0$

we want to simulate **orthogonality** of $a \in A, b \in B$

concatenate ${a_1}^A, \dots, {a_d}^A$, padded with a new symbol 2

$$VG(a) := {a_1}^A\, 2 \dots 2\, {a_2}^A\, 2 \dots 2\, {a_3}^A\, 2 \dots 2\, {a_4}^A$$

$$VG(b) := {b_1}^B\, 2 \dots 2\, {b_2}^B\, 2 \dots 2\, {b_3}^B\, 2 \dots 2\, {b_4}^B$$

- no LCS matches symbols in ${a_i}^A$ with symbols in ${b_j}^B$ where $i \neq j$
- some LCS matches all 2's

Proof: Vector Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i: \; a_i \cdot b_i = 0$

we want to simulate **orthogonality** of $a \in A, b \in B$

concatenate $a_1{}^A, \dots, a_d{}^A$, padded with a new symbol 2

$$VG(a) := a_1{}^A \, 2 \dots 2 \, a_2{}^A \, 2 \dots 2 \, a_3{}^A \, 2 \dots 2 \, a_4{}^A$$

$$VG(b) := b_1{}^B \, 2 \dots 2 \, b_2{}^B \, 2 \dots 2 \, b_3{}^B \, 2 \dots 2 \, b_4{}^B$$

- $LCS\big(VG(a), VG(b)\big) = (d-1)4d + \sum_{i=1}^{d} LCS(a_i{}^A, b_i{}^B)$

#2's

$= f(a_i \cdot b_i)$

$LCS\big(VG(a), VG(b)\big) = C + 2$ if $a \perp b$

$LCS\big(VG(a), VG(b)\big) \leq C$ otherwise

where $C = (d-1)4d + 2d - 2$

Proof: Normalized Vectors Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

add a $(d+1)$-st coordinate:

$a_{d+1} := 0$

$b_{d+1} := 1$

this does not change $a \perp b$

still holds: $\exists C$:

$LCS\big(VG(a), VG(b)\big) = C + 2$ if $a \perp b$

$LCS\big(VG(a), VG(b)\big) \le C$ otherwise

define vector:

$s := (0, \dots, 0, 1) \in \{0,1\}^{d+1}$

$LCS\big(VG(s), VG(b)\big) = C$

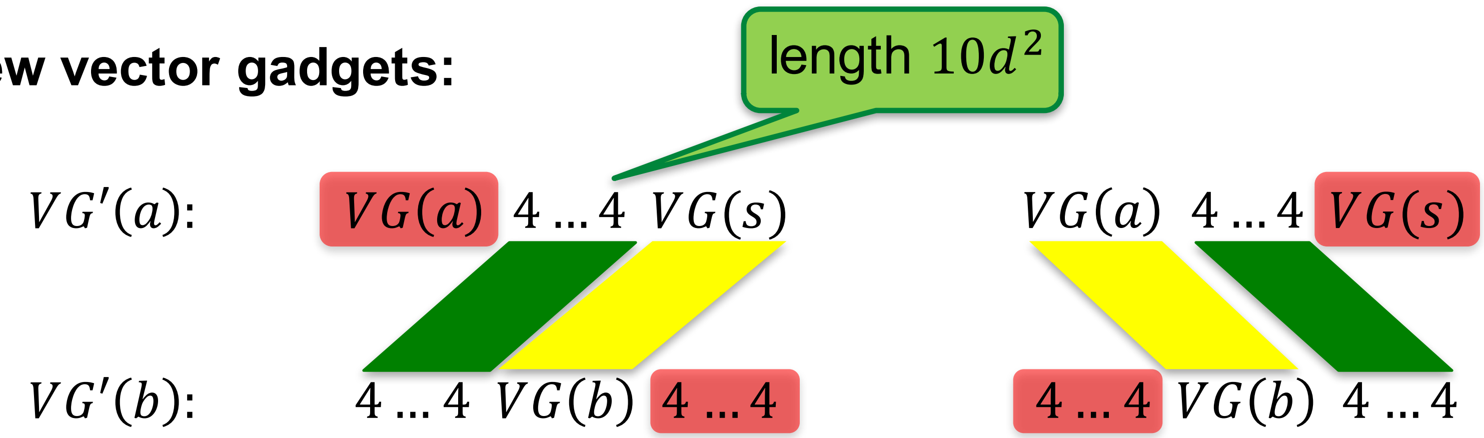
aim for $\max\{LCS\big(VG(a), VG(b)\big), LCS\big(VG(s), VG(b)\big)\}$

this takes only 2 values, depending on whether $a \perp b$

Proof: Normalized Vectors Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i: \; a_i \cdot b_i = 0$

new vector gadgets:



$$LCS(VG'(a), VG'(b)) = 10d^2 + \max\{LCS(VG(a), VG(b)), LCS(VG(s), VG(b))\}$$

$$LCS(VG'(a), VG'(b)) = \begin{cases} C' + 2 & \text{if } a \perp b \\ C' & \text{otherwise} \end{cases}$$

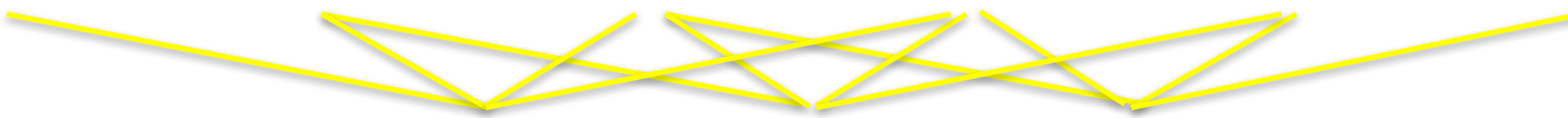
write VG for VG'

Proof: OR-Gadget

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

fresh symbol 3, want to construct: in the picture: $n = 3$

$$VG(A[1])\ 3 \dots 3\ VG(A[2])\ 3 \dots 3\ VG(A[3])\ 3 \dots 3\ VG(A[1])\ 3 \dots 3\ VG(A[2])\ 3 \dots 3\ VG(A[3])$$



$$3 \dots \dots \dots \dots \dots 3\ VG(B[1])\ 3 \dots 3\ VG(B[2])\ 3 \dots 3\ VG(B[3])\ 3 \dots \dots \dots \dots \dots 3$$

length $100d^2$

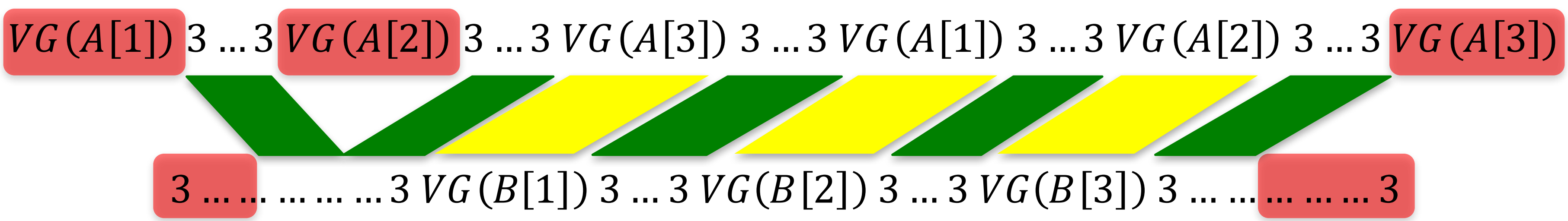
length $100d^2 \cdot 2n$

Proof: OR-Gadget

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

fresh symbol 3, want to construct:

in the picture: $n = 3$



can align $VG(B[j])$ with $VG(A[\Delta + j \bmod n])$ for any offset Δ

$$LCS \geq \underbrace{(2n-1)100d^2}_{\text{\#3's in upper string}} + \max_{\Delta} \sum_{j=1}^{n} LCS(VG(A[\Delta + j \bmod n]), VG(B[j]))$$

#3's in upper string

maximize over offset

need normalization!

If there is an orthogonal pair, some offset Δ aligns this pair, and we get

$$LCS \geq (2n-1)100d^2 + nC + 2$$

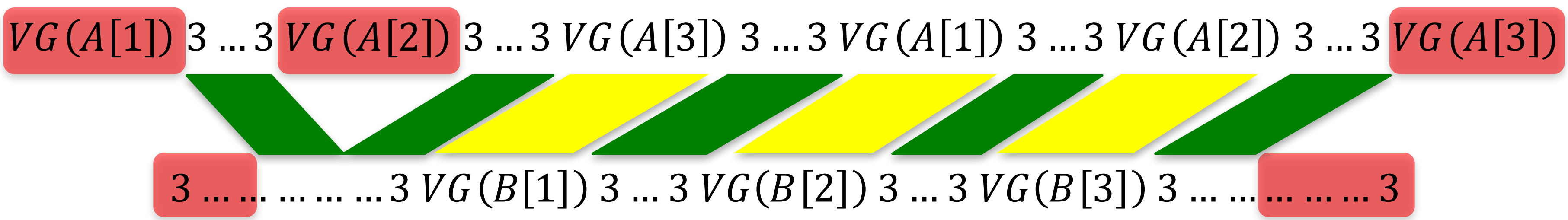
max planck institut informatik

Proof: OR-Gadget

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

fresh symbol 3, want to construct:

in the picture: $n = 3$



if an orthogonal pair exists then $LCS \geq (2n-1)100d^2 + nC + 2$

Claim: otherwise: $LCS \leq (2n-1)100d^2 + nC$

this finishes the proof:

- ✔ equivalent to OV instance
- ✔ length $O(d^2 n)$

OV-Hardness Result

OV

sets $A, B \subseteq \{0,1\}^d$

of size n

reduction

$\longrightarrow$

time $O(d^2 n)$

LCS

strings x, y

of length $O(d^2 n)$

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm $\Leftarrow$ $O(n^{2-\varepsilon})$ algorithm

Thm: Longest Common Subsequence has no $O(n^{2-\varepsilon})$ algorithm unless the OV-Hypothesis fails.

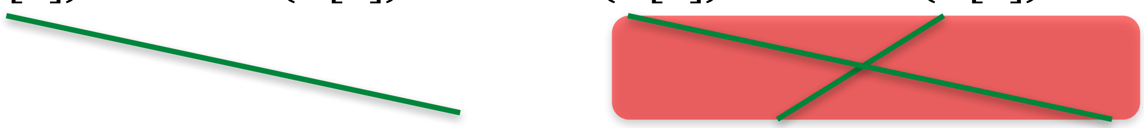
[B.,Künnemann'15+ Abboud,Backurs,V-Williams'15]

Proof of Claim

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

Claim: if no orthogonal pair exists: $LCS \leq (2n-1)100d^2 + nC$

$$VG(A[1])\ 3 \ldots 3\ VG(A[2])\ 3 \ldots 3\ VG(A[3])\ 3 \ldots 3\ VG(A[1])\ 3 \ldots 3\ VG(A[2])\ 3 \ldots 3\ VG(A[3])$$



$$3 \ldots \ldots \ldots \ldots \ldots 3\ VG(B[1])\ 3 \ldots 3\ VG(B[2])\ 3 \ldots 3\ VG(B[3])\ 3 \ldots \ldots \ldots \ldots \ldots 3$$

consider how an LCS matches the $VG(B[j])$

- no crossings

Proof of Claim

OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i$: $a_i \cdot b_i = 0$

Claim: if no orthogonal pair exists: $LCS \leq (2n-1)100d^2 + nC$

$VG(A[1])\ 3 \ldots 3\ VG(A[2])\ 3 \ldots 3\ VG(A[3])\ 3 \ldots 3\ VG(A[1])\ 3 \ldots 3\ VG(A[2])\ 3 \ldots 3\ VG(A[3])$

$3 \ldots \ldots \ldots \ldots \ldots 3\ VG(B[1])\ 3 \ldots 3\ VG(B[2])\ 3 \ldots 3\ VG(B[3])\ 3 \ldots \ldots \ldots \ldots \ldots 3$

$$LCS \leq (2n-1)100d^2 + \sum_{i=1}^{n} \begin{cases} 0 & \text{if } VG(B[j]) \text{ is not matched} \\ C & \text{if } VG(B[j]) \text{ is matched to one} \\ |VG(B[j])| - |3 \ldots 3| & \text{if } VG(B[j]) \text{ is matched to} > 1 \end{cases}$$

non-orthogonal

#3's in upper string

could match VG completely, but loose many 3's

≤ 0

Extensions

similar problems:

edit distance

dynamic time warping

…

alphabet size:

longest common subsequence and edit distance
are even hard on *binary* strings, i.e., alphabet $\{0,1\}$

longest common subsequence of $\boldsymbol{k}$ **strings** takes time $\Omega(n^{k-\varepsilon})$

Summary

reduction SETH → OV

introduced k-OV

OV-hardness for Fréchet distance

OV-hardness for longest common subsequence