Complexity Theory of Polynomial-Time Problems

Lecture 4: The polynomial method
Part II: All-pairs shortest paths

Sebastian Krinninger

Overview on APSP Algorithms

- Floyd-Warshall algorithm: $O(n^3)$
  Inserts one node at a time
  n iterations, each taking time $O(n^2)$
- Faster algorithms for sparse graphs
  - Directed graphs:
    - Single-source shortest paths: $O(m + n \log n)$ (Dijkstra with Fibonacci heap/Hollow heap)
    - $\Rightarrow$ All-pairs shortest paths: $O(mn + n^2 \log n)$, improved to $O(mn + n^2 \log\log n)$ [Pettie 02]
  - Undirected graphs:
    - Single-source shortest paths: $O(m)$ [Thorup 97]
    - $\Rightarrow$ All-pairs shortest paths: $O(mn)$
- Pseudopolynomial algorithms

- Today: Fastest "general-purpose" algorithm

History of slightly subcubic algorithms

Running Time	Author(s)	Year(s)
n^3	Floyd, Warshall	1962
$n^3 / \log^{1/3} n$	Fredman	1975
$n^3 / \log^{1/2} n$	Dobosiewicz, Takaoka	1990, 1991
$n^3 / \log^{5/7} n$	Han	2004
$n^3 / \log n$	Takaoka, Zwick, Chan	2004, 2005
$n^3 / \log^{5/4} n$	Han	2006
$n^3 / \log^2 n$	Chan, Han/Takaoka	2007, 2012
$n^3 / \underbrace{2^{\Omega(\log n)^{1/2}}}$	Williams	2014

Grows faster than any polylogarithmic factor

Problem definition: desired output

- Can create instances such that for every pair of nodes u, v shortest path from u to v consists of $\Omega(n)$ nodes
- $\Rightarrow$ Cannot output all shortest paths explicitly in time $o(n^3)$

- Distance matrix: output size n^2
- **Shortest path matrix $\mathbf{SP}$:** output size n^2

  For every pair of nodes u, v, $\mathrm{SP}[u, v] =$ next node on shortest path from u to v

Machine model: Real RAM

Floyd-Warshall: $O(n^3)$ with only additions and comparisons

$\Omega(n^3)$ lower bound if only additions and comparisons allowed [Kerr 70]

Real RAM:

- Additions and comparisons of reals: unit cost
- Other operations: logarithmic cost

Tools

Tool 1: Razborov-Smolensky

Represent AND of d variables $x_1 \wedge \cdots \wedge x_d$ by **low-degree** polynomial

- Parameter q
- For every $i = 1, \dots, q, j = 1, \dots, d$: Set $r_{i,j} = 0$ or 1 with probability $\frac{1}{2}$

$$A(x_1, \dots, x_d) = \bigwedge_{i=1}^{q} \left(1 \oplus \bigoplus_{j=1}^{d} r_{i,j} \cdot (x_j \oplus 1) \right)$$

Lemma: $\Pr_{r_{i,j}}[A(x_1, \dots, x_d) = x_1 \wedge \cdots \wedge x_d] \geq 1 - \frac{1}{2^q}$

By **distributive law**, A can be written as XOR of $(1 + d)^q$ monomials

$$A(x_1, \dots, x_d) = \bigwedge_{i=1}^{q} \left(1 \oplus \bigoplus_{j=1}^{d} r_{i,j} \cdot (x_j \oplus 1) \right)$$

Lemma: $\Pr_{r_{i,j}}[A(x_1, \dots, x_d) = x_1 \wedge \cdots \wedge x_d] \geq 1 - \frac{1}{2^q}$

Proof:

$x_1 \wedge \cdots \wedge x_d = 1$: Each x_j must be 1. Clearly, $A(x_1, \dots, x_d) = 1$

$x_1 \wedge \cdots \wedge x_d = 0$: First, fix some i

S: subsets of x_j's that are 0

S': subsets of x_j's that are 0 and additionally $r_{i,j} = 1$

Bad event: i-th component of outer AND is $1 \Leftrightarrow |S'|$ is even

Each subset of S has same probability of being picked as S'

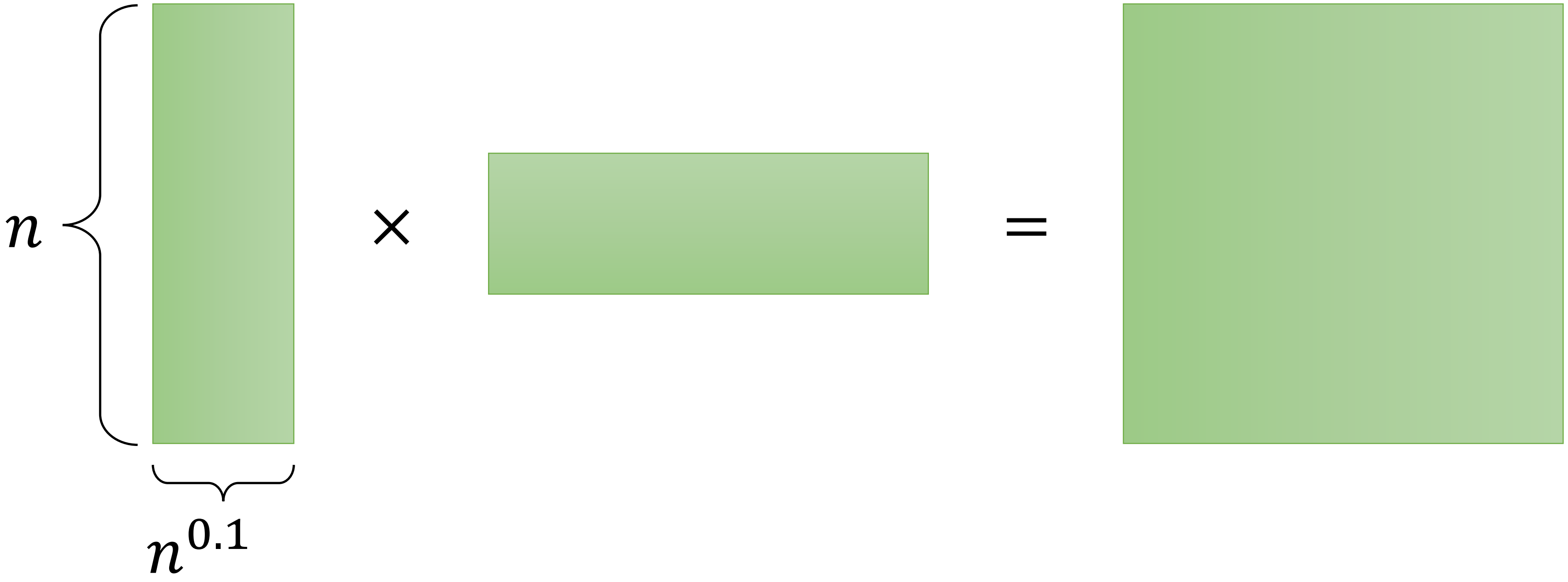
$$\Pr[|S'| \text{ is odd}] = \Pr[|S'| \text{ is even}] = \frac{1}{2}$$

$A(x_1, \dots, x_d) = 1$ only if bad event happens for each component of outer AND

$\Rightarrow$ Error probability $\leq \frac{1}{2^q}$

Observation: For every set S, #odd subsets = #even subsets (by binary encoding of the $2^{|S|}$ subsets)

Tool 2: Fast rectangular matrix multiplication



Theorem: There is an algorithm for multiplying an $n \times n^{0.17}$ matrix with an $n^{0.17} \times n$ matrix in time $O(n^2 \log^2 n)$.

Also works for finite fields such as F_2!

Fast evaluation of polynomial

Given: Polynomial $P(x[1], \dots, x[d], y[1], \dots, y[d])$ over F_2

- With $m \leq n^{0.1}$ monomials
- Two sets of inputs:

$$X = \{x_1, \dots, x_n\} \subseteq \{0,1\}^d \qquad Y = \{y_1, \dots, y_n\} \subseteq \{0,1\}^d$$

$$(x_i = (x_i[1], \dots, x_i[d])) \qquad (y_j = (y_j[1], \dots, y_j[d]))$$

Lemma: There is an algorithm for evaluating P on all pairs $(x_i, y_j) \in X \times Y$ (simultaneously) in time $O(n^2 \mathrm{poly}(\log n))$.

Restrictions of monomials

Shape of polynomial P:

- $P = p_1 + \cdots + p_m$
- each p_k is a **monomial**

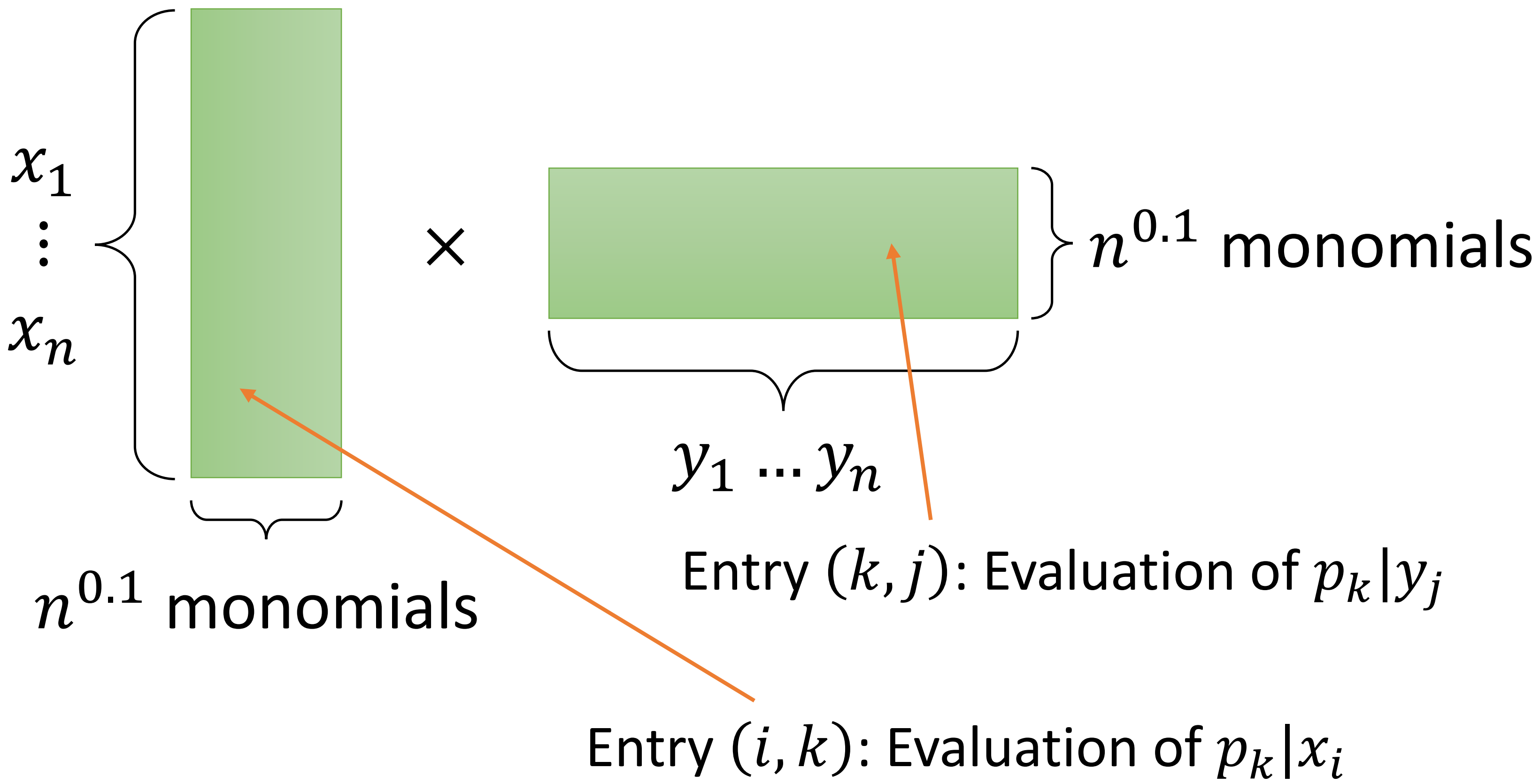
Define

- $p_k|x$: restriction of k-th monomial to variables $x[1], \dots, x[d]$
- $p_k|y$: restriction of k-th monomial to variables $y[1], \dots, y[d]$
- (empty product $= 1$)

“Inner product”

- $P = p_1|x \cdot p_1|y + \cdots + p_m|x \cdot p_m|y$

Reduction to matrix multiplication



Result matrix $R[i, j]$: Evaluation of P under $x_i = (x_i[1], \dots, x_i[d])$
and $y_j = (y_j[1], \dots, y_j[d])$

Tool 3: Union Bound and Chernoff Bound

Union Bound:

$$\Pr[A \cup B] \leq \Pr[A] + \Pr[B]$$

(Variant of) **Chernoff Bound:**

Let $X_1, \ldots, X_k$ be independent 0/1-valued random variables such that $0 < E[X_i] < 1$.

Then, the random variable $X = \sum_{i=1}^{k} X_i$ satisfies:

$$\Pr[X < (1 - \delta)E[X]] \leq e^{-\delta^2 E[X]/2}$$

for every $0 \leq \delta \leq 1$

Solving the Problem

Min-plus matrix product

We give an algorithm for the following problem:

- Given: $n \times d$ integer matrix A and $d \times n$ integer matrix B
- Output: $n \times n$ matrix C such that
$$C[i,j] = \min_{k \in \{1,\dots,d\}} (A[i,k] + B[k,j])$$

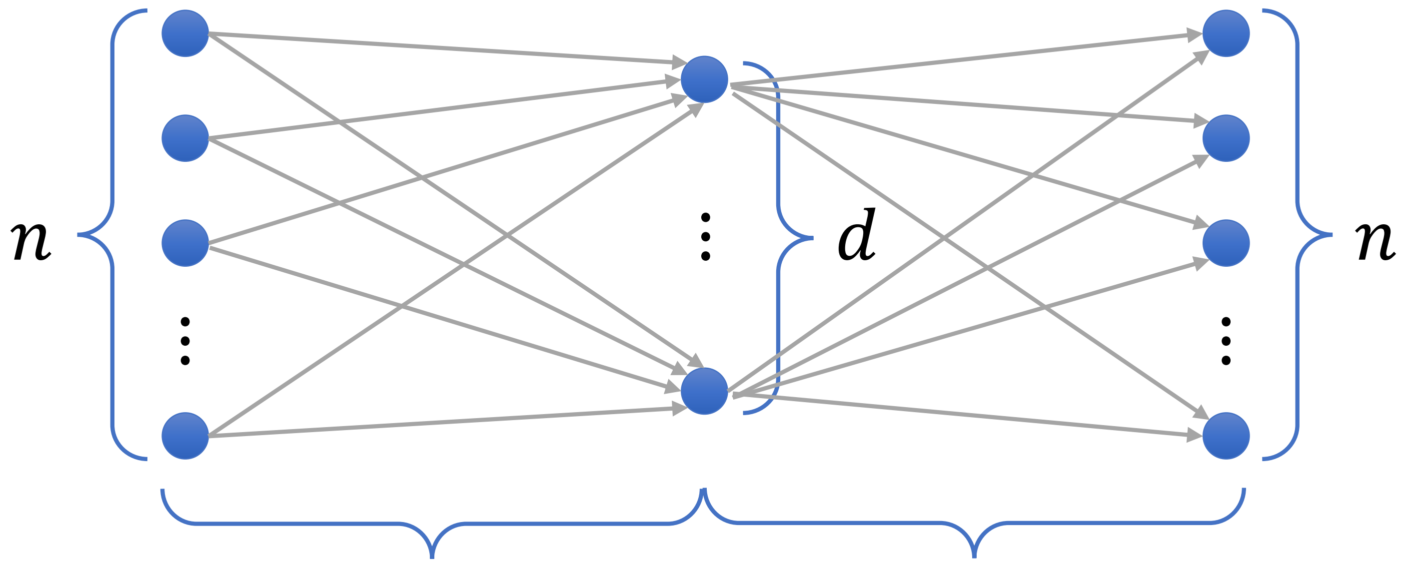
Matrix multiplication in *min-plus semiring:*

- min is addition
- $+$ is multiplication
- 0 is 1-element
- ∞ is 0-element

k^* such that $A[i,k^*] + B[k^*,j] = \min_{k \in \{1,\dots,d\}} (A[i,k] + B[k,j])$ is a **witness** for i, j

Tripartite graph for min-plus product

$$C[i,j] = \min_{1 \le k \le d} (A[i,k] + B[k,j])$$



$w(i,k) = A[i,k]$ $\quad$ $w(k,j) = B[k,j]$ (no edge if weight is ∞)

1. Min-plus product = APSP in tripartite graph
2. If $A = B = G$: $G \times G$ = matrix of 2-hop distances

APSP and min-plus product are “equivalent”

In general: G^i = matrix distances using **exactly** i hops

Distance matrix D:

$$D = I + G + G^2 + \cdots + G^{n-1} = (G + I)^{n-1}$$

$+$ is entry-wise minimum

Identity matrix I: 0 at diagonal, ∞ otherwise

Repeated squaring: Compute $(G+I)^2, (G+I)^4, (G+I)^8, \ldots,$

$\Rightarrow$ $O(\log n)$ min-plus products for distances, shortest paths through witnesses

Even **stronger** relationship known:

Theorem: APSP on n nodes can be solved in time $O(T(n))$ if and only if min-plus product on $n \times n$ matrices can be solved in time $O(T(n))$.

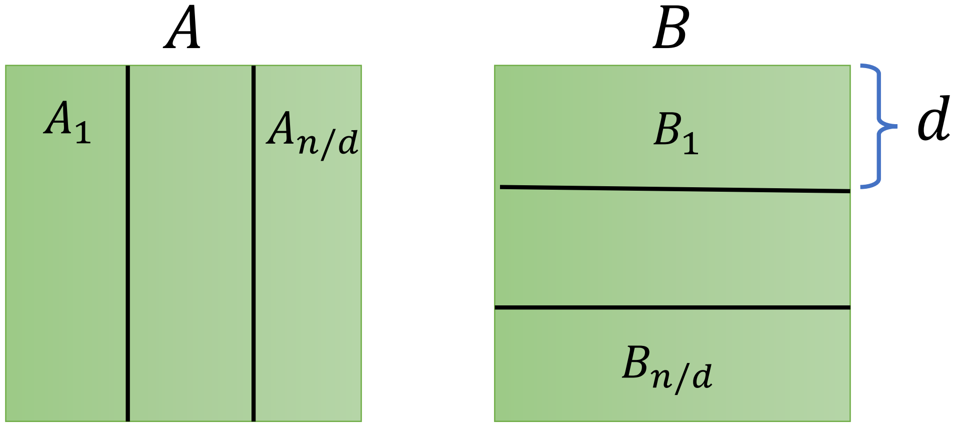
Step 1: Divide into subproblems

Overall algorithm

1. Set $d = 2^{\sqrt{\log n/100}}$
2. Divide problem into $\frac{n}{d}$ subproblems
3. Solve each subproblem in time $O\big(n^2 \text{poly}(\log n)\big)$
4. Merge solutions in time $O(n^3/d)$

Total time:

$$O\left(\frac{n^3}{d}\text{poly}(\log n)\right) = n^3/2^{\Omega(\log n)^{1/2}}$$



For every $k = 1, \dots, \frac{n}{d}$:

- Compute product C_k of A_k and B_k ($n \times d$ matrix with $d \times n$ matrix)

Return: $\min\big(C_1, \dots, C_{n/d}\big)$

(entry-wise minimum)

Subproblem

We solve the following subproblem:

- Given: $n \times d$ matrix A and $d \times n$ matrix B
- Output: $n \times n$ matrix W of witnesses such that

$$W[i,j] = \arg\min_{k\in\{1,\ldots,d\}} (A[i,k] + B[k,j])$$

From witnesses in W we can easily reconstruct values of min-plus product $\min_{k\in\{1,\ldots,d\}} (A[i,k] + B[k,j])$ in time $O(n^2)$

Step 2: Preprocess input of subproblem

Enforce unique minimum

For every entry i, k of A:

$$A^*[i,k] := A[i,k] \cdot (n+1) + k$$

For every entry k, j of B

$$B^*[k,j] := B[k,j] \cdot (n+1)$$

<u>Running time:</u>
$O(\log n)$ **additions** per entry
(add to itself for $O(\log n)$ times)
$\Rightarrow O(nd \log n)$

Fix some pair i, j and define k^* as smallest $k' \in \{1, \ldots, d\}$ such that

$$A[i,k'] + B[k',j] = \min_{k \in \{1,\ldots,d\}} (A[i,k] + B[k,j])$$

Claim: k^* is unique minimum of $A^*[i,k] + B^*[k,j]$ over $k^* \in \{1, \ldots, d\}$

$\Rightarrow$ Work with A^* and B^* instead of A and B to ensure unique minima

Proof of Claim: k^* is unique minimum of $A^*[i,k] + B^*[k,j]$ over $k^* \in \{1, \dots, d\}$

Let $k \neq k^*$. We show that $A^*[i,k] + B^*[k,j] > A^*[i,k^*] + B^*[k,j]$

or equivalently

(1) $(A[i,k] + B[k,j]) \cdot (n+1) + k > (A[i,k^*] + B[k^*,j]) \cdot (n+1) + k^*$

Case 1: $A[i,k] + B[k,j] = A[i,k^*] + B[k,j]$

Then $k^* < k$ because k^* is smallest index assuming min value

(1) follows immediately

Case 2: $A[i,k] + B[k,j] > A[i,k^*] + B[k,j]$

$\Rightarrow A[i,k] + B[k,j] \geq A[i,k^*] + B[k,j] + 1$ (integers!)

$\Rightarrow (A[i,k] + B[k,j]) \cdot (n+1) + k$

$$\geq (A[i,k^*] + B[k^*,j]) \cdot (n+1) + k + n + 1$$
$$> (A[i,k^*] + B[k^*,j]) \cdot (n+1) + k^*$$

Fredman's trick: Get rid of weights

Construct $n \times d^2$ matrix A' and $d^2 \times n$ matrix B'

- $A'[i,(k,\ell)] \coloneqq A^*[i,k] - A^*[i,\ell]$
- $B'[(k,\ell),j] \coloneqq B^*[\ell,j] - B^*[k,j]$

Idea: Compare alternatives k and ℓ without taking sums

Observation: $A'[i,(k,\ell)] \leq B'[(k,\ell),j]$

$\Leftrightarrow A^*[i,k] + B^*[k,j] \leq A^*[i,\ell] + B^*[\ell,j]$

Fredman's trick continued

For every pair k, ℓ **sort** set $S_{k,\ell} \coloneqq \{A'[i,(k,\ell)], B'[(k,\ell),i] \mid i = 1, \dots, n\}$

Breaking ties:

- Precedence of A'-entries over B'-entries
- Otherwise arbitrarily

$O(nd^2 \log n) \le O(n^2)$

Define matrices A'' and B'':

- $A''[i,(k,\ell)] = \mathrm{rank}(A'[i,(k,\ell)]; S_{k,\ell})$
- $B''[(k,\ell),j] = \mathrm{rank}(B'[(k,\ell),j]; S_{k,\ell})$

(replace each value by **rank** in $S_{k,\ell}$)

$\Rightarrow$ Every entry needs $1 + \log n$ bits

(no weight dependence!)

Properties:

1. Entries of A'' and B'' from $\{1, \dots, 2n\}$
2. Comparisons preserved:
   $A'[i,(k,\ell)] \le B'[(k,\ell),j]$ iff
   $A''[i,(k,\ell)] \le B''[(k,\ell),j]$
3. For every i, j there is unique k^* such that for all ℓ:
   $A''[i,(k^*,\ell)] \le B''[(k^*,\ell),j]$

Footnote on running time: A' and B' do not need to be computed explicitly. No subtractions necessary!

Step 3: Design circuit for subproblem

Circuit for min-plus product

Circuit with 0/1 as inputs

Gates:

- Boolean functions: AND, OR
- XOR (i.e., sum modulo 2)

Circuit only outputs 1 bit! ⇒ Compute result bit-per-bit

For every pair i, j and every $b \in \{1, \dots, \log n\}$:

Design circuit $C_b(A''[i,*], B''[*,j])$ computing b-th bit of unique k^* for which
$A''[i,(k^*,\ell)] \leq B''[(k^*,\ell),j]$ for all ℓ

Input: Each bit of i-th row of A'' and j-th colum of B''

Structure of circuit

Goal: For every i, j, compute k^* s.t. $\forall \ell$: $A''[i,(k^*,\ell)] \le B''[(k^*,\ell),j]$

$$C_b(A''[i,*], B''[*,j]) = \bigvee_{\substack{k \in \{1,\dots,d\},\\ b\text{th bit of } k \text{ is } 1}} \bigwedge_{\ell=1}^{d} \underbrace{[A''[i,(k,\ell)] \le B''[(k,\ell),j]]}_{\text{1 iff comparison true (to be specified)}}$$

Claim: $C_b(\cdot,\cdot) = b$-th bit of k^* for which $\forall \ell$: $A''[i,(k^*,\ell)] \le B''[(k^*,\ell),j]$

Proof:

- Big AND returns 1 if and only if $k = k^*$ (uniqueness of minimum)
- If b-th bit of k^* is 1: Big OR includes k^* and thus returns 1
- If b-th bit of k^* is 0: Big OR does not include k^* and thus returns 0

Step 4: Represent circuit by polynomial

Outer OR

$$C_b(A''[i,*], B''[*,j]) = \bigvee_{\substack{k\in\{1,\ldots,d\},\\ b\text{th bit of } k \text{ is } 1}} \bigwedge_{\ell=1}^{d} [A''[i,(k,\ell)] \leq B''[(k,\ell),j]]$$

May be replaced by $\oplus$ due to uniqueness:
AND outputs 1 for **exactly one** k

Polynomial for outer circuit

Fixing i, j, and k, we want to replace the following circuit by a polynomial:

$$\bigwedge_{\ell=1}^{d} \underbrace{\left[A''[i,(k,\ell)] \leq B''[(k,\ell),j]\right]}_{=:\,LEQ_{k,\ell}(\cdot,\cdot)}$$

Apply **Razborov-Smolensky** with $p = 3 + \log d$:

$$\bigwedge_{x=1}^{p} \left(1 \oplus \bigoplus_{\ell=1}^{d} r_{x,\ell} \cdot \left(LEQ_{k,\ell}(A''[i,*], B''[*,j]) \oplus 1\right) \right)$$

- Error probability for specific k: $\leq \frac{1}{2^p} = \frac{1}{8d}$
- Error probability for all k: $\leq d \cdot \frac{1}{8d} = \frac{1}{8}$ (union bound)

Less-or-equal-circuit for two numbers a and b

May be replaced by XOR: at most one of inner expressions is true

$$LEQ(a,b) = \underbrace{\left(\bigwedge_{i=1}^{t}(1 \oplus a_i \oplus b_i)\right)}_{= 1 \text{ iff } a = b} \vee \bigvee_{i=1}^{t} \underbrace{\left((1 \oplus a_i) \wedge b_i \wedge \bigwedge_{j=1}^{i-1} 1 \oplus a_j \oplus b_j\right)}_{= 1 \text{ iff}}$$

- First $i - 1$ bits of a and b equal,
- i-th bit of $a = 0$, and
- i-th bit of $b = 1$

Polynomial for LEQ circuit

$$LEQ(a,b) = \left(\bigwedge_{i=1}^{t} (1 \oplus a_i \oplus b_i)\right) \oplus \bigoplus_{i=1}^{t} \left((1 \oplus a_i) \wedge b_i \wedge \bigwedge_{j=1}^{i-1} 1 \oplus a_j \oplus b_j\right)$$

Apply Razborov/Smolensky with $q = 3 + 2\log d + \log(t+1)$:

$$\bigoplus_{t+1} \left(\bigwedge_{q} \left(\bigoplus_{\leq t} \underbrace{(\text{"}2 \oplus \text{gates"})}_{}\right)\right)$$

at most one a_i, at most one b_i, at most one constant

Additional trick: For every entry a of A'' and every entry b of B'':

Precompute XOR of a_i's and XOR of b_i's: additional time $O(nd^2tq) \leq O(n^2)$

Introduce *new variables* for these combinations for later evaluation

New form: $LEQ'(a,b) = \bigoplus_{t+1} \left(\bigwedge_{q} (\text{"}2 \oplus \text{gates"})\right)$

Polynomial for LEQ circuit cont'd

$$LEQ'(a,b) = \bigoplus_{t+1} \left(\bigwedge_{q} (\text{"2} \oplus \text{gates"}) \right)$$

Expansion (distributive law): → polynomial over F_2 with

- degree $\leq q$
- #monomials: $m \leq (t+1) \cdot 3^q$ monomials

Error probability: For each application of Raz/Smol: Error prob. $\leq \frac{1}{2^q}$

By union bound:

- For comparing a fixed pair (a, b): error probability $\leq \frac{t+1}{2^q}$
- For all d^2 comparisons: error probability $\leq \frac{d^2(t+1)}{2^q} \leq \frac{d^2(t+1)}{2^{3+2\log d+\log(t+1)}} = \frac{1}{8}$

Final polynomial

$$P_b(A''[i,*], B''[*,j]) = \bigoplus_{\substack{k=1,\ldots,d \\ b\text{th bit of } k \text{ is } 1}} \bigwedge_{x=1}^{p} \underbrace{\left(1 \oplus \bigoplus_{\ell=1}^{d} r_{x,\ell} \cdot \left(\underbrace{LEQ'_{k,\ell}(A''[i,*], B''[*,j])}_{\text{XOR with } m \leq (t+1)\cdot 3^q \text{ monomials}} \oplus 1 \right) \right)}_{\text{XOR with } \leq (d+1)m \text{ monomials}}$$

Apply distributive law: #monomials bounded by

$$M \leq d \cdot \big((d+1)m\big)^p = d \cdot \big((d+1)m\big)^{2+\log d}$$

Error probability: $\leq \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$

The calculation $d = 2^{\sqrt{\log n/100}}$ $p = 3 + \log d$ $q = 3 + 2\log d + \log(t+1)$

#monomials $M \le d \cdot \big((d+1)m\big)^p = d \cdot \big((d+1)m\big)^{3+\log d}$

$$= d \cdot \big((d+1)\cdot(t+1)\cdot 3^q\big)^{3+\log d}$$

$$= d \cdot \left((d+1)\cdot(t+1)\cdot 3^{3+2\log d+\log(t+1)}\right)^{3+\log d}$$

Claim: $M \le n^{0.1}$

Taking logarithms:

$$\log M \le \log d + (3+\log d)(\log(d+1) + \log(t+1) + (3 + 2\log d + \log(t+1))\cdot \log 3) \quad d \ge t$$

$$\le \log d + (3+\log d)(\log(d+1) + \log(d+1) + (3 + 2\log d + \log(d+1))\cdot 2)$$

$$\le \log d + (3\log d + \log d)(2\log d + 2\log d + (3\log d + 2\log d + 2\log d)\cdot 2)$$

$$= \log d + 4\log d\,(4\log d + (7\log d)\cdot 2) = \log d + 76\log^2 d \le 100\log^2 d$$

$$= 100\left(\frac{\sqrt{\log n}}{100}\right)^2 \le 0.1\log n$$

Step 5: Fast evaluation of polynomial

Fast evaluation of polynomial

For every $b \in \{1, \ldots, \log n\}$:

Generate probabilistic polynomial P_b with the following properties

- P_b is XOR of $M \leq n^{0.1}$ monomials
- Variables of P_b can partitioned into two subsets X and Y
- For every pair i, j: if
  - variables of X evaluated according to i-th row of A'' and
  - variables of Y evaluated according to j-th column of B'',
  - then P_b returns b-th bit of $\arg\min_{k \in \{1,\ldots,d\}}(A''[i,(k,\ell)] \leq B''[(k,\ell),j])$ with probability $\geq \frac{3}{4}$

$\Rightarrow$ (Fast Evaluation Lemma):

Can evaluate P_b for all n^2 pairs i, j in time $O(n^2 \mathrm{poly}(\log n))$

Result matrix R_b with entries $R_b[i, j]$

Step 6: Amplify success probability

Majority amplification

For all pairs i, j and every $b \in \{1, \dots, \log n\}$:

$$R_b[i,j] = C_b(A''[i,*], B''[*,j]) \text{ with probability} \geq \frac{3}{4}$$

Repeat evaluation with $r = 18 \log n$ different random polynomials
Define $W_b[i,j]$ as majority output of all r evaluations

...still $O(n^2 \text{poly}(\log n))$

Fix pair i, j and $b \in \{1, \dots, \log n\}$

X: Random variable counting how often $R_b[i,j]$ and $C_b(i,j\ )$ agree over all r trials

$$\Pr[W_b[i,j] \neq C_b(i,j)] \leq \Pr\left[X < \frac{r}{2}\right]$$

$$E[X] \geq \frac{3 \cdot r}{4}$$

Bounding success probability

Chernoff: $\Pr[X < (1-\delta)E[X]] \le e^{-\delta^2 E[X]/2}$

Bound error probability using tail bound:

$$\Pr[M_b[i,j] \neq C(i,j,b)] \le \Pr\left[X < \frac{r}{2}\right] \le \Pr\left[X < \frac{4}{6}\mathrm{E}[X]\right] = \Pr\left[X < \left(1 - \frac{1}{3}\right)\mathrm{E}[X]\right]$$

$$\le e^{-\left(\frac{2}{3}\right)^2 \mathrm{E}[X]/2} = e^{-4\mathrm{E}[X]/18} \le e^{-3r/18} = e^{-3\log n} \le 2^{-4\log n} = n^{-4}$$

Majority needs to be correct for all n^2 pairs i,j and $\log d$ bit positions b in all $\frac{n}{d}$ instances of the algorithm:

Union bound:

$$\Pr[\exists i,j,b\colon M_b[i,j] \neq C_b(i,j) \text{ in some instance}] \le \frac{n^3 \log d}{d} \cdot n^{-4} \le \frac{1}{n}$$

Questions?