Fine-Grained Complexity

An Introduction

Antonis Antonopoulos

9/11/2018

CoReLab, NTUA

Table of contents

Introduction

Introduction

- The time hierarchy Theorem states that there are problems solved in $T(n)$, for a constructible function T, but not in $\mathcal{O}\left(T^{1-\varepsilon}(n)\right)$, for $\varepsilon > 0$.
- All known b^n-time kSAT algorithms have: $\lim_{k\to\infty} b = 2$ (all the results are of the form $\mathcal{O}^*\left(2^{n-\frac{cn}{k}}\right)$).
- **NP**-hardness is a useful tool for polynomial time hardness, but it doesn't capture non-polynomial times. For example, an $\mathcal{O}^*\left(2^{\sqrt{n}}\right)$ algorithm for kSAT would be a great advancement. It would be difficult the **P** vs **NP** conjecture to be strong enough to prove exponential lower bounds for kSAT.
- The optimality conjecture of the exponential-time kSAT algorithms is formalized as the Strong Exponential Time Hypothesis (*Impagliazzo, Paturi, 2001*).

Introduction

Exponential Time Hypothesis [ETH]

There exists an $\varepsilon > 0$ such that 3SAT requires $2^{\varepsilon n}$ time.

Strong Exponential Time Hypothesis [SETH]

For all $\delta < 1$ exists a $k \geq 3$ such that kSAT requires $2^{\delta n}$ time.

Introduction

- Other problems in **P** have similar behaviour:

- Other problems in **P** have similar behaviour:
- The Edit Distance problem, for example, admits a classical DP $\mathcal{O}\left(n^2\right)$ algorithm (*Wagner, Fisher 1974*), but till today the best improvement is $\mathcal{O}\left(n^2/\log^2 n\right)$ (*Masek, Paterson 1980*).

- Other problems in **P** have similar behaviour:
- The Edit Distance problem, for example, admits a classical DP $\mathcal{O}\left(n^2\right)$ algorithm (*Wagner, Fisher 1974*), but till today the best improvement is $\mathcal{O}\left(n^2/\log^2 n\right)$ (*Masek, Paterson 1980*).
- Similar observations hold for the Longest Common Subsequence Problem (LCS).

Introduction

- Other problems in **P** have similar behaviour:
- The Edit Distance problem, for example, admits a classical DP $\mathcal{O}\left(n^2\right)$ algorithm (*Wagner, Fisher 1974*), but till today the best improvement is $\mathcal{O}\left(n^2/\log^2 n\right)$ (*Masek, Paterson 1980*).
- Similar observations hold for the Longest Common Subsequence Problem (LCS).
- The 3SUM problem: *Given a set S of integers, are there $x, y, z \in S$ such that $x + y + z = 0$?* We have a trivial $\mathcal{O}\left(n^2 \log n\right)$ algorithm, and the best known runs in $\mathcal{O}\left(n^2(\log\log n)^2/\log^2 n\right)$ (*Baran, Demaine, Patrascu*, 2008).

Introduction

- The Colinearity Problem in Computational Geometry: *Given n points in the plane, are any three colinear?* The best known algorithms runs in $n^{2-o(1)}$.

Introduction

- The Colinearity Problem in Computational Geometry: *Given n points in the plane, are any three colinear?* The best known algorithms runs in $n^{2-o(1)}$.
- The APSP (All-Pairs Shortest Paths) Problem. There is the classical DP $\mathcal{O}\left(n^3\right)$ algorithm (Floyd, Warshall, 1962). Best known algorithm runs in $\mathcal{O}\left(n^3/exp(\sqrt{\log n})\right)$ (Williams 2014), using Circuit Complexity tools, namely the Razborov-Smolensky polynomials.

Introduction

- The Colinearity Problem in Computational Geometry: *Given n points in the plane, are any three colinear?* The best known algorithms runs in $n^{2-o(1)}$.
- The APSP (All-Pairs Shortest Paths) Problem. There is the classical DP $\mathcal{O}\left(n^3\right)$ algorithm (Floyd, Warshall, 1962). Best known algorithm runs in $\mathcal{O}\left(n^3/exp(\sqrt{\log n})\right)$ (Williams 2014), using Circuit Complexity tools, namely the Razborov-Smolensky polynomials.
- For any of these problems, is there a complexity theoretic reason (a.k.a. barrier) to obtaining a better algorithm?

Introduction

- The Colinearity Problem in Computational Geometry: *Given n points in the plane, are any three colinear?* The best known algorithms runs in $n^{2-o(1)}$.
- The APSP (All-Pairs Shortest Paths) Problem. There is the classical DP $\mathcal{O}\left(n^3\right)$ algorithm (Floyd, Warshall, 1962). Best known algorithm runs in $\mathcal{O}\left(n^3/exp(\sqrt{\log n})\right)$ (Williams 2014), using Circuit Complexity tools, namely the Razborov-Smolensky polynomials.
- For any of these problems, is there a complexity theoretic reason (a.k.a. barrier) to obtaining a better algorithm?
- Or else, **is the reason for this hardness the same for all -or even for some- these problems?**

Introduction

Fine-Grained Approach

- Start with a problem solvable in $\mathcal{O}\left(t(n)\right)$, and nothing much better known, and formulate a *hypothesis*.
- Connect this problem to others via *fine-grained reductions*, so that for a problem having an $\mathcal{O}\left(T(n)\right)$ algorithm, obtaining an $\mathcal{O}\left(T^{1-\varepsilon}(n)\right)$ algorithm for $\varepsilon > 0$, would violate the hypothesis.

Introduction

Fine-Grained Approach

- Start with a problem solvable in $\mathcal{O}(t(n))$, and nothing much better known, and formulate a *hypothesis*.
- Connect this problem to others via *fine-grained reductions*, so that for a problem having an $\mathcal{O}(T(n))$ algorithm, obtaining an $\mathcal{O}\left(T^{1-\varepsilon}(n)\right)$ algorithm for $\varepsilon > 0$, would violate the hypothesis.

- **P** vs **NP** is model indepedent, but in a fine-grained analysis, the precise computational model *does* matter.

- We use the *Word RAM model* with $\mathcal{O}(\log n)$ bit words
(i.e. operations on $\mathcal{O}(\log n)$ bit chunks of data in constant time).

Fine-Grained Standard Conjectures

SETH

For all $\delta < 1$ exists a $k \geq 3$ such that kSAT requires $2^{\delta n}$ time.

The 3SUM Conjecture

Any algorithm requires $n^{2-o(1)}$ time to determine whether a set $S \subset \{-n^3, \cdots, n^3\}$ of integers of size $|S| = n$ contains 3 distinct elements $a, b, c \in S$ such that $a + b = c$.

APSP Conjecture

Any algorithm requires $n^{3-o(1)}$ time to compute the distances between every pair of vertices in an n node graph with edge weights in $\{1, \dots, n^c\}$, for some constant c.

- These conjectures can be extended to randomized algorithms.

ETH, SETH and implications

Exponential Time Hypothesis

- We need a more strict formalization to work with:
- Let:

$$s_k = \mathsf{inf}\{c \mid \text{there is a } \mathcal{O}^*\left(2^{c \cdot n}\right) \text{ algorithm for kSAT with } n \text{ variables}\}$$

- Then:

Exponential Time Hypothesis

- We need a more strict formalization to work with:
- Let:

$$s_k = \mathsf{inf}\{c \mid \text{there is a } \mathcal{O}^*\left(2^{c \cdot n}\right) \text{ algorithm for kSAT with } n \text{ variables}\}$$

- Then:

ETH

$$s_3 > 0$$

Exponential Time Hypothesis

- We need a more strict formalization to work with:
- Let:

$$s_k = \mathsf{inf}\{c \mid \text{there is a } \mathcal{O}^*\left(2^{c \cdot n}\right) \text{ algorithm for kSAT with } n \text{ variables}\}$$

- Then:

ETH

$$s_3 > 0$$

SETH

$$\lim_{k \to \infty} s_k = 1$$

The Need for Sparsification

- Can we relate SETH to ETH?
- Recall the classic reduction from **NP**-completeness theory:

Theorem

If 3SAT can be solved in polynomial time, then so can kSAT, for every $k \geq 3$.

- Can we say the same for subexponential time solvability?

The Need for Sparsification

- Can we relate SETH to ETH?
- Recall the classic reduction from **NP**-completeness theory:

Theorem

If 3SAT can be solved in polynomial time, then so can kSAT, for every $k \geq 3$.

- Can we say the same for subexponential time solvability?
- Recall the proof of the above theorem:
  - For every k-clause $C = (x_1 \vee x_2 \vee \cdots \vee x_k)$, we introduce new variables $y_3, y_4, \ldots, y_{k-1}$ and replace C by:

$$(x_1 \vee x_2 \vee y_3) \wedge (\bar{y}_3 \vee x_3 \vee y_4) \wedge (\bar{y}_4 \vee x_4 \vee y_5) \wedge \cdots \wedge (\bar{y}_{k-1} \vee x_{k-1} \vee x_k)$$

The Need for Sparsification

- Can we relate SETH to ETH?
- Recall the classic reduction from **NP**-completeness theory:

Theorem

If 3SAT can be solved in polynomial time, then so can kSAT, for every $k \geq 3$.

- Can we say the same for subexponential time solvability?
- Recall the proof of the above theorem:
  - For every k-clause $C = (x_1 \vee x_2 \vee \cdots \vee x_k)$, we introduce new variables $y_3, y_4, \ldots, y_{k-1}$ and replace C by:

$$(x_1 \vee x_2 \vee y_3) \wedge (\bar{y}_3 \vee x_3 \vee y_4) \wedge (\bar{y}_4 \vee x_4 \vee y_5) \wedge \cdots \wedge (\bar{y}_{k-1} \vee x_{k-1} \vee x_k)$$

- If we use the above reduction, we'll **fail**:
  For each clause we introduce $k - 3$ variables, thus the new formula has $n + (k-3)m$ variables.
- If we can solve 3SAT in subexponential time, suppose $\mathcal{O}\left(2^{\varepsilon n}\right)$ for some $\varepsilon > 0$, then the above reduction algorithm gives us $\mathcal{O}\left(2^{\varepsilon(n+(k-3)m)}\right)$. If $m = \mathcal{O}\left(n^2\right)$, then it's a disaster.

The Need for Sparsification

- We clearly need an intermediate step here:

Theorem (The Sparsification Lemma)

For all $\varepsilon > 0$ and positive q, there is a constant $C = C(\varepsilon, q)$ such that any qCNF formula ϕ with n variables can be expressed as $\phi = \bigvee_{i=1}^{t} \psi_i$, where $t \leq 2^{\varepsilon n}$ and each ψ_i is a qCNF formula over the same variable set as ϕ and at most Cn clauses.
This disjunction can be computed in $\mathcal{O}^*\left(2^{\varepsilon n}\right)$ time.

- Using the above lemma, we can "sparsify" the kCNF formula, in order to avoid the previously mentioned phenomena.

Exponential Time Hypothesis

Theorem

SETH $\Rightarrow$ ETH

Proof.

- Assume, for the sake of contradiction, that $s_3 = 0$.
- So, for every $c > 0$ there exists algorithm A_c solving 3SAT in $\mathcal{O}^*\left(2^{cn}\right)$ time.

Exponential Time Hypothesis

Theorem

SETH $\Rightarrow$ ETH

Proof.

- Assume, for the sake of contradiction, that $s_3 = 0$.
- So, for every $c > 0$ there exists algorithm A_c solving 3SAT in $\mathcal{O}^*\left(2^{cn}\right)$ time.
- Consider the following method for solving *qSAT*:
    - Given formula ϕ, apply the sparsification lemma for some $\varepsilon > 0$.
    - We now have in $\mathcal{O}^*\left(2^{\varepsilon n}\right)$ time at most $2^{\varepsilon n}$ ψ_i's, ϕ is satisfiable iff any of the ψ_i's is, and each ψ_i has at most $C(\varepsilon, q)n$ clauses.
    - Now apply the classic reduction from qSAT to 3SAT on any ψ_i (recall that for every q-clause of ψ_i we introduce $q - 3$ new variables and $q - 2$ new clauses).
    - This results to a 3CNF formula ψ_i' with at most $(1 + qC(\varepsilon, q))n$ variables.

Theorem

SETH $\Rightarrow$ ETH

Proof. (*cont'd*)

- Now, if we apply the algorithm A_δ to ψ'_i, for some $\delta > 0$, we can solve satisfiability of ψ'_i in $\mathcal{O}^*\left(2^{\delta' n}\right)$, for $\delta' = \delta \cdot (1 + qC(\varepsilon, q))$.
- By applying the procedure to all ψ_i's, we can solve satisfiability of ϕ in $\mathcal{O}^*\left(2^{\delta'' n}\right)$, for $\delta'' = \varepsilon + \delta' = \varepsilon + \delta \cdot (1 + qC(\varepsilon, q))$.
- Since $s_3 = 0$, we can choose ε and δ arbitrarily close to 0, so δ'' is arbitrarily close to 0, hence $s_k = 0$ for $k \geq 3$, which contradicts SETH. $\square$

Fine-Grained Reductions

Fine-Grained Reductions

Fine-Grained Reduction

Let $a(n)$, $b(n)$ be nondecreasing functions of n.
Problem A is (a, b)-reducible to problem B ($A \leq_{a,b} B$), if:
$\forall \varepsilon > 0 \, \exists \delta > 0$, an algorithm F with oracle access to B,

- F runs in at most $d \cdot a^{1-\delta}(n)$ time
- F makes at most $k(n)$ oracle queries adaptively
  (the j^{th} instance B_j is a function of $\{B_i, a_i\}_{1 \leq i < j}$)
- The sizes $|B_i| = n_i$ for any choice of oracle answers a_i obey the inequality:

$$\sum_{i=1}^{k(n)} b^{1-\varepsilon}(n_i) \leq d \cdot a^{1-\delta}(n)$$

- Improvements over $b(n)$ for B imply inprovements over $a(n)$ for A.

The Orthogonal Vectors Problem

- A key problem for understanding fine-grained reductions is the Orthogonal Vectors problem.

Orthogonal Vectors (OV)

Let $d = \omega(\log n)$. Given two sets $A, B \subseteq \{0,1\}^*$, with $|A| = |B| = n$, decide whether there exist $a \in A$, $b \in B$ such that $a \cdot b = 0$.

***k*-Orthogonal Vectors** (kOV)

Let $d = \omega(\log n)$. Given sets $A_1, \ldots, A_k \subseteq \{0,1\}^*$, with $|A_i| = n$ for all $i \in [k]$, decide whether there exist $\alpha_1 \in A_1, \ldots, \alpha_k \in A_k$ such that: $\alpha_1 \cdot \alpha_2 \cdots \alpha_k = \sum_{i=1}^{d} \prod_{j=1}^{k} \alpha_j[\,i\,] = 0$.

- Naïve solution in $\mathcal{O}\left(n^k d\right)$ time.
- Best known algorithm runs in $\mathcal{O}\left(n^{k-1/\Theta(d/\log n)}\right)$ time.

The Orthogonal Vectors Problem

- A key problem for understanding fine-grained reductions is the Orthogonal Vectors problem.

Orthogonal Vectors (OV)

Let $d = \omega(\log n)$. Given two sets $A, B \subseteq \{0,1\}^*$, with $|A| = |B| = n$, decide whether there exist $a \in A, b \in B$ such that $a \cdot b = 0$.

***k*-Orthogonal Vectors** (kOV)

Let $d = \omega(\log n)$. Given sets $A_1, \ldots, A_k \subseteq \{0,1\}^*$, with $|A_i| = n$ for all $i \in [k]$, decide whether there exist $\alpha_1 \in A_1, \ldots, \alpha_k \in A_k$ such that: $\alpha_1 \cdot \alpha_2 \cdots \alpha_k = \sum_{i=1}^{d} \prod_{j=1}^{k} \alpha_j[\, i \,] = 0$.

- Naïve solution in $\mathcal{O}\left(n^k d\right)$ time.
- Best known algorithm runs in $\mathcal{O}\left(n^{k-1/\Theta(d/\log n)}\right)$ time.

The *kOV* Hypothesis

There is no (randomized) algorithm that can solve kOV in $n^{k-\varepsilon} poly(d)$ time for constant $\varepsilon > 0$.

Fine-Grained Reductions

Theorem (Williams 2005)

SAT $\leq_{2^n, n^k}$ kOV

Fine-Grained Reductions

Theorem (Williams 2005)

SAT $\leq_{2^n, n^k}$ kOV

Proof.

- Let $F(n, m)$ be the given formula.
- We can assume, due to the Sparsification Lemma, that F has $\mathcal{O}(n)$ clauses.
- Split the n variables into k sets $V_1, \ldots, V_k$.
- For every $j \in [k]$, create a set A_j containing a vector $\alpha^j(\phi)$ for each of the $N = 2^{n/k}$ partial t.a.'s., where:

$$\alpha^j(\phi)[c] = 0, \text{ iff the } c^{th} \text{ clause of } F \text{ is satisfied by } \phi.$$

- If for some $\alpha_1(\phi_1), \alpha_2(\phi_2), \ldots, \alpha_k(\phi_k)$: $\sum_c \prod_j \alpha_j(\phi_j)[c] = 0$, then for every c there is some vector $\alpha_j(\phi_j)$ that is 0 in c, so ϕ_j satisfies c.
- Thus, the concatenation $\bigcirc_{\ell=1}^{k} \phi_\ell$ satisfies all clauses.

Fine-Grained Reductions

Theorem (Williams 2005)

SAT $\leq_{2^n, n^k}$ kOV

Proof. (cont'd)

- Conversely, if ϕ satisfies all clauses, then $\phi_j = \phi \restriction_{V_j}$ and so $\sum_c \prod_j \alpha_j(\phi_j)[c] = 0$.
- So, if we can solve kOV in $N^{k-\varepsilon} poly(m)$ time in $\{0,1\}^m$, then we can solve kSAT in $(2^{n/k})^{k-\varepsilon} poly(m) = 2^{n(1-\varepsilon')} poly(m)$, contradicting SETH. $\square$

Fine-Grained Reductions

Theorem (Williams 2005)

SAT $\leq_{2^n, n^k}$ kOV

Proof. (cont'd)

- Conversely, if ϕ satisfies all clauses, then $\phi_j = \phi \restriction_{V_j}$ and so $\sum_c \prod_j \alpha_j(\phi_j)[c] = 0$.
- So, if we can solve kOV in $N^{k-\varepsilon} poly(m)$ time in $\{0,1\}^m$, then we can solve kSAT in $(2^{n/k})^{k-\varepsilon} poly(m) = 2^{n(1-\varepsilon')} poly(m)$, contradicting SETH. $\square$

- It is simply to see that kOV $\leq_{n^k, n^{k-1}}$ (k − 1)OV $\cdots \leq_{n^3, n^2}$ 2OV.
- So, 2OV is the hardest of these problems.
- The above reduction can be routed through kDOMINATING SET.

Thank You!