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* $x_1, \dots, x_N$
clauses $C_1, \dots, C_M$ are an OR over *literals*
decide whether an assignment of $x_1, \dots, x_N$ *satisfies* ALL clauses
unbounded *clause width*

= variable or negated variable

= number of literals per clause

k-SAT: clause width bounded by k
thus $M \le N^k$

max planck institut informatik

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]

max planck institut informatik

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 $O(2^{(1-\varepsilon)N})$

$\Downarrow$

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

best-known algorithm for CNF-SAT: [Calabro,Impagliazzo,Paturi'06]
$O(2^{(1-x)N})$ where $x = \Theta(1/\log(M/N))$

max planck institut informatik

Satisfiability Hypotheses

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

$\Updownarrow$

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)

$\forall \varepsilon > 0 \colon \exists k \geq 3$: **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

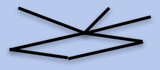
OV not in time $O(\mathrm{poly}(d)\ n^{2-\varepsilon})$

Reminder: Definition of Reductions

transfer hardness of one problem to another one by reductions

problem P

instance I



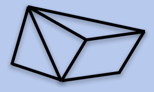
size n

reduction $\longrightarrow$

$r(n)$ 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

max planck institut informatik

SETH-Hardness for OV

CNF-SAT

N variables

M clauses

reduction $\longrightarrow$

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

OV

sets A, B

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

in dimension $d = M$

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

Thm: SETH implies OVH [Williams'05]

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

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

max planck institut informatik

SETH-Hardness for OV

CNF-SAT

N variables

M clauses

reduction $\longrightarrow$

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: $sat(u, C) = 1$ $\quad$ 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 $x_1, \ldots, x_{N/2}$ $\quad$ $V :=$ assignments of $x_{N/2+1}, \ldots, x_N$

$\cong \{1, \ldots$ $\quad$ $\cong \{1, \ldots$

we say

iff ∃

or ∃

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

what if we split into k parts?

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

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

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

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

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

SETH-Hardness for k-OV

CNF-SAT
N variables
M clauses

reduction →
time $O(M 2^{N/k})$

k-OV
sets $A_1, \ldots, A_k$
of size $n = 2^{N/k}$
in dimension $d = M$

k-OrthogonalVectors:

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

Task: Decide whether there are $a^{(1)} \in A_1, \ldots, a^{(k)} \in A_k$ such that $\forall 1 \le i \le d$: $\prod_{j=1}^{k} a^{(j)}{}_i = 0$

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

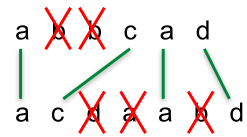
Thm: k-OV has no $O(n^{k-\varepsilon})$ algorithm unless SETH fails. [Williams,Patrascu'10]

III. Longest Common Subsequence

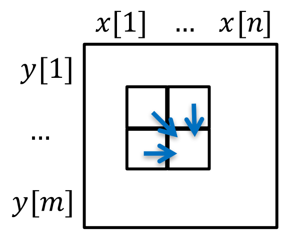
Longest Common Subsequence (LCS)

given strings x, y of length $n \ge m$, 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 $\quad$ *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 $O(d^2 n)$

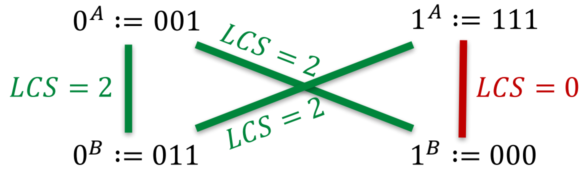
$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm $\Leftarrow$ $O(n^{2-\varepsilon})$ algorithm

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 $b_i{}^B$

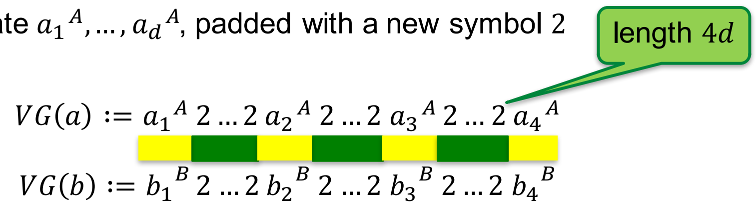
$LCS(a_i{}^A, b_i{}^B)$ can be written as $f(a_i \cdot b_i)$, with $f(0) > f(1)$

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



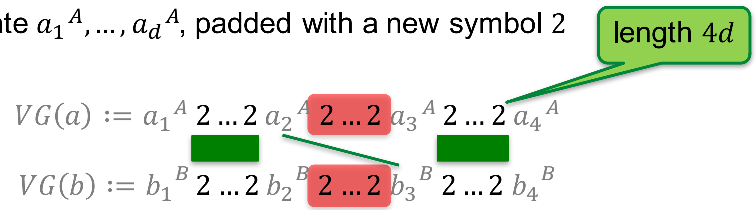
- no LCS matches symbols in $a_i{}^A$ with symbols in $b_j{}^B$ where $i \neq j$

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 $\forall i$: $a_i \cdot b_i = 0$

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

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



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

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 $\forall i: \ a_i \cdot b_i = 0$

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

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

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

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

#2's

$= f(a_i \cdot b_i)$

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

$LCS(VG(a), VG(b)) \leq C$ 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: $\exists C$:

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

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

define vector:

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

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

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

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

Proof: Normalized Vectors Gadgets

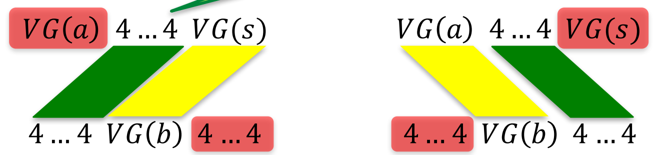
OV: Given $A, B \subseteq \{0,1\}^d$ of size n each
Are there $a \in A, b \in B$ such that $\forall i: \ a_i \cdot b_i = 0$

new vector gadgets:

length $10d^2$

$VG'(a)$: $VG(a)\ 4 \dots 4\ VG(s)$

$VG'(b)$: $4 \dots 4\ VG(b)\ 4 \dots 4$



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

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

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: $n = 3$

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

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

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: $n = 3$

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

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

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

$$LCS \geq (2n-1)100d^2 + \max_{\Delta} \sum_{j=1}^{n} LCS(VG(A[\Delta + j \bmod n]), VG(B[j]))$$

#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

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

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: $n = 3$

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

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

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 → time $O(d^2 n)$

LCS
strings x, y
of length $O(d^2 n)$

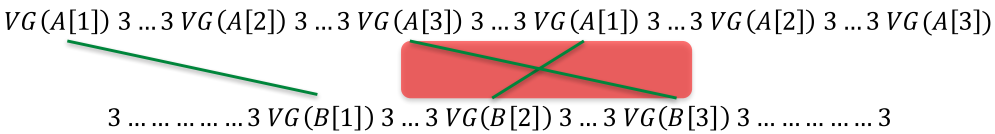
$O(n^{2-\varepsilon}\mathrm{poly}(d))$ algorithm $\Leftarrow$ $O(n^{2-\varepsilon})$ algorithm

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



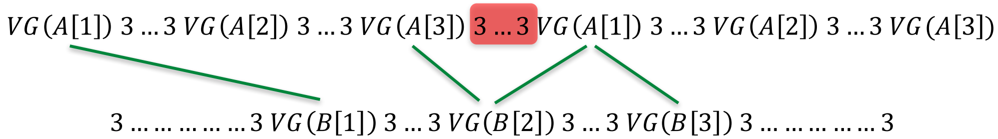
consider how an LCS matches the $VG(B[j])$

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



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

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 $\Omega(n^{k-\varepsilon})$