All Pairs Bottleneck Paths in Truly Subcubic Time

Virginia Vassilevska

STOC

June 13, 2007

joint work with Ryan Williams and Raphael Yuster

Introduction

Introduction

There are strong connections between the complexity of fundamental graph problems and the complexity of matrix multiplication over a ring.

Introduction

There are strong connections between the complexity of fundamental graph problems and the complexity of matrix multiplication over a ring.

Subcubic algorithms for some special cases of APSP have been obtained using fast matrix multiplication (Seidel95, Galil and Margalit97, Shoshan and Zwick99, Zwick02).

The best running time for APSP so far is $O(n^3/\log^2 n)$ by Chan.

Introduction

There are strong connections between the complexity of fundamental graph problems and the complexity of matrix multiplication over a ring.

Subcubic algorithms for some special cases of APSP have been obtained using fast matrix multiplication (Seidel95, Galil and Margalit97, Shoshan and Zwick99, Zwick02).

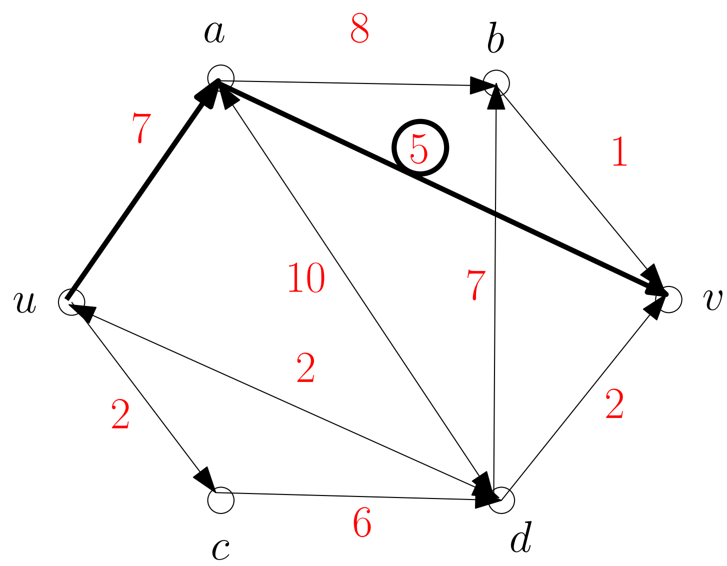
The best running time for APSP so far is $O(n^3/\log^2 n)$ by Chan.

This talk: truly subcubic algorithm for APBP – studied alongside APSP.

Bottleneck paths - definitions

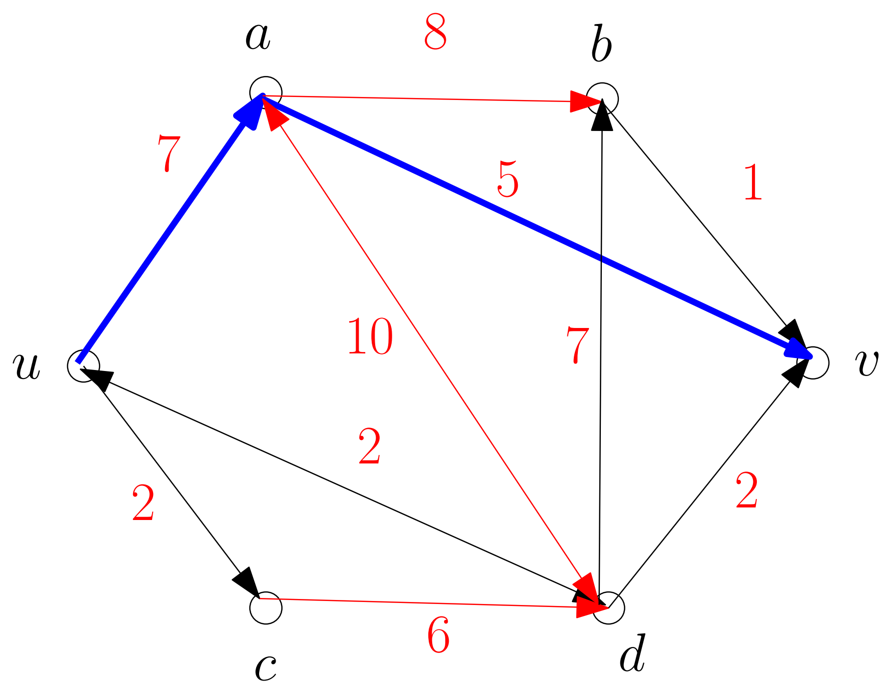
Given: graph $G = (V, E)$ with arbitrary edge weights $w : E \longrightarrow \mathbb{R}$.

The bottleneck edge of a path in G from vertex u to vertex v is the edge of smallest weight on the path.



Maximum bottleneck paths

In many applications (e.g. max flow), the path of maximum bottleneck is needed.



$ab : 8$	$bu : -\infty$
$ac : 2$	$ub : 7$
$ad : 10$	$cd : 6$
$au : 2$	$dc : 2$
$av : 5$	$ud : 7$
$bc : -\infty$	$du : 2$
$cb : 6$	$uv : 5$

In this talk we will consider the all pairs max bottlenecks problem: for all pairs of vertices s and t in the graph, find the weight of the maximum bottleneck edge on a path from s to t.

All pairs bottleneck paths – related work

- Pollack 1960: introduced APBP and showed a cubic algorithm.
- Hu 1961: undirected, edge weighted – max spanning tree. $\leftarrow O(n^2)$
- Shapira, Yuster, Zwick 2007: directed, node weighted in $O(n^{2.58})$.
- this work: directed, edge weighted in $O(n^{2.79})$.

MaxMin product

The MaxMin product of two $n \times n$ matrices A and B is

$$(A \bullet B)[i,j] = \max_k \min\{A[i,k], B[k,j]\}.$$

MaxMin product

The MaxMin product of two $n \times n$ matrices A and B is

$$(A \bullet B)[i,j] = \max_k \min\{A[i,k], B[k,j]\}.$$

Adjacency matrix for weighted graph $G = (V, E, w)$:

$A[i,j] = w_{ij}$, $w_{ii} = \infty$, $w(i,j) = -\infty$ if $(i,j) \notin E$.

$(A \bullet A)[i,j]$ is the maximum bottleneck edge weight over all paths of length ≤ 2 from i to j.

$\underbrace{A \bullet A \bullet \ldots \bullet A}_{n \text{ times}}$: the maximum bottleneck weights for all vertex pairs.

MaxMin product

MaxMin product

The MaxMin product is used to compute all pairs maximum bottleneck paths (APBP), similar to how one uses distance product for APSP.

MaxMin product

The MaxMin product is used to compute all pairs maximum bottleneck paths (APBP), similar to how one uses distance product for APSP.

Moreover: computing the MaxMin product of two $n \times n$ matrices takes the same time (asymptotically) as computing all pairs bottleneck weights in an n vertex graph. [AhoHopcroftUllman74]

MaxMin product

The MaxMin product is used to compute all pairs maximum bottleneck paths (APBP), similar to how one uses distance product for APSP.

Moreover: computing the MaxMin product of two $n \times n$ matrices takes the same time (asymptotically) as computing all pairs bottleneck weights in an n vertex graph. [AhoHopcroftUllman74]

This work: first truly subcubic algorithm for the MaxMin product.

MaxMin product in subcubic time

MaxMin: $C = (A \bullet B)[i,j] = \max_k \min\{A[i,k], B[k,j]\}$

We will proceed as follows:

MaxMin product in subcubic time

MaxMin: $C = (A \bullet B)[i,j] = \max_k \min\{A[i,k], B[k,j]\}$

We will proceed as follows:

1. compute for all i, j, $a_{ij} = \max_k \{A[i,k] \mid A[i,k] \leq B[k,j]\}$,
2. compute for all i, j, $b_{ij} = \max_k \{B[k,j] \mid B[k,j] \leq A[i,k]\}$,

MaxMin product in subcubic time

MaxMin: $C = (A \bullet B)[i,j] = \max_k \min\{A[i,k], B[k,j]\}$

We will proceed as follows:

1. compute for all i, j, $a_{ij} = \max_k \{A[i,k] \mid A[i,k] \leq B[k,j]\}$,
2. compute for all i, j, $b_{ij} = \max_k \{B[k,j] \mid B[k,j] \leq A[i,k]\}$,
3. set for all i, j, $C[i,j] = \max\{a_{ij}, b_{ij}\}$.

Dominance product

want to compute for all i, j, $a_{ij} = \max_k \{A[i,k] \mid A[i,k] \leq B[k,j]\}$

Dominance product

want to compute for all i, j, $a_{ij} = \max_k \{A[i,k] \mid A[i,k] \leq B[k,j]\}$

We will use the so–called dominance product of $n \times n$ matrices A and B:

$$(A \otimes B)[i,j] = |\{k \ : \ A[i,k] \leq B[k,j]\}|.$$

Dominance product

want to compute for all i, j, $a_{ij} = \max_k \{A[i,k] \mid A[i,k] \leq B[k,j]\}$

We will use the so–called dominance product of $n \times n$ matrices A and B:

$$(A \otimes B)[i,j] = |\{k \ : \ A[i,k] \leq B[k,j]\}|.$$

Thm. (Matousek) Dominance Product can be computed in $O(n^{(3+\omega)/2})$ time, where ω is the exponent of fast matrix multiplication.

Dominance product

want to compute for all i, j, $a_{ij} = \max_k\{A[i,k] \mid A[i,k] \leq B[k,j]\}$

We will use the so–called dominance product of $n \times n$ matrices A and B:

$$(A \otimes B)[i,j] = |\{k \ : \ A[i,k] \leq B[k,j]\}|.$$

Thm. (Matousek) Dominance Product can be computed in $O(n^{(3+\omega)/2})$ time, where ω is the exponent of fast matrix multiplication. $\leftarrow O(n^{2.69})$

MaxMin product in subcubic time

We want $a_{ij} = \max_k\{A[i,k] \mid A[i,k] \leq B[k,j]\}$.

1. Take the rows of A and sort the entries of each row.
2. Bucket the entries of each row of A, in their sorted order into s roughly equal buckets.

$$A = \begin{pmatrix} 10 & -1.1 & 5.1 & 3.2 \\ 2 & 3 & 7 & 1 \\ 0 & -1 & -2 & -3 \\ 7 & 2.1 & 4 & 2.1 \end{pmatrix}$$

$$\begin{array}{lllll} \text{row } 1: & A[1,2], & A[1,4], & A[1,3], & A[1,1] \\ \text{row } 2: & A[2,4], & A[2,1], & A[2,2], & A[2,3] \\ \text{row } 3: & A[3,4], & A[3,3], & A[3,2], & A[3,1] \\ \text{row } 4: & A[4,4], & A[4,2], & A[4,3], & A[4,1] \end{array}$$

MaxMin product in subcubic time

3. For each bucket b create a matrix $A(b)$ containing only the elements in bucket b and ∞ in all other entries.

$$A(1) = \begin{pmatrix} \infty & -1.1 & \infty & 3.2 \\ 2 & \infty & \infty & 1 \\ \infty & \infty & -2 & -3 \\ \infty & 2.1 & \infty & 2.1 \end{pmatrix} \quad A(2) = \begin{pmatrix} 10 & \infty & 5.1 & \infty \\ \infty & 3 & 7 & \infty \\ 0 & -1 & \infty & \infty \\ 7 & \infty & 4 & \infty \end{pmatrix}$$

MaxMin product in subcubic time

Recall, $(A \otimes B)[i,j] = |\{k : A[i,k] \leq B[k,j]\}|$.

4. Compute $A(b) \otimes B$ for each bucket b.

$$A(2) \otimes A = \begin{pmatrix} 10 & \infty & 5.1 & \infty \\ \infty & 3 & 7 & \infty \\ 0 & -1 & \infty & \infty \\ 7 & \infty & 4 & \infty \end{pmatrix} \otimes \begin{pmatrix} 10 & -1.1 & 5.1 & 3.2 \\ 2 & 3 & 7 & 1 \\ 0 & -1 & -2 & -3 \\ 7 & 2.1 & 4 & 2.1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 2 & 1 & 2 & 2 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

This tells us for every bucket b and each i, j, the number of coords k such that $A[i,k]$ is in bucket b and $A[i,k] \leq B[k,j]$.

This step takes $O(sn^{\frac{3+\omega}{2}})$.

MaxMin product in subcubic time

MaxMin product in subcubic time

5. For each i, j we find the largest bucket b in which there is an entry $A[i,k]$ such that $A[i,k] \leq B[k,j]$.

MaxMin product in subcubic time

5. For each i, j we find the largest bucket b in which there is an entry $A[i, k]$ such that $A[i, k] \leq B[k, j]$.

   For each i, j, search that bucket for k - there are at most $O(n/s)$ entries we have to go through for each pair i, j.

   This step takes $O(n^3/s)$ and explicitly finds witnesses.

MaxMin product in subcubic time

5. For each i, j we find the largest bucket b in which there is an entry $A[i,k]$ such that $A[i,k] \leq B[k,j]$.

   For each i, j, search that bucket for k - there are at most $O(n/s)$ entries we have to go through for each pair i, j.

   This step takes $O(n^3/s)$ and explicitly finds witnesses.

6. The overall runtime is maximized for $s = n^{\frac{3-\omega}{4}}$ and the runtime is then $O(n^{\frac{9+\omega}{4}}) = O(n^{2.85})$.

MaxMin product in subcubic time

5. For each i, j we find the largest bucket b in which there is an entry $A[i, k]$ such that $A[i, k] \leq B[k, j]$.

   For each i, j, search that bucket for k - there are at most $O(n/s)$ entries we have to go through for each pair i, j.

   This step takes $O(n^3/s)$ and explicitly finds witnesses.

6. The overall runtime is maximized for $s = n^{\frac{3-\omega}{4}}$ and the runtime is then $O(n^{\frac{9+\omega}{4}}) = O(n^{2.85})$.

7. You can do slightly better by using sparse dominance $\longrightarrow O(n^{2.79})$.

Sparse dominance

Theorem: Let A and B be $n \times n$ matrices with entries from a totally ordered set. Let $S \subseteq [n] \times [n]$ such that $|S| = m \geq n$. Let C be the matrix such that

$$C[i,j] = |\{k \mid (i,k) \in S \text{ and } A[i,k] \leq B[k,j]\}|.$$

There is an algorithm that, given A, B, and S, outputs C in $O(\sqrt{m} \cdot n^{\frac{1+\omega}{2}})$ time.

Intuition: The set S of coordinate pairs contains all entries of A we care about. Comparisons between entries of A not in S and entries of B are ignored.

MaxMin Product

Recall the matrices $A(b)$:

$$A(1) = \begin{pmatrix} \infty & -1.1 & \infty & 3.2 \\ 2 & \infty & \infty & 1 \\ \infty & \infty & -2 & -3 \\ \infty & 2.1 & \infty & 2.1 \end{pmatrix}$$

MaxMin Product

Recall the matrices $A(b)$:

$$A(1) = \begin{pmatrix} \infty & -1.1 & \infty & 3.2 \\ 2 & \infty & \infty & 1 \\ \infty & \infty & -2 & -3 \\ \infty & 2.1 & \infty & 2.1 \end{pmatrix}$$

$A(b)$ has $O(n^2/s)$ finite entries. Each of the s dominance products thus takes $O(n^{\frac{3+\omega}{2}}/\sqrt{s})$, and the running time for the entire algorithm is: $O(n^3/s + \sqrt{s}n^{\frac{3+\omega}{2}})$, minimized for $s = n^{1-\omega/3}$.

Conclusion

Conclusion

Corollary: The MaxMin product of $n \times n$ matrices A and B, and hence APBP can be computed in $O(n^{2+\frac{\omega}{3}}) = O(n^{2.79})$.

Conclusion

Corollary: The MaxMin product of $n \times n$ matrices A and B, and hence APBP can be computed in $O(n^{2+\frac{\omega}{3}}) = O(n^{2.79})$.

In $O(n^{2+\frac{\omega}{3}} \log n)$ time one can obtain a witness matrix from which one can obtain actual paths in time linear in their length.

Conclusion

Corollary: The MaxMin product of $n \times n$ matrices A and B, and hence APBP can be computed in $O(n^{2+\frac{\omega}{3}}) = O(n^{2.79})$.

In $O(n^{2+\frac{\omega}{3}} \log n)$ time one can obtain a witness matrix from which one can obtain actual paths in time linear in their length.

Open Problems

1. dominance product, MaxMin product in $O(n^{\omega})$?
2. truly subcubic distance product using dominance product?

Thank You!