Nondeterministic extensions of the Strong Exponential Time Hypothesis and consequences for non-reducibility

Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mikhailin, Ramamohan Paturi, Stefan Schneider

UCSD

2015

Fine-Grained Hardness

- For many problems progress has stalled:

Fine-Grained Hardness

- For many problems progress has stalled:
  - Edit Distance in $\tilde{\Omega}(n^2)$ [WagnerFischer74]

Fine-Grained Hardness

- For many problems progress has stalled:
  - Edit Distance in $\tilde{\Omega}(n^2)$ [WagnerFischer74]
  - 3-SUM in $\tilde{\Omega}(n^2)$

Fine-Grained Hardness

- For many problems progress has stalled:
  - Edit Distance in $\tilde{\Omega}(n^2)$ [WagnerFischer74]
  - 3-SUM in $\tilde{\Omega}(n^2)$
  - CKT-SAT in $\tilde{\Omega}(2^n)$

Fine-Grained Hardness

- For many problems progress has stalled:
  - Edit Distance in $\tilde{\Omega}(n^2)$ [WagnerFischer74]
  - 3-SUM in $\tilde{\Omega}(n^2)$
  - CKT-SAT in $\tilde{\Omega}(2^n)$
  - 3-Points-On-a-Line in $\tilde{\Omega}(n^2)$

Fine-Grained Hardness

- For many problems progress has stalled:
  - Edit Distance in $\tilde{\Omega}(n^2)$ [WagnerFischer74]
  - 3-SUM in $\tilde{\Omega}(n^2)$
  - CKT-SAT in $\tilde{\Omega}(2^n)$
  - 3-Points-On-a-Line in $\tilde{\Omega}(n^2)$
- Proving these lower bounds seems out of reach

Fine-Grained Hardness

- For many problems progress has stalled:
  - Edit Distance in $\tilde{\Omega}(n^2)$ [WagnerFischer74]
  - 3-SUM in $\tilde{\Omega}(n^2)$
  - CKT-SAT in $\tilde{\Omega}(2^n)$
  - 3-Points-On-a-Line in $\tilde{\Omega}(n^2)$
- Proving these lower bounds seems out of reach
- Goal: Explain bounds from common principle

Conditional Lower Bounds

- Conditional lower bounds are an attempt to tackle the problem

Conditional Lower Bounds

- Conditional lower bounds are an attempt to tackle the problem
- Relate hard problems by reductions that respect resources

Conditional Lower Bounds

- Conditional lower bounds are an attempt to tackle the problem
- Relate hard problems by reductions that respect resources
- Goal 1: Explain hardness of as many problems as possible

Conditional Lower Bounds

- Conditional lower bounds are an attempt to tackle the problem
- Relate hard problems by reductions that respect resources
- Goal 1: Explain hardness of as many problems as possible
- Goal 2: Explain hardness using a common principle

Important Conjectures: 3-Sum

- 3-SUM: Given n integers $a_1, \ldots, a_n \in [-M, M]$ find i, j, k such that $a_i + a_j + a_k = 0$

Important Conjectures: 3-Sum

- 3-SUM: Given n integers $a_1, \ldots, a_n \in [-M, M]$ find i, j, k such that $a_i + a_j + a_k = 0$
- Simple algorithm: $\tilde{O}(n^2)$

Important Conjectures: 3-Sum

- 3-SUM: Given n integers $a_1, \ldots, a_n \in [-M, M]$ find i, j, k such that $a_i + a_j + a_k = 0$
- Simple algorithm: $\tilde{O}(n^2)$
- Fastest known algorithms: $O(n^2 / \operatorname{polylog} n)$ [BDP08, GP14]

Important Conjectures: 3-Sum

- 3-SUM: Given n integers $a_1, \ldots, a_n \in [-M, M]$ find i, j, k such that $a_i + a_j + a_k = 0$
- Simple algorithm: $\tilde{O}(n^2)$
- Fastest known algorithms: $O(n^2 / \operatorname{polylog} n)$ [BDP08, GP14]
- 3-SUM conjecture: There is no $O(n^{2-\varepsilon})$ algorithm for 3-Sum

Important Conjectures: All-Pairs Shortest Path

- APSP: Given a weighted directed graph, find the distance of every pair u, v

Important Conjectures: All-Pairs Shortest Path

- APSP: Given a weighted directed graph, find the distance of every pair u, v
- Dynamic Programming: $O(n^3)$

Important Conjectures: All-Pairs Shortest Path

- APSP: Given a weighted directed graph, find the distance of every pair u, v
- Dynamic Programming: $O(n^3)$
- Fastest known algorithms: $O(n^3/2^{\sqrt{\log n}})$ [W14]

Important Conjectures: All-Pairs Shortest Path

- APSP: Given a weighted directed graph, find the distance of every pair u, v
- Dynamic Programming: $O(n^3)$
- Fastest known algorithms: $O(n^3/2^{\sqrt{\log n}})$ [W14]
- All-pairs shortest path conjecture: There is no $O(n^{3-\varepsilon})$ algorithm for APSP

Important Conjectures: Strong Exponential Time Hypothesis

- k-SAT: Given a k-CNF, find a satisfying assignment

Important Conjectures: Strong Exponential Time Hypothesis

- k-SAT: Given a k-CNF, find a satisfying assignment
- Brute force: $\tilde{O}(2^n)$

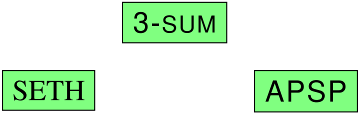
Important Conjectures: Strong Exponential Time Hypothesis

- k-SAT: Given a k-CNF, find a satisfying assignment
- Brute force: $\tilde{O}(2^n)$
- Fastest known algorithm: $O(2^{(1-\frac{c}{k})n})$ [PPSZ98]

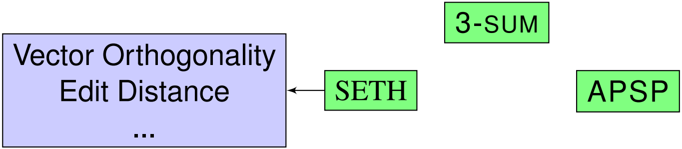
Important Conjectures: Strong Exponential Time Hypothesis

- k-SAT: Given a k-CNF, find a satisfying assignment
- Brute force: $\tilde{O}(2^n)$
- Fastest known algorithm: $O(2^{(1-\frac{c}{k})n})$ [PPSZ98]
- Strong Exponential Time Hypothesis: For every $s > 0$, there is a k such that k-SAT cannot be solved in time $2^{(1-s)n}$

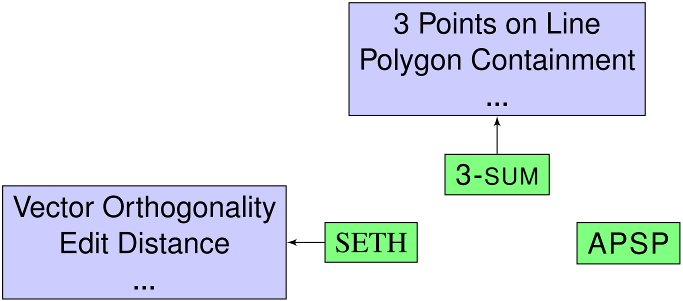
Conditional Lower Bounds



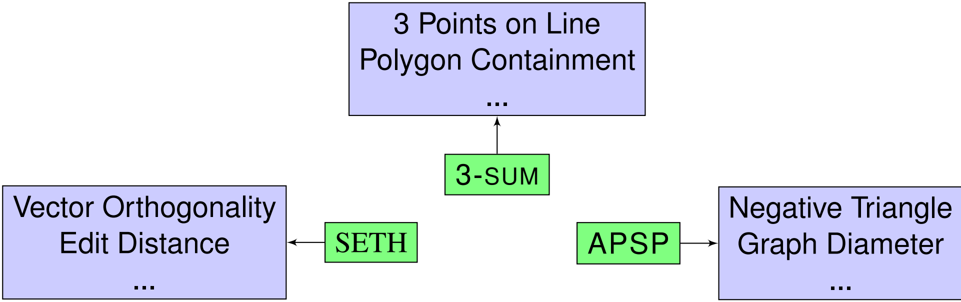
Conditional Lower Bounds



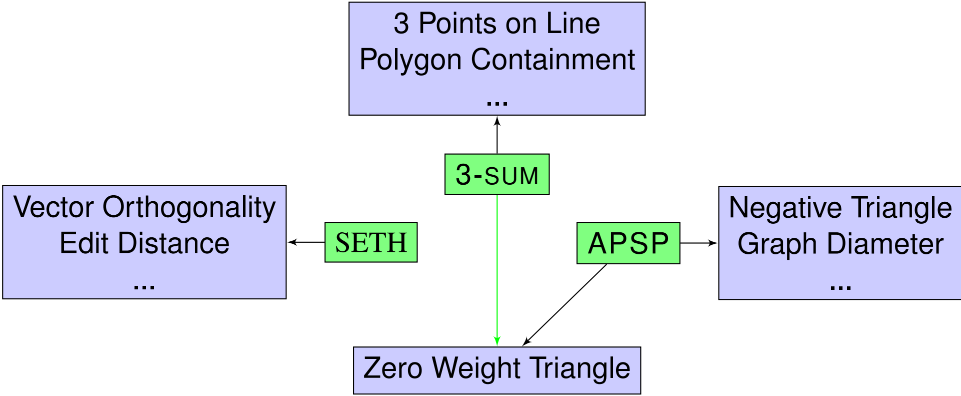
Conditional Lower Bounds



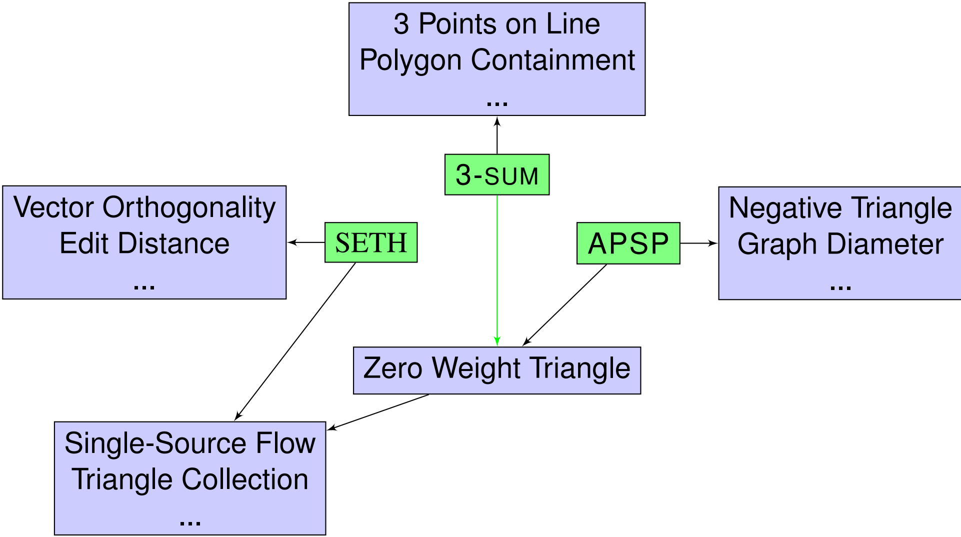
Conditional Lower Bounds



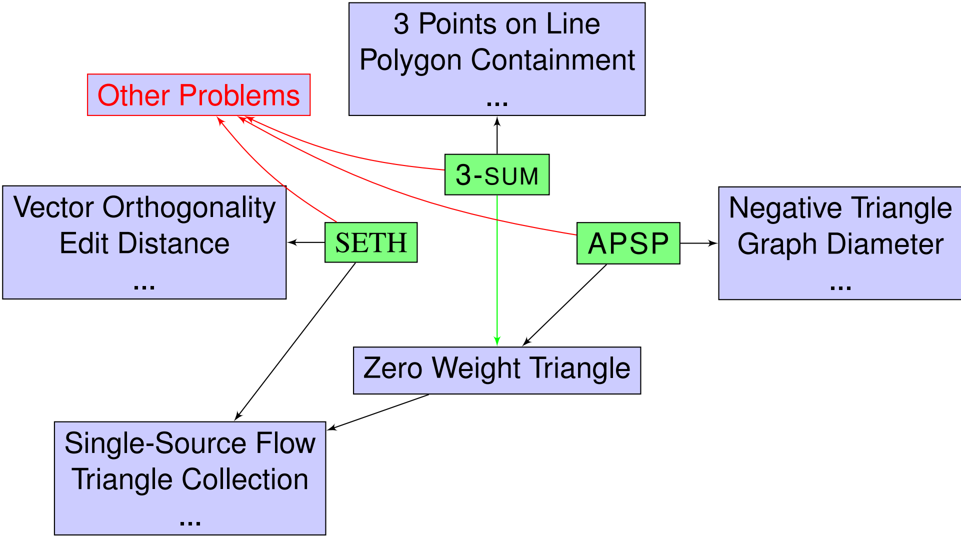
Conditional Lower Bounds



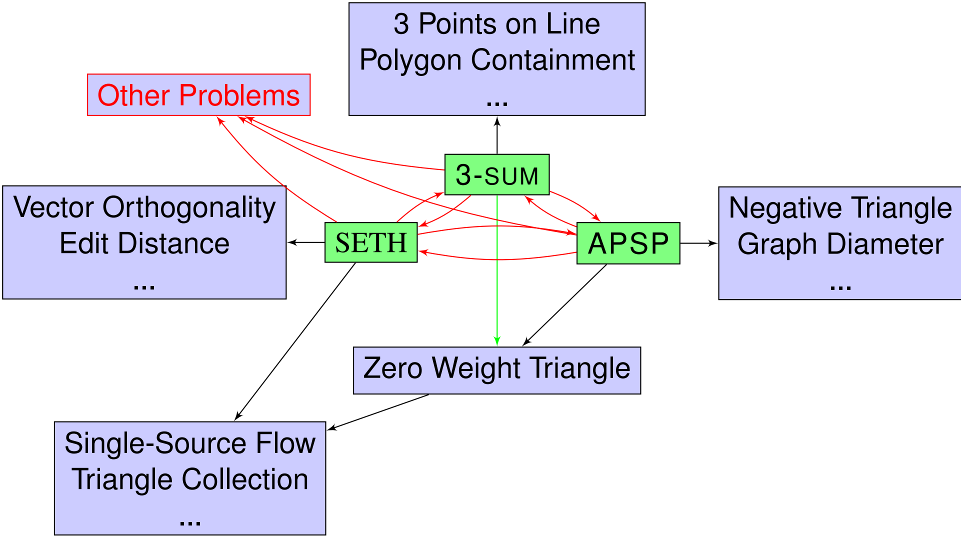
Conditional Lower Bounds



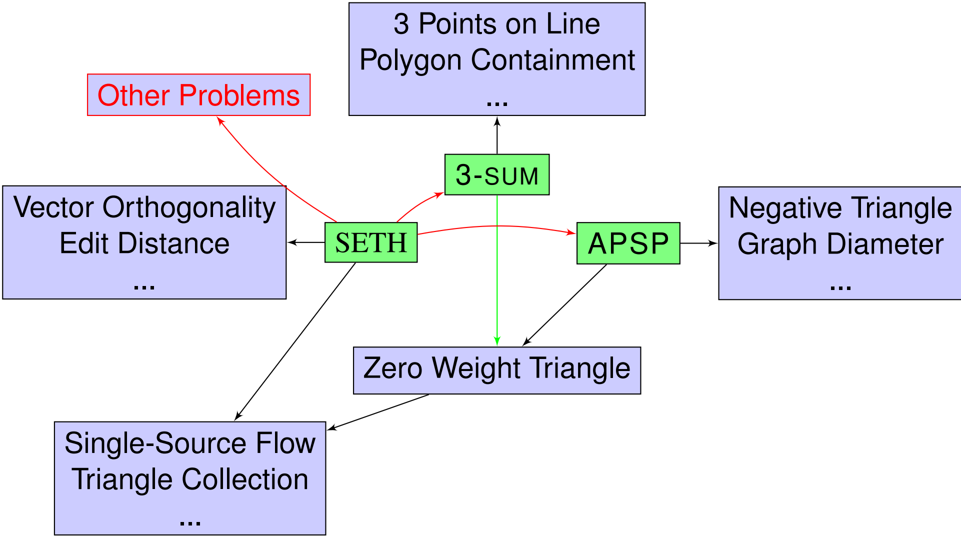
Conditional Lower Bounds



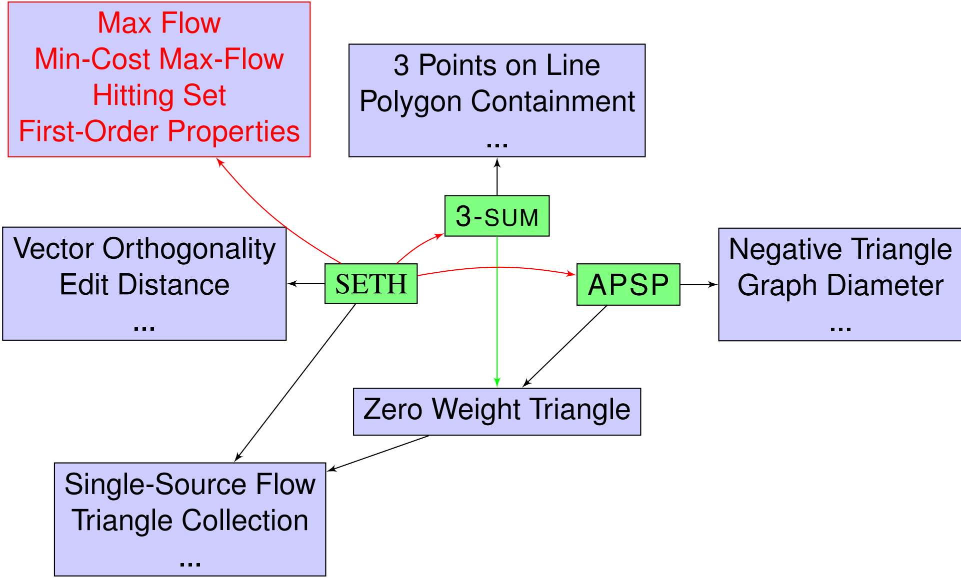
Conditional Lower Bounds



Conditional Lower Bounds



Conditional Lower Bounds



Fine-Grained Reductions

- Formalized in [VWW10]

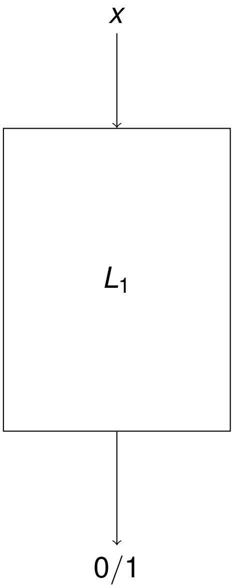
Fine-Grained Reductions

- Formalized in [VWW10]
- Fine-grained reduction from (L_1, T_1) to (L_2, T_2)

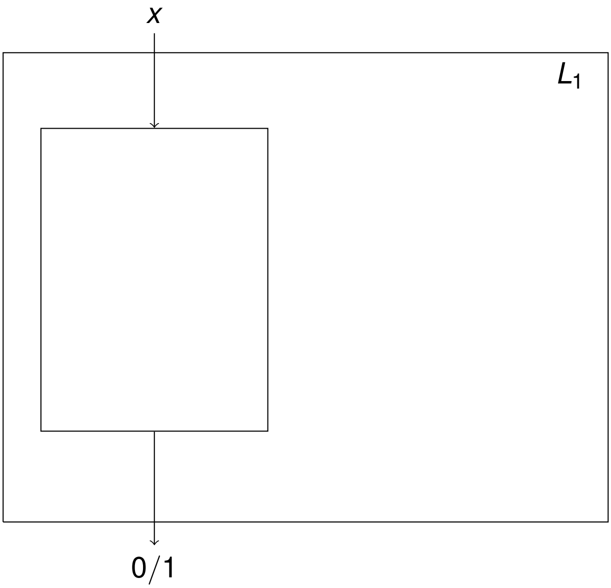
Fine-Grained Reductions

- Formalized in [VWW10]
- Fine-grained reduction from (L_1, T_1) to (L_2, T_2)
- Turing reduction that respects resources

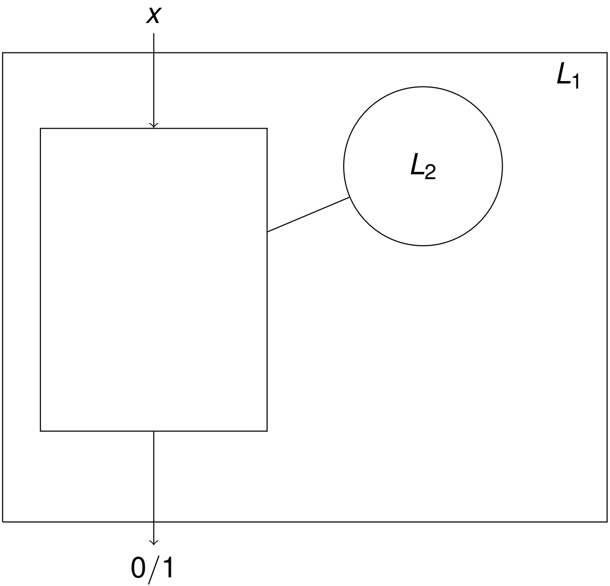
Fine-Grained Reduction



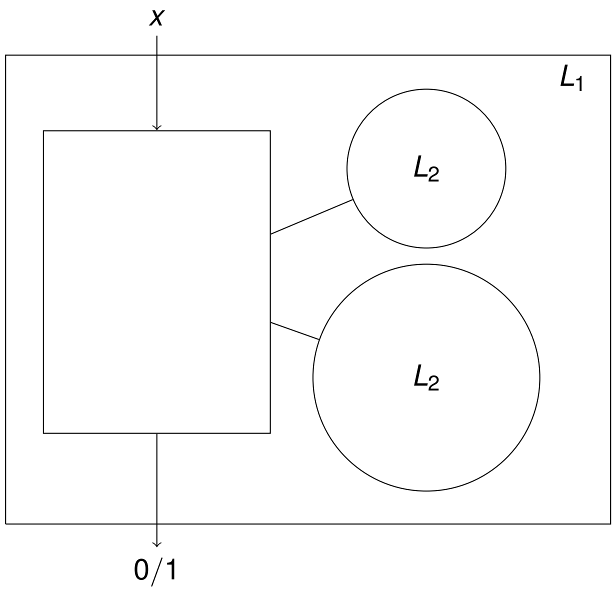
Fine-Grained Reduction



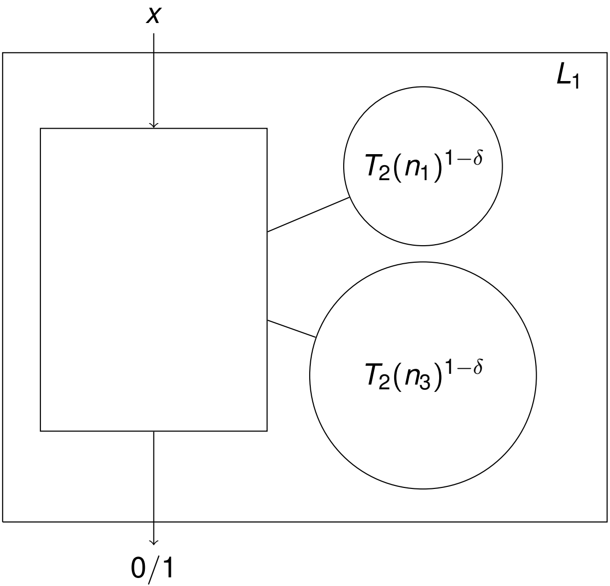
Fine-Grained Reduction



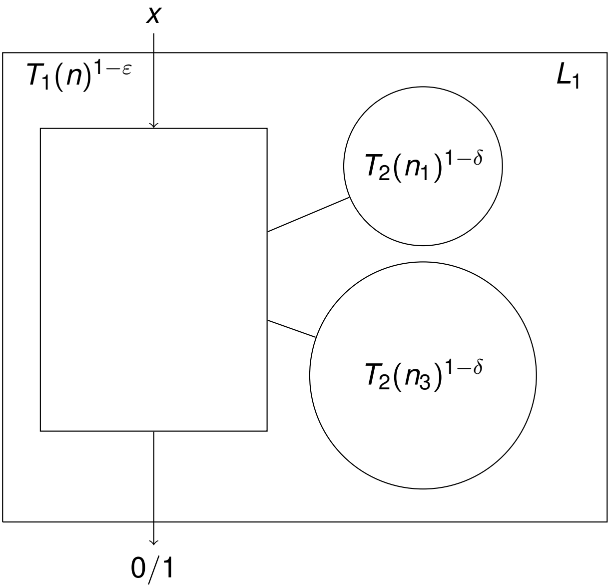
Fine-Grained Reduction



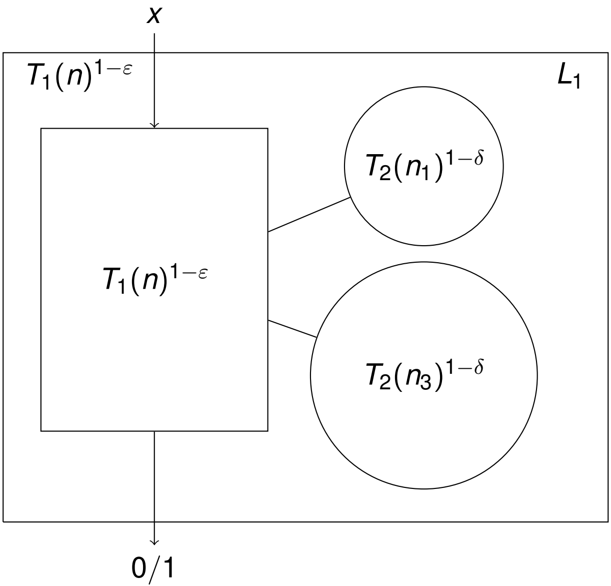
Fine-Grained Reduction



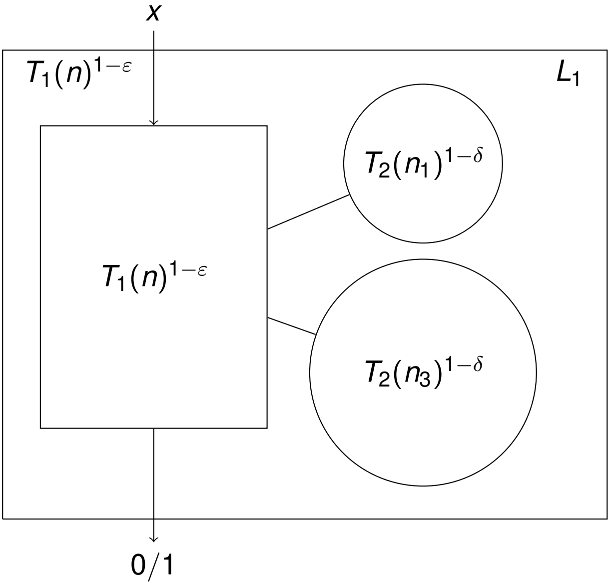
Fine-Grained Reduction



Fine-Grained Reduction



Fine-Grained Reduction



$\sum_i T_2(n_i)^{1-\delta}$
$\leq T_1(n)^{1-\varepsilon}$

Proof Idea

Lemma (Basic Idea)

Every SETH*-hard problem has property X.* 3-SUM *and* APSP *do not have property X.*

NSETH

Nondeterministic Strong Exponential Time Hypothesis

For every $s > 0$, there is a k such that k-SAT cannot be solved in *co-nondeterministic* time $2^{(1-s)n}$

Property X

Det.	Nondet.	Co-Nondet.	X	Example
T	$T^{1-\varepsilon}$	T	yes	CNF-SAT
T	T	$T^{1-\varepsilon}$	yes	DNF-TAUT
T	T	T	yes	Exact-Max-SAT
T	$T^{1-\varepsilon}$	$T^{1-\varepsilon}$	no	3-SUM

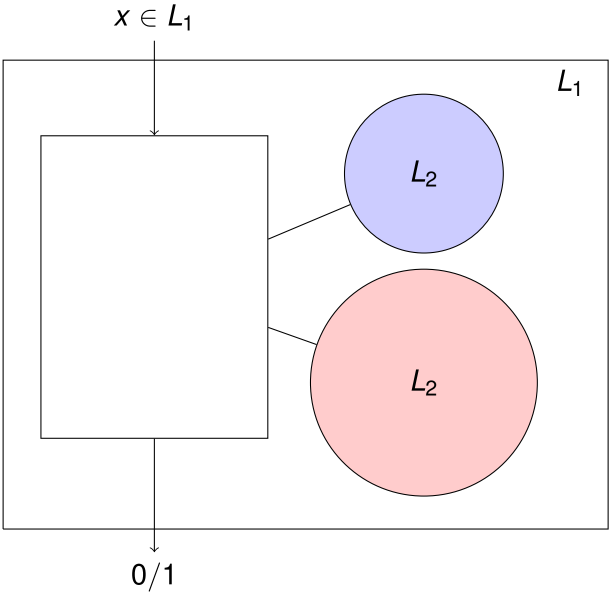
Property X

Lemma

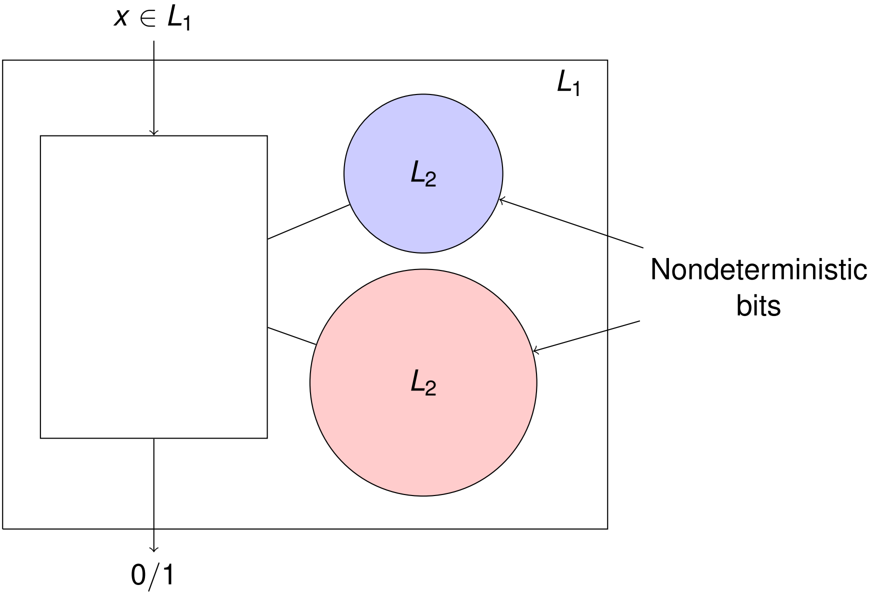
Assuming NSETH, any problem that is SETH-hard with time T under deterministic reductions either

- *Cannot be solved in nondeterministic time $T^{(1-s)}$*
- *Cannot be solved in co-nondeterministic time $T^{(1-s)}$*

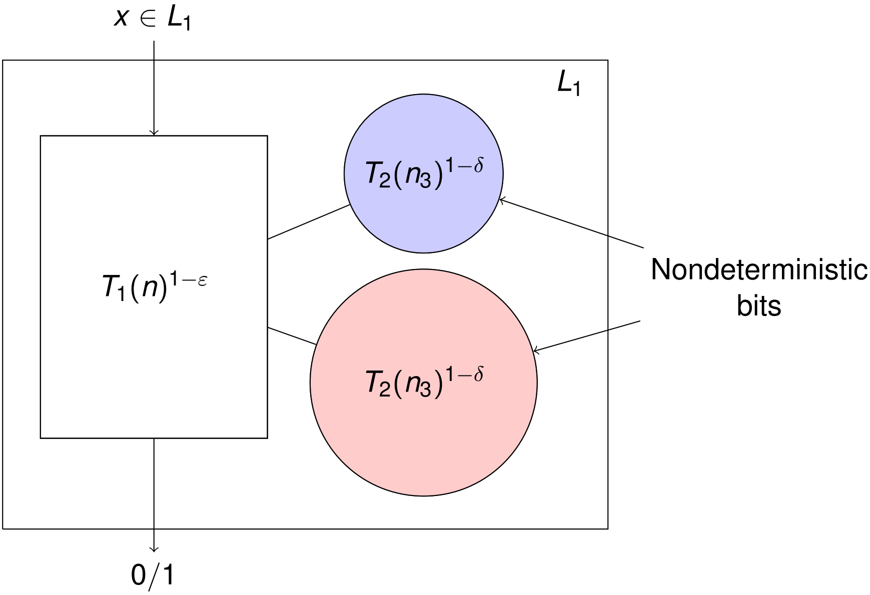
Property X



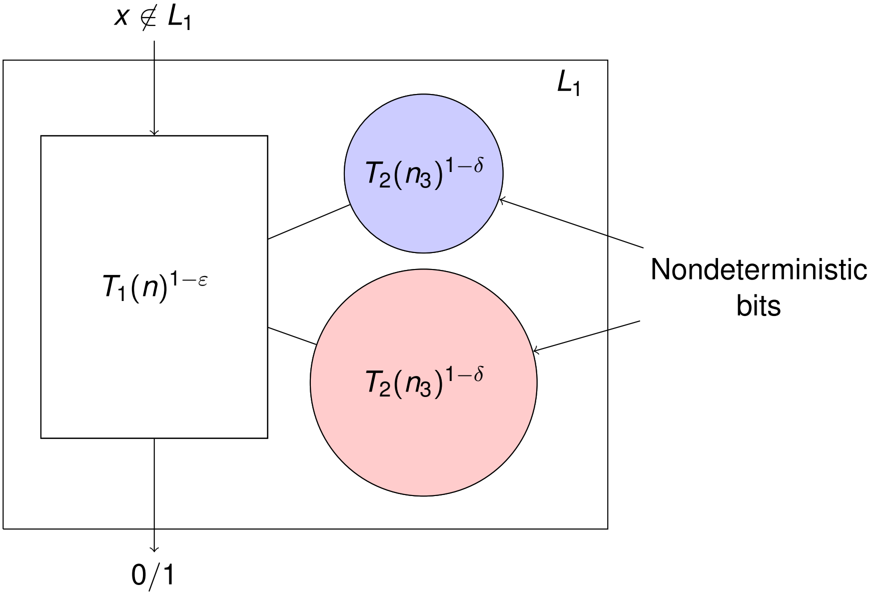
Property X



Property X


$x \in L_1$
L_1
$T_2(n_3)^{1-\delta}$
$T_1(n)^{1-\varepsilon}$
Nondeterministic bits
$T_2(n_3)^{1-\delta}$
$0/1$

Property X


$x \notin L_1$
L_1
$T_2(n_3)^{1-\delta}$
$T_1(n)^{1-\varepsilon}$
Nondeterministic bits
$T_2(n_3)^{1-\delta}$
$0/1$

Outline

- $\neg$NSETH implies interesting circuit lower bounds

Outline

- $\neg$NSETH implies interesting circuit lower bounds
- Problems with fast nondeterministic and co-nondeterministic algorithms ($\neg X$)

Outline

- $\neg$NSETH implies interesting circuit lower bounds
- Problems with fast nondeterministic and co-nondeterministic algorithms ($\neg X$)
- First-order graph properties

$\neg$SETH and $\neg$NSETH Imply Circuit Lower Bounds

The work of [JVM13] uses a two-part strategy to show $\neg\text{SETH} \implies$ Circuit Lower Bounds:

1. A tight implication from $\mathcal{C}$-CKT-SAT algorithms to $\mathcal{C}$ lower bounds
2. Decomposition of $\mathcal{C}$-circuits into $\bigvee$ *CNF* form

"$\neg\text{NSETH} \implies$ Circuit Lower Bounds" is implicit in [JVM13], following from their technical contributions and the proofs of [Williams 2013].

¬SETH and ¬NSETH are Algorithmic

- ETH is false: for every $\epsilon > 0$, 3-SAT is in time $2^{\epsilon n}$

¬SETH and ¬NSETH are Algorithmic

- ETH is false: for every $\epsilon > 0$, 3-SAT is in time $2^{\epsilon n}$
- SETH is false: there is a $\delta < 1$ such that for every k, k-SAT is in time $2^{\delta n}$

¬SETH and ¬NSETH are Algorithmic

- ETH is false: for every $\epsilon > 0$, 3-SAT is in time $2^{\epsilon n}$
- SETH is false: there is a $\delta < 1$ such that for every k, k-SAT is in time $2^{\delta n}$
- NETH is false: for every $\epsilon > 0$, 3-TAUT is in nondeterministic time $2^{\epsilon n}$

¬SETH and ¬NSETH are Algorithmic

- ETH is false: for every $\epsilon > 0$, 3-SAT is in time $2^{\epsilon n}$
- SETH is false: there is a $\delta < 1$ such that for every k, k-SAT is in time $2^{\delta n}$
- NETH is false: for every $\epsilon > 0$, 3-TAUT is in nondeterministic time $2^{\epsilon n}$
- NSETH is false: there is a $\delta < 1$ such that for every k, k-TAUT is in nondeterministic time $2^{\delta n}$

Reducing $\mathcal{C}$-CKT-SAT to k-SAT?

For $\mathcal{C}$:

- Linear-size circuits
- Linear-size series-parallel circuits

There are decompositions: $\forall C \in \mathcal{C}$, we have $C = \bigvee CNF_k$.
Execute the faster k-SAT algorithm on each "leaf" of the decomposition.

Outline

- The following problems have fast nondeterministic and co-nondeterministic algorithms:

- The following problems have fast nondeterministic and co-nondeterministic algorithms:
  - Max-Flow ($O(m)$)

Outline

- The following problems have fast nondeterministic and co-nondeterministic algorithms:
  - Max-Flow ($O(m)$)
  - Min-Cost Max Flow ($O(m)$)

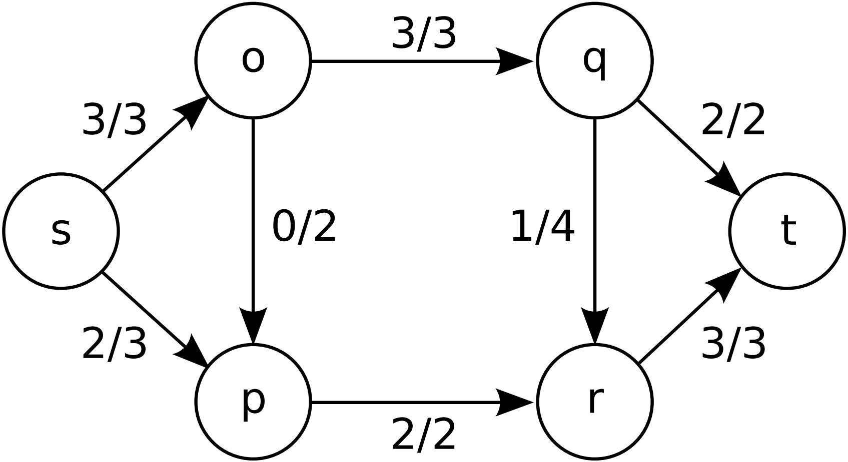
Outline

- The following problems have fast nondeterministic and co-nondeterministic algorithms:
  - Max-Flow ($O(m)$)
  - Min-Cost Max Flow ($O(m)$)
  - 3-SUM ($\tilde{O}(n^{3/2})$)

Outline

- The following problems have fast nondeterministic and co-nondeterministic algorithms:
  - Max-Flow ($O(m)$)
  - Min-Cost Max Flow ($O(m)$)
  - 3-SUM ($\tilde{O}(n^{3/2})$)
  - All-Pairs Shortest Path ($\tilde{O}(n^{2.69})$)

Maximum Flow Problem



1

[1] Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Max_flow.svg#/media/File:Max_flow.svg

Maximum Flow Problem

- Is there a flow of value at least k

Maximum Flow Problem

- Is there a flow of value at least k
- Fastest known algorithm: $\tilde{O}(mn)$ [Orlin13]

Maximum Flow Problem

- Is there a flow of value at least k
- Fastest known algorithm: $\tilde{O}(mn)$ [Orlin13]
- $\tilde{O}(m)$ approximation algorithms for undirected case [KLOS14]

Maximum Flow Problem

- Is there a flow of value at least k
- Fastest known algorithm: $\tilde{O}(mn)$ [Orlin13]
- $\tilde{O}(m)$ approximation algorithms for undirected case [KLOS14]
- Can we prove a $\tilde{\Omega}(mn)$ lower bound under SETH?

Maximum Flow Problem

- There is a $O(m)$ nondeterministic algorithm

Maximum Flow Problem

- There is a $O(m)$ nondeterministic algorithm
- There is a $O(m)$ co-nondeterministic algorithm (Min-cut/Max-flow theorem)

Maximum Flow Problem

- There is a $O(m)$ nondeterministic algorithm
- There is a $O(m)$ co-nondeterministic algorithm (Min-cut/Max-flow theorem)
- Max-Flow is not SETH-hard at time $O(m^{1+\varepsilon})$ under deterministic reductions (assuming NSETH)

Maximum Flow Problem

- There is a $O(m)$ nondeterministic algorithm
- There is a $O(m)$ co-nondeterministic algorithm (Min-cut/Max-flow theorem)
- Max-Flow is not SETH-hard at time $O(m^{1+\varepsilon})$ under deterministic reductions (assuming NSETH)
- Disproving this statement implies new lower bounds for linear size circuits

Min-Cost Maximum Flow Problem

- Edges have a capacity and a cost per unit flow

Min-Cost Maximum Flow Problem

- Edges have a capacity and a cost per unit flow
- Is there a flow of value more than k or of value k and cost at most c?

Min-Cost Maximum Flow Problem

- Edges have a capacity and a cost per unit flow
- Is there a flow of value more than k or of value k and cost at most c?
- Fastest algorithm: $\tilde{O}(m^2)$ [Orlin88]

Min-Cost Maximum Flow Problem

- Edges have a capacity and a cost per unit flow
- Is there a flow of value more than k or of value k and cost at most c?
- Fastest algorithm: $\tilde{O}(m^2)$ [Orlin88]
- Special cases: Max-Flow, Min-Cost Perfect Bipartite Matching

Min-Cost Maximum Flow Problem

- Simple $O(m)$ nondeterministic algorithm

Min-Cost Maximum Flow Problem

- Simple $O(m)$ nondeterministic algorithm
- $O(m)$ co-nondeterministic algorithm based on Klein's cycle canceling algorithm:

Min-Cost Maximum Flow Problem

- Simple $O(m)$ nondeterministic algorithm
- $O(m)$ co-nondeterministic algorithm based on Klein's cycle canceling algorithm:
  - There is a flow of the same value and smaller cost if and only if there is a negative cost cycle in the residual graph

Min-Cost Maximum Flow Problem

- Simple $O(m)$ nondeterministic algorithm
- $O(m)$ co-nondeterministic algorithm based on Klein's cycle canceling algorithm:
  - There is a flow of the same value and smaller cost if and only if there is a negative cost cycle in the residual graph
  - Co-Nondeterministic $O(m)$ algorithm for negative cycles

Certifying Negative Cycles

- Co-nondeterministic algorithm follows from properties Bellman-Ford algorithm

Certifying Negative Cycles

- Co-nondeterministic algorithm follows from properties Bellman-Ford algorithm
- Potential: $p : V \to \mathbb{R}$ such that for all $(u, v) \in E p(u) + l(u, v) \geq p(v)$

Certifying Negative Cycles

- Co-nondeterministic algorithm follows from properties Bellman-Ford algorithm
- Potential: $p : V \to \mathbb{R}$ such that for all $(u, v) \in E p(u) + l(u, v) \geq p(v)$
- There is a negative weight cycle if and only if there is no potential

Min-Cost Maximum Flow Problem

- Nondeterministically guess the min-cost max flow

Min-Cost Maximum Flow Problem

- Nondeterministically guess the min-cost max flow
- Certify that it is a max flow by guessing a cut

Min-Cost Maximum Flow Problem

- Nondeterministically guess the min-cost max flow
- Certify that it is a max flow by guessing a cut
- Certify that it is minimum cost by guessing a potential

Min-Cost Maximum Flow Problem

- Nondeterministically guess the min-cost max flow
- Certify that it is a max flow by guessing a cut
- Certify that it is minimum cost by guessing a potential
- Time complexity: $O(m)$

Min-Cost Maximum Flow Problem

- Min-Cost Max-Flow is not SETH-hard at time $O(m^{1+\varepsilon})$ under deterministic reductions (assuming NSETH)

Min-Cost Maximum Flow Problem

- Min-Cost Max-Flow is not SETH-hard at time $O(m^{1+\varepsilon})$ under deterministic reductions (assuming NSETH)
- Disproving this statement implies new lower bounds for linear size circuits

Co-Nondeterministic List and Count Algorithm

- Embed original problem into one that is much easier, but may have some false positive solutions

Co-Nondeterministic List and Count Algorithm

- Embed original problem into one that is much easier, but may have some false positive solutions
- Nondeterministically list all false positives

Co-Nondeterministic List and Count Algorithm

- Embed original problem into one that is much easier, but may have some false positive solutions
- Nondeterministically list all false positives
- Check that all of them are indeed false positives

Co-Nondeterministic List and Count Algorithm

- Embed original problem into one that is much easier, but may have some false positive solutions
- Nondeterministically list all false positives
- Check that all of them are indeed false positives
- Count number of solutions and check that given list is complete

3-Sum problem

- 3-Sum: Given n integers $a_1, \ldots, a_n \in [-M, M]$ find i, j, k such that $a_i + a_j + a_k = 0$

3-Sum problem

- 3-Sum: Given n integers $a_1, \ldots, a_n \in [-M, M]$ find i, j, k such that $a_i + a_j + a_k = 0$
- $M = \text{poly}(n)$

3-Sum problem

- 3-Sum: Given n integers $a_1, \ldots, a_n \in [-M, M]$ find i, j, k such that $a_i + a_j + a_k = 0$
- $M = \text{poly}(n)$
- Simple algorithm: $\tilde{O}(n^2)$

3-Sum problem

- 3-Sum: Given n integers $a_1, \ldots, a_n \in [-M, M]$ find i, j, k such that $a_i + a_j + a_k = 0$
- $M = \text{poly}(n)$
- Simple algorithm: $\tilde{O}(n^2)$
- FFT based counting algorithm: $\tilde{O}(M)$

3-Sum problem

- 3-Sum: Given n integers $a_1, \ldots, a_n \in [-M, M]$ find i, j, k such that $a_i + a_j + a_k = 0$
- $M = \text{poly}(n)$
- Simple algorithm: $\tilde{O}(n^2)$
- FFT based counting algorithm: $\tilde{O}(M)$
- Fast nondeterministic algorithm

List and Count Algorithm for 3-SUM

- Nondeterministically pick a prime p such that there are t solutions modulo p

List and Count Algorithm for 3-SUM

- Nondeterministically pick a prime p such that there are t solutions modulo p
- Nondeterministically guess t triples (i_q, j_q, k_q).

List and Count Algorithm for 3-SUM

- Nondeterministically pick a prime p such that there are t solutions modulo p
- Nondeterministically guess t triples (i_q, j_q, k_q).
- Check that
  $\forall q \leq t : a[i_q] + a[j_q] + a[k_q] = 0 (mod \;\; p)$,
  but
  $a[i_q] + a[j_q] + a[k_q] \neq 0$.

List and Count Algorithm for 3-SUM

- Nondeterministically pick a prime p such that there are t solutions modulo p
- Nondeterministically guess t triples (i_q, j_q, k_q).
- Check that
  $\forall q \leq t : a[i_q] + a[j_q] + a[k_q] = 0(mod \;\; p)$,
  but
  $a[i_q] + a[j_q] + a[k_q] \neq 0$.
- Count number of solutions of 3-SUM(mod p) and check that it is equal to t.

Analysis

- Nondeterministically listing all the false positive can be done in linear time: $\tilde{O}(t)$

Analysis

- Nondeterministically listing all the false positive can be done in linear time: $\tilde{O}(t)$
- Counting all the false positive can be done by FFT-based algorithm in time $\tilde{O}(p)$

Analysis

- Nondeterministically listing all the false positive can be done in linear time: $\tilde{O}(t)$
- Counting all the false positive can be done by FFT-based algorithm in time $\tilde{O}(p)$
- One can always pick $t, p = \tilde{O}(n^{3/2})$
  The running time is $\tilde{O}(n^{3/2})$

3-SUM modulo p

- Consider the first $n^{3/2}$ primes, $\leq \tilde{O}(n^{3/2})$

3-SUM modulo p

- Consider the first $n^{3/2}$ primes, $\leq \tilde{O}(n^{3/2})$
- $a_i + a_j + a_k$ has at most $\log(3M) = O(\log n)$ prime factors

3-SUM modulo p

- Consider the first $n^{3/2}$ primes, $\leq \tilde{O}(n^{3/2})$
- $a_i + a_j + a_k$ has at most $\log(3M) = O(\log n)$ prime factors
- On average, a prime p has $O\left(\frac{n^3 \log(n)}{n^{3/2}}\right) = \tilde{O}(n^{3/2})$ false positives

3-SUM modulo p

- Consider the first $n^{3/2}$ primes, $\leq \tilde{O}(n^{3/2})$
- $a_i + a_j + a_k$ has at most $\log(3M) = O(\log n)$ prime factors
- On average, a prime p has $O\left(\frac{n^3 \log(n)}{n^{3/2}}\right) = \tilde{O}(n^{3/2})$ false positives
- There is a prime $p \leq \tilde{O}(n^{3/2})$ such that $t \leq \tilde{O}(n^{3/2})$ solutions

3-SUM

- 3-SUM is not SETH-hard at time $O(n^{3/2+\varepsilon})$ under deterministic reductions (assuming NSETH)

- 3-SUM is not SETH-hard at time $O(n^{3/2+\varepsilon})$ under deterministic reductions (assuming NSETH)
- Disproving this statement implies new lower bounds for linear size series-parallel circuits

- Better co-nondeterministic for 3-SUM will give nontrivial co-nondeterministic for SUBSET SUM. This will prove that SUBSET SUM is not SETH -hard at time $O(2^{n/2})$ under deterministic reductions (assuming NSETH).

3-SUM

- Better co-nondeterministic for 3-SUM will give nontrivial co-nondeterministic for SUBSET SUM. This will prove that SUBSET SUM is not SETH -hard at time $O(2^{n/2})$ under deterministic reductions (assuming NSETH).

- Nice coincidence: decision tree complexity of 3-SUM is also $\tilde{O}(n^{3/2})$.

All-Pairs Shortest Path Problem

- APSP: Given a weighted directed graph, find the distance of every pair u, v

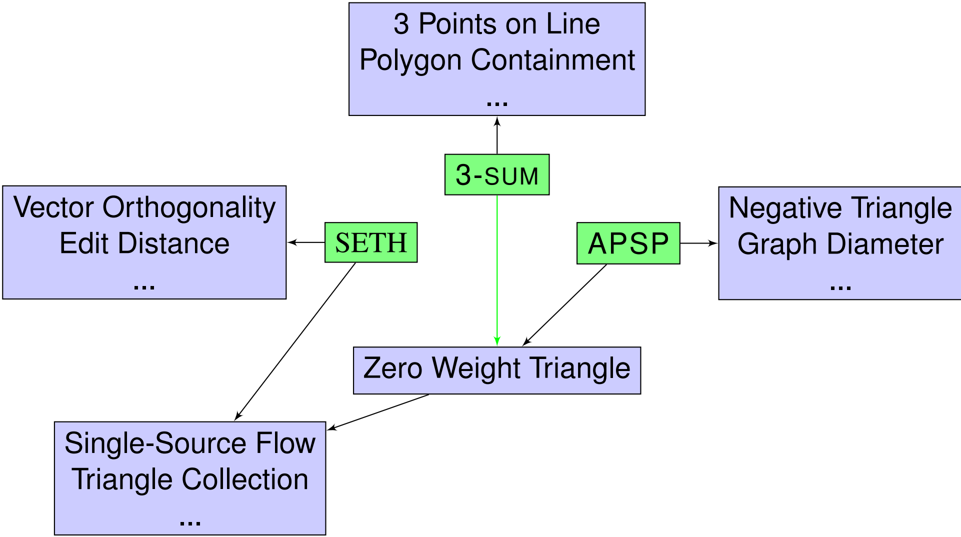
All-Pairs Shortest Path Problem

- APSP: Given a weighted directed graph, find the distance of every pair u, v
- Dynamic Programming: $O(n^3)$ [FloydWarshall62]

All-Pairs Shortest Path Problem

- APSP: Given a weighted directed graph, find the distance of every pair u, v
- Dynamic Programming: $O(n^3)$ [FloydWarshall62]
- We give a co-nondeterministic algorithm for Zero-Weight Triangle

Zero-Weight Triangle



Zero-Weight Triangle

- ZWT: Given a weighted graph with all weights $\in [-M, M]$, find a triangle of total weight equal to 0.

Zero-Weight Triangle

- ZWT: Given a weighted graph with all weights $\in [-M, M]$, find a triangle of total weight equal to 0.
- Trivial algorithm : $O(n^3)$

Zero-Weight Triangle

- ZWT: Given a weighted graph with all weights $\in [-M, M]$, find a triangle of total weight equal to 0.
- Trivial algorithm : $O(n^3)$
- ZWT modulo p in time $\tilde{O}(pn^\omega)$

List and Count Algorithm for ZWT

- Nondeterministically pick t and a prime p

List and Count Algorithm for ZWT

- Nondeterministically pick t and a prime p
- Nondeterministically guess t triples (x_i, y_i, z_i) of vertices

List and Count Algorithm for ZWT

- Nondeterministically pick t and a prime p
- Nondeterministically guess t triples (x_i, y_i, z_i) of vertices
- Check that
  $\forall i \leq t : W[i,j] + W[j,k] + W[j,i] = 0(mod\ p)$,
  but
  $W[i,j] + W[j,k] + W[j,i] \neq 0$.

List and Count Algorithm for ZWT

- Nondeterministically pick t and a prime p
- Nondeterministically guess t triples (x_i, y_i, z_i) of vertices
- Check that
  $\forall i \leq t : W[i,j] + W[j,k] + W[j,i] = 0 (mod\ p)$,
  but
  $W[i,j] + W[j,k] + W[j,i] \neq 0$.
- Count number of solutions of ZWT $(mod\ p)$ and check that it is equal to t.

Counting Solutions

- Define matrix A with $A[i,j] = x^{W[i,j]} \mod p$

Counting Solutions

- Define matrix A with $A[i,j] = x^{W[i,j]} \mod p$
- Compute A^3 in time $\tilde{O}(pn^{\omega})$

Counting Solutions

- Define matrix A with $A[i,j] = x^{W[i,j]} \mod p$
- Compute A^3 in time $\tilde{O}(pn^\omega)$
- For all entries $A^3[i,i]$ sum the coefficients of x^0, x^p, x^{2p}

Analysis

- Same argument as for 3-SUM gives $t \leq \tilde{O}(\frac{n^3}{p})$

Analysis

- Same argument as for 3-SUM gives $t \leq \tilde{O}(\frac{n^3}{p})$
- Nondeterministically listing all the false positive can be done in linear time: $\tilde{O}(t)$

Analysis

- Same argument as for 3-SUM gives $t \leq \tilde{O}(\frac{n^3}{p})$
- Nondeterministically listing all the false positive can be done in linear time: $\tilde{O}(t)$
- Counting takes time $\tilde{O}(pn^\omega)$

Analysis

- Same argument as for 3-SUM gives $t \leq \tilde{O}(\frac{n^3}{p})$
- Nondeterministically listing all the false positive can be done in linear time: $\tilde{O}(t)$
- Counting takes time $\tilde{O}(pn^\omega)$
- Pick $p = n^{0.31}$

Analysis

- Same argument as for 3-SUM gives $t \leq \tilde{O}(\frac{n^3}{p})$
- Nondeterministically listing all the false positive can be done in linear time: $\tilde{O}(t)$
- Counting takes time $\tilde{O}(pn^{\omega})$
- Pick $p = n^{0.31}$
- The running time is $O(n^{2.69})$

Analysis

- Same argument as for 3-SUM gives $t \leq \tilde{O}(\frac{n^3}{p})$
- Nondeterministically listing all the false positive can be done in linear time: $\tilde{O}(t)$
- Counting takes time $\tilde{O}(pn^{\omega})$
- Pick $p = n^{0.31}$
- The running time is $O(n^{2.69})$
- It immediately yields $O(n^{2.9})$ for APSP [VW09]

Analysis

- Same argument as for 3-SUM gives $t \leq \tilde{O}(\frac{n^3}{p})$
- Nondeterministically listing all the false positive can be done in linear time: $\tilde{O}(t)$
- Counting takes time $\tilde{O}(pn^\omega)$
- Pick $p = n^{0.31}$
- The running time is $O(n^{2.69})$
- It immediately yields $O(n^{2.9})$ for APSP [VW09]
- Nondeterministic reduction yields $O(n^{2.69})$ for APSP

- All-Pairs Shortest Path is not SETH-hard at time $O(n^{2.69+\varepsilon})$ under deterministic reductions (assuming NSETH)

All-Pairs Shortest Path

- All-Pairs Shortest Path is not SETH-hard at time $O(n^{2.69+\varepsilon})$ under deterministic reductions (assuming NSETH)
- Disproving this statement implies new lower bounds for linear size circuits

Quantifier Structures of Hard Problems

- Many SETH-hard problems have similar quantifier structures:

Quantifier Structures of Hard Problems

- Many SETH-hard problems have similar quantifier structures:
  - Orthogonal Vectors ($O(n^2)$)
    $(\exists v_1)(\exists v_2)(\forall i)\,[(v_1[i] = 0) \vee (v_2[i] = 0)]$

Quantifier Structures of Hard Problems

- Many SETH-hard problems have similar quantifier structures:
  - Orthogonal Vectors ($O(n^2)$)
    $(\exists v_1)(\exists v_2)(\forall i)\,[(v_1[i] = 0) \vee (v_2[i] = 0)]$
  - Graph k-Dominating Set ($O(n^k)$)
    $(\exists v_1)\ldots(\exists v_k)(\forall v_{k+1})\left[\bigvee_{i=1}^{k} E(v_i, v_{k+1})\right]$

Quantifier Structures of Hard Problems

- Many SETH-hard problems have similar quantifier structures:
  - Orthogonal Vectors ($O(n^2)$)
    $(\exists v_1)(\exists v_2)(\forall i)\,[(v_1[i] = 0) \vee (v_2[i] = 0)]$
  - Graph k-Dominating Set ($O(n^k)$)
    $(\exists v_1)\ldots(\exists v_k)(\forall v_{k+1})\left[\bigvee_{i=1}^{k} E(v_i, v_{k+1})\right]$
- Problems with other quantifier structures:

Quantifier Structures of Hard Problems

- Many SETH-hard problems have similar quantifier structures:
  - Orthogonal Vectors ($O(n^2)$)
    $(\exists v_1)(\exists v_2)(\forall i)\,[(v_1[i] = 0) \vee (v_2[i] = 0)]$
  - Graph k-Dominating Set ($O(n^k)$)
    $(\exists v_1)\ldots(\exists v_k)(\forall v_{k+1})\left[\bigvee_{i=1}^{k} E(v_i, v_{k+1})\right]$
- Problems with other quantifier structures:
  - k-Clique (Solvable in time $O(n^{\omega k/3})$)
    $(\exists v_1)\ldots(\exists v_k)\left[\bigwedge_{i\neq j} E(v_i, v_j)\right]$

Quantifier Structures of Hard Problems

- Many SETH-hard problems have similar quantifier structures:
  - Orthogonal Vectors ($O(n^2)$)
    $(\exists v_1)(\exists v_2)(\forall i)\,[(v_1[i] = 0) \vee (v_2[i] = 0)]$
  - Graph k-Dominating Set ($O(n^k)$)
    $(\exists v_1)\ldots(\exists v_k)(\forall v_{k+1})\left[\bigvee_{i=1}^{k} E(v_i, v_{k+1})\right]$
- Problems with other quantifier structures:
  - k-Clique (Solvable in time $O(n^{\omega k/3})$)
    $(\exists v_1)\ldots(\exists v_k)\left[\bigwedge_{i\neq j} E(v_i, v_j)\right]$
  - Hitting Set (not known to be SETH-hard)
    $(\exists H)(\forall S)(\exists x)\,[(u \in H) \wedge (u \in S)]$

Quantifier Structures of Hard Problems

- First-order formula φ with k quantifiers

$$\varphi = (\exists x_1)(Q_2 x_2)\ldots(Q_k x_k)\psi$$

Each $Q_i \in \{\exists, \forall\}$.

Quantifier Structures of Hard Problems

- First-order formula φ with k quantifiers

$$\varphi = (\exists x_1)(Q_2 x_2)\dots(Q_k x_k)\psi$$

  Each $Q_i \in \{\exists, \forall\}$.
- Model checking problem on graphs

Quantifier Structures of Hard Problems

- First-order formula φ with k quantifiers

$$\varphi = (\exists x_1)(Q_2 x_2)\dots(Q_k x_k)\psi$$

  Each $Q_i \in \{\exists, \forall\}$.
- Model checking problem on graphs
  - Input: Sparse graph G, given by its edge list of size m.

Quantifier Structures of Hard Problems

- First-order formula φ with k quantifiers

  $$\varphi = (\exists x_1)(Q_2 x_2)\dots(Q_k x_k)\psi$$

  Each $Q_i \in \{\exists, \forall\}$.
- Model checking problem on graphs
  - Input: Sparse graph G, given by its edge list of size m.
  - Output: Whether $G \models \varphi$.

Quantifier Structures of Hard Problems

- First-order formula φ with k quantifiers

$$\varphi = (\exists x_1)(Q_2 x_2)\ldots(Q_k x_k)\psi$$

  Each $Q_i \in \{\exists, \forall\}$.
- Model checking problem on graphs
  - Input: Sparse graph G, given by its edge list of size m.
  - Output: Whether $G \models \varphi$.
  - Can be done in $O(m^{k-1})$

Quantifier Structures of Hard Problems

- First-order formula φ with k quantifiers

$$\varphi = (\exists x_1)(Q_2 x_2)\dots(Q_k x_k)\psi$$

  Each $Q_i \in \{\exists, \forall\}$.
- Model checking problem on graphs
  - Input: Sparse graph G, given by its edge list of size m.
  - Output: Whether $G \models \varphi$.
  - Can be done in $O(m^{k-1})$
- Our results:

Quantifier Structures of Hard Problems

- First-order formula φ with k quantifiers

$$\varphi = (\exists x_1)(Q_2 x_2) \ldots (Q_k x_k)\psi$$

  Each $Q_i \in \{\exists, \forall\}$.
- Model checking problem on graphs
  - Input: Sparse graph G, given by its edge list of size m.
  - Output: Whether $G \models \varphi$.
  - Can be done in $O(m^{k-1})$
- Our results:
  - All SETH-hard problems have

$$Q_1 = Q_2 = \cdots = Q_{k-1} = \exists, Q_k = \forall$$

Quantifier Structures of Hard Problems

- First-order formula φ with k quantifiers

$$\varphi = (\exists x_1)(Q_2 x_2)\ldots(Q_k x_k)\psi$$

  Each $Q_i \in \{\exists, \forall\}$.
- Model checking problem on graphs
  - Input: Sparse graph G, given by its edge list of size m.
  - Output: Whether $G \models \varphi$.
  - Can be done in $O(m^{k-1})$
- Our results:
  - All SETH-hard problems have

$$Q_1 = Q_2 = \cdots = Q_{k-1} = \exists, Q_k = \forall$$

  - If NSETH holds, there is no reduction from this quantifier structure to other quantifier structures.

Quantifier Structures of Hard Problems

$$\varphi = (\exists x_1)(Q_2 x_2) \ldots (Q_k x_k)\psi$$

Quantifier structure	**Result**	**Hardness**
$\exists \ldots \exists\forall$	If solvable in $O(m^{k-1-\epsilon})$ co-nondeterministic time, then NSETH is false.	SETH-hard
All k quantifiers are $\exists$'s	solvable in $O(m^{k-1.5})$ time	Easy
More than one $\forall$'s	faster co-nondeterministic algorithms	Not SETH-hard under NSETH
Exactly one $\forall$, but not at Q_k	faster co-nondeterministic algorithms	Not SETH-hard under NSETH

Example: Hitting Set

- Decide if $\exists S \forall T \exists x\, (x \in S \land x \in T)$

Example: Hitting Set

- Decide if $\exists S \forall T \exists x\,(x \in S \wedge x \in T)$
- Exactly one $\forall$ quantifier, but not at Q_k.

Example: Hitting Set

- Decide if $\exists S \forall T \exists x \, (x \in S \wedge x \in T)$
- Exactly one $\forall$ quantifier, but not at Q_k.
- $O(m)$ nondeterministic algorithm

Example: Hitting Set

- Decide if $\exists S \forall T \exists x \, (x \in S \wedge x \in T)$
- Exactly one $\forall$ quantifier, but not at Q_k.
- $O(m)$ nondeterministic algorithm
  - Guess S, enumerate T, guess x.

Example: Hitting Set

- Decide if $\exists S \forall T \exists x \, (x \in S \wedge x \in T)$
- Exactly one $\forall$ quantifier, but not at Q_k.
- $O(m)$ nondeterministic algorithm
  - Guess S, enumerate T, guess x.
- $O(m)$ co-nondeterministic algorithm

Example: Hitting Set

- Decide if $\exists S \forall T \exists x \, (x \in S \wedge x \in T)$
- Exactly one $\forall$ quantifier, but not at Q_k.
- $O(m)$ nondeterministic algorithm
  - Guess S, enumerate T, guess x.
- $O(m)$ co-nondeterministic algorithm
  - Decide if $\forall S \exists T \forall x \, (x \notin S \vee x \notin T)$ nondeterministically

Example: Hitting Set

- Decide if $\exists S \forall T \exists x\,(x \in S \wedge x \in T)$
- Exactly one $\forall$ quantifier, but not at Q_k.
- $O(m)$ nondeterministic algorithm
  - Guess S, enumerate T, guess x.
- $O(m)$ co-nondeterministic algorithm
  - Decide if $\forall S \exists T \forall x\,(x \notin S \vee x \notin T)$ nondeterministically
  - For each S, guess T, for each $(x \in S)$, check if $(x \in T)$

Example: Hitting Set

- Decide if $\exists S \forall T \exists x \, (x \in S \wedge x \in T)$
- Exactly one $\forall$ quantifier, but not at Q_k.
- $O(m)$ nondeterministic algorithm
  - Guess S, enumerate T, guess x.
- $O(m)$ co-nondeterministic algorithm
  - Decide if $\forall S \exists T \forall x \, (x \notin S \vee x \notin T)$ nondeterministically
  - For each S, guess T, for each $(x \in S)$, check if $(x \in T)$
  - If none element x in S is also in T, accept.

Example: Hitting Set

- Decide if $\exists S \forall T \exists x \, (x \in S \wedge x \in T)$
- Exactly one $\forall$ quantifier, but not at Q_k.
- $O(m)$ nondeterministic algorithm
  - Guess S, enumerate T, guess x.
- $O(m)$ co-nondeterministic algorithm
  - Decide if $\forall S \exists T \forall x \, (x \notin S \vee x \notin T)$ nondeterministically
  - For each S, guess T, for each $(x \in S)$, check if $(x \in T)$
  - If none element x in S is also in T, accept.
  - Otherwise, reject.

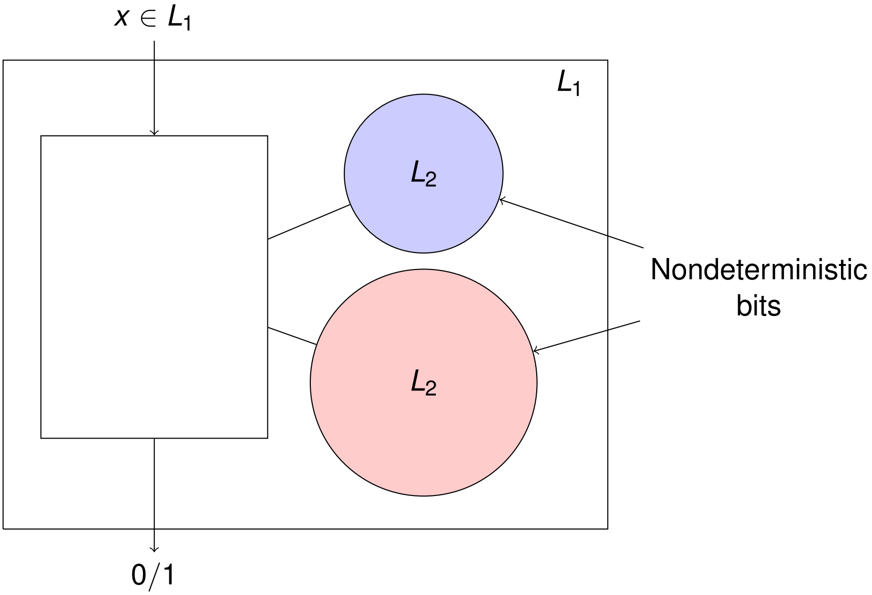
Example: Hitting Set

- Decide if $\exists S \forall T \exists x\, (x \in S \wedge x \in T)$
- Exactly one $\forall$ quantifier, but not at Q_k.
- $O(m)$ nondeterministic algorithm
  - Guess S, enumerate T, guess x.
- $O(m)$ co-nondeterministic algorithm
  - Decide if $\forall S \exists T \forall x\, (x \notin S \vee x \notin T)$ nondeterministically
  - For each S, guess T, for each $(x \in S)$, check if $(x \in T)$
  - If none element x in S is also in T, accept.
  - Otherwise, reject.
- Hitting Set $\leq_{FGR}$ Orthogonal Vectors [AVWW16]

Example: Hitting Set

- Decide if $\exists S \forall T \exists x \, (x \in S \land x \in T)$
- Exactly one $\forall$ quantifier, but not at Q_k.
- $O(m)$ nondeterministic algorithm
  - Guess S, enumerate T, guess x.
- $O(m)$ co-nondeterministic algorithm
  - Decide if $\forall S \exists T \forall x \, (x \notin S \lor x \notin T)$ nondeterministically
  - For each S, guess T, for each $(x \in S)$, check if $(x \in T)$
  - If none element x in S is also in T, accept.
  - Otherwise, reject.
- Hitting Set $\leq_{FGR}$ Orthogonal Vectors [AVWW16]
- Orthogonal Vectors $\not\leq_{FGR}$ Hitting Set, under NSETH.

Randomized Reductions



Randomized Reductions

- The argument does not extend to randomized reductions

Randomized Reductions

- The argument does not extend to randomized reductions
- Argument only gives a fast Merlin-Arthur algorithm for SAT

Randomized Reductions

- The argument does not extend to randomized reductions
- Argument only gives a fast Merlin-Arthur algorithm for SAT
- MASETH?

Randomized Reductions

- The argument does not extend to randomized reductions
- Argument only gives a fast Merlin-Arthur algorithm for SAT
- MASETH?
- There is a $\tilde{O}(2^{n/2})$ MA algorithm for CNF-SAT [Williams]

Randomized Reductions

- The argument does not extend to randomized reductions
- Argument only gives a fast Merlin-Arthur algorithm for SAT
- MASETH?
- There is a $\tilde{O}(2^{n/2})$ MA algorithm for CNF-SAT [Williams]
- Zero-error reductions are ok

- Idea: Consider a stronger hypothesis that rules out randomized reductions

Randomized Reductions

- Idea: Consider a stronger hypothesis that rules out randomized reductions

Non-Uniform Nondeterministic Strong Exponential Time Hypothesis

For every $s > 0$, there is a k such that k-SAT does not have $2^{(1-s)n}$ size nondeterministic circuits

Summary

- Introduced Nondeterministic Strong Exponential Time Hypothesis

Summary

- Introduced Nondeterministic Strong Exponential Time Hypothesis
- If NSETH is false, then new circuit lower bounds follow

Summary

- Introduced Nondeterministic Strong Exponential Time Hypothesis
- If NSETH is false, then new circuit lower bounds follow
- If NSETH is true, then non-reducibility results follow

Summary

- Introduced Nondeterministic Strong Exponential Time Hypothesis
- If NSETH is false, then new circuit lower bounds follow
- If NSETH is true, then non-reducibility results follow
- If there is a deterministic fine-grained reduction from vector orthogonality to hitting set, then we have new lower bounds for linear size circuits

Open Questions

- Deal with randomized reductions

Open Questions

- Deal with randomized reductions
- Find exponential time problems not hard under SETH

Open Questions

- Deal with randomized reductions
- Find exponential time problems not hard under SETH
- Adapt the framework to dynamic problems

Open Questions

- Deal with randomized reductions
- Find exponential time problems not hard under SETH
- Adapt the framework to dynamic problems
- Consequences of NETH

Open Questions

- Deal with randomized reductions
- Find exponential time problems not hard under SETH
- Adapt the framework to dynamic problems
- Consequences of NETH
- Every APSP-hard problem has property X, CNFSAT and 3-SUM do not

Thank You!