ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Bachelor Project

Lower bounds in P based on popular conjectures

Author:

Zeineb Sahnoun

Under the direction of:

Professor Michael Kapralov

Spring 2018

Contents

Introduction

In general, for problems that deal with large inputs, an efficient solution would ideally run in linear time on the size of the input. For those kind of problems a polynomial time algorithm is not always practical. Indeed, in practice even n^3 or n^2 algorithms are often impractical on realistic sizes of problems. Hence, it is convinient to look at the lower bounds we get under certain conjectures to convince ourselves that a better solution may not be possible.

In theoretical computer science, we don't know how to prove any lower bounds unconditionally. We can prove under certain hypothesis, that an algorithm is as hard as something else. We generally use largely agreed upon conjectures of which we present the strong exponential time (SETH), the triangle detection (TRIANGLE), and the boolean matrix multiplication (BMM) in what follows.

We are going to present an overview of some already known lower bounds in P of exact and approximate problems and the reductions that were used to prove these bounds.

Chapter 1

Conjectures used

The Strong Exponential Time Hypothesis (SETH) k-SAT is the problem of determining if there exists an assignment to the variables of a k-CNF formula (one that has k literals on each clause) which sets the formula to True. The exponential time hypothesis is an unproven computational hardness assumption which states that k-SAT cannot be solved in subexponential time in the worst case.

Conjecture 1.1. *(SETH) For every $\epsilon > 0$, there exists a k, such that SAT on k-CNF formulas on n variables cannot be solved in time $O(2^{(1-\varepsilon)n})$*

Lemma 1.1. *(Sparsification Lemma) If there is an $\epsilon > 0$ such that for all $k \geq 3$ we can solve k-SAT on n variables and $c_{k,\varepsilon} \cdot n$ clauses in $O(2^{(1-\varepsilon)n})$ time, then SETH is false. [6]*

Triangle detection (TRIANGLE) The best known algorithm for triangle detection in a graph relies on fast matrix multiplication and runs in time $O(min\{m^{1.41}, n^w\})$ in an m-edge, n-node graph. The lack of alternative algorithms has lead to the conjecture that there may not be a linear time algorithm for triangle finding.

Conjecture 1.2. *There is a constant $\delta > 0$, such that any algorithm requires $m^{1+\delta-o(1)}$ time in expectation to detect whether an m edge graph contains a triangle.*

Boolean Matrix Multiplication (BMM) The boolean product of two $n \times n$ boolean matrices A and B is the matrix C with entries $C_{i,j} = \vee_k(A_{i,k} \wedge B_{k,j})$. We can compute the boolean product of two boolean matrices by treating them as integer matrices and applying fast matrix multiplication algorithms. The most efficient of which runs in $O(n^{2.373})$ operations.

The problem with such algebraic algorithms is that they have high constant factors making them provide an advantage only for matrices so large that cannot be processed by modern hardware. The current best combinatorial algorithm for BMM runs in $\hat{O}(\frac{n^3}{\log^4 n})$ time [3]. Finding a matrix multiplication problem that is both efficient in theory and practice is still an open problem.

Conjecture 1.3. *(No truly subcubic combinatorial BMM) Any combinatorial algorithm requires $n^{3-o(1)}$ time in expectation to compute the boolean product of two $n \times n$ matrices.*

Remark 1.1. (Relationship between BMM and Triangle detection) Any $O(n^{3-o(1)})$ combinatorial algorithm for finding a triangle can be converted to an $O(n^{3-o(1)})$ combinatorial algorithm for BMM [4]. Hence **Conjecture 1.3** is equivalent to saying "there exists no combinatorial triangle finding algorithm in n node graphs that runs in subcubic time".

Chapter 2

Strong lower bounds in P for dynamic problems

2.1 Preliminaries

Many static algorithmic problems have meaningful dynamic versions (stated in terms of changing input). This is in particular the case for the *maximum bipartite matching problem* and the *single source reachability problem* that we consider below.

Maximum Cardinality Bipartite Matching problem

Definition 2.1. Let $G = (E, V)$ be an undirected graph. A matching in G is a subset of edges $M \subseteq E$ such that at most one edge is incident to each vertex in V. (Edges $e \in M$ have no common vertices at endpoints).

- A *maximal matching* is a matching in G that covers one of the end-points of each edge $e \in E$ of the graph.
- A *maximum matching* on the other hand is a maximal matching which has the largest possible number of edges.
- A *perfect matching is a* maximum matching which covers all vertices of a graph.
- An *alternating path* is a path in which the edges belong alternatively to the matching and not to the matching.
- An *augmenting path* is an alternating path that starts from and ends on non matched vertices.

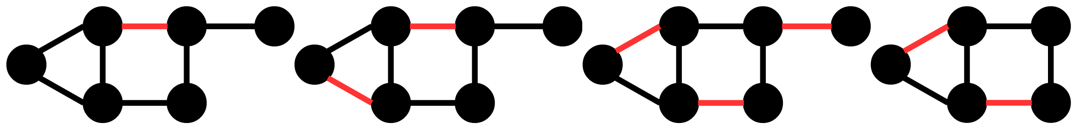
Figure 2.1: From left to right: Matching, Maximal matching, Perfect matching, Maximum matching

Definition 2.2. A *bipartite graph* is an undirected graph whose vertices can be divided into two disjoint and independent sets U and V such that every edge $e \in E$ connects a vertex in U to one in V .

The static maximum bipartite matching problem can be treated as a network flow problem and solved using Ford-Fulkerson in $O(Ef)$ running time where f is the maximum flow of the graph. An improvement is the Hopcroft-Karp algorithm which is guaranteed to terminate on any instance of the graph and has running time $O(\sqrt{V}E)$ independent of the flow. Mucha and Sankowski improved this result for dense enough graphs by giving an $\tilde{O}(V^{w})^3$ time algorithm where $\omega < 2.373$ is the matrix multiplication exponent.

We consider the dynamic version of the maximum bipartite matching problem. Any update can genrate at most one augmenting path. A trivial approach consists in finding the augmenting path in $O(E)$ time using BFS. The best known algorithm, by Sankowski uses fast matrix multiplication and runs in $\mathcal{O}(\mathcal{V}^{1.495})$ which only improves upon the trivial approach for sufficiently dense graphs.

Single Source Reachability problem (ss-reach)

Definition 2.3. In a directed graph $G = (E, V)$, with vertex set V and edge set E, a vertex $s \in V$ can reach a vertex $t \in V$ (or t is reachable from s) if there exists a sequence of vertices $v0 = s, v1, v2, \ldots, vk = t$ such that edges $(v_{i-1}, v_i) \in E$ for all $1 \leq i \leq k$.

In the static setting, linear time algorithms such that breadth first search or iteratice deepening depth first search can be used to solve the single source reachability problem.

In the dynamic setting, the trivial algorithm that recomputes the reachability after each update or at each query is still the best known in the case of sparse graphs. For dense enough graphs, Sankowski obtained update time $O(V^{1.495})$ and query time $O(1)$ for st-Reach and $O(V^{1.495})$ query time for ss-Reach. His results rely on the fast matrix multiplication as it's the case for the BP-Match problem.

Questions arise The fast matrix multiplication algorithm is theoretically efficient but uses mathemathical machinery which makes it have high constant factors making it impractical in real life. The lack of efficient dynamic algorithms for BP-Match and ss-Reach gives rise to these two questions :

- Is the use of fast matrix multiplication inherent for solving dynamic BPMatch and ss-Reach?
- Can we find a combinatorial algorithm which improves upon the $O(E)$ trivial approaches ?

In what follows, we will be interested in dynamic *bipartite perfect matchings* (BPMatch) and dynamic *reachability between two fixed vertices* (st-Reach). We will answer the questions above going through reductions from the conjectures introduced to get strong lower bounds on the running time of BP-Match and st-Reach.

2.2 Lower bounds for BPMatch and st-Reach

We first reduce the dynamic bipartite perfect matching problem (BPMatch) to the dynamic reachability problem between two fixed vertices (st-Reach) $st - Reach \leq_p BPMatch$.

Theorem 2.1. *If fully dynamic BPMatch can be solved in preprocessing, update and query times $p(n,m), u(n,m), q(n,m)$ respectively then fully dynamic st- Reach can also be solved with preprocessing, update and query times $p(O(n),O(m)), u(O(n),O(m)), q(O(n),O(m))$.*

Proof. Given a directed graph $G = (E_G, V_G)$ with $|E_G| = n$ nodes, $|V_G| = m$ edges and $s, t \in V_G$, we create an undirected bipartite graph $H = (E_H, V_H)$, in which there exists a perfect matching iff there is a path from s to t in G.

Vertices of the new graph $H = (E_H, V_H)$ are made of two copies of the vertices of G, as follows: $V_H = V_{in} \cup V_{out}$ where $V_{in} = \{v_{in} | v \in V_G, v \neq s\}$ and $V_{out} = \{v_{out} | v \in V_G, v \neq t\}$.

Edges E_H are defined as follows: $\forall (u,v) \in E_G$ where $u \neq s, v \neq t$, we create an edge $(u_{out}, v_{in}) \in E_H$. We also create an edge $(u_{in}, u_{out}) \in E_H$, between the two copies of each node $u \neq s, t$. Hence $E_H = \{(u_{out}, v_{in}) | (u,v) \in E_G\} \cup \{(u_{in}, u_{out}) | u \in V_G, u \neq s, t\}$. And we have $|E_H| = O(n)$ nodes, $|V_H| = O(m)$ edges.

We claim: There exists a path from s to t in G iff there exists a perfect matching in H:

" $\Rightarrow$ " Assume there exists a path in G $s \to v1 \to v2 \dots vk \to t$ and consider edge set $M = \{(s_{out}, v_{1in}), (v_{1out}, v_{2in}) \dots (v_{kout}, t_{in})\} \cup \{(u_{in}, u_{out}) | u \in V_G \setminus \{s, t, v_1 \dots v_k\}\}$.

Every node of H appears exactly once in M $\Longrightarrow$ M is a perfect matching

" $\Leftarrow$ " Assume M is a perfect matching in H.

We claim $S = \{(u, v) | (u_{out}, v_{in}) \in M, u \neq v\} \subset E_G$ contains a path from s to t in G:

Define $d_{in}(v) = |\{(u, v) \in S\}|$, $d_{out}(v) = |\{(v, u) \in S\}|$ as the in-degree and out-degree "in the matching M" of some vertex v. We know $d_{out}(v), d_{in}(v) \leq 1$ since M is a matching.

For vertices s and t we have $d_{in}(s) = d_{out}(t) = 0$ since $s_{in}, t_{out} \notin V_H$ and $d_{out}(s) = d_{in}(t) = 1$ since M is a perfect matching and so s_{out}, t_{in}should have been matched.

$\forall v \neq s$, if $d_{in}(v) = 1$ then $d_{out}(v) = 1$ since v_{out} should be matched too and is not matched to v_{in} (from the construction of the set S)

$\Longrightarrow$Hence S must contain a path from s to t (along with possibly disjoint cycles). □

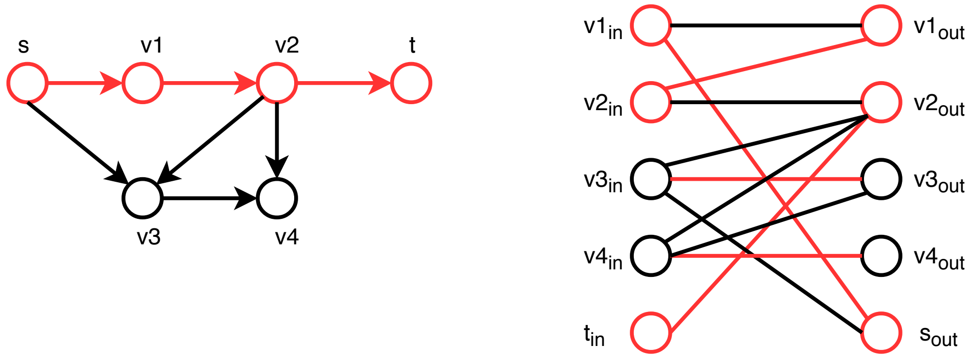

Figure 2.2: Given the directed graph on the left, we create the bipartite graph on the right as described. The st-path and the perfect matching are coloured in red. We see that vertices v3 and v4 which are not part of the st-path are matched "horizontally"($v_{3,in}$ with $v_{3,out}$ and $v_{4,in}$ with $v_{4,out}$)

We will now reduce the triangle detection problem (which is a static problem) to the dynamic st-Reach problem $Triangle \leq_p st-Reach$. For that we will define different "stages" on the graph, and at each stage we will use updates and queries from the st-Reach problem.

Theorem 2.2. *Suppose st- Reach has fully dynamic algorithm with preprocessing time p(m,n) update time u(m,n) and query time q(m,n) then triangle detection can be solved in O(n.(u(m,n)+q(m,n))+p(m,n)) time.*

Proof. Let G be the graph input of triangle detection G=(V,E).

We create a 4-partite graph H of partitions A, B, C and A' each of which containing a copy of each vertex of G (for vertex v create $v_A \in A$, $v_B \in B$, $v_C \in C$, $v_{A'} \in A'$). We also create vertex s which we will connect to vertices in the partition A at different stages and t which we connect to vertices in the partition A'.

For each edge $(u, v) \in E$, we create edges $(u_A, v_B), (u_B, v_C), (u_C, v_{A'}) \in H$

Now, we say that each node $u \in V$ has a stage. In the stage of u, we add edges (s, u_A) and s $(u_{A'}, t)$, we then run the query if t is reachable from s.

If it is the case then it means that there exists some $v_B \in B, w_C \in C$ such that (u,v), (v,w), (w,u) $\in E$ meaning that G has a triangle containing u.

If no st-path was found in the stage of u, we remove edges $(s, u_A), (u_{A'}, t)$ and we move on to a next stage.

We repeat the process at most O(n) times, and each time we perform a constant number of updates/queries. Hence the total runtime for triangle detection assuming dynamic st- Reach is : O(n(u(m,n)+ p(m,n)) + p(m,n)). □

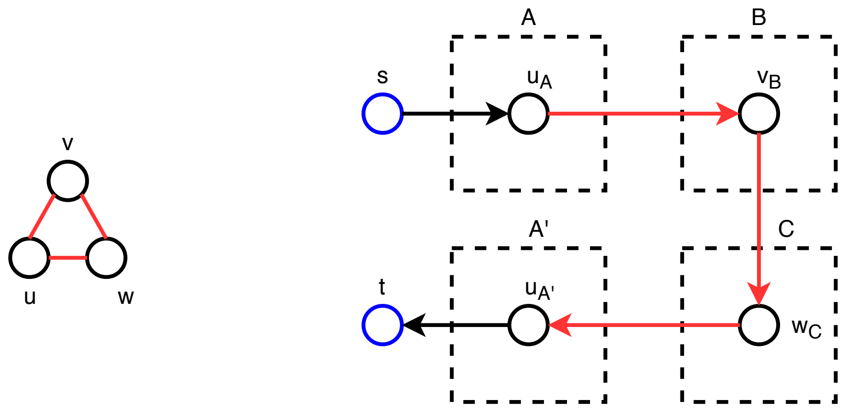

Figure 2.3: Given an undirected graph, we create a 4-partite graph as described in the proof. On the figure, the graph is represented in the stage of vertex u (we connect s to u_A and $u_{A'}$ to t).We only represented the vertices that are part of the st-path for readability.

Using **Theorem 2.2**, we can say: if for some $\varepsilon > 0$, st-reach is solvable in $O(m^{1+\delta-\varepsilon})$ processing time, update and query times $O(m^{2\delta+\varepsilon})$ then triangle detection is solvable in $O(nm^{2\delta-\varepsilon} + m^{1+\delta-\varepsilon}) = O(nm^{2\delta-\varepsilon})$. Where $\delta >$ 0 is some constant for which Triangle detection is not in $m^{1+\delta}$ time. We want to express the running time we get for triangle detection only in terms of the number of edges m of a graph, in order to be able to compare our result to the lower bound on **conjecture 1.2.** For that we use the theorem below.

Theorem 2.3. *If there is an $O(nm^{\varepsilon})$ time algorithm for triangle detection for graphs on n nodes and m edges, then there is an $O(m^{1+\varepsilon/2})$ time algorithm for triangle detection for graphs on m edges and arbitrary number of nodes.*

Proof. Suppose there exists an $O(nm^{\varepsilon})$ time algorithm for triangle detection.

Let $\triangle = m^{\epsilon/2}$. We can find any triangle that contains a node of degree at most Δ in $O(m\Delta) = O(m^{1+\varepsilon/2})$: we loop over all edges in the graph, and over all neighbors of low degree u or v as detailed in the algorithm below.

Hence in $O(m^{1+\varepsilon/2})$ we can find any triangle which has a node of degree at most Δ.

If no triangle is found, then it means any triangle only has higher degree nodes.

We can have at most $O(\frac{m}{\Delta}) = O(m^{1-\varepsilon/2})$ such nodes.

And we can find a triangle on those nodes on $O(\frac{m}{\Delta}m^{\varepsilon}) = O(m^{1+\varepsilon/2})$ time, from our first claim.

Algorithm 2.1 Detecting a triangle which contains a node of degree at most Δ

```
begin
For all edges (u,v) in $E$
 If degree(u)$\leq \Delta$
  For all w $\in$ Adj [u]
   If (w,u) $\in E$ and (w,v) $\in E$
    Return true
  end for
 Else if degree(v)$\leq \Delta$
  for all w$\in$Adj [v]
   If (w,u)$\in E$ and (w,v)$\in E$
    Return true
  end for
end for
Return false
end
```

We conclude: If there is an $O(nm^{\varepsilon})$ algorithm for triangle detection for graphs on n nodes and m edges, then there is an $O(m^{1+\varepsilon/2})$ time algorithm for triangle detection for graphs on m edges and arbitrary number of nodes. □

We are finally able to derive the lower bounds we get from our reductions:

- ***If fully dynamic st-Reach or fully dynamic BPMatch can be solved in*** $O(m^{1+\delta-o(1)})$ ***processing time, update and query times*** $O(m^{2\delta+o(1)})$ ***then it will falsify the triangle conjecture (conjecture 1.2).***

Also, we already mentioned that any $O(n^{3-o(1)})$ combinatorial algorithm for finding a triangle can be converted to an $O(n^{3-o(1)})$ combinatorial algorithm for BMM (**remark 1.1**), and since all of our reductions are combinatorial, our lower bounds become (we apply m=n^2 and we set $\delta = \frac{1-\varepsilon}{2}$ where $\delta > 0$ is some constant for which Triangle detection is not in m$^{1+\delta}$ time.):

- ***If fully dynamic st-Reach or fully dynamic BPMatch can be solved in*** $O(n^{3-o(1)})$ ***processing time, update and query times*** $O(n^{2-o(1)})$ ***then it will falsify the Boolean Matrix Multiplication conjecture (conjecture 1.3).***

Chapter 3

Hardness of approximation in P

3.1 Preliminaries

Bichromatic Maximum Inner Product problem (MAX-IP)

Definition 3.1. Given two sets A,B, each of N binary vectors in $\{0,1\}^d$, return a pair (a,b) $\in$ A $\times$ B that maximizes the inner product a $\cdot$ b.

The MAX-IP problem can be seen as finding the most correlated pair in a dataset (the pair with largest overlap).

A naive approach which runs in $O(N^2 \cdot d)$ would be to compare every pair of vectors (a,b) $\in$ A $\times$ B and to evaluate the overlap of their elements. A SETH lower bound for this problem is: Assuming SETH, we cannot solve MAX-IP in exactly $N^{2-\varepsilon} \cdot 2^{o(d)}$. [5]

As for approximate solutions, even for a subquadratic running time of $O(N^{2-\varepsilon})$, all known algorithms suffer from polynomial $N^{g(\varepsilon)}$ approximation factors. It is of interest to design approximate but near-linear time solutions for MAX-IP (for example, the goal of the locality sensitive hashing problem is to solve a version of this problem).

Questions arise

- Is there an $O(N^{1+\varepsilon})$ algorithm for computing an f(ε)- approximation to Bichromatic Max Inner Product ?

In order to solve this question, when trying to adapt the PCP Theorem (which is usually used for NP-Hardness-of-approximation frameworks) to P, we are faced with the fact that when starting from SETH conjecture, all known PCPs incur a blowup in the size of the proof ($\pi(\phi, x)$ requires n´ $\gg$ n bits). [2]

- Is there a PCP-like theorem for fine-grained complexity?

3.2 Lower bound for approximate MAX-IP

MA communication protocol for Set Disjointness

Set disjointness is very hard for randomized communication, and hard even for non-deterministic communication. We introduce a MA (Merlin-Arthur) style communication protocol for Set Disjointness with sublinear communication and near-polynomial soundness.[2]

Theorem 3.1. *For $T \le n$, there exists a computationally efficient MA-protocol for Set Disjointness in which:*

1. Merlin sends Alice O(n log n/T) bits;
2. Alice and Bob jointly toss log n + O(1) coins;
3. Bob sends Alice O(T log n) bits.
4. Alice returns Accept or Reject.

Completeness:

If the sets are disjoint, there is a message from Merlin such that Alice always accepts

Soundness:

If the sets have a non-empty intersection or Merlin sends the wrong message, Alice rejects with probability $\geq \frac{1}{2}$

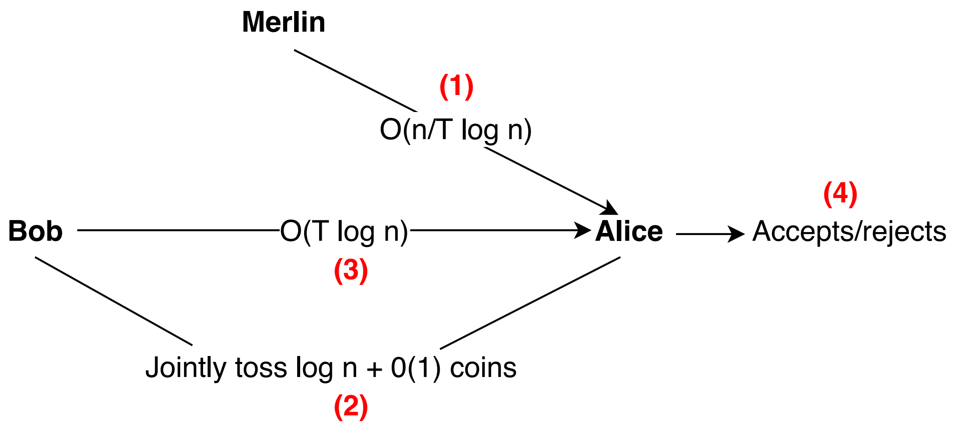

Figure 3.1: MA communication protocol

Proof. **Arithmetization**

Assume T|n.

Let q be a prime number st. 4N<q<8N.

$\mathbb{F}_q$ prime field of size q.

We identify [n]×[T] with the universe [n] over which Alice and Bob want to compute set disjointness.

Alice's input can now be represented as T functions $\psi_{\alpha,t} : [\frac{n}{T}] \rightarrow$ {0,1} such that: $\psi_{\alpha,t}(i) = 1 \Leftrightarrow$ the element corresponding to (i,t) is in Alice's set.

We define $\psi_{\beta,t}$ similarly for Bob's set.

Alice's and Bob's sets are disjoint iff $\psi_{\alpha,t}(i)\psi_{\beta,t}(i) = 0 \; \forall(i,t) \in [\frac{n}{T}] \times [T]$.

Extend $\psi_{\alpha,t}(i), \psi_{\beta,t}(i)$ to polynomials $P_{\alpha,t}, P_{\beta,t} : \mathbb{F}_q \rightarrow \mathbb{F}_q$ of degree at most $\frac{n}{T} - 1$.

Define $P_t(i) = P_{\alpha,t}(i) \cdot P_{\beta,t}(i)$ (has degree at most $2(\frac{n}{T} - 1)$) and $P(i) = \sum_{t=1}^{T} P_t(i)$ (has degree at most $2(\frac{n}{T} - 1)$).

The sets are disjoint iff P$_t$(i)=0 $\forall(i,t) \in [\frac{n}{T}] \times [T] \Leftrightarrow$ P(i)=0 $\forall i \in [\frac{n}{T}]$.

Protocol

Merlin knows $\psi_{\alpha,t}, \psi_{\beta,t} \; \forall t \in [T]$. He sends Alice a polynomial φ allegedly equal to P (degree at most $2(\frac{n}{T} - 1)$). The polynomial is uniquely specified by $2(\frac{n}{T} - 1)$ coefficients in $\mathbb{F}_q$ Each coefficient is log|$\mathbb{F}_q$|=log n Hence Merlin sends Alice O(n log n/T) bits

Alice and Bob jointly draw i∈ $\mathbb{F}_q$ uniformly at random → toss log|$\mathbb{F}_q$|=log n coins.

Bob sends Alice P$_{\beta,t}\forall t \in [T]$ (require 0(T log n) bits)

Alice Accepts iff:

$$\varphi(i) = 0 \; \forall i \in [\frac{n}{T}] \tag{3.1}$$

$$\varphi(i) = \sum_{t=1}^{T} P_{\alpha,t}(i) P_{\beta,t}(i) \tag{3.2}$$

Analysis

Completeness If the sets are disjoint, Merlin can send the true Ψ, and Alice always accepts.

Soundness If the sets are not disjoint, Merlin must send a different low degree polynomial. By the Schwartz-Zippel Lemma, since Ψ and φ have degree less than 2 $\frac{n}{T} \leq \frac{q}{2}$, if they are distinct they must differ on at least half of |$\mathbb{F}_q$| . Hence, (3.2) is false with probability ≥ 2 . □

Corollary 3.1. *There exists a computationally efficient MA-protocol for Set Disjointness s.t.:*

1. Merlin sends Alice o(n) bits;
2. Alice and Bob jointly toss o(n) coins;
3. Bob sends Alice o(n) bits.
4. Alice returns Accept or Reject.

Completeness:

If the sets are disjoint, there is a unique message from Merlin such that Alice always accepts

Soundness:

if the sets have a non-empty intersection or Merlin sends the wrong message, Alice rejects with probability $\geq 1 - (\frac{1}{2})^{n^{1-o(1)}}$.

Proof. Let T be a small super-logarithmic function of n, e.g. T = $\log^2$ n.

We repeat the protocol from Theorem 3.4 R = $\frac{n}{T^2}$ times to amplify the soundness (notice it is sufficient to only repeat steps 2-4).

Thus Merlin still sends O($\frac{n}{T}$ $\cdot$ log n) = o(n) bits, Alice and Bob toss a total of O(R $\cdot$ log n) = o(n) coins, and Bob sends a total of O(R $\cdot$ T $\cdot$ log n) = o(n) bits. □

Distributed non deterministic PCP theorem

Alice and Bob share a k-SAT formula φ.

Alice has an assignment α ∈ $\{0, 1\}^{n/2}$ to the first half of the variables, and Bob has an assignment β ∈ $\{0, 1\}^{n/2}$ to the second half.

We want a protocol where Alice locally computes a string π´(α) ∈ $\{0, 1\}^{n'}$, Bob locally computes π´(β) ∈ $\{0, 1\}^{n'}$, such that π (α; β) = (π´ (α) , π´ (β)) is a valid probabilistically checkable proof that x = (α,β) satisfies φ.

We present the distributed non deterministic PCP theorem below, which enumerates over all possible messages from Merlin and Bob. Thus incuring only a subexponential blowup in n (the number of k-SAT variables) and since N = $2^{n/2}$ it only incurs a subpolynomial blowup in N (The number of gadgets).[2]

Theorem 3.2. *Let φ be a Boolean CNF formula with n variables and $m = O(n)$ clauses. There is a non-interactive protocol where:*

- Alice has partial assignment $a \in \{0,1\}^{n/2}$, advice $\mu \in \{0,1\}^{o(n)}$

⇨ Outputs $a^{\alpha,\mu} \in \{0, 1\}^{2^{o(n)}}$

- Bob has partial assignment $\beta \in \{0, 1\}^{n/2}$

⇨ Outputs $b^{\beta} \in \{0, 1\}^{2^{o(n)}}$

- Verifier knows ϕ tasses o(n) coins, reads o(n) bits from b^{β} non adaptively, one bit from $a^{\alpha,\mu}$ adaptively

⇨ Returns accept/reject

***Completeness**: If the combined assignment (a,β) satisfies φ, there exists $\mu*$ such that the verifier always accepts.*

***Soundness**: if ϕ is unsatisfiable, $\forall \mu$, the verifier rejects with prob $\geq 1 - \frac{1}{2}^{n^{1-o(1)}}$*

Proof. $\forall$ partial assignment $\alpha, \beta \in \{0,1\}^{n/2}$, consider sets S_α, $T_\beta \subseteq [m]$ where $j \in S_\alpha$ iff none of the literals in the jth clause is set to 1 in α (either all set to 0 or not specified by the partial assignment)

The same applies for T_β.

Now, (α, β) satisfies ϕ iff $S_\alpha$and T_β are disjoint (we use **corollary 3.1**)

Constructing the PCP

Bob's PCP is the list of all messages he can send on the MA communication protocol given coin tosses (of the protocol) ($|L| \leq 2^{o(n)}$: all outcomes of Bob and Alice coin tosses)

$\forall l \in L$, let b_l^β be the message Bob sends on input T_β and coin tosses l

Alice's PCP is: $\forall l \in L$, she writes the messages she will accept from Bob

Let $|K|=2^{o(n)}$ be all possible messages of Bob

$\forall k \in K, l \in L$, μ message from Merlin, if Alice Accepts set a $a_{l,b_l^\beta}^{\alpha,\mu} = 1$ Otherwise $a_{l,b_l^\beta}^{\alpha,\mu} = 0$

Verifier chooses $l \in L$ randomly, reads b_l^β and then $a_{l,b_l^\beta}^{\alpha,\mu}$ Accepts iff $a_{l,b_l^\beta}^{\alpha,\mu} = 1$ (Alice accepts) (S_α,T_β are disjoint) (Alice and Bob partial assignments satisfy ϕ). The probability that the verifier accepts is exactly the probqbility that Alice accepts in the MA communication protocol. □

Next thing, we abstract the prover-verifier formulation by reducing the PCP to an orthogonal vectors problem : PCP-Vectors.

Definition 3.2. (PCP-Vectors). We define PCP-Vectors to be the problem which input consists of two sets of vectors A$\subset\Sigma^{L\times K}$ and B$\subset\Sigma^{L}$. The goal is to find vectors a $\in$ A and b $\in$ B that maximize s(a,b)$\triangleq$ Pr $\left[\bigvee_{k\in K}(a_{l,k}=b_l)\right]$

Theorem 3.3. *Let $\epsilon > 0$ be any constant, and let (A, B) be an instance of PCP-Vectors with N vectors and parameters $|L|,|K|,|\Sigma| = N^{o(1)}$. Then, assuming SETH, $O(N^{2-\varepsilon})$ algorithms cannot distinguish between:*

Completeness: *there exist a$*$, b$*$ such that s (a$*$, b$*$) = 1;*

Soundness: *for every a$\in$A, b$\in$B, we have s(a,b) $\leq \frac{1}{2}^{(logN)^{1-o(1)}}$.*

Proof. We want to reduce the satisfiability problem to the PCP-vectors problem.

Let ϕ b a CNF formula with n variables and m = O(n) clauses (w.l.o.g using sparsification lemma)

Let B be the set of vectors generated by Bob in the disributed non deterministic PCP (**theorem 3.5**) such that b$_l^{\beta}$ is a single element of Σ.

Since the verifier picks l$\in$ $|L|$ at random, reads b$^{\beta}{}_l$ and reads a$^{\alpha}{}_l$ and then if $a^{\alpha,\mu}_{l,b^{\beta}_l}=1$ he accepts,

Let $\hat{A}$ be the set of vectors generated by Alice such that $\hat{a}^{\alpha,\mu}_{l,b^{\beta}_l}$ is a single element of Σ.

$\forall$l,k, we modify $\hat{a}^{\alpha,\mu}_{l,b^{\beta}_l}$ as follows:

$a^{\alpha,\mu}_{l,b^{\beta}_l}=k$ if $\hat{a}^{\alpha,\mu}_{l,b^{\beta}_l}$ =1 and $a^{\alpha,\mu}_{l,b^{\beta}_l}=\perp$ otherwise.

We get that the probability that the verifier accepts is s($a^{\alpha,\mu}_{l,b^{\beta}_l}$, b$_l^{\beta}$)

So we used PCP vectors to represent the satisfiability problem with the distributed non deterministic PCP theorem. □

Using the above we are able to derive the following statement.

Theorem 3.4. *Assuming SETH, for any $\epsilon > 0$, given two collections A,B, each of N subsets of a universe [m], where $m = N^{o(1)}$ and all subsets $b \in B$ have size k, no $O(N^{2-\varepsilon})$ time algorithm can distinguish between the cases:*

Completeness *there exist $a \in A$, $b \in B$ such that $b \subseteq a$*

Soundness *for every $a \in A$, $b \in B$ we have $|a \bigcap b| \leq \frac{k}{2}^{(logN)1-o(1)}$.*

Proof. Let A´,B´ be an instance of PCP-Vectors.

We map each vector a´ $\in$ A´ $\subseteq \Sigma^{L\times K}$ to a subset a $\subseteq$ U = L $\times$ Σ in the following way:

For all l $\in$ [L], k $\in$ [K] and $\sigma \in \Sigma$ we add (l,σ) to a iff there is a k $\in$ K such that a$\acute{}_{l,k} = \sigma$.

We each vector b$\acute{}$ $\in$ B to a subset b $\subseteq$ U = L $\times$ Σ by adding the element (l,σ) to b iff b$\acute{}_{l} = \sigma$.

Note that the universe has size |U| = |L|·|Σ| = N$^{o(1)}$ and the set b has size |L|.

Now, $\forall$a'$\in$ A', b'$\in$ B', s(a',b')=$\frac{|a \cap b|}{L}$ (Probability that a row a'$_l$ selected at random from a' contains b'$_l$)

Hence, finding max[s(a',b')] reduces to finding max[a$\cap$b]

And we get : s(a' , b') = 1 $\Leftrightarrow$ | a $\cap$ b | = | L | $\Leftrightarrow$ b $\subseteq$ a

s(a' , b') $\leq \frac{1}{2}^{logN^{1-o(1)}}$ $\Leftrightarrow$ | a $\cap$ b | = | L | $\frac{1}{2}^{logN^{1-o(1)}}$ $\square$

Conclusion

As we have seen, triangle conjecture can be used to derive lower bounds in P on dynamic st-Reach and BPMatch problems. BMM conjecture also provides lower bounds on combinatorial algorithms for these dynamic problems. We also used SETH conjecture, to derive lower bounds in P on the approximate MAX-IP problem.

These are some of many other lower bounds on problems in P that can be derived using popular conjectures. So much so that today, P complexity class can be seen as a set of conjecture-based complexity subclasses (SETH-hard class, TRIANGLE-hard class...). At the same time, there still exists unclassified problems. To classify all of P, new conjectures might be needed.

On the other hand, no formal relationship is known between these conjectures. As far as we know, any subset of the above conjectures could be false, and the rest could still be true. Hence finding connections between the conjectures, or barriers for relating them, could be of interest.

Bibliography

[1] A. Abboud and V. V. Williams, Popular conjectures imply strong lower bounds for dynamic problems, 2014.

[2] A. Abboud, A. Rubinstein and R. Williams, Distributed PCP Theorems for Hardness of Approximation in P, 2017.

[3] Yu H. (2015) An Improved Combinatorial Algorithm for Boolean Matrix Multiplication. In: Halldórsson M., Iwama K., Kobayashi N., Speckmann B. (eds) Automata, Languages, and Programming. ICALP 2015. Lecture Notes in Computer Science, vol 9134. Springer, Berlin, Heidelberg

[4] V. V. Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In Proc. FOCS, pages 645–654, 2010.

[5] R. Ryan Williams, A new algorithm for optimal 2-constraint satisfaction and its implications, Theoretical Computer Science 348 (2005), no. 2–3, 357–365.

[6] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane, Which problems have strongly exponential complexity?, J. Comput. Syst. Sci. 63 (2001), no. 4, 512–530.

[7] https://people.csail.mit.edu/virgi/conclusions-amir.pdf