CONSEQUENCES OF FASTER ALIGNMENT OF SEQUENCES

UNIVERSITY OF HAIFA

AMIR ABBOUD STANFORD

VIRGINIA VASSILEVSKA WILLIAMS STANFORD

OREN WEIMANN UNIVERSITY OF HAIFA

Thursday, July 10th, 2014

FASTER ALGORITHMS?

Some classic problems on sequences have $\tilde{O}(n)$ algorithms:

- ✓ Exact Pattern Matching
- ✓ Pattern Matching with don't cares
- ✓ Longest Common Substring

While other classic problems don't have $O(n^{2-\varepsilon})$ algorithms:

- ➢ Local Alignment
- ➢ Edit Distance
- ➢ Longest Common Subsequence (LCS)

Isn't quadratic time efficient enough?

LOCAL ALIGNMENT

$O(n^2)$ is not that efficient…

Input: Two (DNA) sequences of length n.

AGCCCGTCTACGTGCAACCGGGGAAAGTATA
AAACGTGACGAGAGAGAGAACCCATTACGAA

Output: The optimal alignment of two substrings.

C C G – T C T A C G
C C C A T – T A C G

+1 +1 -0.5 -1 +1 -1 +1 +1 +1 +1 = +4.5

	A	C	G	T	–
A	+1	-1.4	-1.8	-0.7	-1
C	-1.4	+1	-0.5	-1	-1
G	-1.8	-0.5	+1	-1.9	-1
T	-0.7	-1	-1.9	+1	-1
-	-1	-1	-1	-1	-∞

Solved daily on huge sequences: $n = 3 \cdot 10^9$ for the human genome.

Algorithms:

Smith-Waterman dynamic programming $O(n^2)$.

Compression tricks $O\left(\frac{n^2}{\log n}\right)$.

LOCAL ALIGNMENT

When $n = 3 \cdot 10^9$, $O(n^2/\log n)$ is too slow!

In practice? Heuristics.

Most cited paper in the 90s:

BLAST: Basic Local Alignment Search Tool

A *heuristic* algorithm for Local Alignment.

Can we find an $O(n \log n)$ algorithm??

(that would probably be efficient…)

How about $O(n^{1.5})$ or even $O(n^{1.8})$?

Today: Theoretical evidence that the answer is "no"!

HARDNESS FOR EASY PROBLEMS

How can we prove that a problem requires $\sim n^2$ time?

Prove NP-Hardness?	***$O(n^4)$ vs $O(n \log n)$?***
Unconditional Lower bounds ?	***No superlinear bounds***
Lower bounds for classes of algorithms ?	***Not a complete answer.***

IDEA: REDUCTIONS

Theorem: Problem X is NP-hard

X is in P → → *Every NP-complete problem is in P*

<u>Conclusion</u>: **X is probably not in P…**

OUR APPROACH

A surprising algorithm for problem Y → → *Unexpected breakthroughs in different areas*

<u>Conclusion</u>: **Such algorithm is unlikely…**

A refined version of NP-hardness…

MAIN RESULT

“Theorem”:

If Local Alignment can be solved in $n^{1.99}$ time, then:

- 3-SUM can be solved in $n^{1.99}$ time!

  Refuting the 3-SUM conjecture

- CNF-SAT can be solved in 1.99^n time!

- Max-4-Clique can be solved in $n^{3.99}$ time!

3SUM

Most famous example of this approach

Input: A list of n numbers

-15	-6	33	8	1	-21	4	-30	7	...	107

Output: Are there 3 numbers that sum to 0?

Trivial: $O(n^3)$, Simple: $O(n^2)$, Best: $O(n^2 / \log^2 n)$

[STOC 10': Patrascu] The 3SUM Conjecture:

3SUM cannot be solved in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$.

[Gajentaan – Overmars 95'] and many others:

- A long list of *3SUM-hard* problems.

3SUM-HARD PROBLEMS

The 3SUM Conjecture:
3SUM cannot be solved in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$.

The 3SUM conjecture implies the following lower bounds:

Computational Geometry

[C.G. 95': Gajentaan -- Oevrmars]
- 3-Points-On-A-Line requires $n^{2-o(1)}$ time.

[SODA 01': Barequet – Har Peled]
- Polygon Containment requires $n^{2-o(1)}$ time.

Graph Algorithms

[STOC 10': Patrascu] and [STOC 09': Vassilevska – Williams]
- Zero-Triangle requires $n^{3-o(1)}$ time.

[ICALP 13': A. -- Lewi]
- Zero-4-Path requires $n^{3-o(1)}$ time.

Stringology

[ICALP 14': Amir – Chan -- Lewenstein – Lewenstein]
- A lower bound for Jumbled Pattern Matching.

MAIN RESULT

"Theorem":

If Local Alignment can be solved in $n^{1.99}$ time, then:

- 3-SUM can be solved in $n^{1.99}$ time!

  Refuting the 3-SUM conjecture

- CNF-SAT can be solved in 1.99^n time!

  Refuting the Strong Exponential Time Hypothesis (SETH)

- Max-4-Clique can be solved in $n^{3.99}$ time!

THE STRONG EXPONENTIAL TIME HYPOTHESIS

Very useful for proving lower bounds…

CNF-SAT: Given a CNF formula on n variables and m clauses, is it satisfiable?

[01': Impagliazzo – Paturi -- Zane]
The Strong Exponential Time Hypothesis (SETH):
"CNF-SAT cannot be solved in $(2-\varepsilon)^n \, poly(m)$ time."

There are faster algorithms for k-SAT but they become $\sim 2^n$ as k grows.

SETH HARDNESS

The Strong Exponential Time Hypothesis (SETH):

"CNF-SAT cannot be solved in $2^{(1-\varepsilon)n}\ poly(m)$ time."

Theorem(s): *The SETH implies the following lower bounds:*

[SODA 10': Patrascu -- Williams]

- k-Dominating-Set requires $n^{k-o(1)}$ time.

[STOC 13': Roditty – Vassilevska Williams]

- A $\left(\frac{3}{2} - \varepsilon\right)$-approximation for the diameter requires $(mn)^{1-o(1)}$ time.

[FOCS 14': A. – Vassilevska Williams]

- Dynamic Reachability requires $m^{1-o(1)}$ amortized update time.

[FOCS 14': Bringmann]

- Computing the Frechet distance requires $n^{2-o(1)}$ time.

MAIN RESULT

<u>"Theorem":</u>

If Local Alignment can be solved in $n^{1.99}$ time, then:

- 3-SUM can be solved in $n^{1.99}$ time!

  Refuting the 3-SUM conjecture

  Computational Geometry

- CNF-SAT can be solved in 1.99^n time!

  Refuting the Strong Exponential Time Hypothesis (SETH)

  Satisfiability Algorithms

- Max-4-Clique can be solved in $n^{3.99}$ time!

  A longstanding open problem

  Graph Algorithms

<u>Bottom line</u>: Local Alignment probably requires $\sim n^2$ to solve optimally, and we should settle for heuristics in practice…

PLAN

- Motivation
- Main Results
- Other Results
- Proof examples:
  - CNF-SAT to LCS*
  - Sketch: 3-SUM to Local Alignment
- Open problems

MORE RESULTS

The conjectures imply tight lower bounds for:

- Edit Distance with gap penalties
- Normalized LCS
- Multiple Local Alignment
- Partial Match

- LCS*

  The simplest problem that requires $n^{2-o(1)}$ *time?*

LCS*

The Longest Common Substring with don't cares problem (LCS*)

Input: Two string of length n, containing don't care characters *.

S = RESEARCH_P*P*RS_ARE_*OOL

T = GO*GLE_SE*R*H_IS_U*EFUL

Output: The longest common substring.

Theorem: The SETH implies that LCS* on ***binary*** strings requires $n^{2-o(1)}$ time!

CNF-SAT TO LCS*

Theorem: The SETH implies that LCS* on ***binary*** strings requires $n^{2-o(1)}$ time!

Proof: $O(n^{2-\varepsilon})$ alg for LCS* => $2^{\left(1-\frac{\varepsilon}{2}\right)n}$ alg for CNF-SAT

Given a CNF formula with m clauses

$$\varphi(x_1, \ldots, x_n) = \underbrace{(\neg x_1 \vee x_{17} \vee \cdots \vee x_{10})}_{C_1} \wedge \cdots \wedge \underbrace{(x_2 \vee x_5 \vee x_{21})}_{C_m}$$

Split the variables and enumerate over partial assignments

$U_1 = \{x_1, \ldots, x_{n/2}\}$ $\quad$ $U_2 = \{x_{n/2+1}, \ldots, x_n\}$

$$\alpha = \begin{pmatrix} x_1 = T \\ x_2 = F \\ \vdots \\ x_{n/2} = T \end{pmatrix} \qquad \beta = \begin{pmatrix} x_{n/2+1} = F \\ x_{n/2+2} = F \\ \vdots \\ x_n = T \end{pmatrix}$$

There are $N = 2^{n/2}$ such α's and β's

Goal of alg: find a pair such that $(\alpha \cdot \beta)$ sat φ.

CNF-SAT TO LCS*

Theorem: The SETH implies that LCS* on ***binary*** strings requires $n^{2-o(1)}$ time!

Proof: $O(n^{2-\varepsilon})$ alg for LCS* => $2^{\left(1-\frac{\varepsilon}{2}\right)n}$ alg for CNF-SAT

$$\varphi \text{ is satisfiable} \Leftrightarrow \exists \alpha, \beta : \forall C_i : (\alpha \cdot \beta) \text{ sat } C_i$$

Idea: construct strings S, T of length $\sim(2^{n/2}m)$ such that

$$\text{LCS*}(S,T) = m \Leftrightarrow \exists \alpha, \beta : \forall C_i : (\alpha \cdot \beta) \text{ sat } C_i$$

Done: we get a $\left(2^{n/2}m\right)^{2-\varepsilon} = 2^{\left(1-\frac{\varepsilon}{2}\right)n} poly(m)$ alg for CNF-SAT

CNF-SAT TO LCS*

Theorem: The SETH implies that LCS* on ***binary*** strings requires $n^{2-o(1)}$ time!

Proof: Construct strings S, T of length $O(2^{n/2}m)$ such that

$$\text{LCS*}(S, T) = m \Leftrightarrow \exists \alpha, \beta : \forall C_i : (\alpha \cdot \beta) \text{ sat } C_i$$

Define strings of length m:

$$T_\alpha = [0 \ \ * \ \ * \ \ 0 \ \ * \ \ \cdots \ \ 0]$$

$$T_\alpha[i] = \begin{cases} * & \alpha \text{ sat } C_i \\ 0 & \text{otherwise} \end{cases}$$

$$S_\beta = [0 \ \ 0 \ \ 1 \ \ 0 \ \ 1 \ \ \cdots \ \ 0]$$

$$S_\beta[i] = \begin{cases} 0 & \beta \text{ sat } C_i \\ 1 & \text{otherwise} \end{cases}$$

Then: $\mathrm{T}_\alpha \equiv S_\beta \Leftrightarrow \forall C_i : (\alpha \cdot \beta) \text{ sat } C_i$

Construct S,T in $O(2^{n/2}\, m)$ time:

$$T = [\cdots \ T_{\alpha 1} \cdots] \ \$ \ [\cdots T_{\alpha 2} \cdots] \ \$ \ \cdots \ \$ \ [\cdots T_{\alpha N} \cdots]$$

$$S = [\cdots \ S_{\beta 1} \cdots] \ \# \ [\cdots S_{\beta 2} \cdots] \ \# \ \cdots \ \# \ [\cdots S_{\beta N} \cdots]$$

∎

3-SUM TO LOCAL ALIGNMENT

3-SUM on n numbers

-15	-6	33	$\boldsymbol{x} \in [\pm \boldsymbol{n}^3]$	-30	7	…	107

$|\Sigma| \sim n^3$?

↓ [ESA 14': A. – Lewi – Williams]

$n^{o(1)}$ instances of 3-Vector-SUM on n vectors

		$\boldsymbol{v_x = (x_1, \dots, x_d)}$	…	

$|\Sigma| \sim \log n$

$x_i \in [\pm \log n]$ and $d = O\left(\frac{\log n}{\log \log n}\right)$

$\exists v_a, v_b, v_c : v_a + v_b + v_c = (0, \dots, 0)$?

$|\Sigma| \sim n^{\varepsilon} \log n$

↓ Hashing…

Define substrings of length d:
$S_x = [\dots,\ '(h(x), x_i)',\ \dots]$

Σ contains pairs
$(h(x), x_i)$

3-SUM TO LOCAL ALIGNMENT

Define substrings of length d:
$S_x = [\ldots, \ '(h(x), x_i)', \ldots]$

Σ contains pairs
$(h(x), x_i)$

Our scoring matrix enforces that:

$(h(x), x_i)$ and $(h(y), y_i)$ will "match" iff:

$x_i + y_i + z_i = 0$ where z is determined by $h(x), h(y)$

CONCLUSION

The reductions explain the lack of progress
and prove that new ideas are required for faster algorithms

*"An **opportunity** to solve many famous open problems while working on your favorite problem!"*

- Subquadratic Edit Distance?
- Subquadratic LCS?
- Subcubic Protein Folding?
- Subcubic Tree Edit Distance?

Thank You!
Questions?