6.1420 Subcubic time for structured Min-Plus product; Exact Triangle and Counting Exact Triangles are equivalent

Lecture 15 (notes by V) **Date:** October 29, 2024

Our goal today is twofold:

1. Show that for some structured matrices, their Min-Plus product can be computed in truly subcubic time
2. Use similar techniques to show that Exact Triangle is equivalent to its counting version.

1 Faster Min-Plus for some matrices

We will use fast matrix multiplication to show that in some natural cases the Min-Plus product of two $n \times n$ matrices can be computed in truly subcubic time.

Recall that the Min-Plus product of two $n \times n$ matrices A and B is the $n \times n$ matrix C with $C[i,j] = \min_k A[i,k] + B[k,j]$.

The (normal) matrix product of two matrices A and B, on the other hand, is the matrix C with $C[i,j] = \sum_k (A[i,k] + B[k,j])$.

We define ω to be the smallest real number such that $n \times n$ matrices can be multiplied (in the normal $(+,\cdot)$ way) in time $O(n^{\omega+\varepsilon})$ for all $\varepsilon > 0$.

Determining the value of ω is an active research area. The current record (as of 2024) is $\omega < 2.37134$ [1].

The actual definition of ω so far ignores the sizes of the matrix entries. We will actually need to consider their bit complexity however. Thus, we will say that two integer matrices with entries in $\{-M, \dots M\}$ can be multiplied in time $O((\log M) \cdot n^{\omega+\varepsilon})$ for all $\varepsilon > 0$. Slightly abusing notation, we will from now on say:

> Any two $n \times n$ matrices with entries in $\{-M, \dots M\}$ can be multiplied in $O((\log M) \cdot n^{\omega})$ time.

In the above, you should always think of ω really being $\omega + \varepsilon$, for all $\varepsilon > 0$. We omit the ε so we don't have to keep writing it.

We will use the fact that $\omega < 3$ to solve Min-Plus product faster in some integer matrices.

The first case is when both input matrices have bounded entries. Then we will relax this to when only one has bounded entries and the second can be arbitrary.

1.1 Min-Plus product of two matrices with bounded entries

Suppose that we are given A and B with entries in $\{-M, \dots, M\} \cup \{\infty\}$ and we want to compute their Min-Plus product. Here M is some small integer, $M << n$.

The first notice that we can replace ∞ by $3M+1$, forming new matrices A' and B' with entries in $\{-M \dots, 3M+1\}$. This is because if either $A[i,k] = \infty$ or $B[k,j] = \infty$, then $A'[i,k] + B'[k,j] \geq (3M+1) - M = 2M+1$, whereas if both $A[i,k] < \infty, B[k,j] < \infty$, we have that $A'[i,k] + B'[k,j] \leq 2M$, so no previously infinite entries could mess up the Min-Plus product computation and

$$(A' \star B')[i,j] = (A \star B)[i,j] \leq 2M \text{ whenever } (A \star B)[i,j] < \infty,$$
$$\text{and } (A' \star B')[i,j] \geq 2M+1 \text{ whenever } (A \star B)[i,j] = \infty.$$

Now, let's negate all entries of A' and B' obtaining $A'' = -A, B'' = -B$. We now want to compute the **Max-Plus product** C'' of A'', B'' defined as

$$C''[i,j] = \max_k A''[i,k] + B''[k,j].$$

Notice that A'' and B'' have entries in $\{-3M-1,\ldots,M\}$. If we add $3M+1$ to all entries of A'' and B'' we will get matrices X and Y with entries in $\{0,\ldots,W\}$ for $W=4M+1$. If we can compute the Max-Plus product Z of X and Y quickly, then with only an $O(n^2)$ overhead we can obtain from Z the Max-Plus product of A'' and B'' by subtracting $2(3M+1)$ from every entry of Z. Then with an additional $O(n^2)$ overhead we can obtain the Min-Plus product of A and B.

We give a faster algorithm for this problem:

Claim 1. *The Max-Product of two $n\times n$ matrices with entries in $\{0,\ldots,W\}$ can be computed in $\tilde{O}(Wn^\omega)$ time.*

If the claim is true, we get as a corollary:

Corollary 1.1. *The Min-Plus product of two $n\times n$ matrices with entries in $\{-M,\ldots,M\}\cup\{\infty\}$ can be computed in $\tilde{O}(Mn^\omega)$.*

Let's prove the claim. Given two $n\times n$ matrices X and Y with entries in $\{0,\ldots,W\}$, let's form new matrices X' and Y' with $X'[i,k]=(n+1)^{X[i,k]}$ and $Y'[k,j]=(n+1)^{Y[k,j]}$.

Consider the (normal) matrix product Z' of X' and Y':

$$Z'[i,j]=\sum_k (n+1)^{X[i,k]+Y[k,j]}.$$

Suppose that Z is the Max-Plus product of X and Y so that $Z[i,j]=\max_k X[i,k]+Y[k,j]$.

Then:

1. As the sum is at least as large as any individual summand,
$$Z'[i,j]=\sum_k (n+1)^{X[i,k]+Y[k,j]}\geq (n+1)^{Z[i,j]}.$$
2. Clearly, for all k, $Z[i,j]\geq X[i,k]+Y[k,j]$. Thus,
$$Z'[i,j]=\sum_k (n+1)^{X[i,k]+Y[k,j]}\leq n\cdot(n+1)^{Z[i,j]}<(n+1)^{Z[i,j]+1}.$$

Hence, if we have Z', we can compute Z in only $O(n^2)$ additional time as follows. For every i,j, find the unique integer T for which $Z'[i,j]\in[(n+1)^T,(n+1)^{T+1})$.

How fast can we compute Z' from X' and Y'? The entries of X' and Y' are bounded above by $(n+1)^W$. Thus, we can compute Z' in time $O(Wn^\omega\log n)$.

Thus, the total time to compute the Max-Plus product is $O(Wn^\omega\log n)$.

1.2 Min-Plus product of two matrices where only one has bounded entries

Now let's consider the case when B is an arbitrary integer matrix and A has bounded entries. Since we want to be able to read the input in truly subcubic time, we will assume that the entries of B have at most $n^{1-\delta}$ bits for some $\delta>0$, but otherwise they are arbitrary integers in $[2^{n^{1-\delta}}]$. (Here technically we want $1-\delta\leq(\omega-1)/2$ since we are aiming for a running time of $O(n^{(3+\omega)/2})$ and the input size would be $n^{3-\delta}$.)

We will consider two cases for A. The first case is when all entries are in $\{-M,\ldots M\}$. In the second case we will also allow infinite entries (which could signify the absence of an edge, for instance, if A is a generalized adjacency matrix).

A has entries in $\{-M, \ldots, M\}$. We want to compute the Min-Plus product C of A and B. Let's fix some output index pair i, j. Suppose that k is a *witness* for the Min-Plus product entry $C[i, j]$, i.e. $A[i, k]+B[k, j] = C[i, j]$.

Then, for all ℓ, $A[i, k] + B[k, j] \leq A[i, \ell] + B[\ell, j]$. We get that

$$B[k, j] \leq (A[i, \ell] - A[i, k]) + B[\ell, j].$$

Since $A[i, k], A[i, \ell] \in \{-M, \ldots, M\}$, we obtain that for the witness k we have that for all ℓ:

$$B[k, j] \leq 2M + B[\ell, j].$$

Let's sort all columns of B in $O(n^2 \log n)$ time. Now, consider the jth column in light of the above. Let $B[s_j, j]$ be the minimum entry in the jth column of B. From our discussion above, we get:

For any $i \in [n]$ and k which is a witness of the i, j Min-Plus product entry, $B[k, j] \leq 2M + B[s_j, j]$.

Now, form the matrix B' as follows:

$$B'[\ell, j] = \begin{cases} B[\ell, j] - B[s_j, j], & \text{if } B[\ell, j] - B[s_j, j] \leq 2M \\ \infty, & \text{otherwise.} \end{cases}$$

The claim is that $A \star B' = A \star B$. This is because, for every i, j if $(A \star B)[i, j] = A[i, k] + B[k, j]$, by our discussion above, $B[k, j] \leq 2M + B[s_j, j]$. Thus, $B[k, j] - B[s_j, j] \leq 2M$ and $B'[k, j] = B[k, j] - B[s_j, j]$.

Let S_j be the indices ℓ for which $B[\ell, j] - B[s_j, j] \leq 2M$. We have shown that the witness k for the Min-Plus product entry $C[i, j]$ is in S_j. Since for all $\ell \in S_j$, $A[i, \ell] + B[\ell, j] \geq A[i, k] + B[k, j] = C[i, j]$:

$$(A \star B')[i, j] = \min_{\ell} A[i, \ell]+B'[\ell, j] = \min_{\ell \in S_j} A[i, \ell]+B[\ell, j]-B[s_j, j] = A[i, k]+B[k, j]-B[s_j, j] = C[i, j]-B[s_j, j].$$

We can compute $A \star B'$ in $\tilde{O}(Mn^\omega)$ time because A has entries in $\{-M, \ldots, M\}$ as given and B' has entries that are either ∞ or are in $\{0, \ldots, 2M\}$. This is because we subtracted the *smallest* entry of column j from the other entries, thus obtaining nonnegative numbers, and then we set to ∞ every entry that was bigger than $2M$.

After we compute $A \star B'$ in $\tilde{O}(Mn^\omega)$ time, we can obtain C by simply adding $B[s_j, j]$ to every entry in the jth column of $A \star B'$.

The total running time is $\tilde{O}(Mn^\omega)$.

A has entries in $\{-M, \ldots, M\} \cup \{\infty\}$. In the above discussion we crucially used the fact that *all* entries of A are in $\{-M, \ldots, M\}$. We now want to allow ∞ entries in A as well.

As before, let's consider some $i, j \in [n]$ and let k be a witness of the Min-Plus product entry $C[i, j]$ so that $C[i, j] = A[i, k] + B[k, j]$.

For any ℓ for which $A[i, \ell]$ is *finite*, we can proceed as before to obtain:

> If k is a witness for $C[i, j]$, then for any ℓ for which $A[i, \ell]$ is *finite*, $B[k, j] \leq 2M + B[\ell, j]$.

Think about why the infinite entries no longer allow us to subtract the smallest entry in a column of B from every other entry in the column: $A[i, \ell]$ might be finite but for some $i' \neq i$, $A[i', \ell]$ might not be. We need to change our approach.

Ok, let's sort the columns of B in $O(n^2 \log n)$ time.

Let L_j be the sorted list of entries of the jth column of B. Let d be a parameter. Split L_j in the sorted order into n/d buckets of size roughly d. Let L_{jb} be the bth bucket for column j. Let S_{jb} and M_{jb} be the smallest and largest elements in bucket b, respectively.

Since the buckets are in sorted order of the elements, we necessarily have that if $b < b'$ then $M_{jb} \leq S_{jb'}$.

We will deal with "small" and "large" buckets separately. A bucket b for column j is small if $M_{jb} - S_{jb} \leq 2M$. It is large otherwise.

Let's deal with the **small** buckets first. We will form n/d matrices, one for each choice of $b \in [n/d]$. Define

$$B_b[k,j] = \begin{cases} B[k,j] - S_{jb}, & \text{if } L_{jb} \text{ is a small bucket for } j \text{ and } B[k,j] \in L_{jb}, \\ \infty, & \text{otherwise.} \end{cases}$$

Notice that for every b, B_b has entries in $\{0, \ldots, 2M\}$.

Compute the Min-Plus product of A and B_b, for each choice of b and set

$$C'[i,j] = \min_b \left(S_{j,b} + (A \star B_b)[i,j] \right).$$

The running time is $\tilde{O}(Mn^{\omega+1}/d)$.

For every i, j if the witness k for $C[i,j]$ has $B[k,j]$ in a small bucket L_{jb} for column j, then $C[i,j] = C'[i,j]$.

Thus, what remains to do is to deal with the **large** buckets.

First, we will compute for every i, j, b whether there is some k such that $A[i,k] < \infty$ and $B[k,j] \in L_{jb}$, i.e. is there a potential witness k that is in the bth bucket for column j of B.

We do this as follows. Create A' and B'_b:

$$A'[i,k] = \begin{cases} 1, & \text{if } A[i,k] < \infty, \\ 0, & \text{otherwise.} \end{cases}$$

$$B'_b[k,j] = \begin{cases} 1, & \text{if } B[k,j] \in L_{jb}, \\ 0, & \text{otherwise.} \end{cases}$$

We multiply $A' \cdot B'_b$ for every b, in total time $O(n^{\omega+1}/d)$.

Note that $A' \cdot B'_b[i,j] > 0$ if and only if there is some k with $A[i,k] < \infty$ and $B[k,j] \in L_{jb}$.

What do we want to do? For every i, j and b for which there is some k with $A[i,k] < \infty$ and $B[k,j] \in L_{jb}$, we want to check if one of these k is a witness for $C[i,j]$. We could brute force and look at all $B[k,j] \in L_{jb}$ (where L_{jb} is a large bucket), compute $A[i,k] + B[k,j]$ and take the minimum. There are $O(d)$ entries in L_{jb}. However, we cannot afford to look at all n/d buckets for each i, j as this would cost $O(n^3)$ time.

We will show that it suffices to look at only two large buckets!

Fix i, j. Consider three large buckets $L_{jb_1}, L_{j,b_2}, L_{j,b_3}$, for $b_1 < b_2 < b_3$ such that we have determined that

- there is some k_1 with $A[i,k_1] < \infty$ and $B[k_1,j] \in L_{j,b_1}$,
- there is some k_2 with $A[i,k_2] < \infty$ and $B[k_2,j] \in L_{j,b_2}$,
- there is some k_3 with $A[i,k_3] < \infty$ and $B[k_3,j] \in L_{j,b_3}$.

We have the following:

$$A[i,k_1] + B[k_1,j] \leq M + M_{jb_1} \leq M + S_{jb_2}.$$

The first inequality is because $A[i,k_1]$ is finite and because $B[k_1,j] \in L_{jb_1}$ and M_{jb_1} is the largest entry in L_{jb_1}. The second inequality is because $b_1 < b_2$ and because the buckets are in sorted order so that $M_{jb_1} \leq S_{jb_2}$.

We also have:

$$A[i,k_3] + B[k_3,j] \geq -M + S_{jb_3} \geq -M + M_{jb_2}.$$

The first inequality is because $A[i,k_3]$ is finite and because $B[k_3,j] \in L_{jb_3}$ and S_{jb_3} is the smallest entry in L_{jb_3}. The second inequality is because $b_2 < b_3$ and because the buckets are in sorted order so that $M_{jb_2} \leq S_{jb_3}$.

Finally, since L_{jb_2} is a large bucket, $M_{jb_2} - S_{jb_2} > 2M$. We get:

$$A[i,k_3] + B[k_3,j] \geq -M + M_{jb_2} > M + S_{jb_2} \geq A[i,k_1] + B[k_1,j].$$

Thus, k_3 cannot be a witness for the Min-Plus product entry $C[i,j]$, and so it suffices to look at the first two large buckets L_{jb_1} and L_{jb_2} (in the sorted order)!

For fixed i,j and each of the smallest two large buckets L_{jb_1} and L_{jb_2} for j such that $A'B'_{b_1}[i,j] > 0, A'B'_{b_2}[i,j] > 0$ (i.e. there are k_1,k_2 with $A[i,k_1] < \infty, A[i,k_2] < \infty, B[k_1,j] \in L_{jb_1}, B[k_2,j] \in L_{jb_2}$), go through all $O(d)$ choices of $B[k,j] \in L_{jb_1} \cup L_{jb_2}$ and take the min over all the $O(d)$ corresponding values $A[i,k] + B[k,j]$. Let this be $C''[i,j]$.

If the ij witness is in a large bucket, then $C''[i,j] = C[i,j]$.

Set the Min-Plus product entry $C[i,j]$ to be the min of $C'[i,j]$ and $C''[i,j]$.

The total running time is within polylogs,

$$Mn^{3+\omega}/d + n^2 d.$$

We set $d = \sqrt{M}n^{(3-\omega)/2}$ to obtain the running time $O(\sqrt{M}n^{(3+\omega)/2})$.

We have obtained the following Theorem by [3].

Theorem 1.1. *The Min-Plus product of an $n \times n$ matrix A with entries in $\{-M,\ldots,M\} \cup \{\infty\}$ with an $n \times n$ matrix B with polylog(n) bit integer entries can be computed in $\tilde{O}(\sqrt{M}n^{(3+\omega)/2})$ time.*

Improving this running time is an open problem, even when $M = 0$.

2 Equality Product (optional)

We now take a short detour to another type of matrix product called the *Equality Product*.

Given two $n \times n$ integer matrices A and B their equality product is the $n \times n$ matrix E given by

$$E[i,j] = |\{k \mid A[i,k] = B[k,j]\}|.$$

This product is a generalization of the (normal) product of two *binary* matrices. In the latter we are just computing the number of k for which $A[i,k]$ and $B[k,j]$ are both zero. Since it is not hard to show that multiplying matrices with b-bit entries can be reduced to $\tilde{O}(b)$ instances of binary matrix multiplication, the equality product is a true generalization of matrix multiplication, and it hence must require at least $n^{\omega-o(1)}$ time. It is actually believed to be much harder, in that there is no $O(n^{2.5-\varepsilon})$ time algorithms for it.

Here we present an $O(n^{(3+\omega)/2})$ time algorithm for it. The original running time is essentially due to Matoušek [4] who gave this running time for the dominance product problem which is known to be equivalent to equality product.

(On the problem set you gave an algorithm with a different truly subcubic running time via a different approach.)

Theorem 2.1. *Equality product of $n \times n$ matrices is in $O(n^{(3+\omega)/2})$ time.*

Proof. Let A and B be the two given $n \times n$ matrices. For every fixed k, sort the set $\{A[i,k]\}_i \cup \{B[k,j]\}_j$, obtaining a list L_k of $2n$ numbers.

The total running time so far is $O(n^2 \log n)$.

For each value w that appears in the matrices A, B, let L_{kw} be the set of $A[i,k]$ and $B[k,j]$ that are equal to w.

Set C to be the all 0s matrix to begin with. We will fill it out and in the end it will be the equality product of A and B.

Let d be a parameter. We will deal with all L_{kw} that are small, i.e. for which $0 \leq |L_{kw}| < d$.

For every $i,k \in [n]$, if $A[i,k]$ is in L_{kw} and $|L_{kw}| < d$ (here $w = A[i,k]$), then go over all the at most d $B[k,j] \in L_{kw}$ and increment $C[i,j]$.

The running time so far is $O(n^2 d)$.

The remaining L_{kw} that have not been handled are those for which $|L_{kw}| \geq d$. For each fixed k, the number of such buckets is $\leq n/d$ since they are disjoint.

For each k, let the large buckets be $L_{kw_1}, \ldots, L_{kw_{n/d}}$. We will deal with them using matrix multiplication.

For each $b \in [n/d]$, let's create matrices A_b, B_b:

$$A_b[i,k] = \begin{cases} 1, & \text{if } A[i,k] \in L_{kw_b}, \\ 0, & \text{otherwise.} \end{cases}$$

$$B_b[k,j] = \begin{cases} 1, & \text{if } B[k,j] \in L_{kw_b}, \\ 0, & \text{otherwise.} \end{cases}$$

We then multiply $A_b B_b$ and add the result to C.

Since $A_b B_b[i,j] = \{k \mid A[i,k] = B[k,j] \in L_{kw_b}\}$, we get that the result is correct.

The total running time is

$$O(n^2 d + n^{\omega+1}/d),$$

and setting $d = n^{(\omega-1)/2}$ we get the running time $O(n^{(3+\omega)/2})$. □

3 Exact Triangle and Exact Triangle Count (optional)

In this section we will use the following fact a lot:

Fact 3.1. *For $x > 1$, $(1-1/x)^x \leq 1/e \leq (1-1/x)^{x-1}$.*

Recall the **Exact Triangle** problem: Given a tripartite graph $G = (V, E)$ with node parts A, B, C of size n each and edge weights in $[\pm U]$ where U has $\text{poly}(\log n)$ bits, are there $a \in A, b \in B, c \in C$ so that $w(a,b) + w(b,c) + w(a,c) = 0$?

Let's define the following **AE Exact Triangle Count** problem: Given a tripartite graph $G = (V, E)$ with node parts A, B, C of size n each and edge weights in $[\pm U]$ where U has $\text{poly}(\log n)$ bits, for every $a \in A, b \in B$, compute **the number** of $c \in C$ so that $w(a,b) + w(b,c) + w(a,c) = 0$.

It's easy to see that AE Exact Triangle Count is at least as hard as Exact Triangle. We will show that they are actually subcubically equivalent.

Theorem 3.1 ([2])**.** *AE Exact Triangle Count is (n^3, n^3)-fine-grained equivalent to Exact Triangle.*

Since AE Exact Triangle Count is at least as hard as Exact Triangle, we only need to show that we can reduce AE Exact Triangle Count to Exact Triangle.

A few lectures ago we showed that Exact Triangle is (n^3, n^3)-fine-grained equivalent to the **AE Exact Triangle** problem: Given a tripartite graph $G = (V, E)$ with node parts A, B, C of size n each and edge weights in $[\pm U]$ where U has $\text{poly}(\log n)$ bits, for every $a \in A, b \in B$, return some $c \in C$ so that $w(a,b) + w(b,c) + w(a,c) = 0$, or determine that no such c exists.

In fact we showed that:

(From lecture 6): If Exact Triangle can be solved in $O(n^{3-\varepsilon})$ time for $\varepsilon > 0$, the AE Exact Triangle can be solved in $O(n^{3-\varepsilon/3})$ time.

We will thus strive to reduce AE Exact Triangle Count to AE Exact Triangle.

Let $G = (A \cup B \cup C, E, w)$ be an instance of AE Exact Triangle Count.

For every $a \in A, b \in B$, let's consider the set $W_{a,b}$ of *witnesses* $c \in C$, i.e. those c such that $w(a,c) + w(b,c) = -w(a,b)$. We want to compute $|W_{a,b}|$ for every $a \in A, b \in B$.

Let's pick a parameter L. We will compute $|W_{a,b}|$ for those a, b with $|W_{a,b}| < L$ ("small count") separately from the $|W_{a,b}|$ for the a, b with $|W_{a,b}| \geq L$ ("large count").

We begin with a randomized reduction from small-count AE Exact Triangle Count to AE Exact Triangle.

Claim 2. *If AE Exact Triangle is in $O(n^{3-\varepsilon})$ time, then in $\tilde{O}(L^{\varepsilon} n^{3-\varepsilon})$ time we can list all $c \in W_{a,b}$ for every a, b with $|W_{a,b}| < L$, with high probability.*

We begin with a probabilistic claim.

Claim 3. *Assume that for some s, for a pair $(a,b) \in A \times B$ we have $|W_{a,b}| = q$ where $L/2^{s+1} \leq q < L/2^s$. Suppose we pick $2^s(n/L)$ elements from $C = \{1, \ldots, n\}$ uniformly at random with replacement, and let S be the set of elements picked. Then, for any fixed $c \in W_{a,b}$, with probability at least $\frac{2^s}{eL} = \Omega(2^s/L)$, $W_{a,b} \cap S = \{c\}$.*

Proof. Let $S = \{s_1, \ldots, s_{2^s(n/L)}\}$ where s_i is the ith element drawn from C to form S. As $|W_{a,b}| = q$,

$$Pr(|S \cap W_{a,b}| = \{c\}) = \sum_{i=1}^{2^s(n/L)} Pr(s_i = c) \cdot Pr((S \setminus \{s_i\}) \cap W_{a,b} = \emptyset)$$

$$\geq 2^s(n/L) \cdot \frac{1}{n} \cdot \left(1 - \frac{q}{n}\right)^{2^s(n/L)-1} \geq \frac{2^s}{L} \cdot \left(1 - \frac{L}{n2^s}\right)^{(2^s n/L)-1} \geq \frac{2^s}{eL}.$$

□

Now we are ready to prove Claim 2.

Proof of Claim 2. We repeat the following procedure for every integer s from 0 to $\log L - 1$. Repeat $100 \cdot e \cdot (L/2^s) \ln n$ times:
Let S be a random subset of C of size $2^s(n/L)$. Consider the subgraph of G induced by $A \cup B \cup S$. Since $|A| = |B| = n$ and $|S| = 2^s(n/L)$, we partition A and B into $L/2^s$ parts each, of size $2^s(n/L)$, $(A_1, \ldots, A_{L/2^s}), (B_1, \ldots, B_{L/2^s})$. We now create $(L/2^s)^2$ instances of AE Exact Triangle on a subgraph (A_i, B_j, S), where $|A_i| = |B_j| = |S| = 2^s(n/L)$.

Each AE Exact Triangle call produces for some pairs $(a,b) \in A \times B$ an exact triangle (a,b,c). We maintain a list $L_{a,b}$ for every $a \in A, b \in B$ of all $c \in C$ for which we find an exact triangle. If $|L_{a,b}| \geq L$, we disregard the pair a, b since it must have $|W_{a,b}| \geq |L_{a,b}| \geq L$.

We claim that if $W_{a,b} \in [L/2^{s+1}, L/2^s)$, then whp after the $O((L/2^s) \log n)$ repetitions (for the choice of s) we will get $W_{a,b} = L_{a,b}$, whp. It suffices to show that whp, for every a, b, for any fixed $c \in W_{a,b}$, c will be returned as a witness for a, b by some instance of AE Exact Triangle.

Note that if c is the unique witness for a, b in some iteration, i.e. if $S \cap W_{a,b} = \{c\}$, then a, b, c will be returned as an Exact Triangle by the AE Exact triangle instance (A_i, B_j, S) for which $a \in A_i, b \in B_j$ in that iteration.

By Claim 3, the probability that c is not returned in a fixed iteration is at most $1 - \frac{2^s}{eL}$. Hence the probability that (a,b,c) is never returned as an exact triangle in any of the iterations is at most

$$\left(1 - \frac{2^s}{eL}\right)^{100 \cdot e \cdot (L/2^s) \ln n} \leq (1/e)^{100 \ln n} = 1/n^{100}.$$

Hence the probability that there exists a triple a, b, c such that $|W_{a,b}| < L$ and $c \in W_{a,b}$ and a, b, c is not returned as an exact triangle is by a union bound at most $1/n^{97}$. Thus, with probability $\geq 1 - 1/n^{97}$ all exact triangles through edges with a small number of witnesses are listed.

Assuming that AE Exact Triangle is in $O(N^{3-\varepsilon})$ time in N node graphs for $\varepsilon > 0$, the running time is, within polylogs,

$$\sum_{s=0}^{\log L-1} (L/2^s)^3 \cdot (2^s(n/L))^{3-\varepsilon} = \sum_{s=0}^{\log L-1} (L/2^s)^{\varepsilon} n^{3-\varepsilon} \leq \tilde{O}(L^{\varepsilon} n^{3-\varepsilon}).$$

□

We have shown that for any $L \leq n^{1-\delta}$ for $\delta > 0$, if AE Exact Triangle is in $O(n^{3-\varepsilon})$ time, then listing all exact triangles through edges with $< L$ witnesses can be done in truly subcubic time, $O(n^{3-\varepsilon\delta})$.

Now we show that the exact triangles through edges with $\geq L$ witnesses can be done in truly subcubic time using fast matrix multiplication, provided $L \geq n^{(\omega-1)/2+\delta}$ for some $\delta > 0$.

Claim 4. *For every a, b with $|W_{a,b}| > L$, we can compute $|W_{a,b}|$ in $\tilde{O}((n/L)n^{(3+\omega)/2})$ time, whp. This is truly subcubic time as long as $L \geq n^{(\omega-1)/2+\delta}$ for some $\delta > 0$.*

Proof. We will use random sampling to find a witness $c \in W_{a,b}$ for every $a \in A, b \in B$ with $|W_{a,b}| \geq L$. Then we will use Fredman's trick and matrix multiplication to compute $|W_{a,b}|$ with the help of the sampled witness c.

Randomly sample to get a witness for every a, b with $|W_{a,b}| \geq L$: Sample a uniformly random set $R \subseteq C$ of size $100n/L \ln n$. For any fixed $a \in A, b \in B$ with $|W_{a,b}| \geq L$, the probability that $R \cap W_{a,b} = \emptyset$ is at most $(1 - L/n)^{100n/L \ln n} \leq 1/n^{100}$. Hence with probability at least $1 - 1/n^{98}$, for every a, b with $|W_{a,b}| \geq L$, $W_{a,b} \cap R \neq \emptyset$.

For every a, b and $c \in R$ check if $w(a, c) + w(c, b) = -w(a, b)$, i.e. whether $c \in W_{a,b}$. Whp, now for every a, b with $|W_{a,b}| \geq L$ we have found a $c \in W_{a,b}$.

Fix a, b and $c \in R \cap W_{a,b}$.

We want to compute the number of c' such that $w(a, c') + w(c', b) + w(a, b) = w(a, c) + w(c, b) + w(a, b)$, equivalently, the number of c' such that

$$w(a, c') + w(c', b) = w(a, c) + w(c, b).$$

This is exactly the number of cs in $W_{a,b}$. By Fredman's trick we want the number of c' such that

$$w(a, c') - w(a, c) = w(c, b) - w(c', b).$$

Computing $|W_{a,b}|$ using fast matrix multiplication. For every $c \in R$, create two matrices:

$$A_c[a, c'] = w(a, c') - w(a, c), \text{ and } B_c[c', b] = w(c, b) - w(c', b).$$

We equality-multiply $C_c := A_c \cdot B_c$ for all $c \in R$. For every $a \in A, b \in B$ and such that $c \in R \cap W_{a,b}$, we have that C_c is the number of c' such that $w(a, c') + w(c', b) + w(a, b) = w(a, c) + w(c, b) + w(a, b) = 0$.

As equality product can be computed in time $O(n^{(3+\omega)/2})$, the total running time is $\tilde{O}((n/L)n^{(3+\omega)/2})$. □

Now we put the two pieces of our proof together:

Claim 5. *If Exact Triangle is in $O(n^{3-\varepsilon})$ time for some $\varepsilon > 0$, then AE Exact Triangle Count is in $O(n^{3-\delta})$ time for some $\delta > 0$.*

If Exact Triangle is in $O(n^{3-\varepsilon})$ time then AP Exact Triangle is in $O(n^{3-\varepsilon/3})$ time via our reduction from a few lectures ago. Let $\varepsilon' = \varepsilon/3$.

Pick the parameter $L = n^{0.7}$. First compute the counts for all a, b with $|W_{a,b}| < L$ in $\tilde{O}(L^{\varepsilon'} n^{3-\varepsilon'}) = \tilde{O}(n^{3-0.3\varepsilon'})$ time. Then compute the counts for all a, b with $|W_{a,b}| \geq L$ in $\tilde{O}((n/L)n^{(3+\omega)/2}) \leq \tilde{O}(n^{0.3+2.68}) = O(n^{2.98})$ time.

(One can set the parameter L in terms of ω as well to get a better subcubic running time, but we omit this analysis for simplicity.)

References

[1] Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, Renfei Zhou: More Asymmetry Yields Faster Matrix Multiplication. In SODA 2025, to appear.

[2] Timothy M. Chan, Virginia Vassilevska Williams, Yinzhan Xu: Fredman's Trick Meets Dominance Product: Fine-Grained Complexity of Unweighted APSP, 3SUM Counting, and More. STOC 2023: 419-432

[3] Virginia Vassilevska Williams, Yinzhan Xu: Truly Subcubic Min-Plus Product for Less Structured Matrices, with Applications. SODA 2020: 12-29

[4] Jiri Matousek: Computing Dominances in E^n. Inf. Process. Lett. 38(5): 277-278 (1991).