Matrix-Vector Multiplication in Sub-Quadratic Time
(Some Preprocessing Required)

Ryan Williams

Carnegie Mellon University
Introduction

Matrix-Vector Multiplication: Fundamental Operation in Scientific Computing
Introduction

Matrix-Vector Multiplication: Fundamental Operation in Scientific Computing

How fast can $n \times n$ matrix-vector multiplication be?
Introduction

Matrix-Vector Multiplication: Fundamental Operation in Scientific Computing

How fast can $n \times n$ matrix-vector multiplication be?

$\Theta(n^2)$ steps just to read the matrix!
Introduction

Matrix-Vector Multiplication: Fundamental Operation in Scientific Computing

How fast can $n \times n$ matrix-vector multiplication be?

$\Theta(n^2)$ steps just to read the matrix!

**Main Result:** If we allow $O(n^{2+\epsilon})$ preprocessing, then matrix-vector multiplication over any finite semiring can be done in $O(n^2/(\epsilon \log n)^2)$. 
Better Algorithms for Matrix Multiplication

Three of the major developments:
Better Algorithms for Matrix Multiplication

Three of the major developments:

- Arlazarov et al., a.k.a. “Four Russians” (1960’s): $O(n^3 / \log n)$ operations
  
  *Uses table lookups*
  
  *Good for hardware with short vector operations as primitives*
Better Algorithms for Matrix Multiplication

Three of the major developments:

• Arlazarov et al., a.k.a. “Four Russians” (1960’s): \( O\left(\frac{n^3}{\log n}\right) \) operations
  
  *Uses table lookups*

  *Good for hardware with short vector operations as primitives*

• Strassen (1969): \( n^{\log_7 2} = O\left(n^{2.81}\right) \) operations

  *Asymptotically fast, but overhead in the big-O*

  *Experiments in practice are inconclusive about Strassen vs. Four Russians for Boolean matrix multiplication (Bard, 2006)*
Better Algorithms for Matrix Multiplication

Three of the major developments:

• Arlazarov et al., a.k.a. “Four Russians” (1960’s): $O(\frac{n^3}{\log n})$ operations
  
  * Uses table lookups
  
  * Good for hardware with short vector operations as primitives

• Strassen (1969): $n^{\log_2 7} = O(n^{2.81})$ operations
  
  * Asymptotically fast, but overhead in the big-O
  
  * Experiments in practice are inconclusive about Strassen vs. Four Russians for Boolean matrix multiplication (Bard, 2006)

• Coppersmith and Winograd (1990): $O(n^{2.376})$ operations
  
  * Not yet practical
Focus: Combinatorial Matrix Multiplication Algorithms
Focus: *Combinatorial* Matrix Multiplication Algorithms

- Also called *non-algebraic*; let’s call them *non-subtractive*
  
  *E.g.* Four-Russians is combinatorial, Strassen isn’t
Focus: *Combinatorial* Matrix Multiplication Algorithms

- Also called *non-algebraic*; let’s call them *non-subtractive*

  *E.g.* Four-Russians is combinatorial, Strassen isn’t

More Non-Subtractive Boolean Matrix Mult. Algorithms:

- Atkinson and Santoro: $O\left(n^3 / \log^{3/2} n\right)$ on a $(\log n)$-word RAM
- Rytter and Basch-Khanna-Motwani: $O\left(n^3 / \log^2 n\right)$ on a RAM
- Chan: Four Russians can be implemented on $O\left(n^3 / \log^2 n\right)$ on a pointer machine
Main Result

The $O(n^3 / \log^2 n)$ matrix multiplication algorithm can be “de-amortized”
Main Result

The $O(n^3 / \log^2 n)$ matrix multiplication algorithm can be “de-amortized”

More precisely, we can:

- **Preprocess** an $n \times n$ matrix $A$ over a finite semiring in $O(n^{2+\varepsilon})$
- **Such that** vector multiplications with $A$ can be done in $O(n^2 / (\varepsilon \log n)^2)$
Main Result

The $O(n^3 / \log^2 n)$ matrix multiplication algorithm can be “de-amortized”

More precisely, we can:

- **Preprocess** an $n \times n$ matrix $A$ over a finite semiring in $O(n^{2+\varepsilon})$
- **Such that** vector multiplications with $A$ can be done in $O(n^2 / (\varepsilon \log n)^2)$

Allows for “non-subtractive” matrix multiplication to be done *on-line*
Main Result

The $O(n^3 / \log^2 n)$ matrix multiplication algorithm can be “de-amortized”

More precisely, we can:

1. **Preprocess** an $n \times n$ matrix $A$ over a finite semiring in $O(n^{2+\varepsilon})$

2. **Such that** vector multiplications with $A$ can be done in $O(n^2 / (\varepsilon \log n)^2)$

Allows for “non-subtractive” matrix multiplication to be done on-line

Can be implemented on a pointer machine
Main Result

The $O(n^3 / \log^2 n)$ matrix multiplication algorithm can be “de-amortized”

More precisely, we can:

- **Preprocess** an $n \times n$ matrix $A$ over a finite semiring in $O(n^{2+\varepsilon})$
- **Such that** vector multiplications with $A$ can be done in $O(n^2 / (\varepsilon \log n)^2)$

Allows for “non-subtractive” matrix multiplication to be done on-line

Can be implemented on a pointer machine

This Talk: The Boolean case
Preprocessing Phase: The Boolean Case

Partition the input matrix $A$ into blocks of $\lceil \varepsilon \log n \rceil \times \lceil \varepsilon \log n \rceil$ size:

$$A = \begin{bmatrix}
A_{1,1} & A_{1,2} & \cdots & A_{1,\frac{n}{\varepsilon \log n}} \\
A_{2,1} & & \ddots & \\
& \ddots & & \\
A_{\frac{n}{\varepsilon \log n},1} & \cdots & \cdots & A_{\frac{n}{\varepsilon \log n},\frac{n}{\varepsilon \log n}}
\end{bmatrix}$$
Preprocessing Phase: The Boolean Case

Build a graph $G$ with parts $P_1, \ldots, P_{n/(\varepsilon \log n)}, Q_1, \ldots, Q_{n/(\varepsilon \log n)}$

Each part has $2^{\varepsilon \log n}$ vertices, one for each possible $\varepsilon \log n$ vector.
Preprocessing Phase: The Boolean Case

Edges of $G$: Each vertex $v$ in each $P_i$ has exactly one edge into each $Q_j$.

$P_i$, $Q_j$, $v$, $A_{j,i}$
Preprocessing Phase: The Boolean Case

Edges of $G$: Each vertex $v$ in each $P_i$ has exactly one edge into each $Q_j$

Time to build the graph:

\[
\frac{n}{\varepsilon \log n} \cdot \frac{n}{\varepsilon \log n} \cdot 2\varepsilon \log n \cdot (\varepsilon \log n)^2 = O(n^{2+\varepsilon})
\]

- number of $Q_j$
- number of $P_i$
- number of nodes in $P_i$
- matrix-vector mult
- of $A_{j,i}$ and $v$

7-a
How to Do Fast Vector Multiplications

Let $v$ be a column vector. Want: $A \cdot v$. 
How to Do Fast Vector Multiplications

Let $v$ be a column vector. Want: $A \cdot v$.

1. Break up $v$ into $\varepsilon \log n$ sized chunks:

$$v = \begin{bmatrix}
v_1 \\
v_2 \\
\vdots \\
v_{\frac{n}{\varepsilon \log n}}
\end{bmatrix}$$
How to Do Fast Vector Multiplications

(2) For each $i = 1, \ldots, n/(\varepsilon \log n)$, look up $v_i$ in $P_i$. 
How to Do Fast Vector Multiplications

(2) For each $i = 1, \ldots, n/(\varepsilon \log n)$, look up $v_i$ in $P_i$. 

- $P_1$ \hspace{1cm} $Q_1$
- $P_2$ \hspace{1cm} $Q_2$
- $P$ \hspace{1cm} $Q$

$T \tilde{\Omega}(n)$ time.
How to Do Fast Vector Multiplications

(2) For each \( i = 1, \ldots, n/(\varepsilon \log n) \), look up \( v_i \) in \( P_i \).

\[
\begin{align*}
\mathcal{P}_1 & \quad 2\varepsilon \log n & \quad n/\varepsilon \log n \\
\quad & \quad \quad v_1 & \quad \quad \quad \quad \quad Q_1 \\
\mathcal{P}_2 & \quad 2\varepsilon \log n & \quad 2\varepsilon \log n \\
\quad & \quad \quad v_2 & \quad \quad \quad \quad \quad Q_2 \\
\quad & \quad \quad \vdots & \quad \quad \quad \quad \quad \quad \quad \vdots \\
\mathcal{P}_{n/\varepsilon \log n} & \quad 2\varepsilon \log n & \quad 2\varepsilon \log n \\
\quad & \quad \quad v_n/(\varepsilon \log n) & \quad \quad \quad \quad \quad Q_{n/\varepsilon \log n}
\end{align*}
\]

Takes \( \tilde{O}(n) \) time.
How to Do Fast Vector Multiplications

(3) Look up the neighbors of \( v_i \), mark each neighbor found.
How to Do Fast Vector Multiplications

(3) Look up the neighbors of $v_i$, mark each neighbor found.

\[
P_1 \quad 2^{\varepsilon \log n} \quad Q_1 \\
\quad v_1 \\
P_2 \quad 2^{\varepsilon \log n} \quad Q_2 \\
\quad v_2 \\
\vdots \quad \vdots \\
\vdots \quad \vdots \\
\frac{P}{\varepsilon \log n} \quad 2^{\varepsilon \log n} \quad \frac{Q}{\varepsilon \log n} \\
\frac{v_n}{(\varepsilon \log n)} \\
A_{1,1} \cdot v_1 \\
A_{2,1} \cdot v_1 \\
A_{\frac{n}{\varepsilon \log n}, 1} \cdot v_1
\]
How to Do Fast Vector Multiplications

(3) Look up the neighbors of $v_i$, mark each neighbor found.

$$P_1  \quad 2^{\epsilon \log n} \quad Q_1$$

$$P_2  \quad 2^{\epsilon \log n} \quad Q_2$$

$$P_{\frac{n}{\epsilon \log n}}  \quad 2^{\epsilon \log n} \quad Q_{\frac{n}{\epsilon \log n}}$$

$$v_1$$

$$v_2$$

$$v_n / (\epsilon \log n)$$

$A_{1,2} \cdot v_2$

$A_{2,2} \cdot v_2$

$A_{\frac{n}{\epsilon \log n},2} \cdot v_2$
How to Do Fast Vector Multiplications

(3) Look up the neighbors of \( v_i \), mark each neighbor found.

\[
\begin{align*}
\text{P}_1 & \quad 2^{\varepsilon \log n} & \quad \text{Q}_1 \\
\text{P}_2 & \quad 2^{\varepsilon \log n} & \quad \text{Q}_2 \\
\vdots & & \vdots \\
\text{P} \cdot \frac{n}{\varepsilon \log n} & \quad 2^{\varepsilon \log n} & \quad \text{Q} \cdot \frac{n}{\varepsilon \log n} \\
\text{v}_n/(\varepsilon \log n) & & \text{v}_n/(\varepsilon \log n) \\
\end{align*}
\]

\[
A_1, \frac{n}{\varepsilon \log n} \cdot \text{v}_n/(\varepsilon \log n)
\]

\[
A_2, \frac{n}{\varepsilon \log n} \cdot \text{v}_n/(\varepsilon \log n)
\]

Takes \( O \left( \left( \frac{n}{\varepsilon \log n} \right)^2 \right) \)
How to Do Fast Vector Multiplications

(4) For each $Q_j$, define $v'_j$ as the OR of all marked vectors in $Q_j$.

$P_1 \quad Q_1$

$P_2 \quad Q_2$

$P_{\frac{n}{\varepsilon \log n}} \quad Q_{\frac{n}{\varepsilon \log n}}$

$\forall \quad \Rightarrow \quad v'_1$

$\forall \quad \Rightarrow \quad v'_2$

$\forall \quad \Rightarrow \quad v'_{n/(\varepsilon \log n)}$
How to Do Fast Vector Multiplications

(4) For each $Q_j$, define $v'_j$ as the OR of all marked vectors in $Q_j$

$P_1 \quad 2^{\varepsilon \log n} \quad Q_1$

$\bigvee \quad \Rightarrow \quad v'_1$

$P_2 \quad 2^{\varepsilon \log n} \quad Q_2$

$\bigvee \quad \Rightarrow \quad v'_2$

$\vdots \quad \vdots \quad \vdots$

$P \quad \frac{n}{\varepsilon \log n} \quad 2^{\varepsilon \log n} \quad Q \quad \frac{n}{\varepsilon \log n}$

$\bigvee \quad \Rightarrow \quad v'_n/(\varepsilon \log n)$

Takes $\tilde{O}(n^{1+\varepsilon})$ time
How to Do Fast Vector Multiplications

Output \( \mathbf{v}' := \begin{bmatrix} v'_1 \\ v'_2 \\ \vdots \\ v'_{n/\varepsilon \log n} \end{bmatrix} \). Claim: \( \mathbf{v}' = \mathbf{A} \cdot \mathbf{v} \).
How to Do Fast Vector Multiplications

(5) Output \( \mathbf{v}' := \begin{bmatrix} v'_1 \\ v'_2 \\ \vdots \\ v'_{\frac{n}{\varepsilon \log n}} \end{bmatrix} \). \hspace{1cm} \text{Claim:} \ v' = A \cdot v.

Proof: By definition, \( v'_j = \bigvee_{i=1}^{n/(\varepsilon \log n)} A_{j,i} \cdot v_i. \)
How to Do Fast Vector Multiplications

(5) Output $v' := \begin{bmatrix} v'_1 \\ v'_2 \\ \vdots \\ v'_{\frac{n}{\log n}} \end{bmatrix}$. Claim: $v' = A \cdot v$.

Proof: By definition, $v'_j = \bigvee_{i=1}^{n/(\varepsilon \log n)} A_{j,i} \cdot v_i$. 

$Av = \begin{bmatrix} A_{1,1} & \cdots & A_{1,n/(\varepsilon \log n)} \\ \vdots & \ddots & \vdots \\ A_{n/(\varepsilon \log n),1} & \cdots & A_{n/(\varepsilon \log n),n/(\varepsilon \log n)} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_{\frac{n}{\log n}} \end{bmatrix}$
How to Do Fast Vector Multiplications

Output $v' := \begin{bmatrix} v'_1 \\ v'_2 \\ \vdots \\ v'_{\varepsilon \log n} \end{bmatrix}$. Claim: $v' = A \cdot v$.

Proof: By definition, $v'_j = \bigvee_{i=1}^{n/(\varepsilon \log n)} A_{j,i} \cdot v_i$.

$Av = \begin{bmatrix} A_{1,1} & \cdots & A_{1,n/(\varepsilon \log n)} \\ \vdots & \ddots & \vdots \\ A_{n/(\varepsilon \log n),1} & \cdots & A_{n/(\varepsilon \log n),n/(\varepsilon \log n)} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_{\varepsilon \log n} \end{bmatrix}$

$= \left( \bigvee_{i=1}^{n/(\varepsilon \log n)} A_{1,i} \cdot v_i, \ldots, \bigvee_{i=1}^{n/(\varepsilon \log n)} A_{1,n/(\varepsilon \log n)} \cdot v_i \right) = v'$. 
Some Applications

Can quickly compute the neighbors of arbitrary vertex subsets

Let $A$ be the adjacency matrix of $G = (V, E)$.

Let $v_S$ be the indicator vector for a $S \subseteq V$. 
Some Applications

Can quickly compute the neighbors of arbitrary vertex subsets

Let $A$ be the adjacency matrix of $G = (V, E)$.

Let $v_S$ be the indicator vector for a $S \subseteq V$.

**Proposition:** $A \cdot v_S$ is the indicator vector for $N(S)$, the neighborhood of $S$. 

---

17-a
Some Applications

Can quickly compute the neighbors of arbitrary vertex subsets

Let $A$ be the adjacency matrix of $G = (V, E)$.

Let $v_S$ be the indicator vector for a $S \subseteq V$.

**Proposition:** $A \cdot v_S$ is the indicator vector for $N(S)$, the neighborhood of $S$.

**Corollary:** After $O(n^{2+\varepsilon})$ preprocessing, can determine the neighborhood of any vertex subset in $O(n^2 / (\varepsilon \log n)^2)$ time.

(One level of BFS in $o(n^2)$ time)
Graph Queries

Corollary: After $O(n^{2+\epsilon})$ preprocessing, can determine if a given vertex subset is an independent set, a vertex cover, or a dominating set, all in $O(n^2/(\epsilon \log n)^2)$ time.
Graph Queries

Corollary: After $O(n^{2+\varepsilon})$ preprocessing, can determine if a given vertex subset is an independent set, a vertex cover, or a dominating set, all in $O(n^2/(\varepsilon \log n)^2)$ time.

Proof: Let $S \subseteq V$. 
Graph Queries

Corollary: After $O(n^{2+\varepsilon})$ preprocessing, can determine if a given vertex subset is an independent set, a vertex cover, or a dominating set, all in $O(n^2/(\varepsilon \log n)^2)$ time.

Proof: Let $S \subseteq V$.

$$S \text{ is dominating} \iff S \cup N(S) = V.$$
Graph Queries

Corollary: After $O(n^{2+\varepsilon})$ preprocessing, can determine if a given vertex subset is an independent set, a vertex cover, or a dominating set, all in $O(n^2/(\varepsilon \log n)^2)$ time.

Proof: Let $S \subseteq V$.

$S$ is dominating $\iff S \cup N(S) = V.$

$S$ is independent $\iff S \cap N(S) = \emptyset.$
Corollary: After $O(n^{2+\varepsilon})$ preprocessing, can determine if a given vertex subset is an independent set, a vertex cover, or a dominating set, all in $O(n^2/(\varepsilon \log n)^2)$ time.

Proof: Let $S \subseteq V$.

\[
S \text{ is dominating } \iff S \cup N(S) = V.
\]

\[
S \text{ is independent } \iff S \cap N(S) = \emptyset.
\]

\[
S \text{ is a vertex cover } \iff V - S \text{ is independent.}
\]
Graph Queries

Corollary: After $O(n^{2+\varepsilon})$ preprocessing, can determine if a given vertex subset is an independent set, a vertex cover, or a dominating set, all in $O(n^2/(\varepsilon \log n)^2)$ time.

Proof: Let $S \subseteq V$.

- $S$ is dominating $\iff S \cup N(S) = V$.
- $S$ is independent $\iff S \cap N(S) = \emptyset$.
- $S$ is a vertex cover $\iff V - S$ is independent.

Each can be quickly determined from knowing $S$ and $N(S)$. 
Triangle Detection
Triangle Detection

Problem: Triangle Detection

Given: Graph $G$ and vertex $i$.

Question: Does $i$ participate in a 3-cycle, a.k.a. triangle?
Triangle Detection

Problem: Triangle Detection

Given: Graph $G$ and vertex $i$.

Question: Does $i$ participate in a 3-cycle, a.k.a. triangle?

Worst Case: Can take $\Theta(n^2)$ time to check all pairs of neighbors of $i$.
Triangle Detection

Problem: Triangle Detection

Given: Graph $G$ and vertex $i$.

Question: Does $i$ participate in a 3-cycle, a.k.a. triangle?

Worst Case: Can take $\Theta(n^2)$ time to check all pairs of neighbors of $i$

Corollary: After $O(n^{2+\varepsilon})$ preprocessing on $G$, can solve triangle detection for arbitrary vertices in $O(n^2/(\varepsilon \log n)^2)$ time.
Triangle Detection

Problem: **Triangle Detection**

Given: Graph $G$ and vertex $i$.

Question: Does $i$ participate in a 3-cycle, a.k.a. triangle?

Worst Case: Can take $\Theta(n^2)$ time to check all pairs of neighbors of $i$

Corollary: After $O(n^{2+\varepsilon})$ preprocessing on $G$, can solve triangle detection for arbitrary vertices in $O(n^2/(\varepsilon \log n)^2)$ time.

Proof: Given vertex $i$, let $S$ be its set of neighbors (gotten in $O(n)$ time).

$S$ is not independent $\iff$ $i$ participates in a triangle.
Conclusion

A preprocessing/multiplication algorithm for matrix-vector multiplication that builds on lookup table techniques
Conclusion

A preprocessing/multiplication algorithm for matrix-vector multiplication that builds on lookup table techniques

- Is there a preprocessing/multiplication algorithm for sparse matrices? Can we do multiplication in e.g. \(O(m/poly(\log n) + n)\), where \(m\) = number of nonzeros?
Conclusion

A preprocessing/multiplication algorithm for matrix-vector multiplication that builds on lookup table techniques

- Is there a preprocessing/multiplication algorithm for sparse matrices? Can we do multiplication in e.g. \( O\left(\frac{m}{\text{poly}(\log n)} + n\right) \), where \( m \) = number of nonzeros?

- Can the algebraic matrix multiplication algorithms (Strassen, etc.) be applied to this problem?
Conclusion

A preprocessing/multiplication algorithm for matrix-vector multiplication that builds on lookup table techniques

- Is there a preprocessing/multiplication algorithm for sparse matrices? Can we do multiplication in \( O(m/poly(\log n) + n) \), where \( m \) = number of nonzeros?

- Can the algebraic matrix multiplication algorithms (Strassen, etc.) be applied to this problem?

- Can our ideas be extended to achieve non-subtractive Boolean matrix multiplication in \( o(n^3 / \log^2 n) \)?
Thank you!