Flexible Network Vertex Connectivity Algorithm Design

Dynamic Vertex Connectivity

Piotr Sankowski

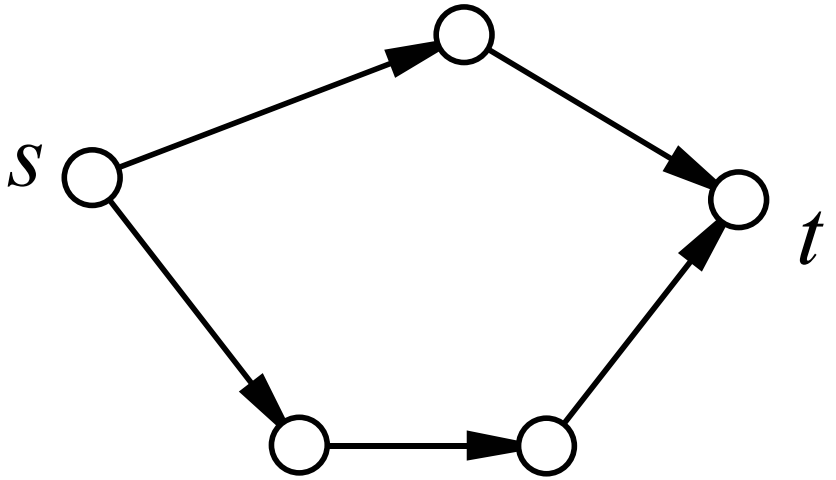
Departamento di Informatica e Sistemistica
University of Rome "La Sapienza"

Outline

- **Introduction**
- Simple Solution
- Dynamic Cycle Cover
- Dynamic s, t-Vertex Connectivity
  - Vertex Connectivity and Maximum Matchings
- Dynamic Matrix Rank
- Static k-Vertex Connectivity
- Dynamic k-Vertex Connectivity
- Conclusions and Open Problems

Dynamic Vertex Connectivity

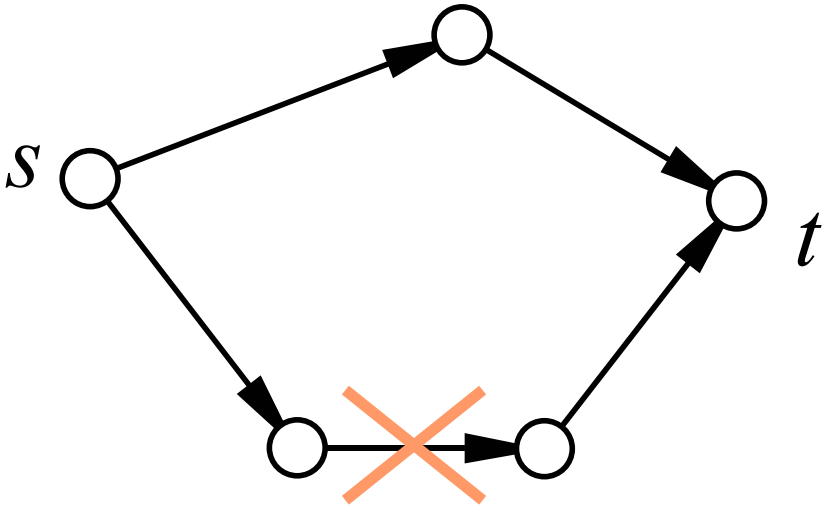
For the dynamic graph:



we want to compute the number of vertex disjoint paths from s to t? 2

Dynamic Vertex Connectivity

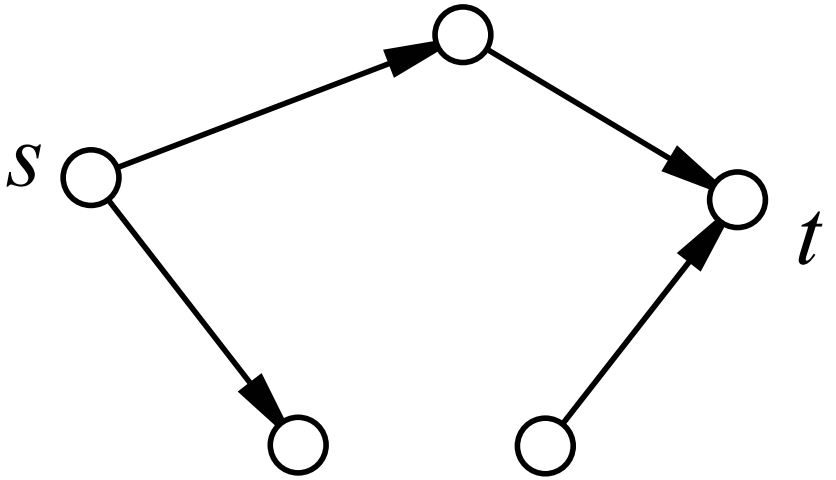
For the dynamic graph:



we want to compute the number of vertex disjoint paths from s to t?

Dynamic Vertex Connectivity

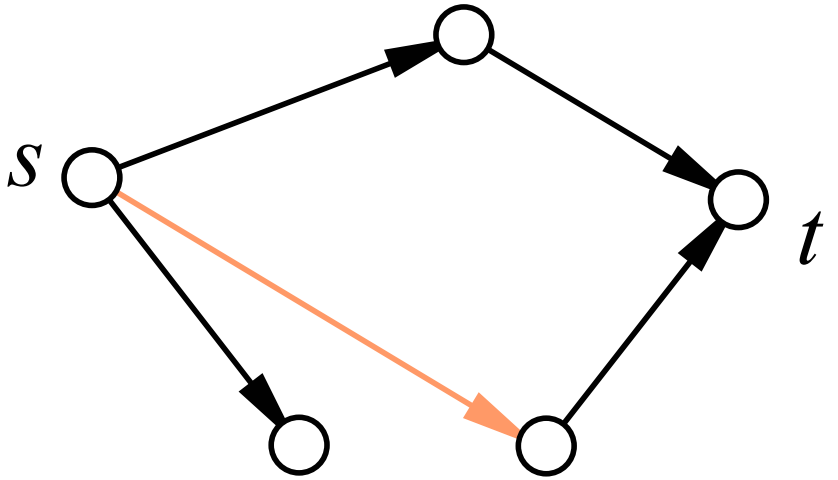
For the dynamic graph:



we want to compute the number of vertex disjoint paths from s to t? 1

Dynamic Vertex Connectivity

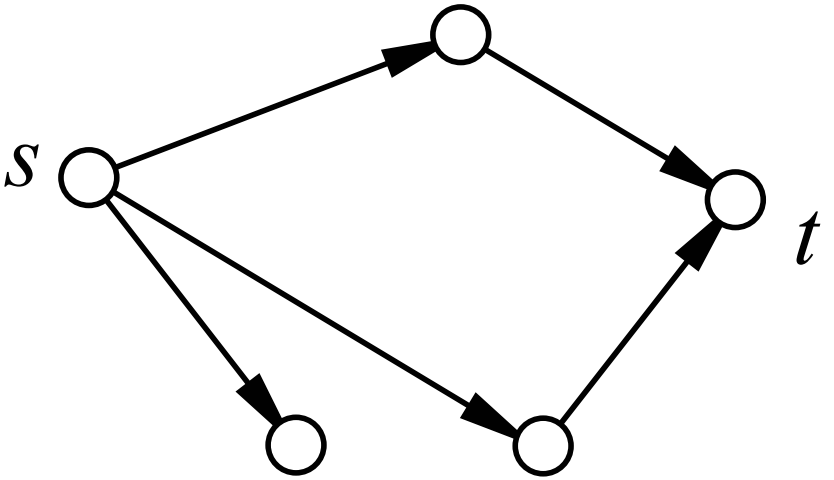
For the dynamic graph:



we want to compute the number of vertex disjoint paths from s to t?

Dynamic Vertex Connectivity

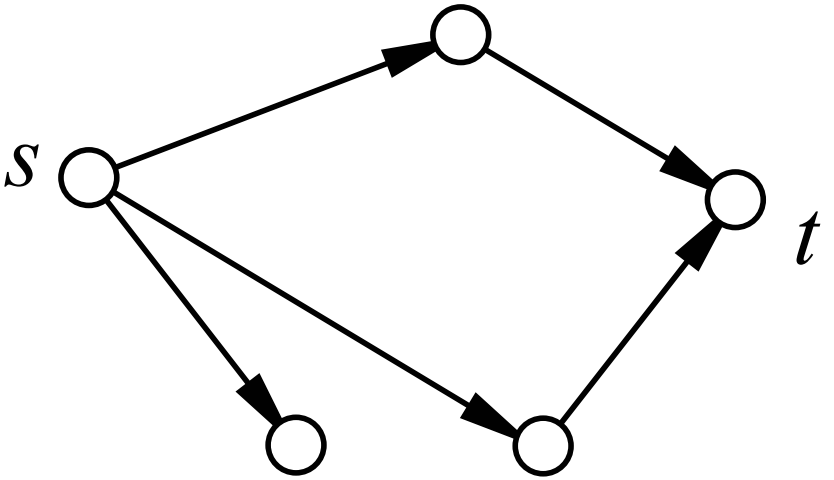
For the dynamic graph:



we want to compute the number of vertex disjoint paths from s to t? 2

Dynamic Vertex Connectivity

For the dynamic graph:



we want to compute the number of vertex disjoint paths from s to t? 2

The s and t vertices can be common for these paths.

Dynamic Vertex Connectivity

An algorithm maintains the directed graph $G = (V, E)$ two vertices s, t and supports the following operations:

- `insert`(e) – inserts the edge e,
- `delete`(e) – deletes the edge e,
- `paths` – computes the number of vertex disjoint paths between s and t,
- k-`connected` – checks if the graph is k-vertex connected.

For `paths` *a simple* $O(m)$ *solution exists.*

For k-`connected` *only solutions for small* k *are known.*

Dynamic Results

Dynamic k-vertex connectivity:

- undirected connectivity and 2-vertex connectivity in $\tilde{O}(1)$ time by Holm *et al.* '00.
- undirected 3-vertex connectivity and 4-vertex connectivity in $\tilde{O}(n)$ time by Eppstein *et al.* '97.
- directed 1-vertex connectivity, i.e., strong connectivity in $O(n^{1.575})$ time using the transitive closure algorithm by Sankowski '04.

Static Results

Static k-vertex connectivity:

Complexity	Author
$\tilde{O}(n^\omega + nk^\omega)$	*(undirected)* Linial, Lovász and Wigderson '88
$\tilde{O}(n^\omega + nk^\omega)$	Cheriyan and Reif '92
$O((k^{5/2} + n)m)$	Gabow '00
$O((k + n^{1/4})n^{3/4}m)$	Gabow '00
$O((k^{5/2} + n)kn)$	*(undirected)* Gabow '00
$O((k + n^{1/4})kn^{7/4})$	*(undirected)* Gabow '00

Our Results

Our results:

- $O(n^{1.495})$ time algorithm for the `paths` problem, i.e., s, t-vertex connectivity,
- $O(n^{1.495})$ time algorithm for the maximum matching problem and for the matrix rank problem,
- $O(n^{1.575} + nk^2)$ time algorithm for the directed k-vertex connectivity,
- these solutions break the input size bound.

Outline

- Introduction
- **Simple Solution**
- Dynamic Cycle Cover
- Dynamic s, t-Vertex Connectivity
  - Vertex Connectivity and Maximum Matchings
- Dynamic Matrix Rank
- Static k-Vertex Connectivity
- Dynamic k-Vertex Connectivity
- Conclusions and Open Problems

Simple Solution

In the algorithm we maintain explicitly the current set of vertex disjoint paths P.

The insertion of an edge e can be realized as follows:

- insert e into the graph,
- if there exists an s, t augmenting path p,
  - augment P with the path p.

After insertion or removal of the edge the number of vertex disjoint paths can change by at most one.

Simple Solution

The removal of an edge e can be handled similarly:

- remove e from the graph,
- remove from P a path containing e if there is any,
- if there exists an s, t augmenting path p,
  - augment P with the path p.

The augmenting path can be found in $O(m)$ time, so we get an $O(m)$ update time algorithm.

We have to beat the input size bound - $O(m)$.

Outline

- Introduction
- Simple Solution
- **Dynamic Cycle Cover**
- Dynamic s, t-Vertex Connectivity
  - Vertex Connectivity and Maximum Matchings
- Dynamic Matrix Rank
- Static k-Vertex Connectivity
- Dynamic k-Vertex Connectivity
- Conclusions and Open Problems

Dynamic Perfect Cycle Cover

A *cycle cover* in a graph G is a set of cycles C such that every vertex belongs to at most one cycle $c \in C$.

A *size* of a cycle cover C is defined as $|C| = \sum_{c \in C} |c|$.

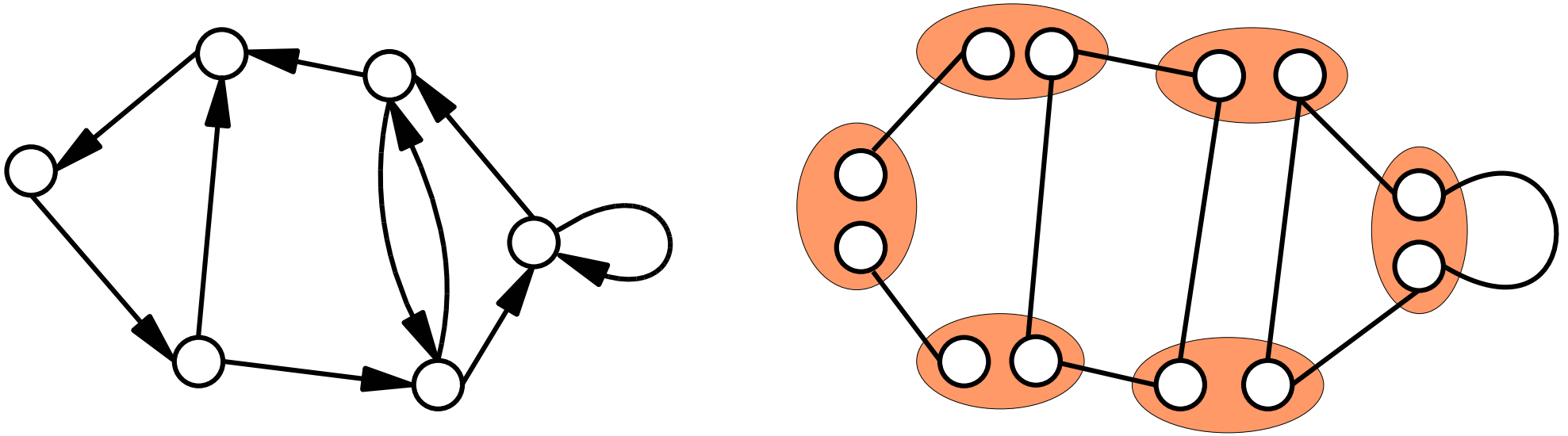
A *perfect cycle cover* is a cycle cover of size $|V|$.

An algorithm for dynamic perfect cycle cover problem maintains the graph $G = (V, E)$ and supports the following operations:

- `insert`(e) – inserts the edge e,
- `delete`(e) – if $G - e$ has a perfect cycle cover then deletes the edge e otherwise returns `FALIURE`.

Perfect Covers = Perfect Matchings

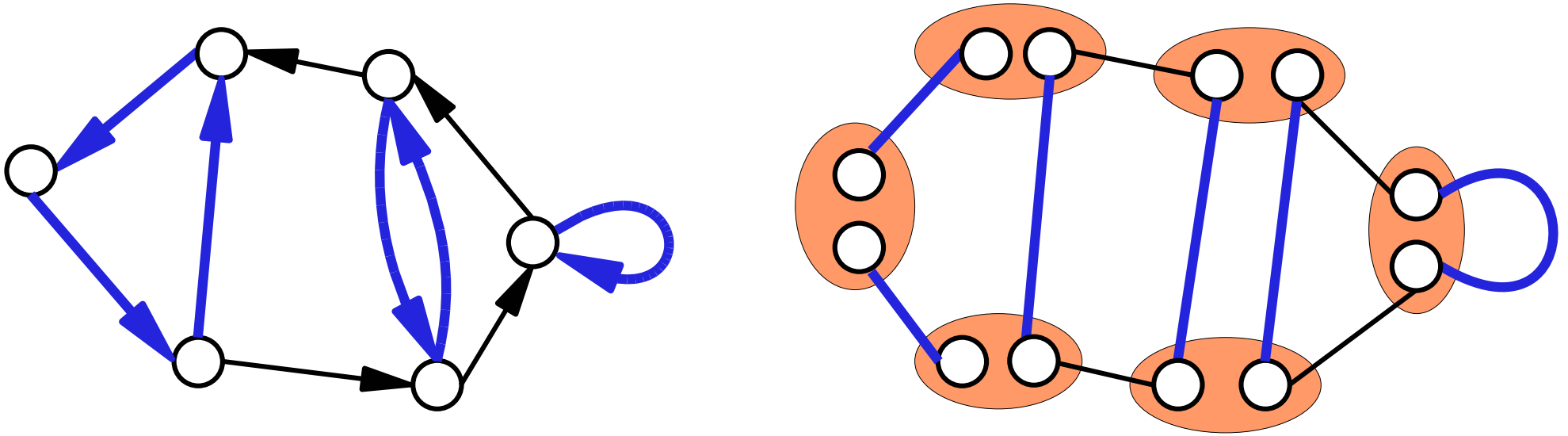
For each vertex we create two copies:



- one for in-edges,
- one for out-edges.

Perfect Covers = Perfect Matchings

For each vertex we create two copies:



- one for in-edges,
- one for out-edges.

Dynamic Perfect Cycle Cover

Theorem 1 (Sankowski '04) *The problems of dynamic cycle cover can be solved with the following costs:*

- `initialization` – $O(n^\omega)$ *time,*
- `insert` *and* `delete` – $O(n^{1.495})$ *time (worst-case).*

The algorithm is randomized.

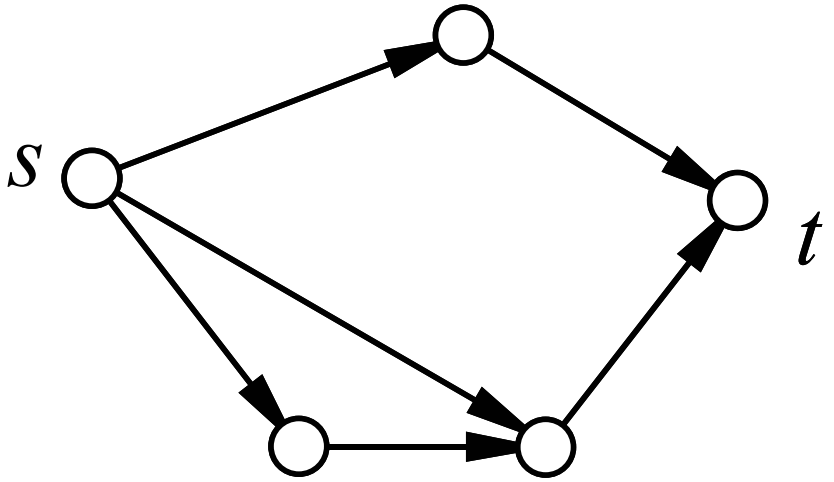
$O(n^\omega)$ is the matrix multiplication time – currently $\omega < 2.38$.

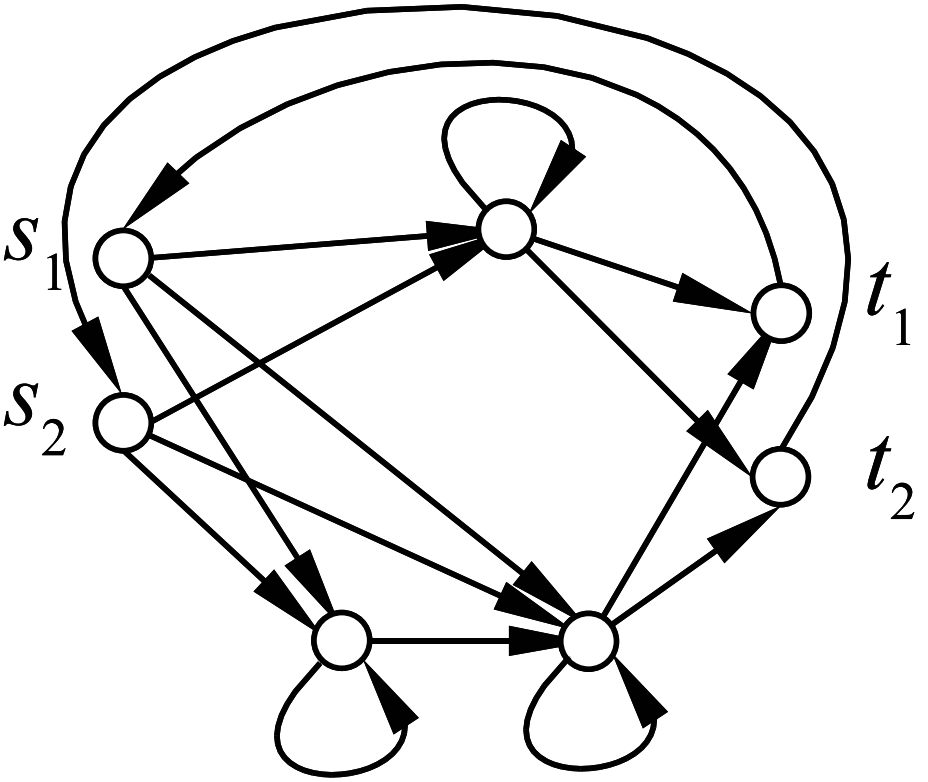
Outline

- Introduction
- Simple Solution
- Dynamic Cycle Cover
- **Dynamic s, t-Vertex Connectivity**
  - Vertex Connectivity and Maximum Matchings
- Dynamic Matrix Rank
- Static k-Vertex Connectivity
- Dynamic k-Vertex Connectivity
- Conclusions and Open Problems

Dynamic s, t-Vertex Connectivity

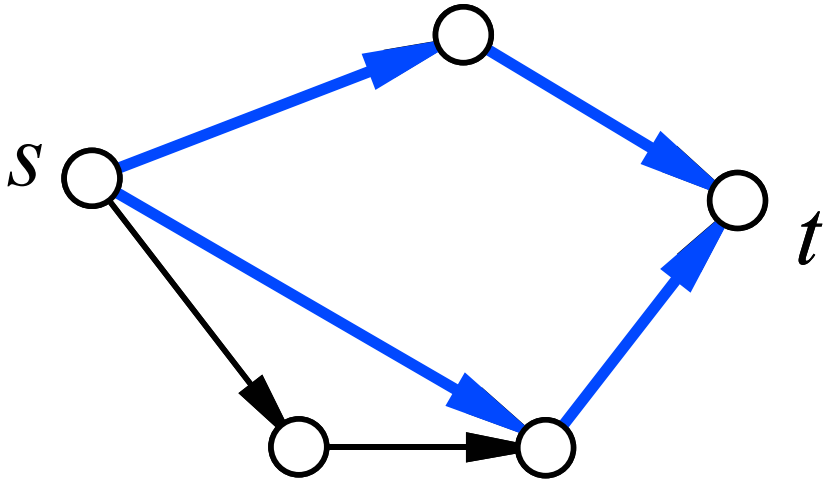
In order to check if there are ≥ 2 s, t-paths:

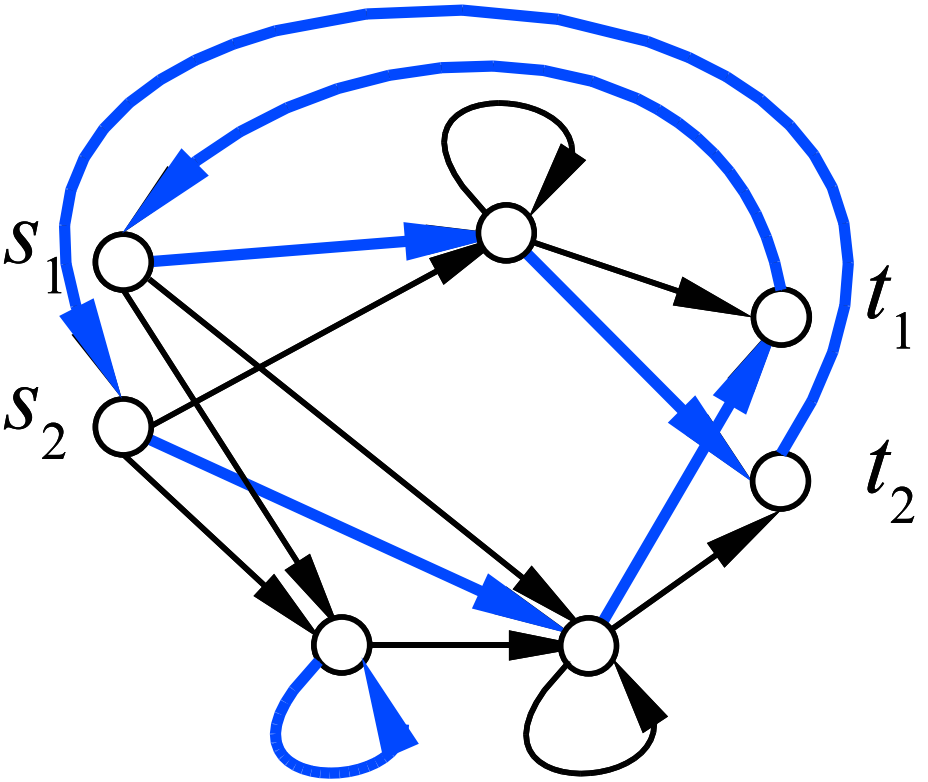




Dynamic s, t-Vertex Connectivity

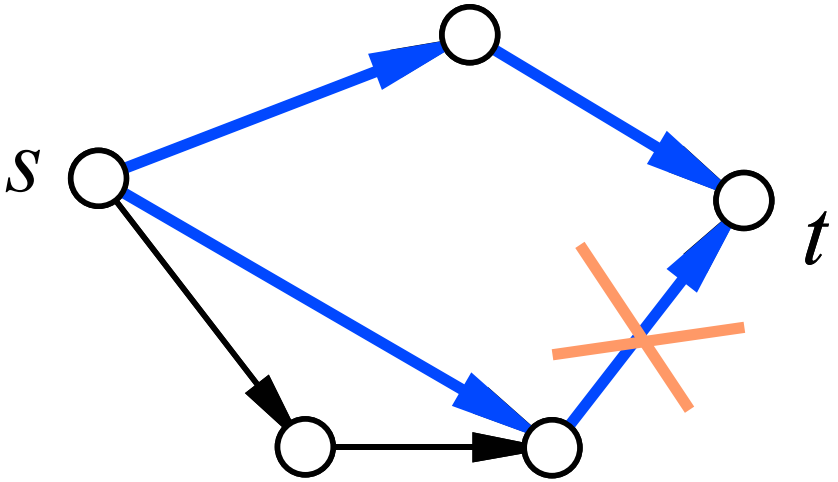
In order to check if there are ≥ 2 s, t-paths:

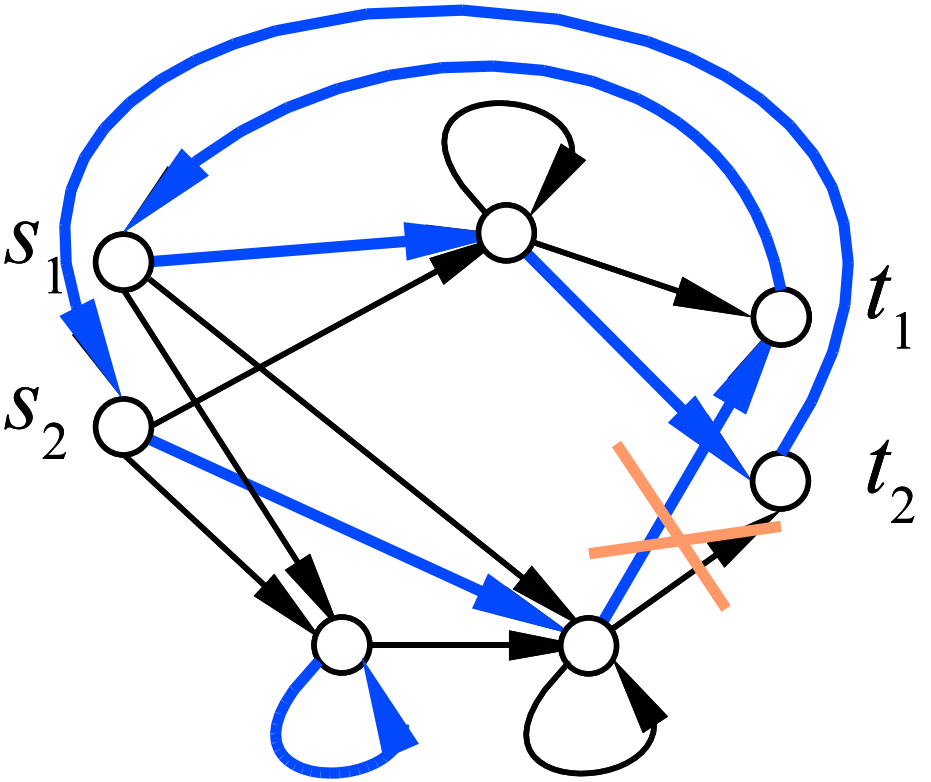




Dynamic s, t-Vertex Connectivity

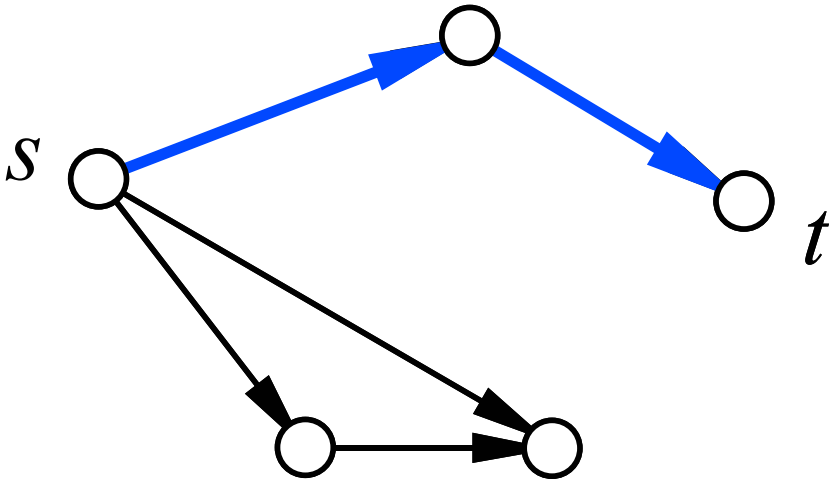
In order to check if there are ≥ 2 s, t-paths:

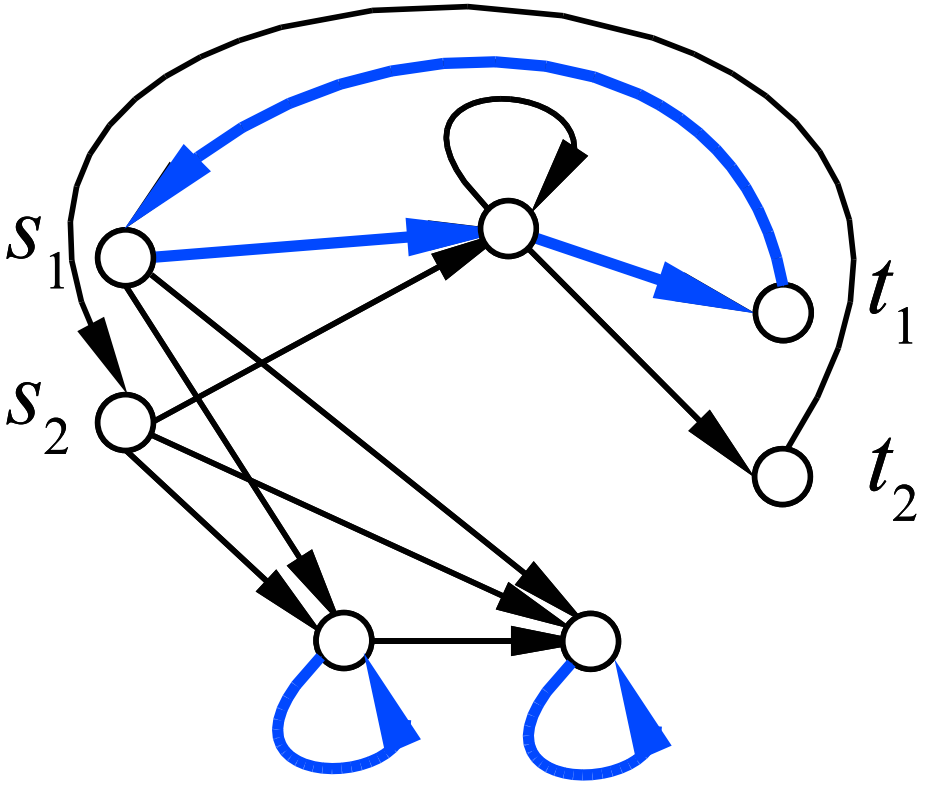




Dynamic s, t-Vertex Connectivity

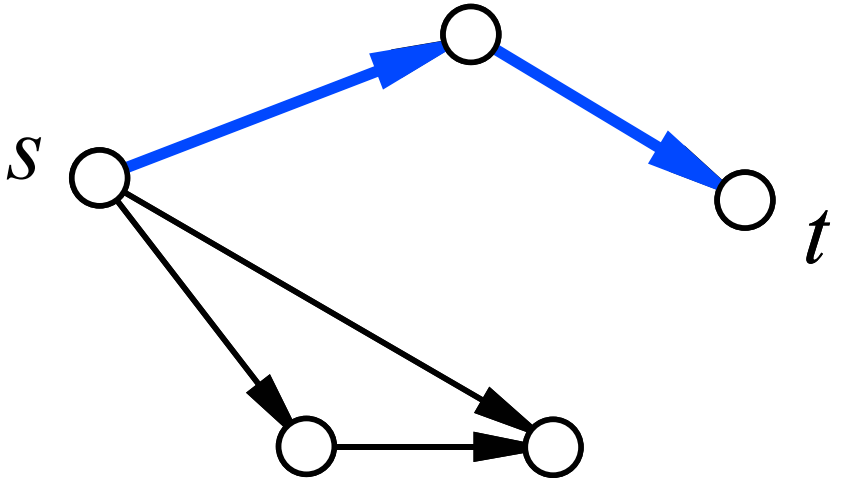
In order to check if there are ≥ 2 s, t-paths:

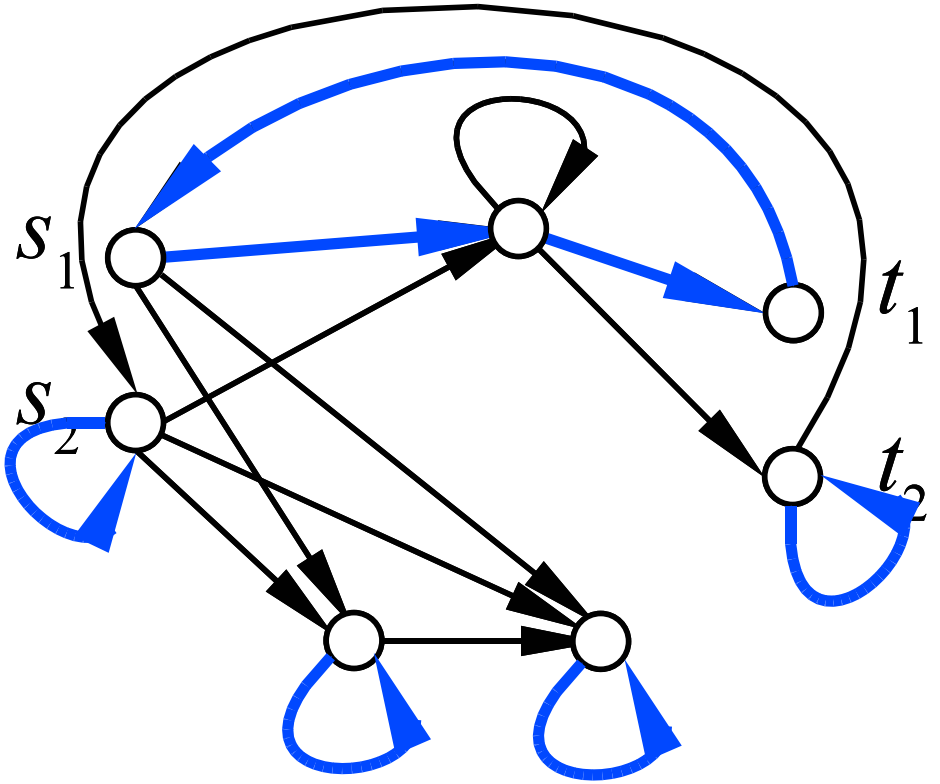




Dynamic s, t-Vertex Connectivity

In order to check if there are ≥ 2 s, t-paths:





Dynamic s, t-Vertex Connectivity

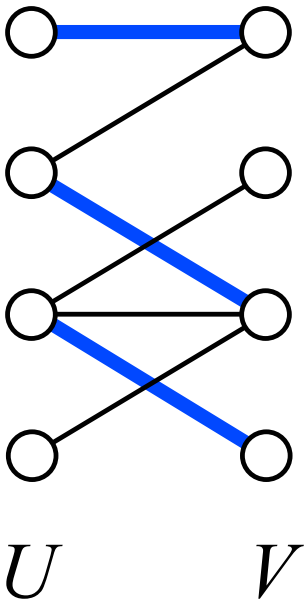
In order to dynamically compute s, t-vertex connectivity, we:

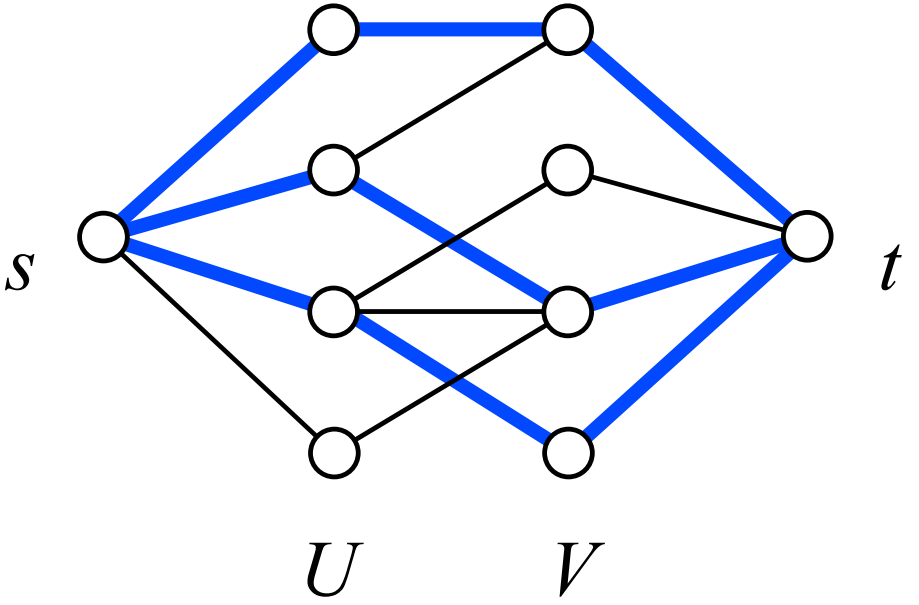
- copy n times the vertices s and t,
- add self-loops to all other vertices,
- add n edges from t to s.

When the s, t-vertex connectivity changes we need to add/delete self loops to the pair of copies of s and t.

Theorem 2 *There exists an algorithm for the problems of computing the number of vertex disjoint s, t-paths that supports updates in $O(n^{1.495})$ time (worst-case).*

Dynamic Maximum Matchings





Theorem 3 *There exists an algorithm for the problems of computing the size of the maximum matching that supports updates in $O(n^{1.495})$ time (worst-case).*

Outline

- Introduction
- Simple Solution
- Dynamic Cycle Cover
- Dynamic s, t-Vertex Connectivity
  - Vertex Connectivity and Maximum Matchings
- **Dynamic Matrix Rank**
- Static k-Vertex Connectivity
- Dynamic k-Vertex Connectivity
- Conclusions and Open Problems

Dynamic Matrix Rank

We have reduced the problem of dynamically computing the maximum matching to the problem of maintaining the perfect matchings.

We can do the same with the matrices, i.e., we enhance any dynamic algorithm that maintains the information on non-singular matrices to support also singular matrices.

A black-box reduction without loss of efficiency, only at the cost of randomization.

Dynamic Matrix Functions

We want to construct an algorithms that for a given $n \times n$ matrix A supports the following operations:

- `update`(i, j, x)`:` changes the element $A_{i,j}$ to x,
- `determinant:` returns $\det(A)$ the determinant of A,
- `rank:` returns $\text{rank}(A)$ the rank of A.

If the algorithm supports only non-singular matrices we cannot solve the `rank` problem.

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

After the change:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -4$$

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

After the change:

$$A = \begin{bmatrix} 2 & 1 & 2 \\ 2 & 2 & 2 \\ 1 & 2 & 2 \end{bmatrix} \qquad \det(A) = 2$$

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

After the change:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix} \qquad \det(A) = 0$$

Dynamic Matrix Functions

We are given the matrix:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} \qquad \det(A) = -2$$

After the change:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix} \qquad \det(A) = 0$$

Algorithm returns FALIURE.

Dynamic Matrix Functions

Theorem 4 (Sankowski '04) *The problem of dynamic matrix determinant with* **non-singular** *updates can be solved with the costs:*

- `initialization` – $O(n^{\omega})$ *time,*
- `update` – $O(n^{1.495})$ *time (worst-case).*

Theorem 5 (Frandsen and Frandsen 2006) *The problem of dynamic matrix rank can be solved with the costs:*

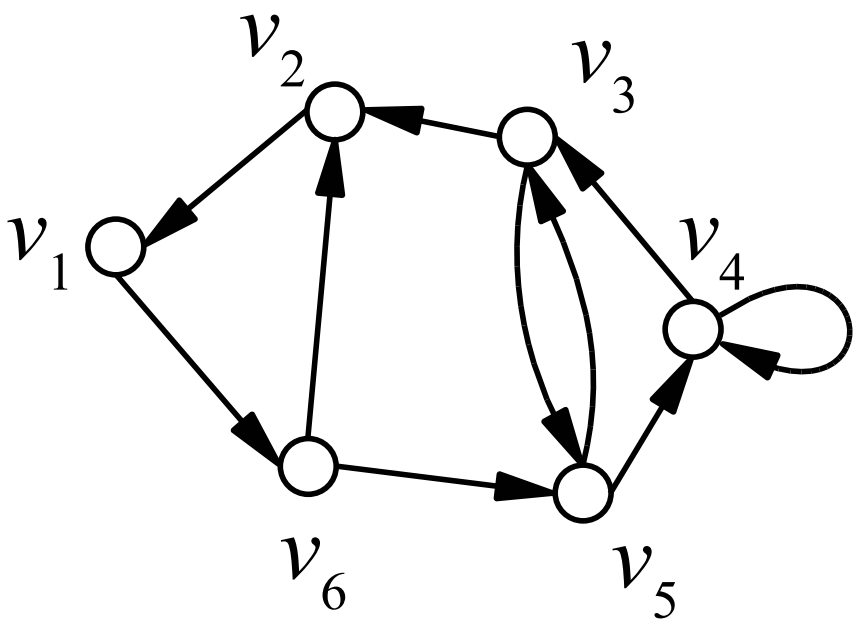
- `initialization` – $O(n^{\omega})$ *time,*
- `update` – $O(n^{1.595})$ *time (worst-case).*

A special more complicated solution.

Determinant = Cycle Cover

Consider the adjacency matrix of the graph:

G

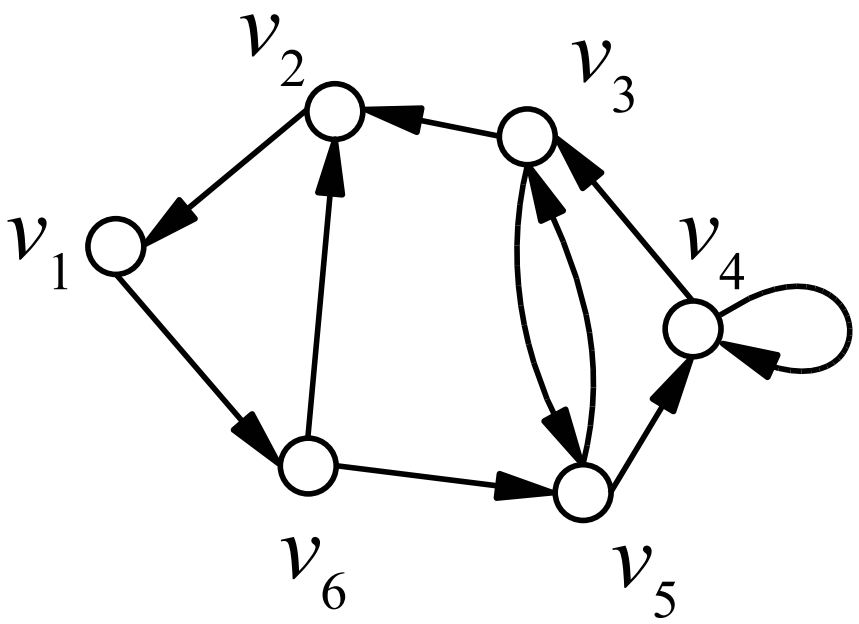


$$\Rightarrow \quad A(G) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & x_{1,6} \\ x_{2,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & x_{3,2} & 0 & 0 & x_{3,5} & 0 \\ 0 & 0 & x_{4,3} & x_{4,4} & 0 & 0 \\ 0 & 0 & x_{5,3} & x_{5,4} & 0 & 0 \\ 0 & x_{6,2} & 0 & 0 & x_{6,5} & 0 \end{pmatrix}$$

Determinant = Cycle Cover

Compute the determinant of the adjacency matrix:

G $\qquad\qquad$ $A(G)$



$$\Rightarrow \quad \det(A(G)) =$$

$$= \; x_{2,1}x_{1,6}x_{6,2}x_{3,5}x_{5,4}x_{4,3} +$$

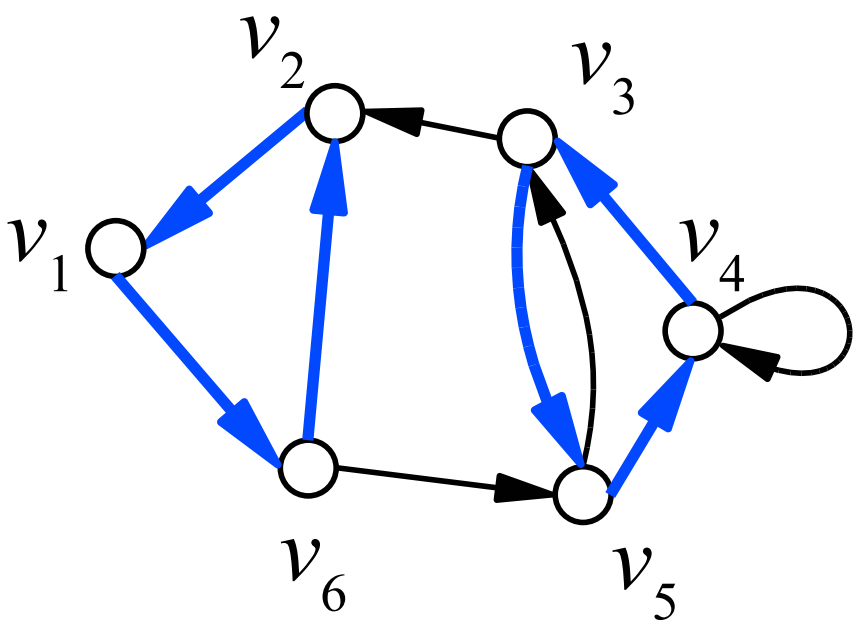
$$- \; x_{2,1}x_{1,6}x_{6,2}x_{3,5}x_{5,3}x_{4,4} +$$

$$- \; x_{2,1}x_{1,6}x_{6,5}x_{5,4}x_{4,3}x_{3,2}.$$

Determinant = Cycle Cover

Compute the determinant of the adjacency matrix:

G $\qquad\qquad$ $A(G)$



$$\Rightarrow \quad \det(A(G)) =$$

$$= \quad x_{2,1}x_{1,6}x_{6,2}x_{3,5}x_{5,4}x_{4,3} +$$

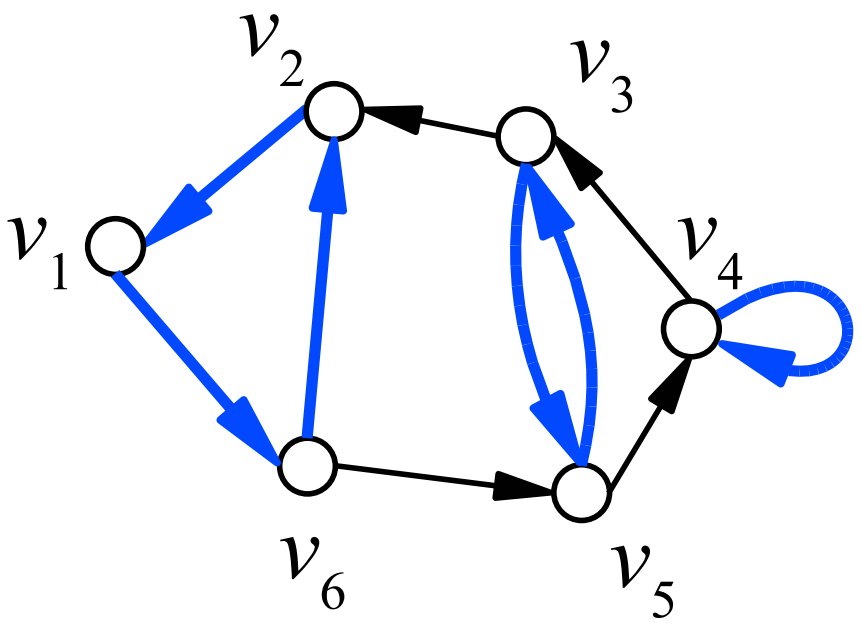
$$- \quad x_{2,1}x_{1,6}x_{6,2}x_{3,5}x_{5,3}x_{4,4} +$$

$$- \quad x_{2,1}x_{1,6}x_{6,5}x_{5,4}x_{4,3}x_{3,2}.$$

Determinant = Cycle Cover

Compute the determinant of the adjacency matrix:

G $\qquad\qquad$ $A(G)$



$$\Rightarrow \quad \det(A(G)) =$$

$$= \quad x_{2,1}x_{1,6}x_{6,2}x_{3,5}x_{5,4}x_{4,3} +$$

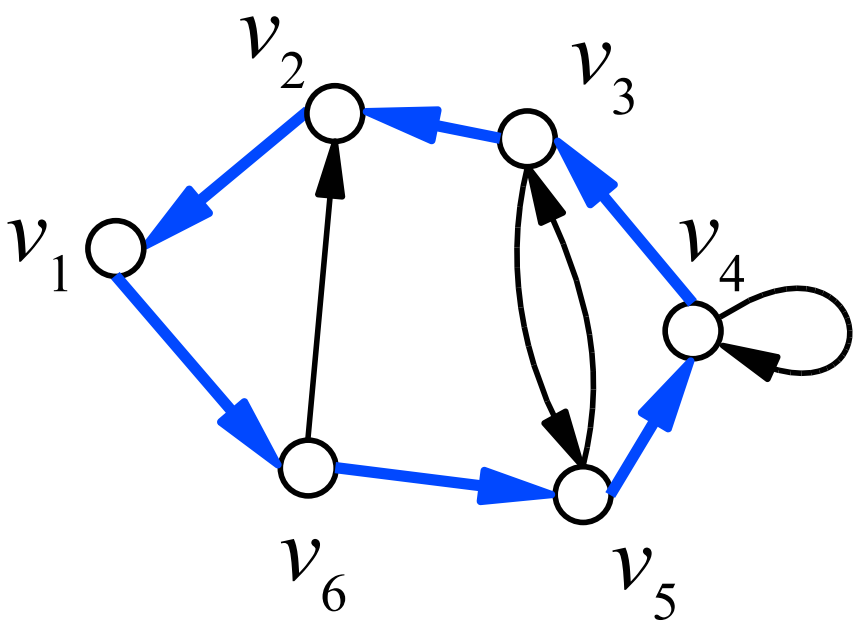
$$- \quad \mathbf{x_{2,1}x_{1,6}x_{6,2}x_{3,5}x_{5,3}x_{4,4}} +$$

$$- \quad x_{2,1}x_{1,6}x_{6,5}x_{5,4}x_{4,3}x_{3,2}.$$

Determinant = Cycle Cover

Compute the determinant of the adjacency matrix:

G $\qquad\qquad$ $A(G)$



$$\Rightarrow \quad \det(A(G)) =$$

$$= \; x_{2,1}x_{1,6}x_{6,2}x_{3,5}x_{5,4}x_{4,3} +$$

$$- \; x_{2,1}x_{1,6}x_{6,2}x_{3,5}x_{5,3}x_{4,4} +$$

$$- \; \mathbf{x_{2,1}x_{1,6}x_{6,5}x_{5,4}x_{4,3}x_{3,2}}.$$

Determinant = Cycle Cover

The determinant of an $n \times n$ matrix A is given as:

$$\det(A(G)) = \sum_{p \in \Pi_n} \sigma(p) \prod_{i=0}^{n} a_{i,p_i}.$$

In each product along p for each vertex i we choose an edge (i, p_i).

The cycles of p give perfect cycle covers of G.

Dynamic Matrix Rank

Let A be an $n \times n$ matrix over the field F. Let $\tilde{X}$ and $\tilde{Y}$ be symbolic matrices of size $n \times n$. Consider the matrix

$$\tilde{A}^{(k)} := \begin{bmatrix} A & \tilde{X} & 0 \\ \tilde{Y} & 0 & I \\ 0 & I & I^{(k)} \end{bmatrix},$$

where $I^{(k)} = \mathrm{diag}(\underbrace{1,\ldots,1}_{k \text{ times}},\underbrace{0,\ldots,0}_{n-k \text{ times}})$.

Lemma 6 *The matrix $A^{(k)}$ is non-singular if and only if* $\mathrm{rank}(A) \geq n - k$.

Dynamic Matrix Rank

Let $G(A)$ be a graph corresponding to the symbolic matrix A and let C be a maximum size cycle cover in $G(A)$, then:

$$\operatorname{rank}(A) = |C|.$$

When A is not symbolic this may be not true.

Can be solved with some randomization.

$$\tilde{A}^{(k)} := \begin{bmatrix} A & \tilde{X} & 0 \\ \tilde{Y} & 0 & I \\ 0 & I & I^{(k)} \end{bmatrix}$$

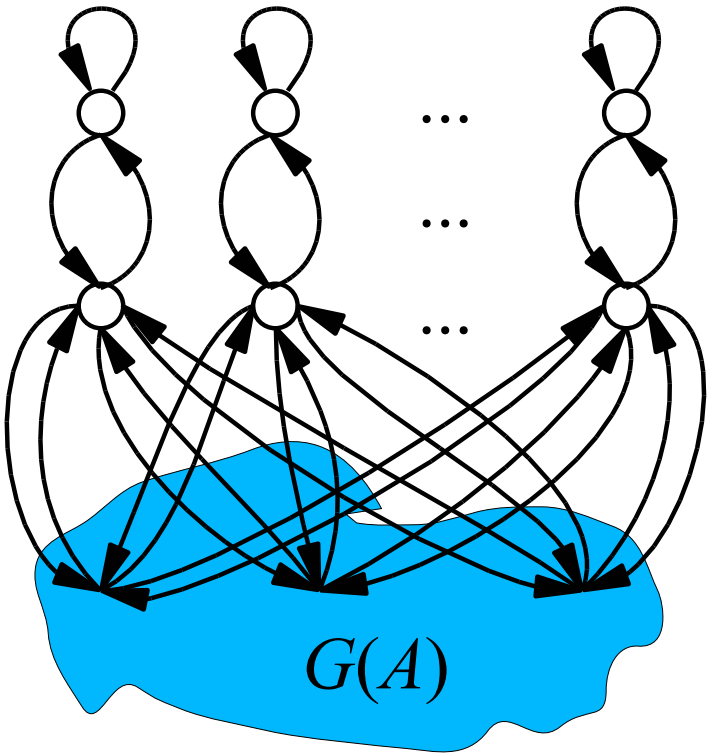
Dynamic Matrix Rank

Consider the matrix $A^{(k)}$ and the corresponding graph:

$\tilde{A}^{(n)}$

$$\begin{bmatrix} A & \tilde{X} & 0 \\ \tilde{Y} & 0 & I \\ 0 & I & I^{(n)} \end{bmatrix} \Rightarrow$$

$G(\tilde{A}^{(k)})$



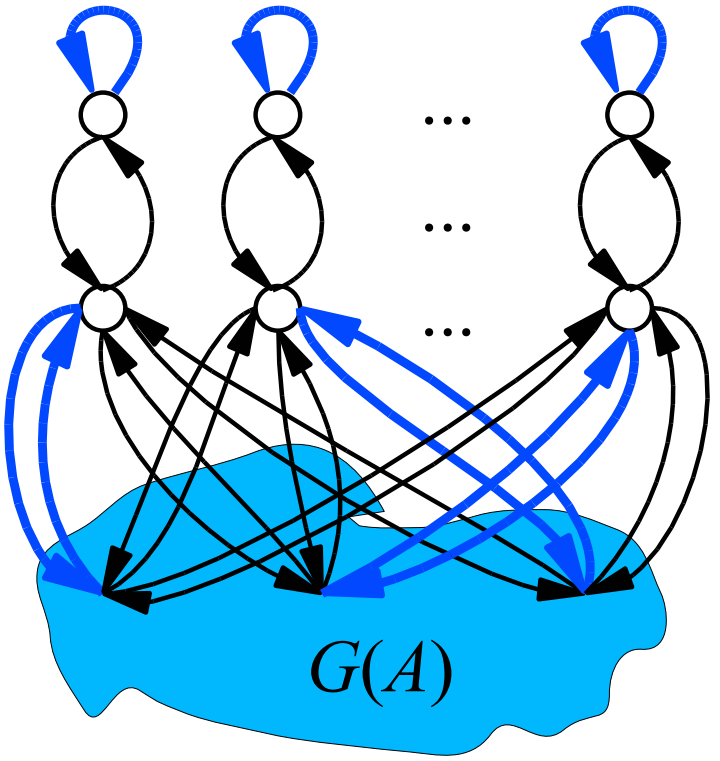
Dynamic Matrix Rank

Consider the matrix $A^{(k)}$ and the corresponding graph:

$$\tilde{A}^{(n)}$$

$$\begin{bmatrix} A & \tilde{X} & 0 \\ \tilde{Y} & 0 & I \\ 0 & I & I^{(n)} \end{bmatrix} \Rightarrow$$

$G(\tilde{A}^{(k)})$



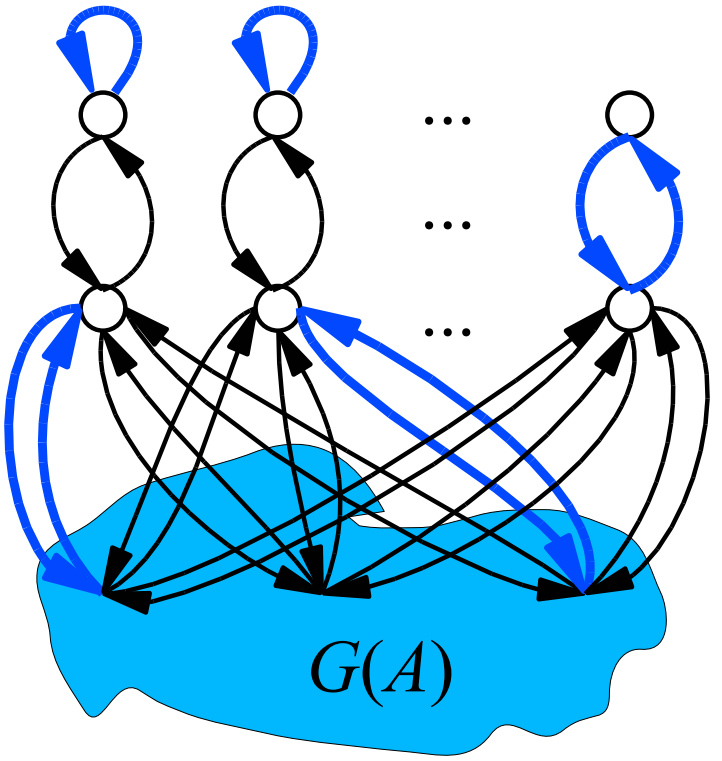
Dynamic Matrix Rank

Consider the matrix $A^{(k)}$ and the corresponding graph:

$$\tilde{A}^{(n-1)}$$

$$\begin{bmatrix} A & \tilde{X} & 0 \\ \tilde{Y} & 0 & I \\ 0 & I & I^{(n-1)} \end{bmatrix} \Rightarrow$$

$$G(\tilde{A}^{(k)})$$



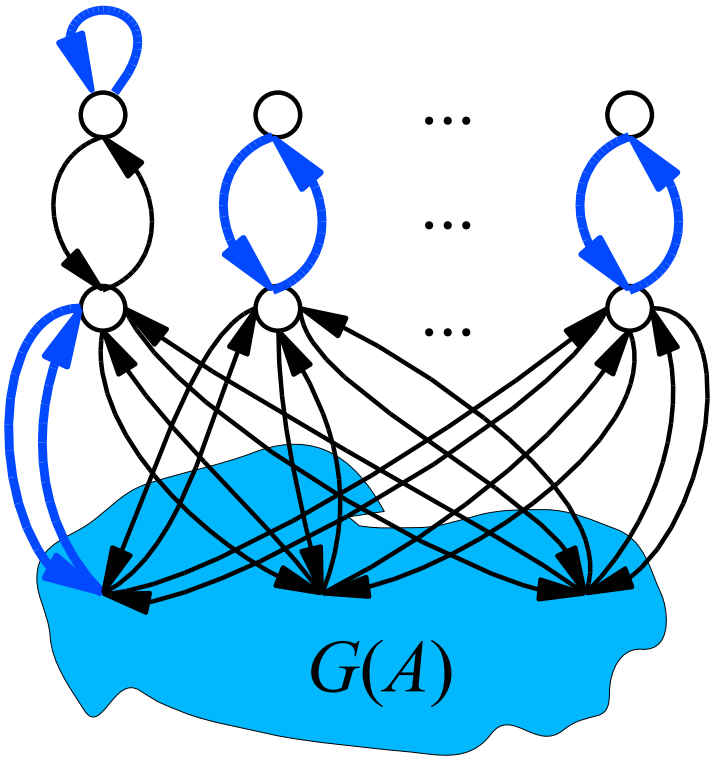
Dynamic Matrix Rank

Consider the matrix $A^{(k)}$ and the corresponding graph:

$$\tilde{A}^{(1)}$$

$$\begin{bmatrix} A & \tilde{X} & 0 \\ \tilde{Y} & 0 & I \\ 0 & I & I^{(1)} \end{bmatrix} \Rightarrow$$

$$G(\tilde{A}^{(k)})$$



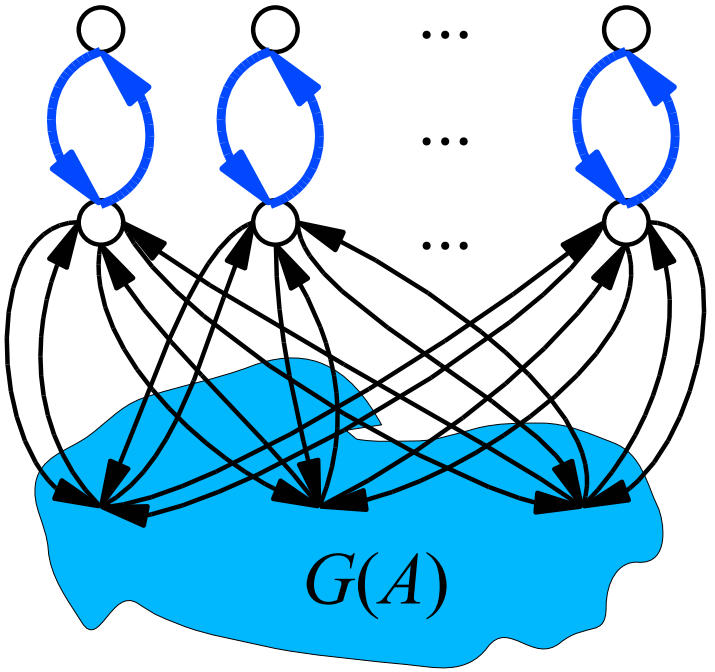
Dynamic Matrix Rank

Consider the matrix $A^{(k)}$ and the corresponding graph:

$$\tilde{A}^{(0)}$$

$$\begin{bmatrix} A & \tilde{X} & 0 \\ \tilde{Y} & 0 & I \\ 0 & I & I^{(0)} \end{bmatrix} \Rightarrow$$

$$G(\tilde{A}^{(k)})$$



Dynamic Transitive Closure

Theorem 7

Dynamic Matrix Inverse
Update $O(n^\alpha)$ operations
Query $O(n^\beta)$ operations
assumes non-singularity

Dynamic Matrix Rank
Update $O(n^\alpha)$ time
Query $O(n^\beta)$ time
randomized

Outline

- Introduction
- Simple Solution
- Dynamic Cycle Cover
- Dynamic s, t-Vertex Connectivity
  - Vertex Connectivity and Maximum Matchings
- Dynamic Matrix Rank
- **Static k-Vertex Connectivity**
- Dynamic k-Vertex Connectivity
- Conclusions and Open Problems

Static Vertex Connectivity

Let us denote by $\Gamma(v)$ the set $\{w : (v, w) \in E\}$.

A *symbolic Laplacian matrix* of the directed graph $G = (V, E)$ is the $n \times n$ matrix $\tilde{A}$ such that

$$\tilde{A}_{i,j} = \begin{cases} z_i & \text{if } i = j, \\ x_{i,j} & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

where $x_{i,j}$ are unique variables corresponding to the edges of G and z_i are unique variables corresponding to the vertices of G.

Static Vertex Connectivity

We say that $X, Y \subseteq V$, $|X| = |Y|$, are k-linked if there exist a set of vertex disjoint paths joining k vertices from X with k vertices from Y.

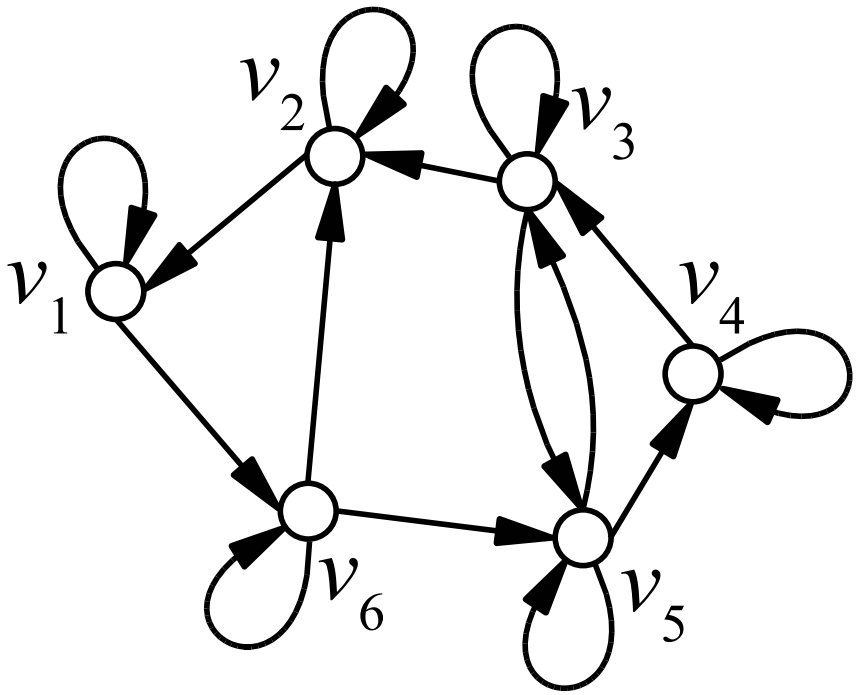
Theorem 8 (Cheriyan and Reif '92) *Let $\tilde{A}$ be the symbolic Laplacian matrix of a graph G. Let $X, Y \subseteq V$ be two k element sets of vertices. Then $\det(\tilde{A}_{\overline{X},\overline{Y}}) \neq 0$ iff X and Y are k-linked.*

In the undirected case proven by Linial, Lovász and Wigderson '88

Static Vertex Connectivity

In order to check if $X = \{v_5, v_6\}$ is k-linked to $Y = \{v_2, v_3\}$ we compute $\det(\tilde{A}_{\overline{X},\overline{Y}})$.

G



$\Rightarrow$

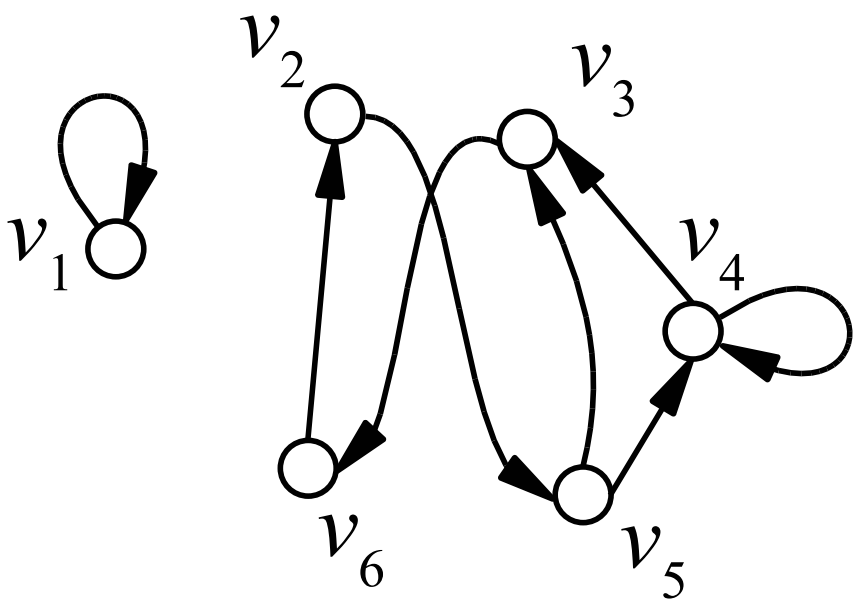
$\tilde{A}(G)$

$$\left(\begin{array}{cccc|cc} z_1 & 0 & 0 & 0 & 0 & x_{1,6} \\ \hline x_{2,1} & z_2 & 0 & 0 & 0 & 0 \\ 0 & x_{3,2} & z_3 & 0 & x_{3,5} & 0 \\ \hline 0 & 0 & x_{4,3} & z_4 & 0 & 0 \\ 0 & 0 & x_{5,3} & x_{5,4} & z_5 & 0 \\ 0 & x_{6,2} & 0 & 0 & x_{6,5} & z_6 \end{array}\right)$$

Static Vertex Connectivity

In order to check if $X = \{v_5, v_6\}$ is k-linked to $Y = \{v_2, v_3\}$ we compute $\det(\tilde{A}_{\overline{X},\overline{Y}})$.

G



$\Leftarrow$

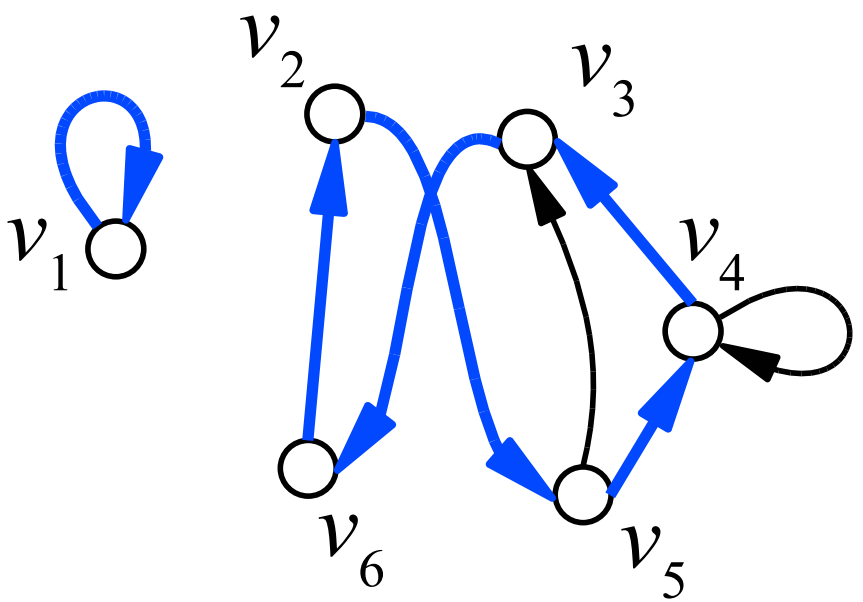
$\tilde{A}(G)_{\overline{X},\overline{Y}}$

$$\left(\begin{array}{cccc|cc} z_1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & x_{4,3} & z_4 & 0 & 0 \\ 0 & 0 & x_{5,3} & x_{5,4} & 0 & 0 \\ 0 & x_{6,2} & 0 & 0 & 0 & 0 \end{array}\right)$$

Static Vertex Connectivity

In order to check if $X = \{v_5, v_6\}$ is k-linked to $Y = \{v_2, v_3\}$ we compute $\det(\tilde{A}_{\overline{X},\overline{Y}})$.

G



$$\det(\tilde{A}(G)_{\overline{X},\overline{Y}}) = - z_1 x_{6,2} x_{5,4} x_{4,3} + + z_1 z_4 x_{6,2} x_{5,3}.$$

Static Vertex Connectivity

The theorem together with Zippel-Schwartz Lemma implies that we can test whether X and Y are k-linked in $O(n^\omega)$ time.

In the k connectivity test we perform many k-link tests. We can use the following.

Lemma 9
$\det(A_{\overline{X},\overline{Y}}) \neq 0$ *iff* $\det((A^{-1})_{X,Y}) \neq 0$.

Thus we can compute the inverse once and than test only if $k \times k$ size matrices are non-singular.

Static Vertex Connectivity

Lemma 10 *If $p(x_1, \ldots, x_m)$ is a non-zero polynomial of degree d with coefficients in a field F, then the probability that p evaluates to 0 on a random element $(s_1, s_2, \ldots, s_m) \in F^m$ is at most $d/|F|$. We call such event false zero.*

Corollary 11 *If a polynomial of degree n is evaluated on random values modulo prime number p of length $(1+c)\log n$, then the probability of false zero is at most $\frac{1}{n^c}$, for any $c > 0$.*

Static Vertex Connectivity

In the algorithm we use the following procedure to test if the given vertex v can be reached with k paths from all other vertices.

Algorithm 12 *TEST-ROOT-k(z, G):*

- *let Y be the set of k predecessors of z,*
- **for all** $v \in V - Y$ **do**
  - *let X be the set of k successors of v,*
  - *(we test if X and Y are k-linked)*
  - **if** $\text{rank}((A^{-1})_{X,Y}) < k$ **then** *return FALIURE,*
- *return SUCCESS.*

Static Vertex Connectivity

Algorithm 13 *TEST-k-CONNECTIVITY(G):*

- *choose a random prime* $p \in [n^6, n^7]$,
- *substitute random elements from* Z_p *into* $\tilde{A}$ *obtaining* A,
- **if** A *is singular* **then** *return FALIURE,*
- *compute* A^{-1} *and let* $k' = \lceil \frac{\log n}{\log n - \log k} \rceil$,
- **for** $i = 1$ *to* k' **do**
  - *choose a random vertex* $y_i \in V$,
  - *call TEST-ROOT-k*(y_i, G) *and TEST-ROOT-k*$(y_i, \mathbf{rev}(G))$,
  - *if either of the calls fails return FAILURE,*
- *return SUCCESS.*

Static Vertex Connectivity

If the graph is k connected:

- then every pair of k element subsets is k-linked,
- thus the algorithm will succeed w.h.p when the graph is k connected.

If the graph is not k connected:

- then there exists a separator $S \subseteq V$ od size $k-1$,
- the probability that from a set of k' random vertices one is $\notin S$ is

$$1 - \left(\frac{k-1}{n}\right)^{\left\lceil \frac{\log n}{\log n - \log k} \right\rceil} \approx 1 - \left(\frac{k}{n}\right)^{\log_{\frac{n}{k}} n} = 1 - \frac{1}{n}.$$

Static Vertex Connectivity

Lemma 14 (Cheriyan and Reif '92) *If G is k-connected than TEST-k-CONNECTIVITY return SUCCESS with probability $> 1 - \frac{1}{n}$ otherwise FAILURE is returned with probability $> 1 - \frac{1}{n}$.*

The algorithm works in time

$$O(n^\omega + \frac{\log n}{\log n - \log k} \cdot (n-k) \cdot k^\omega) = O(n^\omega + nk^\omega).$$

Outline

- Introduction
- Simple Solution
- Dynamic Cycle Cover
- Dynamic s, t-Vertex Connectivity
  - Vertex Connectivity and Maximum Matchings
- Dynamic Matrix Rank
- Static k-Vertex Connectivity
- **Dynamic k-Vertex Connectivity**
- Conclusions and Open Problems

Dynamic Vertex Connectivity

In the static algorithm we use only k-linkage tests.

In the dynamic algorithm we can fix these tests in the beginning and use the same test during the whole run of the algorithm.

- the choice is independent from the graph.

However, we need to do many k-linkage tests at once.

Dynamic Vertex Connectivity

Theorem 15 (Sherman and Morrison '49) *The problem of dynamic matrix inverse with* **non-singular** *updates can be solved with the costs:*

- `initialization` – $O(n^\omega)$ *time,*
- `update` – $O(n^2)$ *time (worst-case),*
- `query` – $O(1)$ *time, i.e., the inverse is maintained explicitly.*

The Laplacian matrix is non-singular.

Thus we can easily obtain an $O(n^2 + nk^\omega)$ time algorithm.

Dynamic Vertex Connectivity

Let $\tilde{X}$ be an $n \times k$ symbolic matrix and $\tilde{Y}$ be a $k \times n$ symbolic matrix and let $\tilde{A}$ be the symbolic Laplacian matrix for G.

Consider the $(n+2k) \times (n+2k)$ matrix $\tilde{A}^{(k,d)}$ defined as follows

$$\tilde{A}^{(k,d)} := \begin{bmatrix} \tilde{A} & \tilde{X} & 0 \\ \tilde{Y} & 0 & I \\ 0 & I & I^{(d)} \end{bmatrix}.$$

Dynamic Vertex Connectivity

$$\tilde{A}^{(k,d)} := \begin{bmatrix} \tilde{A} & \tilde{X} & 0 \\ \tilde{Y} & 0 & I \\ 0 & I & I^{(d)} \end{bmatrix}.$$

Lemma 16 *Let $P, Q \subseteq V$ and $|P| = |Q| = k$. Then the sets P and Q are $(k-d)$-linked iff the matrix $\left(\tilde{A}^{(k,d)}\right)^{-1}_{P,Q}$ is non-singular.*

- d controls the rank of submatrices,
- and allows the control of connectivity.

Dynamic Vertex Connectivity

A *rank one update* is an update of the form $A := A + ab^T$, where a and b are n dimensional vectors.

Lemma 17 *There exists an algorithm for dynamically maintaining the inverse of A that supports rank one updates in $O(n^2)$ time and queries in $O(1)$ time. For each $R, C \subseteq \{1, \ldots, n\}$, where $|R| = |C| = k$, the algorithm maintains determinant of $(A^{-1})_{R,C}$ with additional cost of $O(k^2)$ time per update.*

Dynamic Vertex Connectivity

The matrix A after the rank one update equals:

$$A' = A\left(I + A^{-1}ab^T\right).$$

The inverse of A is recomputed by:

$$A'^{-1} = \left(I + A^{-1}ab^T\right)^{-1} A^{-1}.$$

This takes $O(n^2)$ time. We can recompute the submatrix:

$$(A'^{-1})_{R,C} = \left(\left(I + A^{-1}ab^T\right)^{-1} A^{-1}\right)_{R,C},$$

$$\ldots$$

Dynamic Vertex Connectivity

Lemma 18 *There exists an algorithm for dynamically maintaining the inverse of A that supports element updates in $O(n^{1.575})$ time and queries in $O(n^{0.575})$ time. For each $R, C \subseteq \{1, \ldots, n\}$, where $|R| = |C| = k$, the algorithm maintains determinant of $(A^{-1})_{R,C}$ with additional cost of $O(k^2)$ time per update.*

$$\Downarrow$$

Theorem 19 *There exists an algorithm for dynamically testing if the graph is k vertex connected that supports edge updates in $\tilde{O}(n^{1.575} + nk^2)$ time and queries in $O(1)$ time. The algorithm is randomized.*

Outline

- Introduction
- Simple Solution
- Dynamic Cycle Cover
- Dynamic s, t-Vertex Connectivity
  - Vertex Connectivity and Maximum Matchings
- Dynamic Matrix Rank
- Static k-Vertex Connectivity
- Dynamic k-Vertex Connectivity
- **Conclusions and Open Problems**

Conclusions and Open Problem

- Can the static vertex connectivity be solved in matrix multiplication time? Current time:

$$O(n^\omega + nk^\omega).$$

- We can solve a dynamic "flow-like" problem, i.e., s,t-vertex connectivity in subquadratic $O(n^{1.495})$ time.
  - Can the same be done for the maximum flow problem? Only unit edge weights?
  - For unit edge weights a $O(m)$ time solution exists.

Conclusions and Open Problems

- We can solve the dynamic maximum matching problem in subquadratic $O(n^{1.495})$ time.
  - What about the weighted case? Maximum weighted matching? Maximum weight perfect matching?
- We can solve maximum weighted bipartite matching problem with integer edge weights from the set $1, \ldots, W$ in $O(W^{2.495} n^{1.495})$ time.
  - The used unfolded graph technique works only in the case of bipartite graphs and maximum weighted matchings.

Conclusions and Open Problems

- We can solve the dynamic matrix rank problem and other matrix problems in subquadratic $O(n^{1.495})$ time.
  - What about the characteristic polynomial?
  - *(Recently: $\tilde{O}(n^2)$ time in symmetric case.)*
- We can solve the dynamic directed k-vertex connectivity in $O(n^{1.575} + nk^2)$ time.
  - Static time is $O(n^\omega + nk^\omega)$. In dynamic case it should be $O(n^{1.575} + nk^{1.575})$?
  - What about k-edge connectivity? Line graph?