Space-efficient algorithms for reachability in surface-embedded graphs

Derrick Stolee* N. V. Vinodchandran

University of Nebraska–Lincoln

s-dstolee1@math.unl.edu

http://www.math.unl.edu/∼s-dstolee1/

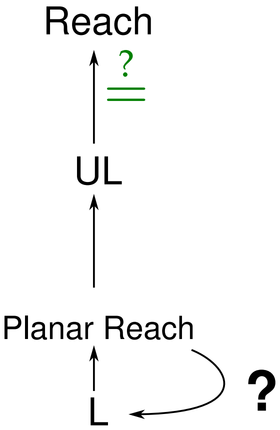
CCC 2012

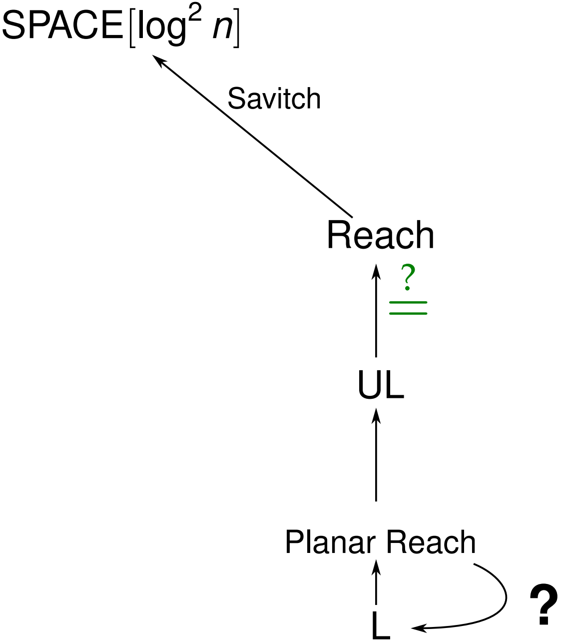
June 29, 2012

Supported by NSF grants DMS-0354008, DMS-0914815, and CCF-0916525, and a CCC Student Travel Grant.

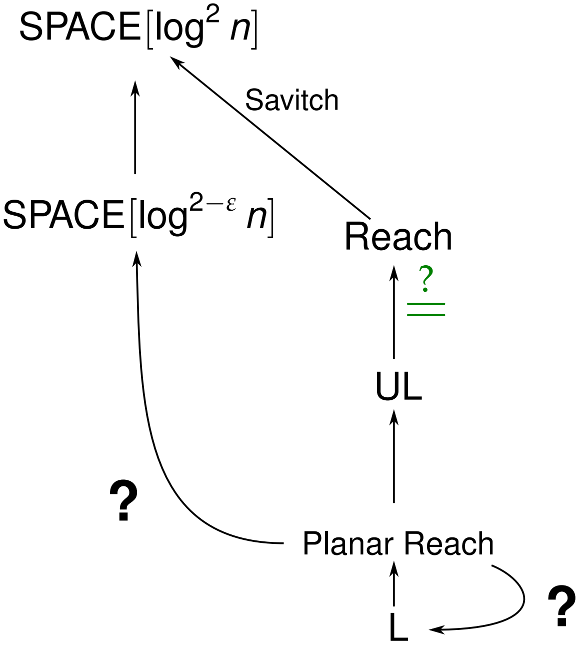

NL
L

$$\begin{tikzcd} \text{Reach} \\ \text{UL} \arrow[u, "\overset{?}{=}"'] \\ \text{Planar Reach} \arrow[u] \\ \text{L} \arrow[u] \end{tikzcd}$$

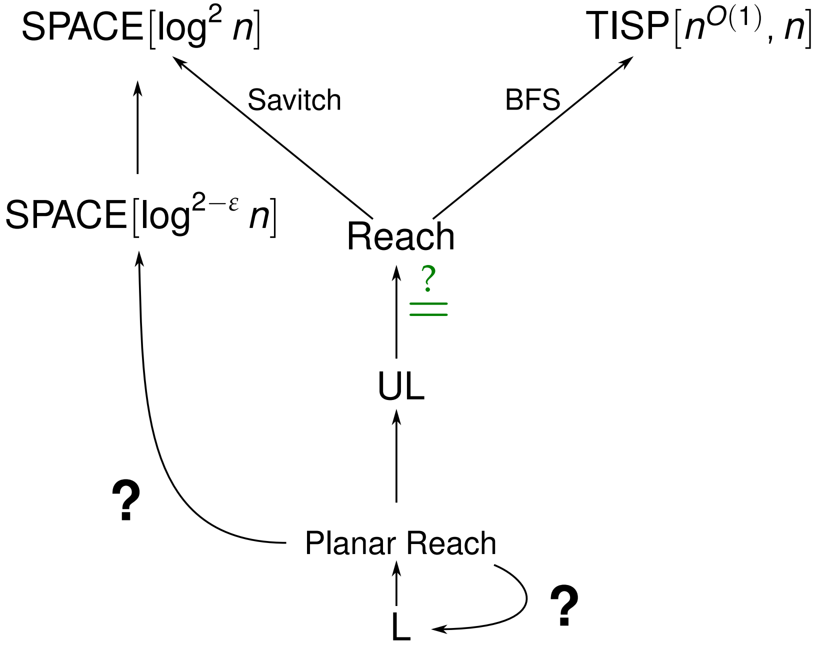

Reach
?
UL
Planar Reach
L
?



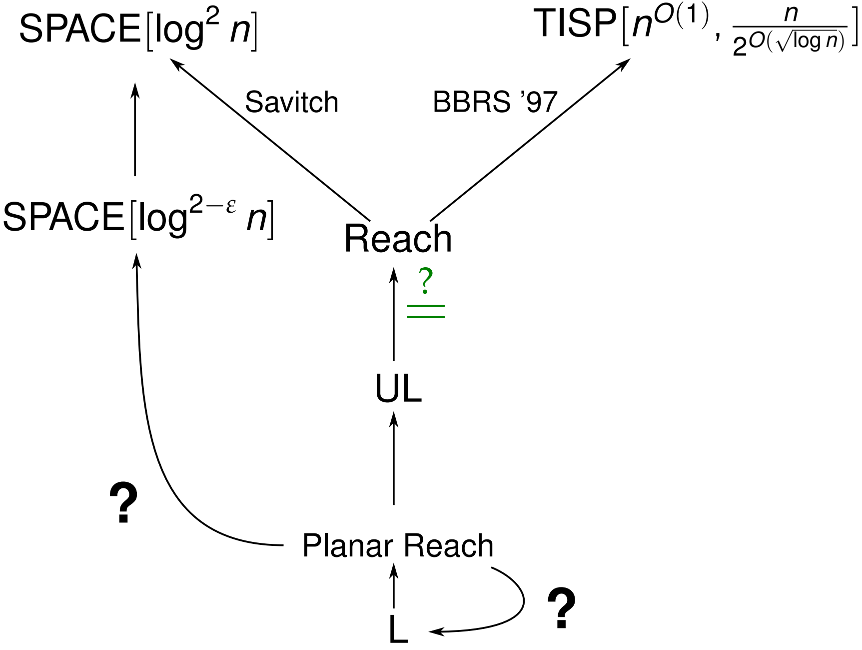
SPACE[log2 n]
Savitch
Reach
?
=
UL
Planar Reach
L
?



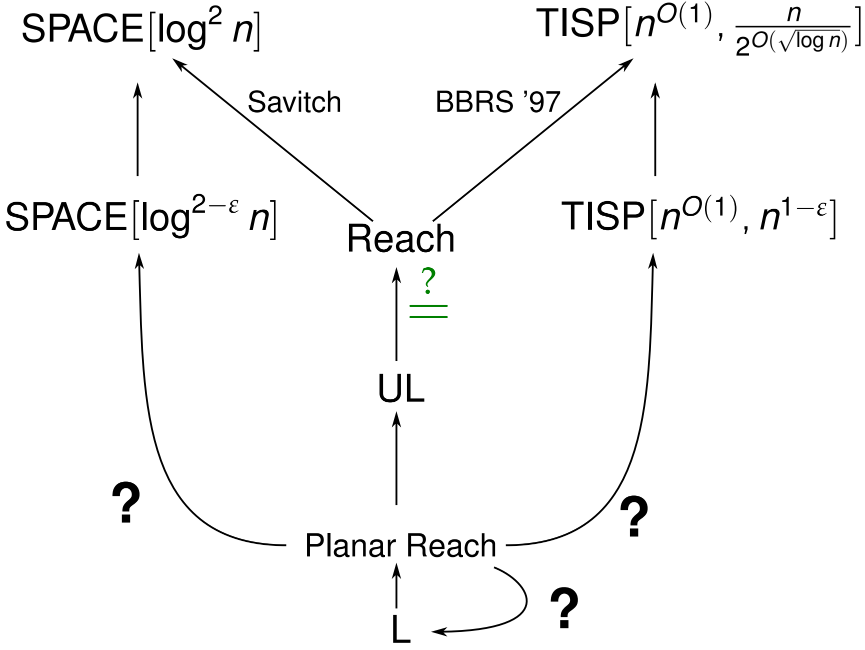
$\mathsf{SPACE}[\log^2 n]$
Savitch
$\mathsf{SPACE}[\log^{2-\varepsilon} n]$
Reach
$\stackrel{?}{=}$
UL
?
Planar Reach
L
?



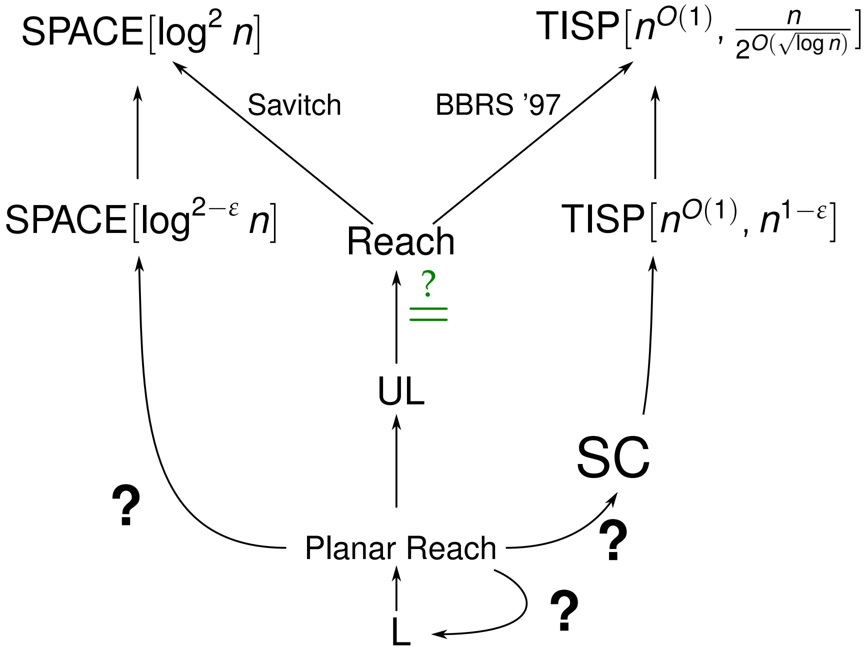
SPACE[log^2 n]
TISP[n^{O(1)}, n]
Savitch
BFS
SPACE[log^{2−ε} n]
Reach
?
=
UL
?
Planar Reach
L
?



$\mathsf{SPACE}[\log^2 n]$
$\mathsf{TISP}[n^{O(1)}, \frac{n}{2^{O(\sqrt{\log n})}}]$
Savitch
BBRS '97
$\mathsf{SPACE}[\log^{2-\varepsilon} n]$
Reach
$\stackrel{?}{=}$
UL
?
Planar Reach
L
?



$\mathsf{SPACE}[\log^2 n]$
$\mathsf{TISP}[n^{O(1)}, \frac{n}{2^{O(\sqrt{\log n})}}]$
Savitch
BBRS '97
$\mathsf{SPACE}[\log^{2-\varepsilon} n]$
Reach
$\mathsf{TISP}[n^{O(1)}, n^{1-\varepsilon}]$
$\stackrel{?}{=}$
UL
?
Planar Reach
?
L
?



SPACE[$\log^2 n$]
TISP[$n^{O(1)}, \frac{n}{2^{O(\sqrt{\log n})}}$]
Savitch
BBRS '97
SPACE[$\log^{2-\varepsilon} n$]
Reach
TISP[$n^{O(1)}, n^{1-\varepsilon}$]
$\stackrel{?}{=}$
UL
SC
?
Planar Reach
?
?
L

Planar and Acyclic Restrictions

1. Reach for **acyclic** digraphs is complete for NL.

Planar and Acyclic Restrictions

1. Reach for **acyclic** digraphs is complete for NL.
2. Reach for **planar** digraphs is in UL, but we believe UL $=$ NL.

   (See Reinhardt and Allender, 2002)

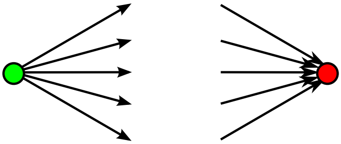
Planar and Acyclic Restrictions

1. Reach for **acyclic** digraphs is complete for NL.
2. Reach for **planar** digraphs is in UL, but we believe UL $=$ NL.
   (See Reinhardt and Allender, 2002)
3. What if we combine **acyclic** and **planar**?

Planar and Acyclic Restrictions

1. Reach for **acyclic** digraphs is complete for NL.
2. Reach for **planar** digraphs is in UL, but we believe UL $=$ NL.
   (See Reinhardt and Allender, 2002)
3. What if we combine **acyclic** and **planar**?

We also bound number of



sources **sinks**

UNDIRECTED REACH in L

(Riengold, STOC 2005)

Planar + Acyclic Reachability in Log-Space

1. Series-parallel graphs
   (Jakoby, Liśkiewicz, Reischuk ’06; Jakoby and Tantau ’07)

Planar + Acyclic Reachability in Log-Space

1. Series-parallel graphs
   (Jakoby, Liśkiewicz, Reischuk '06; Jakoby and Tantau '07)

2. Single-source Single-Sink Planar DAGs
   (Allender, Barrington, Chakraborty, Datta, Roy, '09)

Planar + Acyclic Reachability in Log-Space

1. Series-parallel graphs
   (Jakoby, Liśkiewicz, Reischuk '06; Jakoby and Tantau '07)
2. Single-source Single-Sink Planar DAGs
   (Allender, Barrington, Chakraborty, Datta, Roy, '09)
3. Single-source Multiple-Sink Planar DAGs
   (Allender, Barrington, Chakraborty, Datta, Roy, '09)

Planar + Acyclic Reachability in Log-Space

1. Series-parallel graphs
   (Jakoby, Liśkiewicz, Reischuk ’06; Jakoby and Tantau ’07)
2. Single-source Single-Sink Planar DAGs
   (Allender, Barrington, Chakraborty, Datta, Roy, ’09)
3. Single-source Multiple-Sink Planar DAGs
   (Allender, Barrington, Chakraborty, Datta, Roy, ’09)
4. Log-source Multiple-Sink Planar DAGs
   (Stolee, Bourke, Vinodchandran, ’10)

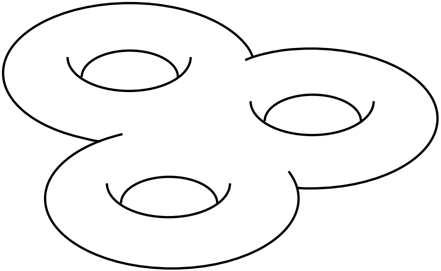
Planar + Acyclic Reachability in Log-Space

1. Series-parallel graphs
   (Jakoby, Liśkiewicz, Reischuk '06; Jakoby and Tantau '07)
2. Single-source Single-Sink Planar DAGs
   (Allender, Barrington, Chakraborty, Datta, Roy, '09)
3. Single-source Multiple-Sink Planar DAGs
   (Allender, Barrington, Chakraborty, Datta, Roy, '09)
4. Log-source Multiple-Sink Planar DAGs
   (Stolee, Bourke, Vinodchandran, '10)
   1. $O(\log n + m)$-space algorithm for m sources.
   2. $O(\log n \cdot \log m)$-space algorithm for m sources.

Surface-embedded graphs

We also extend to graphs embedded in *surfaces of low genus*.

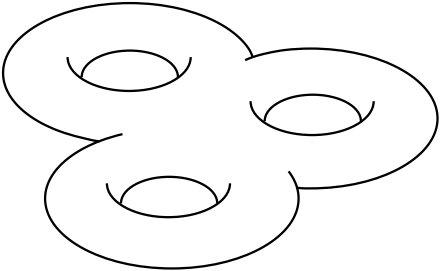
(orientable or non-orientable)



Surface-embedded graphs

We also extend to graphs embedded in *surfaces of low genus*.

(orientable or non-orientable)



$\mathcal{G}(m, g)$ is the class of **acyclic** digraphs with at most m **sources** embedded in a surface of **genus at most** g.

Our Results (Stolee, Vinodchandran, '12)

Theorem (Sub-Savitch) Reachability for graphs of order n in $\mathcal{G}(m, g)$ is in

$$\mathrm{SPACE}[\log n + \log^2(m+g)].$$

Our Results (Stolee, Vinodchandran, '12)

Theorem (Sub-Savitch) Reachability for graphs of order n in $\mathcal{G}(m, g)$ is in

$$\mathsf{SPACE}[\log n + \log^2(m+g)].$$

Theorem (Log-Space) If $m = g = 2^{\sqrt{\log n}}$, reach for $\mathcal{G}(m, g)$ is in L.

Our Results

(Stolee, Vinodchandran, '12)

Theorem (Sub-Savitch) Reachability for graphs of order n in $\mathcal{G}(m, g)$ is in

$$\mathsf{SPACE}[\log n + \log^2(m+g)].$$

Theorem (Log-Space) If $m = g = 2^{\sqrt{\log n}}$, reach for $\mathcal{G}(m, g)$ is in L.

Theorem (Time-Space) Reachability for graphs of order n in $\mathcal{G}(m, g)$ is in

$$\mathsf{TISP}[n^{O(1)}, \log n + m + g].$$

Reduction with Compression

Theorem (Stolee, Vinodchandran, '12) Given a graph $G \in \mathcal{G}(m, g)$ and $u, v \in V(G)$, we can compute in **log-space** a graph G' with vertices u', v' so that

1. There is a path from u to v in G **if and only if** there is a path from u' to v' in G'.
2. G' has $O(m + g)$ vertices.

Our Results (Stolee, Vinodchandran, '12)

Theorem (Sub-Savitch) Reachability for graphs of order n in $\mathcal{G}(m, g)$ is in

$$\mathrm{SPACE}[\log n + \log^2(m + g)].$$

(Run Savitch on G')

Theorem (Log-Space) If $m = g = 2^{\sqrt{\log n}}$, reach for $\mathcal{G}(m, g)$ is in L.

$(\log^2(2^{\sqrt{\log n}}) = \log n)$

Theorem (Time-Space) Reachability for graphs of order n in $\mathcal{G}(m, g)$ is in

$$\mathrm{TISP}[n^{O(1)}, \log n + m + g].$$

(Run Breadth-First-Search)

Our Results

(Stolee, Vinodchandran, '12)

Theorem (Sub-Savitch) Reachability for graphs of order n in $\mathcal{G}(m, g)$ is in

$$\mathrm{SPACE}[\log n + \log^2(m+g)].$$

(Run Savitch on G')

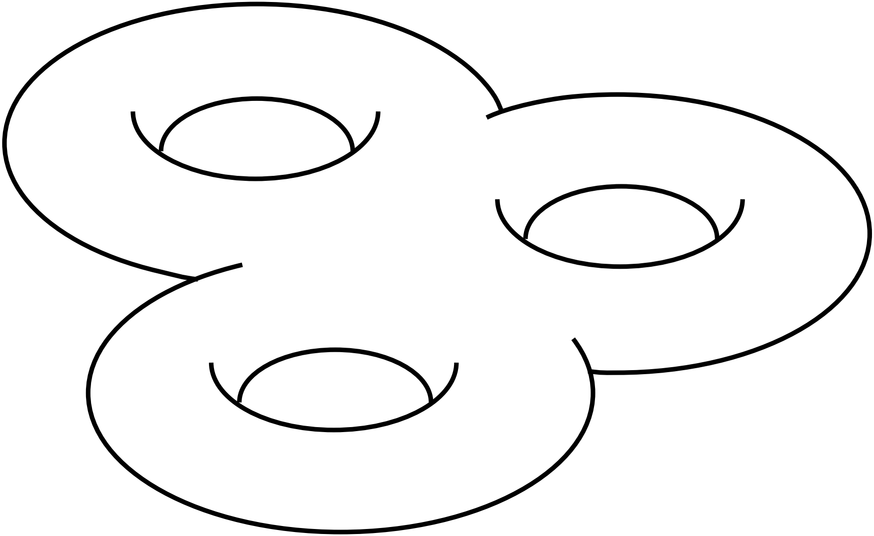
Theorem (Log-Space) If $m = g = 2^{\sqrt{\log n}}$, reach for $\mathcal{G}(m, g)$ is in L.

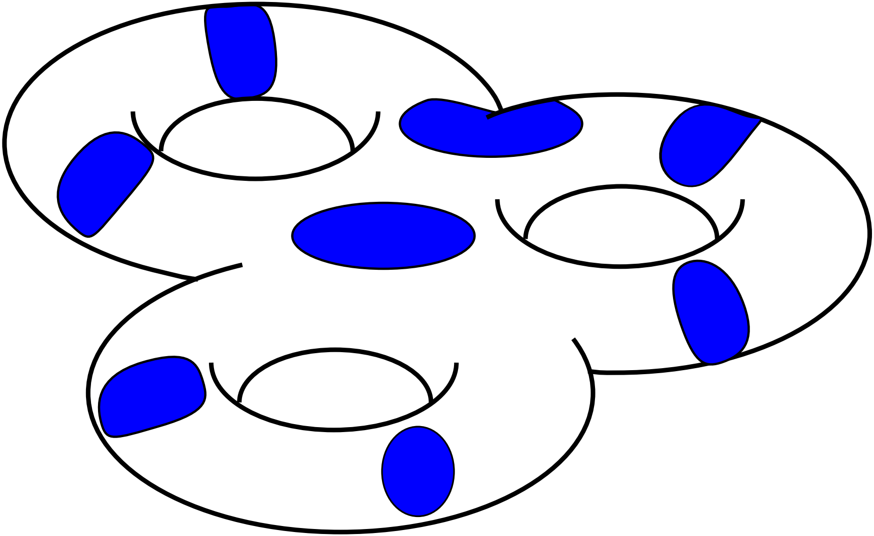
($\log^2(2^{\sqrt{\log n}}) = \log n$)

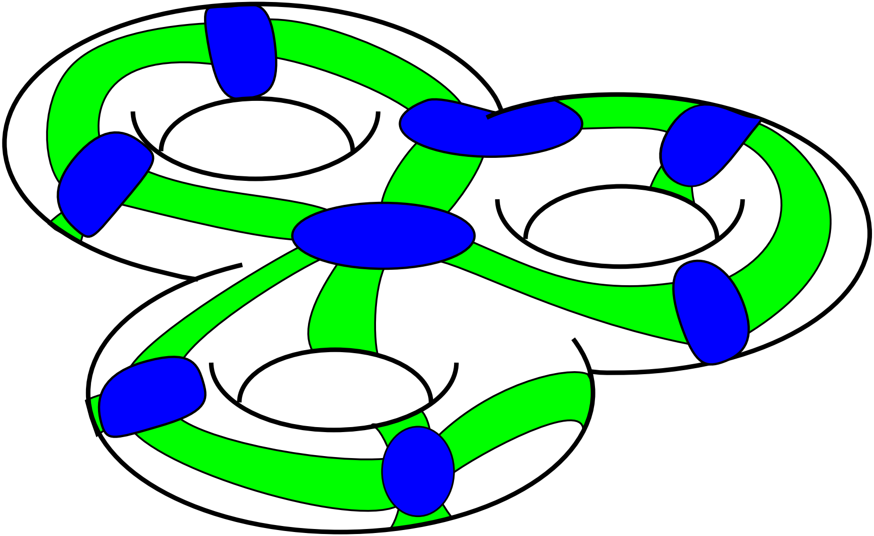
Theorem (Time-Space) Reachability for graphs of order n in $\mathcal{G}(m, g)$ is in

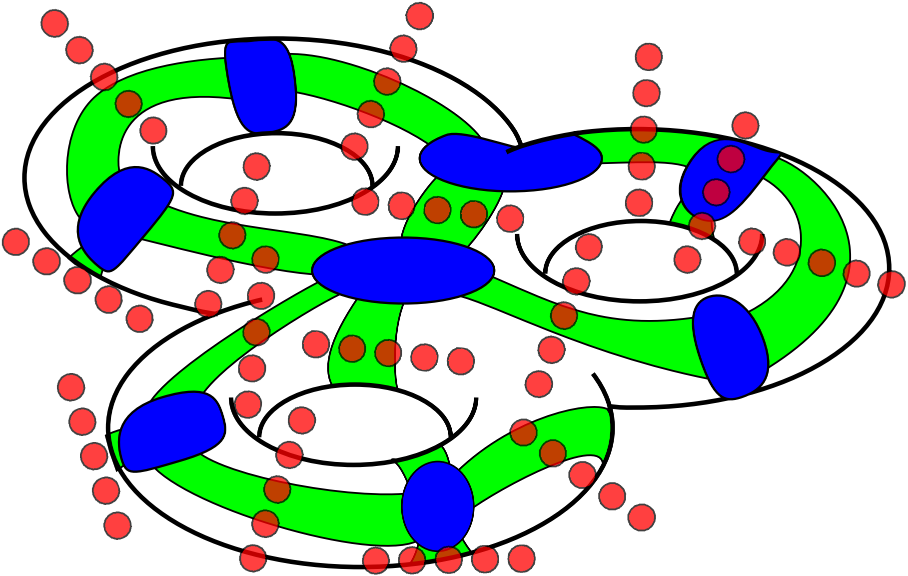
$$\mathrm{TISP}\left[n^{O(1)}, \log n + \frac{m+g}{2^{O(\sqrt{m+g})}}\right].$$

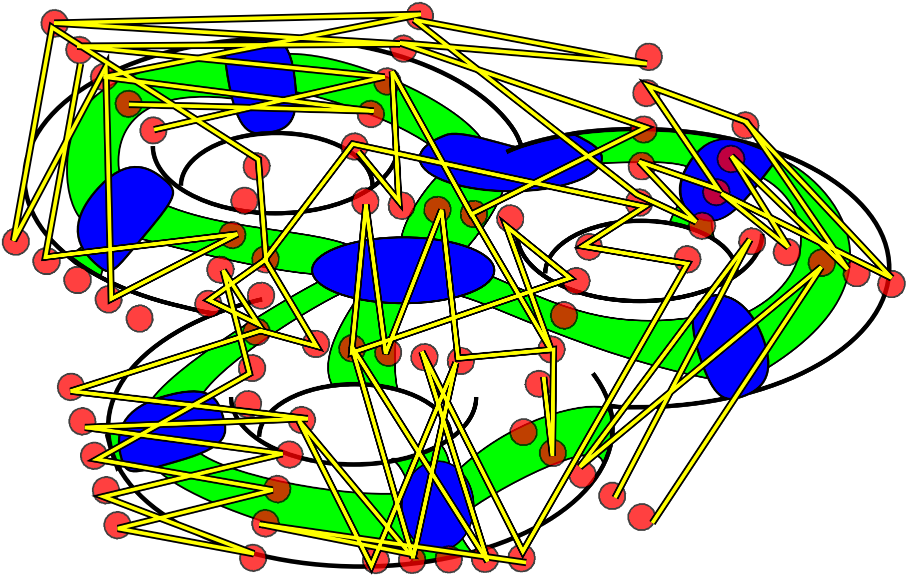
(Run BBRS)

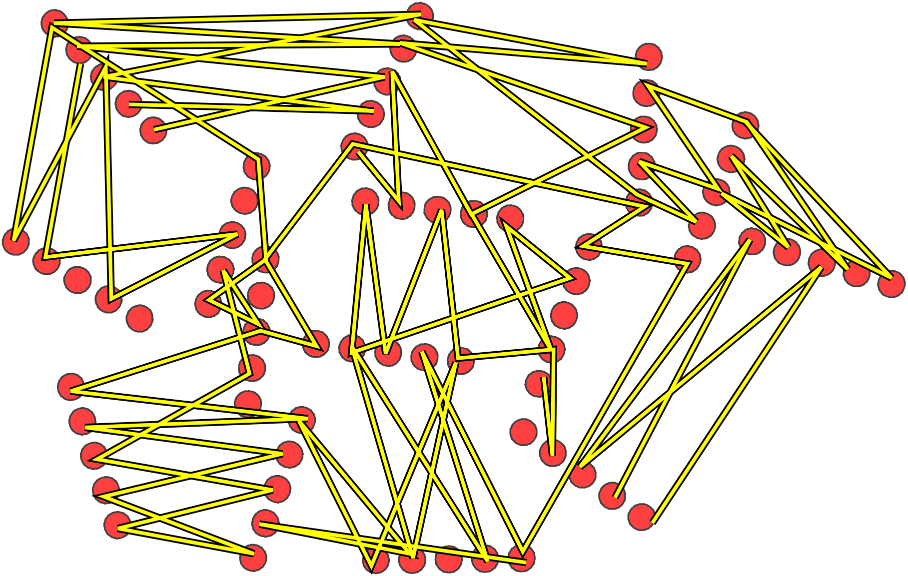












A Note About Embeddings

We take the embedding as **input**.

1. If $g = 0$ (G is planar), we can find an embedding in log-space.
   (Allender and Mahajan 2004; Datta and Prakriya 2011)

A Note About Embeddings

We take the embedding as **input**.

1. If $g = 0$ (G is planar), we can find an embedding in log-space.
   (Allender and Mahajan 2004; Datta and Prakriya 2011)

2. If $g > 0$, it is **hard** to produce an embedding.
   (Thomassen 1989; Chen, Kanchi, and Kanevsky 1997)

A Note About Embeddings

We take the embedding as **input**.

1. If $g = 0$ (G is planar), we can find an embedding in log-space.
   (Allender and Mahajan 2004; Datta and Prakriya 2011)

2. If $g > 0$, it is **hard** to produce an embedding.
   (Thomassen 1989; Chen, Kanchi, and Kanevsky 1997)

3. Kynčl and Vyskočil (2010) reduced reachability on a **fixed surface** to reachability on a planar graph.

A Note About Embeddings

We take the embedding as **input**.

1. If $g = 0$ (G is planar), we can find an embedding in log-space.
(Allender and Mahajan 2004; Datta and Prakriya 2011)

2. If $g > 0$, it is **hard** to produce an embedding.
(Thomassen 1989; Chen, Kanchi, and Kanevsky 1997)

3. Kynčl and Vyskočil (2010) reduced reachability on a **fixed surface** to reachability on a planar graph.

4. We **can** lower number of **sources** by increasing **genus**.

Forest Decomposition

Given $G \in \mathcal{G}(m, g)$ with sources $s_1, \ldots, s_m$ and $u, v \in V(G)$:

Forest Decomposition

Given $G \in \mathcal{G}(m, g)$ with sources $s_1, \ldots, s_m$ and $u, v \in V(G)$:

1. For every vertex $x \in V(G) \setminus \{s_1, \ldots, s_m, u, v\}$, **select an incoming edge**.

Forest Decomposition

Given $G \in \mathcal{G}(m, g)$ with sources $s_1, \ldots, s_m$ and $u, v \in V(G)$:

1. For every vertex $x \in V(G) \setminus \{s_1, \ldots, s_m, u, v\}$, **select an incoming edge**.
2. These edges are **tree edges** and form a forest.

Forest Decomposition

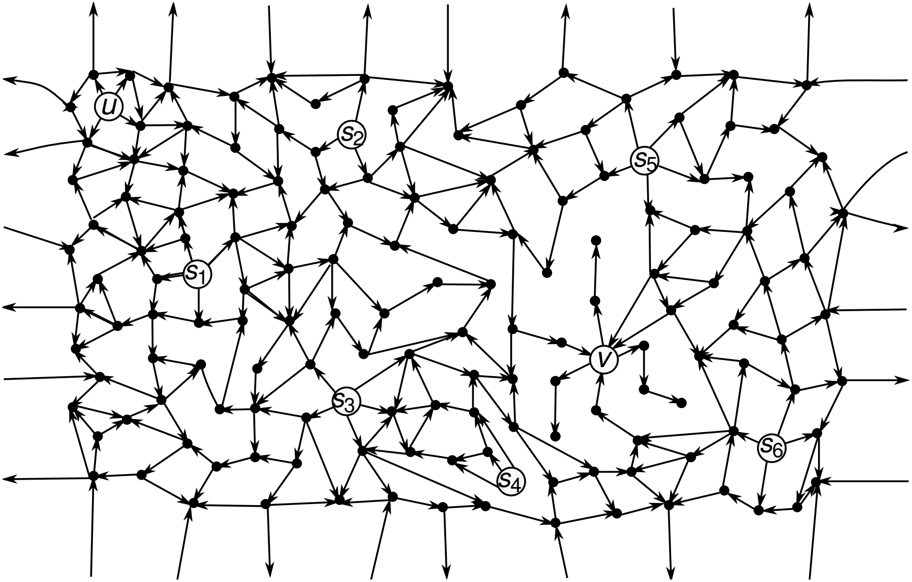
Given $G \in \mathcal{G}(m, g)$ with sources $s_1, \ldots, s_m$ and $u, v \in V(G)$:

1. For every vertex $x \in V(G) \setminus \{s_1, \ldots, s_m, u, v\}$, **select an incoming edge**.
2. These edges are **tree edges** and form a forest.
3. The connected components are trees rooted at $s_1, \ldots, s_m, u, v$:

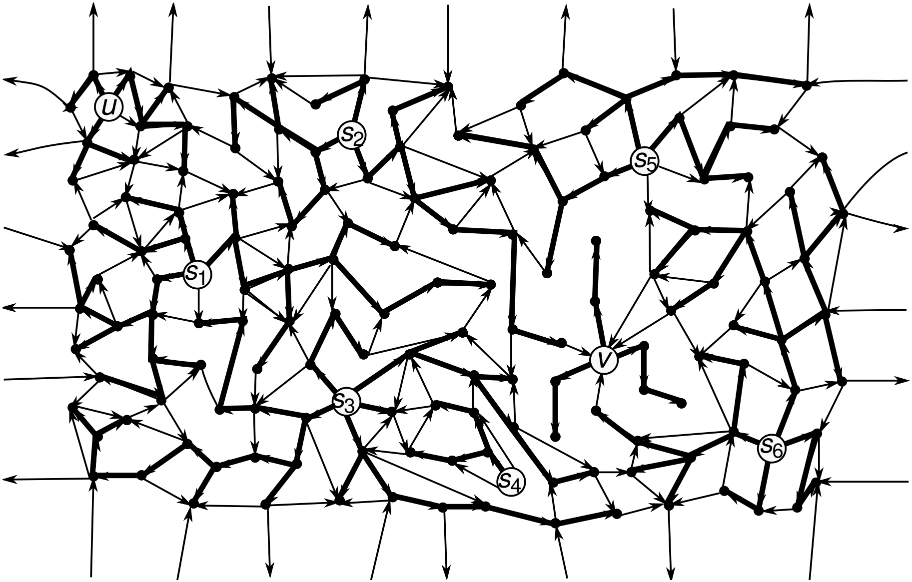
$$T_{s_1}, \ldots, T_{s_m}, T_u, T_v.$$

(We can remove vertices in T_v)

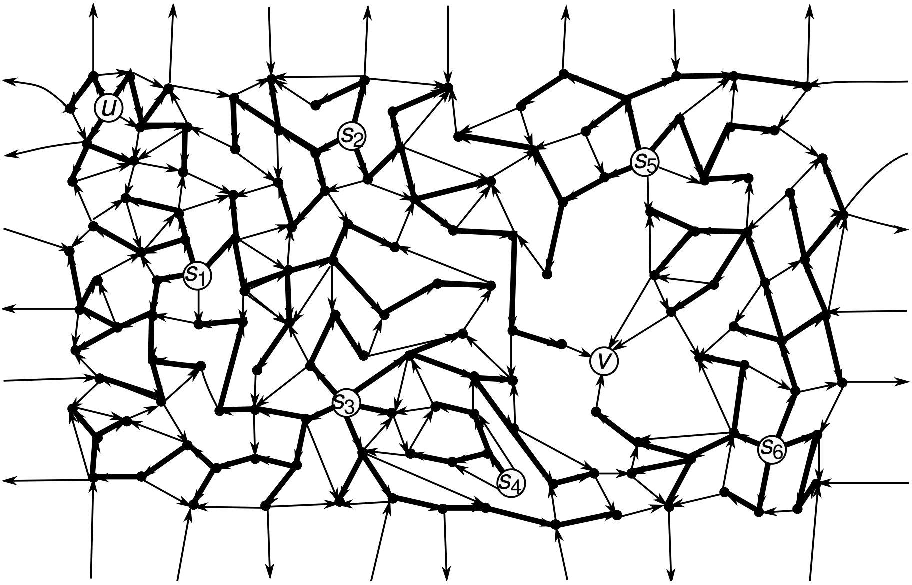
Start with G (Here on the torus).



Select Tree Edges.



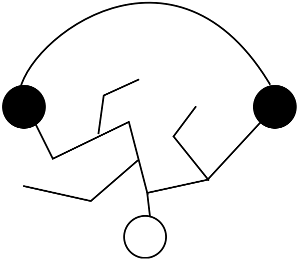
Remove vertices in v's tree.



Local Edges

An edge $x \to y$ is **local** if

1. x and y are within the same source tree T_{s_i}, and
2. The (undirected) tree path from x to y along with the edge xy is a **contractible cycle**.



Contractible Cycles

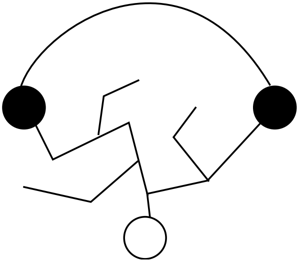
A cycle is **contractible** if:

1. It partitions $s_1, \ldots, s_m, u, v$ *trivially*.
2. The trivial part of the surface is homeomorphic to a disk.

Local Edges

An edge $x \to y$ is **local** if

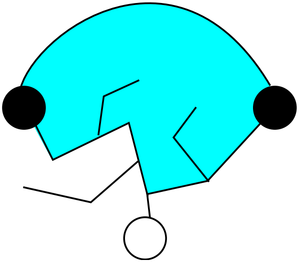
1. x and y are within the same source tree T_{s_i}, and
2. The (undirected) tree path from x to y along with the edge xy is a **contractible cycle**.



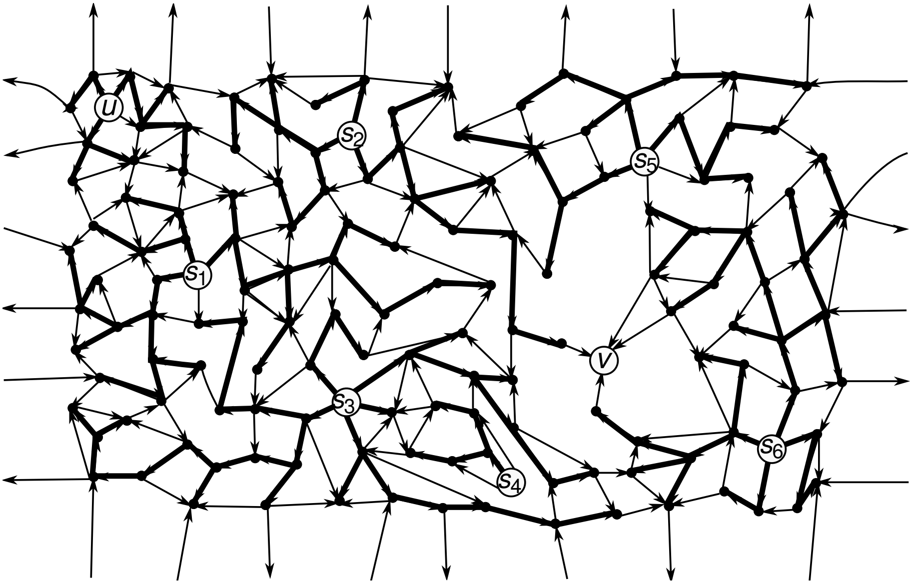
Local Edges

An edge $x \to y$ is **local** if

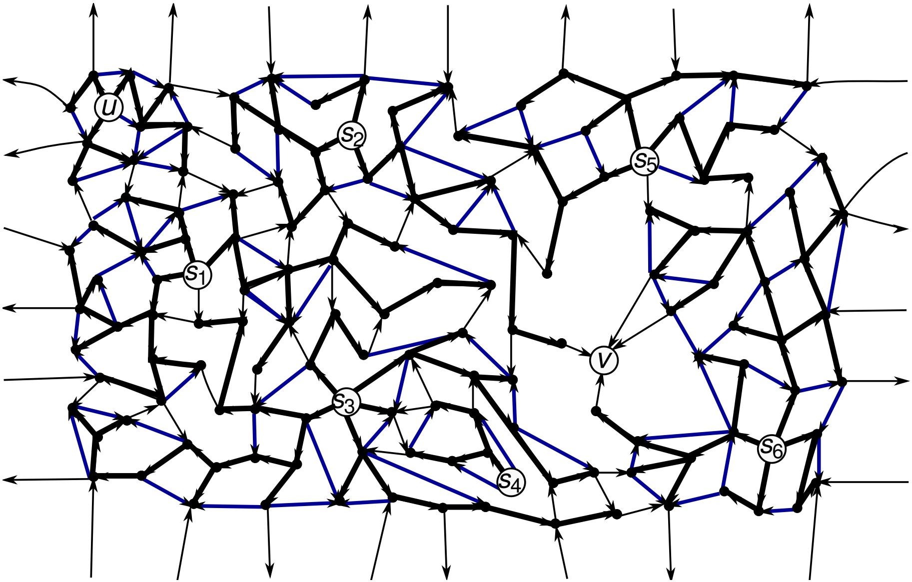
1. x and y are within the same source tree T_{s_i}, and
2. The (undirected) tree path from x to y along with the edge xy is a **contractible cycle**.



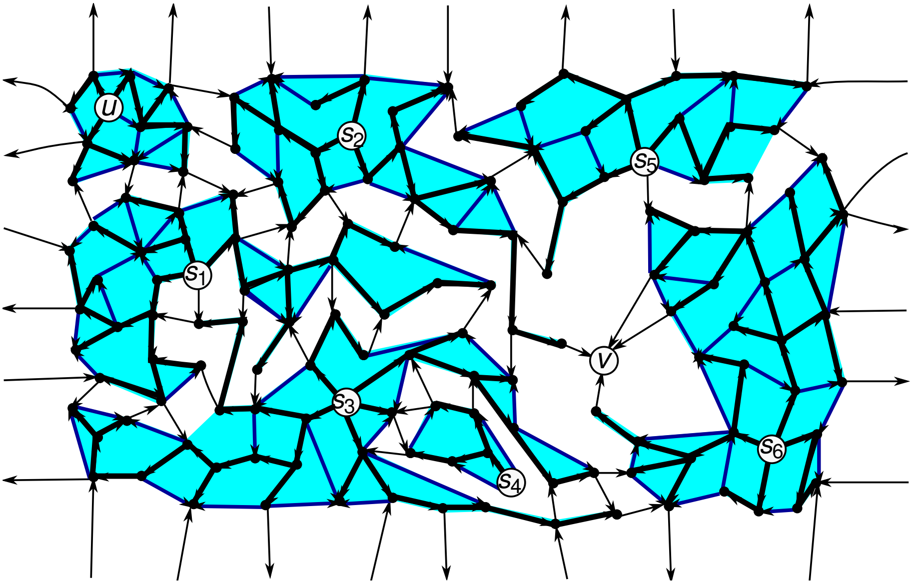
Identify Tree Edges.



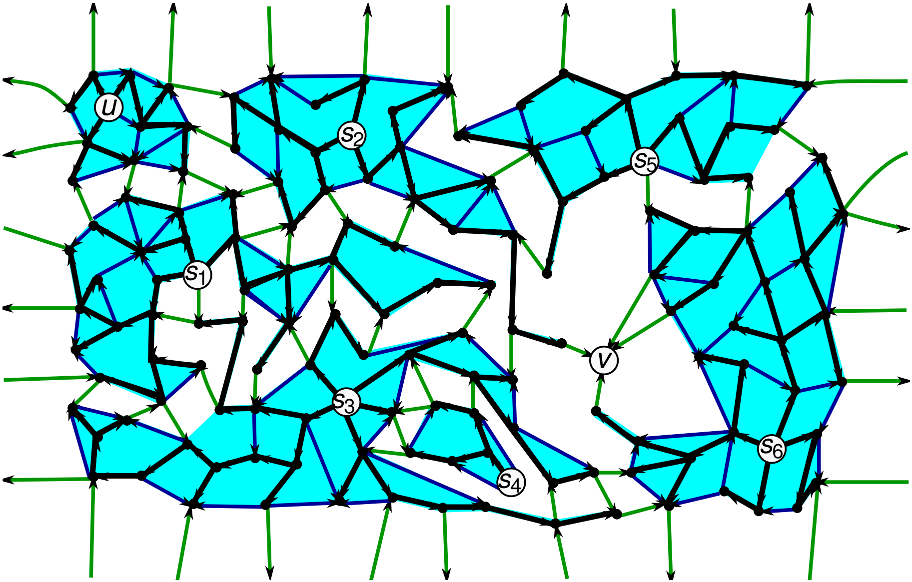
Identify **local edges**.



Mark tree regions $\mathcal{R}[T]$.



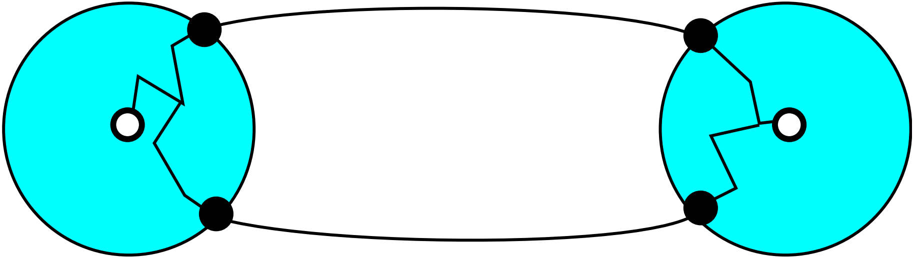
Identify **global edges**.



Topological Equivalence

Two global edges *xy* and *wz* are **topologically equivalent** if

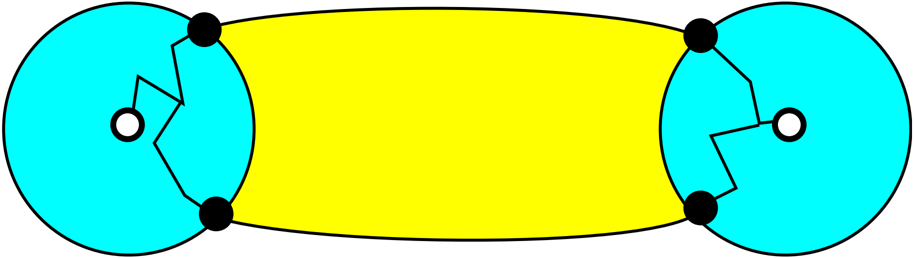
1. The trees *xy* and *wz* span are the same, and
2. The cycle given by the two tree paths between the endpoints is contractible.



Topological Equivalence

Two global edges *xy* and *wz* are **topologically equivalent** if

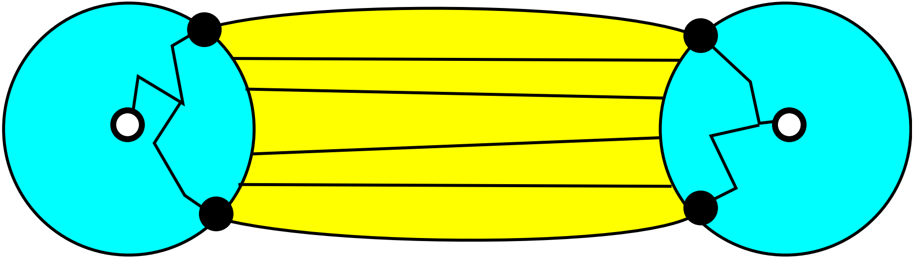
1. The trees *xy* and *wz* span are the same, and
2. The cycle given by the two tree paths between the endpoints is contractible.



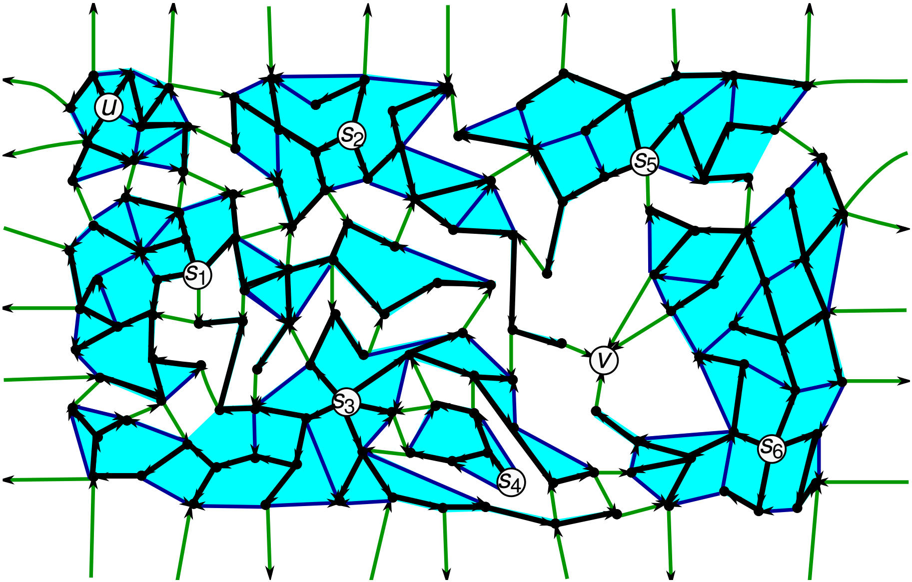
Topological Equivalence

Two global edges xy and wz are **topologically equivalent** if

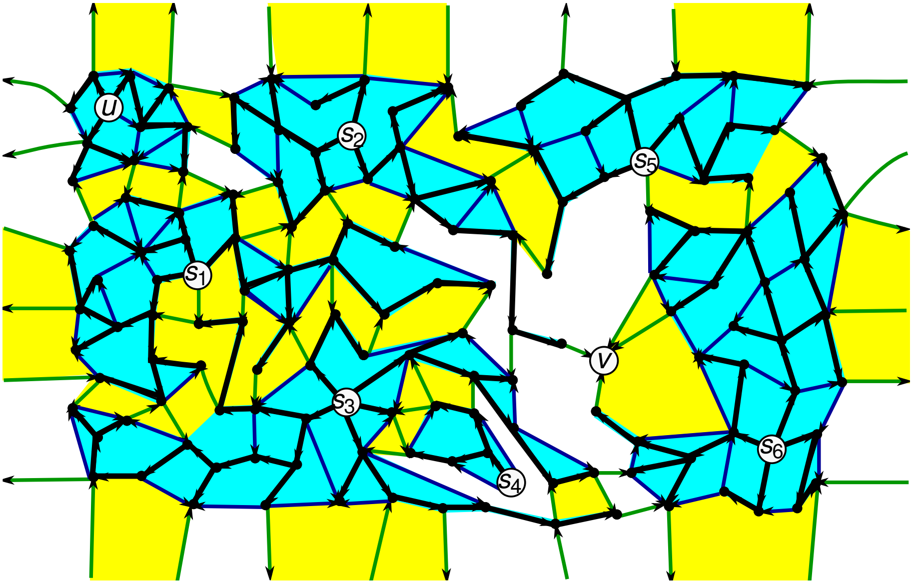
1. The trees xy and wz span are the same, and
2. The cycle given by the two tree paths between the endpoints is contractible.



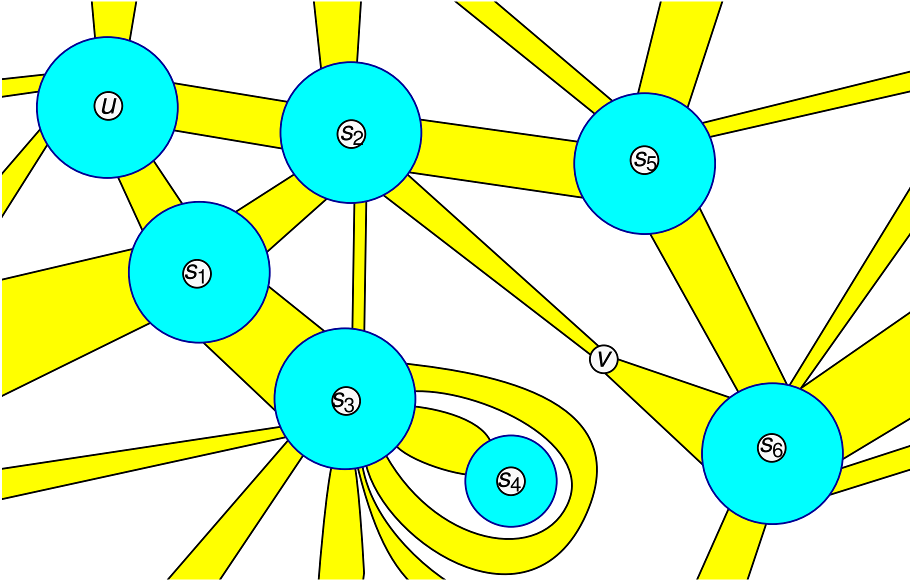
Identify **global edges**.



Mark equivalence class regions $\mathcal{R}[E]$.



A compressed view.



Number of Equivalence Classes

A simple application of **Euler's Formula**:

$$n - e + f = \begin{cases} 2 - 2g & \text{(orientable)} \\ 2 - g & \text{(non-orientable)} \end{cases}$$

shows that the number of equivalence classes is $O(m + g)$.

Number of Equivalence Classes

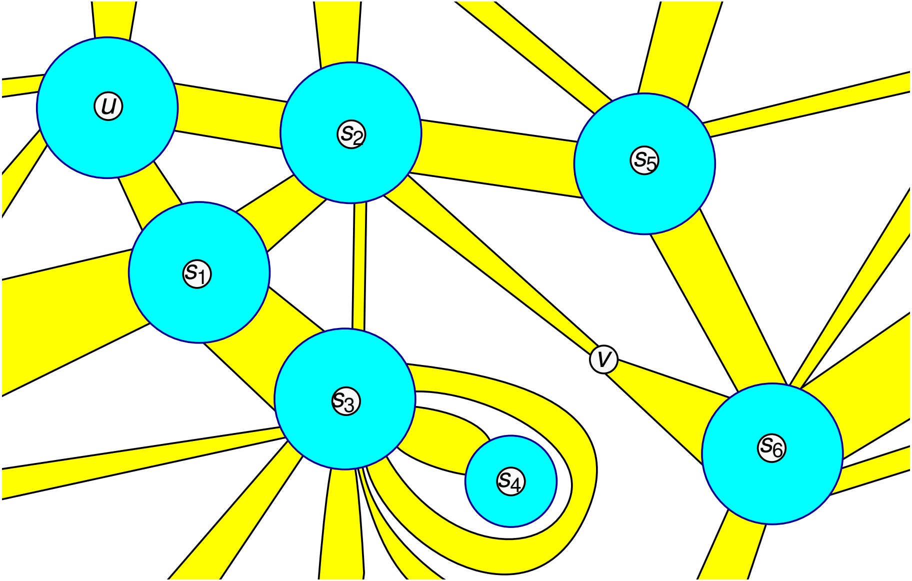
A simple application of **Euler's Formula**:

$$n - e + f = \begin{cases} 2 - 2g & \text{(orientable)} \\ 2 - g & \text{(non-orientable)} \end{cases}$$

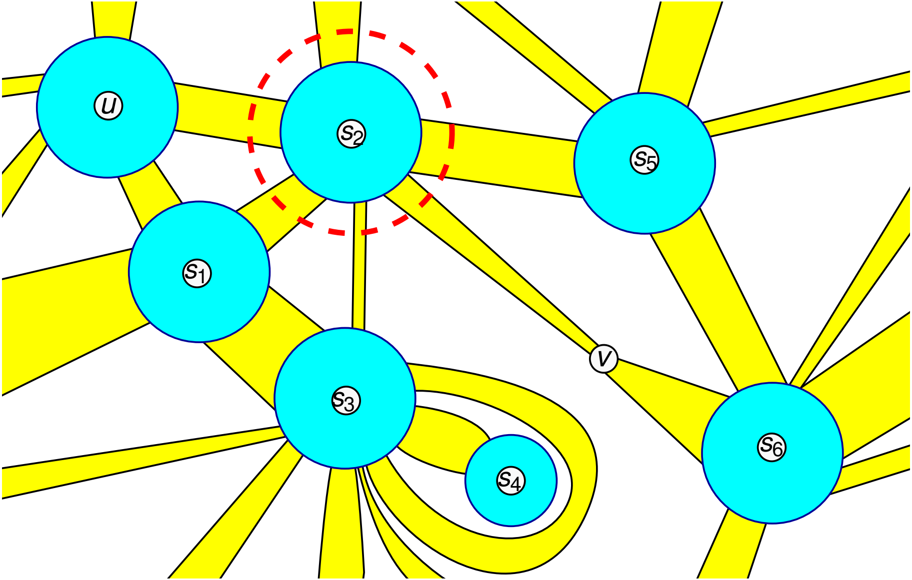
shows that the number of equivalence classes is $O(m + g)$.

We will "blow up" these equivalence classes to form our vertices of G'.

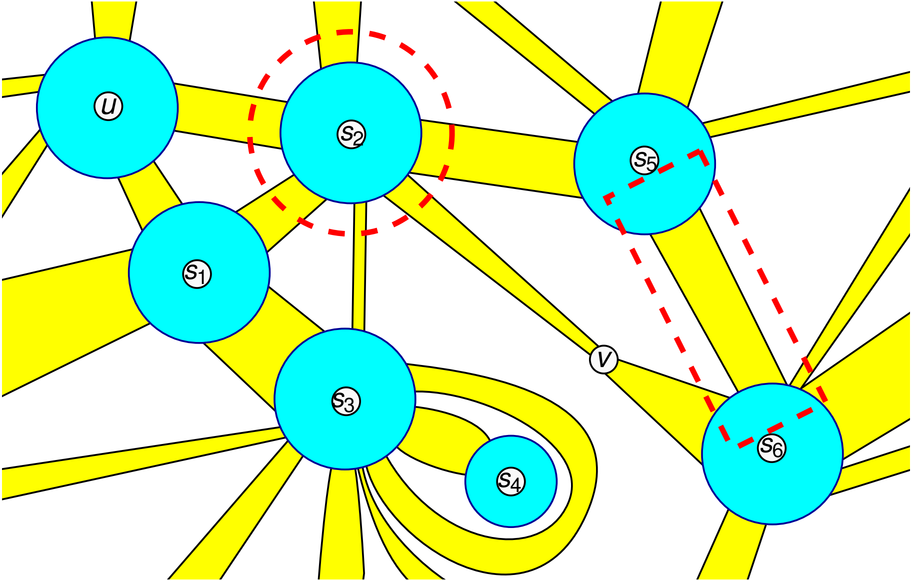
A compressed view.



1. Local reachability within a tree.



2. Global reachability within an equivalence class.



Irreducible Paths

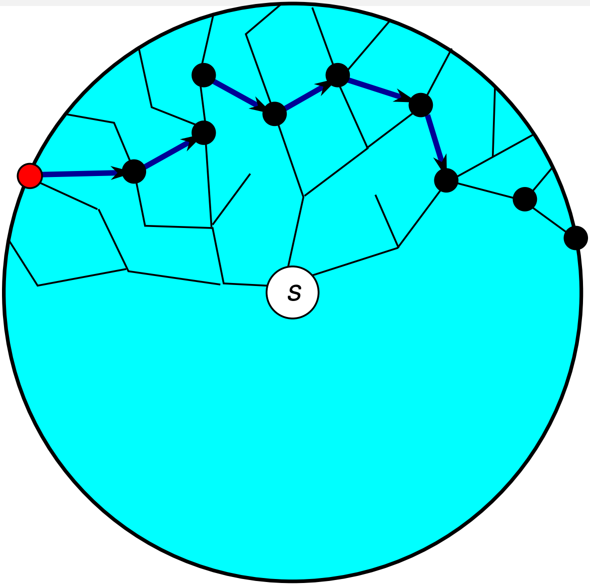
Definition A path P is **irreducible** if for all vertices x, y so that P visits x before y and x is an ancestor of y (with respect to the forest decomposition), then P follows the tree edges from x to y.

Irreducible Paths

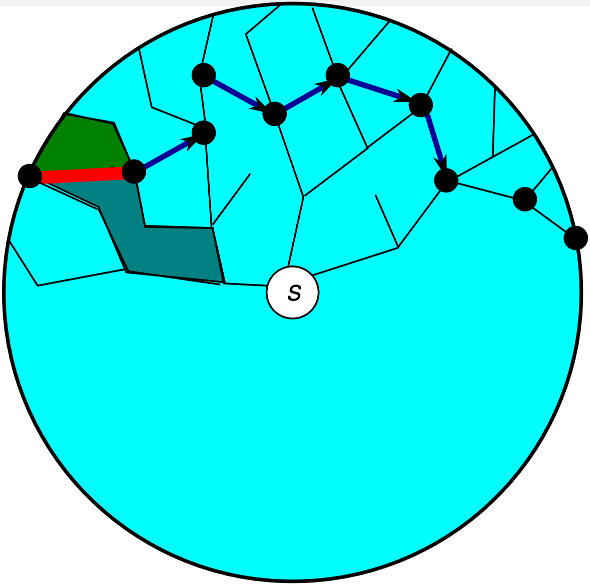
Definition A path P is **irreducible** if for all vertices x, y so that P visits x before y and x is an ancestor of y (with respect to the forest decomposition), then P follows the tree edges from x to y.

Irreducible paths are **nice** because they follow a single **clockwise** or **counterclockwise** direction while traveling through a source tree.

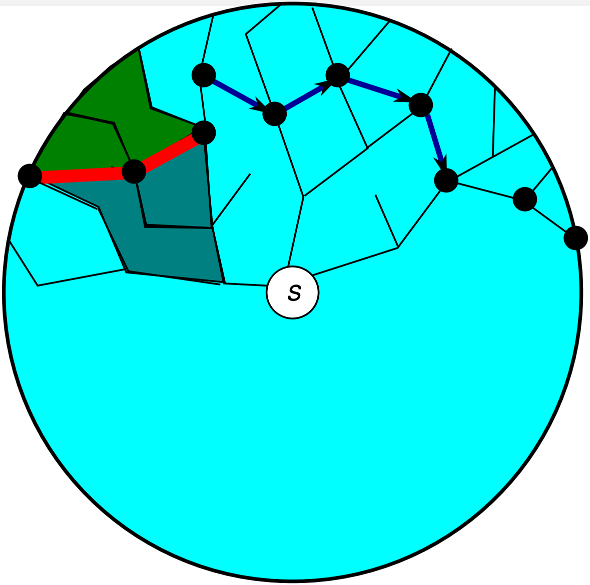
Irreducible Paths in a Source Tree



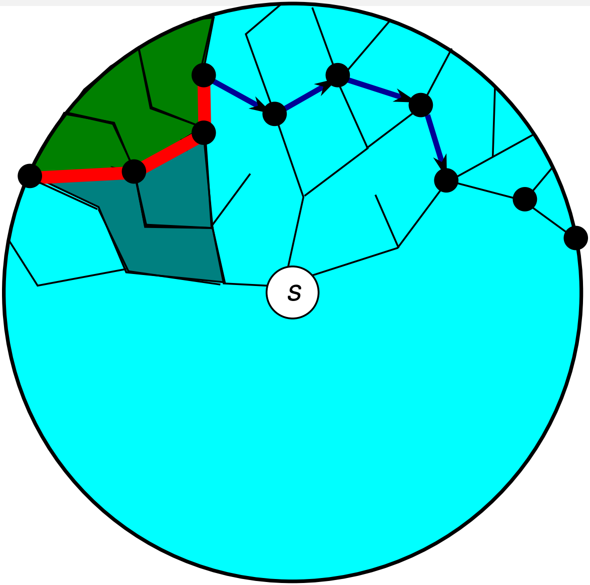
Irreducible Paths in a Source Tree



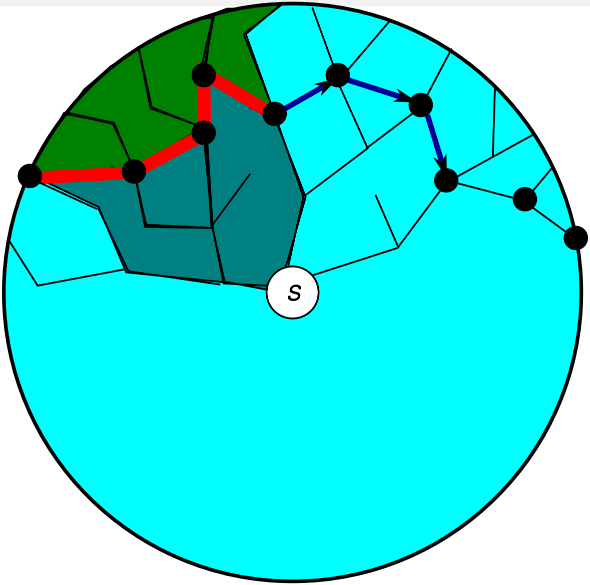
Irreducible Paths in a Source Tree



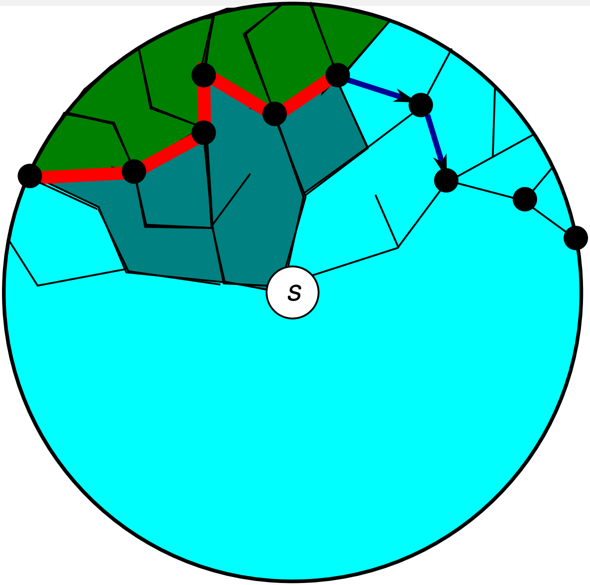
Irreducible Paths in a Source Tree



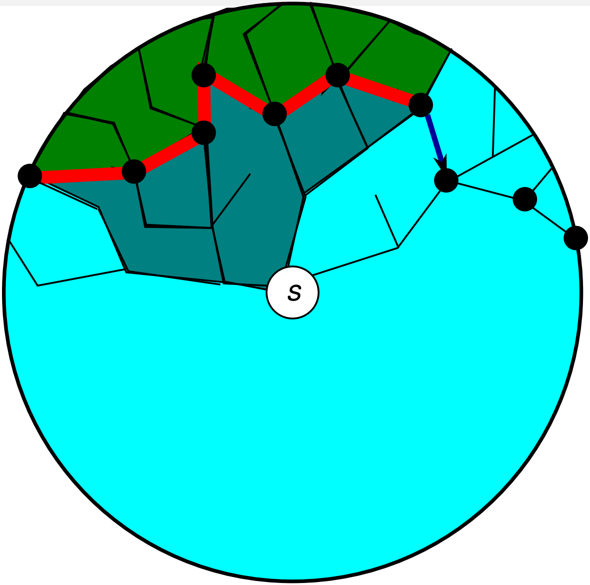
Irreducible Paths in a Source Tree



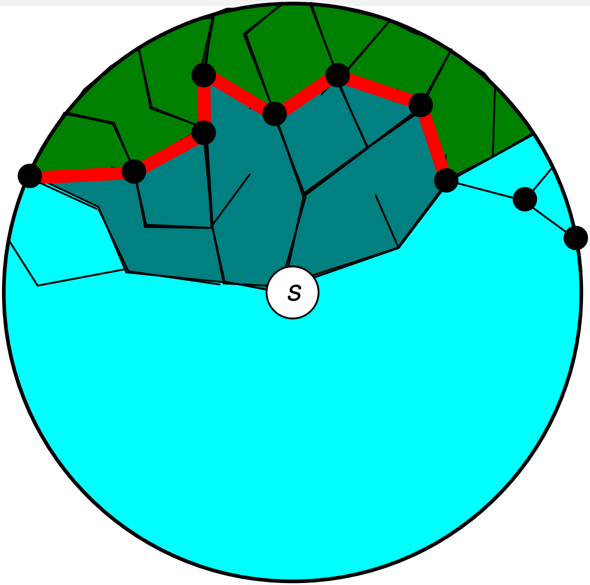
Irreducible Paths in a Source Tree



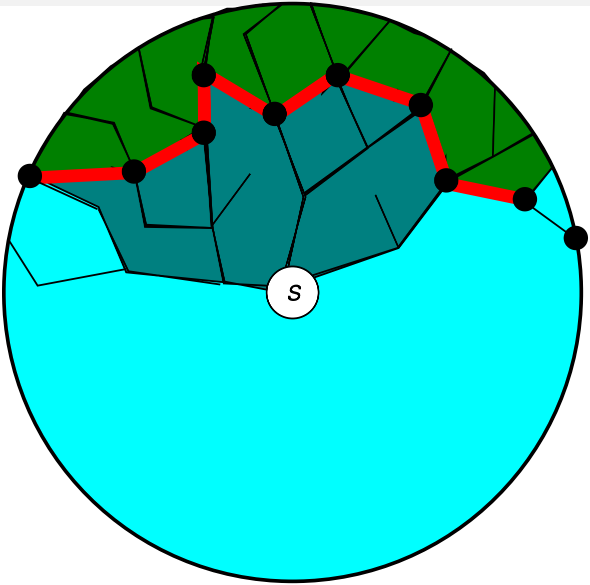
Irreducible Paths in a Source Tree



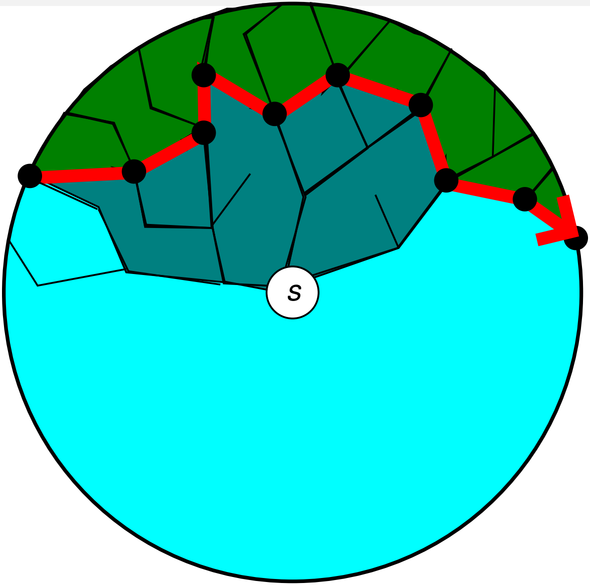
Irreducible Paths in a Source Tree



Irreducible Paths in a Source Tree



Irreducible Paths in a Source Tree



Directional Reachability Within Source Trees

The tree and local edges within a source tree, embedded on the region $\mathcal{R}[T]$ is a *single-source, multiple-sink, planar DAG*.

We can use ABCDR's algorithm as a black box (almost) to find **directional reachability** within source trees.

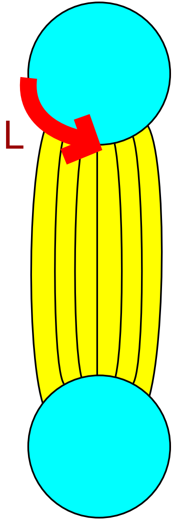
Directional Reachability Within Source Trees

The tree and local edges within a source tree, embedded on the region $\mathcal{R}[T]$ is a *single-source, multiple-sink, planar DAG*.

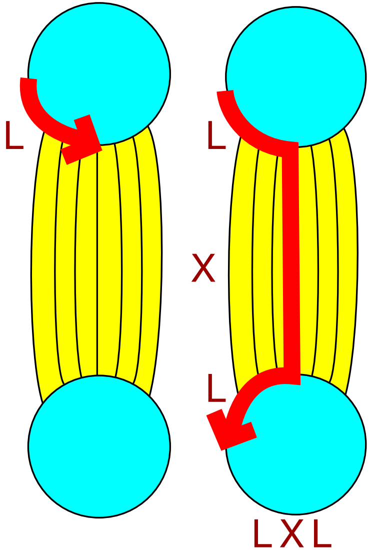
We can use ABCDR's algorithm as a black box (almost) to find **directional reachability** within source trees.

Now, what happens in global edges?

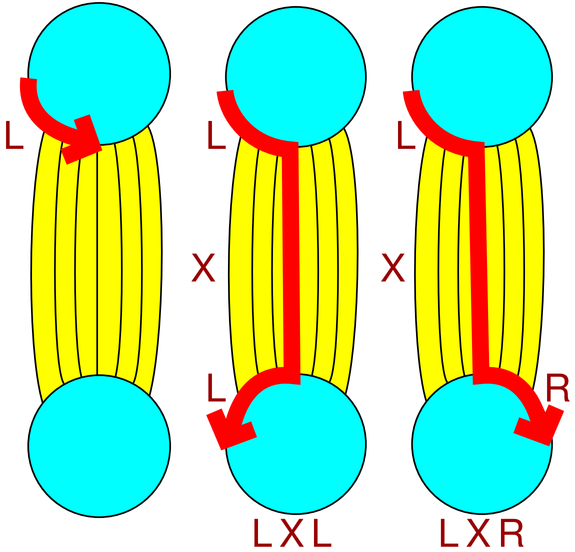
Patterns on an Equivalence Class



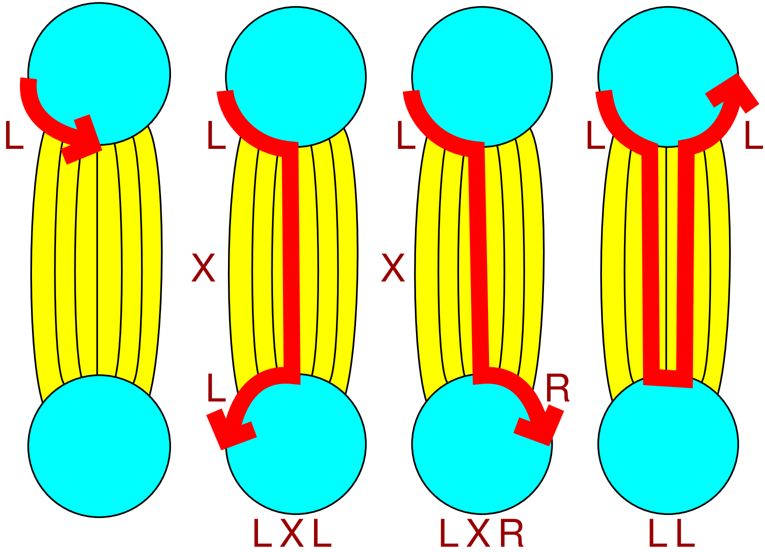
Patterns on an Equivalence Class



Patterns on an Equivalence Class



Patterns on an Equivalence Class



Pattern Descriptions

$$\mathcal{P} = \{\langle \mathsf{L\,X\,L}\rangle, \langle \mathsf{R\,X\,R}\rangle, \langle \mathsf{L\,X\,R}\rangle, \langle \mathsf{R\,X\,L}\rangle, \langle \mathsf{L\,L}\rangle, \langle \mathsf{R\,R}\rangle\}$$

A **pattern node** (denoted **x**, **y**, or **z**) consists of:

1. An equivalence class index i (for the ith class)
2. A pattern from $\mathcal{P}$
3. An entrance tree (A or B)
4. An orientation ($+$ or $-$, depending on A-tree)

Pattern Descriptions

$$\mathcal{P} = \{\langle \mathrm{L\,X\,L} \rangle, \langle \mathrm{R\,X\,R} \rangle, \langle \mathrm{L\,X\,R} \rangle, \langle \mathrm{R\,X\,L} \rangle, \langle \mathrm{L\,L} \rangle, \langle \mathrm{R\,R} \rangle\}$$

A **pattern node** (denoted **x**, **y**, or **z**) consists of:

1. An equivalence class index i (for the ith class)
2. A pattern from $\mathcal{P}$
3. An entrance tree (A or B)
4. An orientation ($+$ or $-$, depending on A-tree)

Subtle: Requires $O(\log(m+g))$ bits to describe a pattern node.

Pattern Descriptions

$$\mathcal{P} = \{\underbrace{\langle \mathrm{L\,X\,L}\rangle, \langle \mathrm{R\,X\,R}\rangle}_{\text{Nesting}}, \underbrace{\langle \mathrm{L\,X\,R}\rangle, \langle \mathrm{R\,X\,L}\rangle, \langle \mathrm{L\,L}\rangle, \langle \mathrm{R\,R}\rangle}_{\text{Full}}\}$$

A **pattern node** (denoted **x**, **y**, or **z**) consists of:

1. An equivalence class index i (for the ith class)
2. A pattern from $\mathcal{P}$
3. An entrance tree (A or B)
4. An orientation ($+$ or $-$, depending on A-tree)

Subtle: Requires $O(\log(m+g))$ bits to describe a pattern node.

Structure of Full Patterns

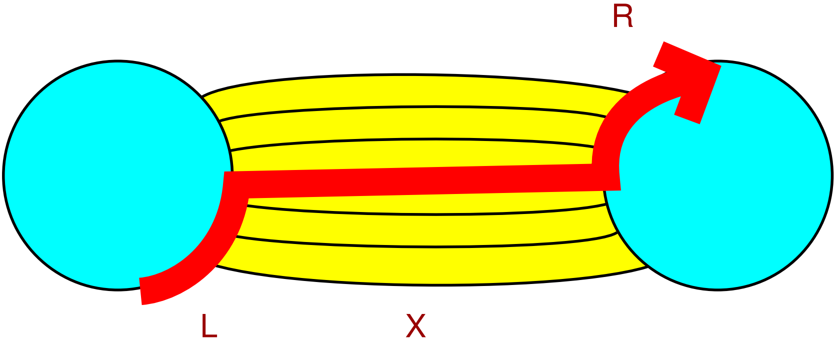
For a pattern node $\mathbf{x}$ that uses a *full pattern* over an equivalence class E_i, there are two edges:

$$e_{\mathbf{x}}^{\text{in}} \qquad \text{and} \qquad e_{\mathbf{x}}^{\text{out}}$$

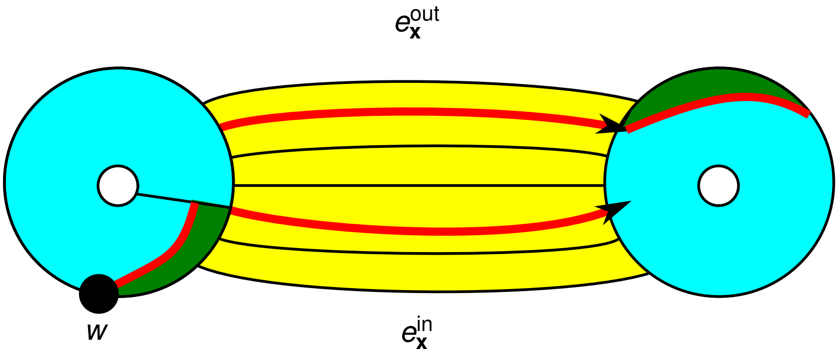
so that a vertex w has an irreducible path using tree, local, and E_i edges inducing $\mathbf{x}$, then

1. w can reach $e_{\mathbf{x}}^{\text{in}}$, and
2. everything w can reach with such paths is reachable from $e_{\mathbf{x}}^{\text{out}}$.

Structure of Full Patterns



Structure of Full Patterns



Intuition

For two pattern descriptions **x** and **y** with matching exit-entrance, we want:

Put an edge $\mathbf{x} \rightarrow \mathbf{y}$

$$\Longleftrightarrow$$

there is a local path from $e_{\mathbf{x}}^{\mathrm{out}}$ to $e_{\mathbf{y}}^{\mathrm{in}}$.

Intuition

For two pattern descriptions **x** and **y** with matching exit-entrance, we want:

Put an edge $\mathbf{x} \to \mathbf{y}$

$$\Longleftrightarrow$$

there is a local path from $e^{\text{out}}_{\mathbf{x}}$ to $e^{\text{in}}_{\mathbf{y}}$.

But nesting patterns mess this up!

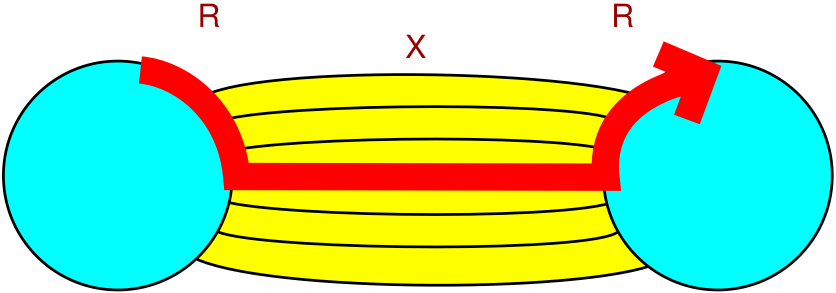
Structure of Nesting Patterns

For a pattern node $\mathbf{x}$ that uses a *nesting pattern* over an equivalence class E_i, we have

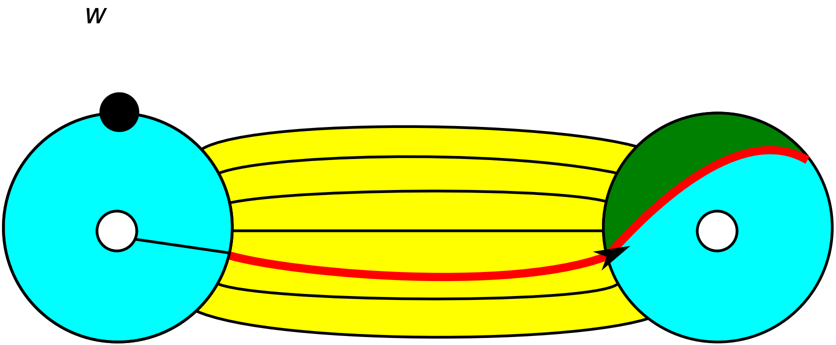
$$e_{\mathbf{x}}^{\text{in}} \qquad = \qquad e_{\mathbf{x}}^{\text{out}}$$

BUT: A vertex w can use the pattern without reaching $e_{\mathbf{x}}^{\text{in}}$!

Structure of Nesting Patterns

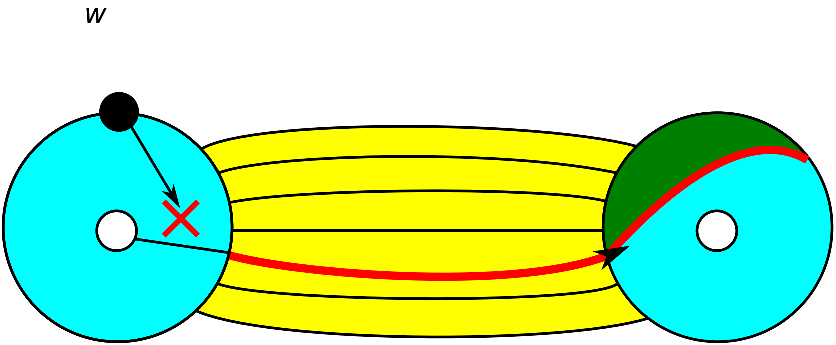


Structure of Nesting Patterns



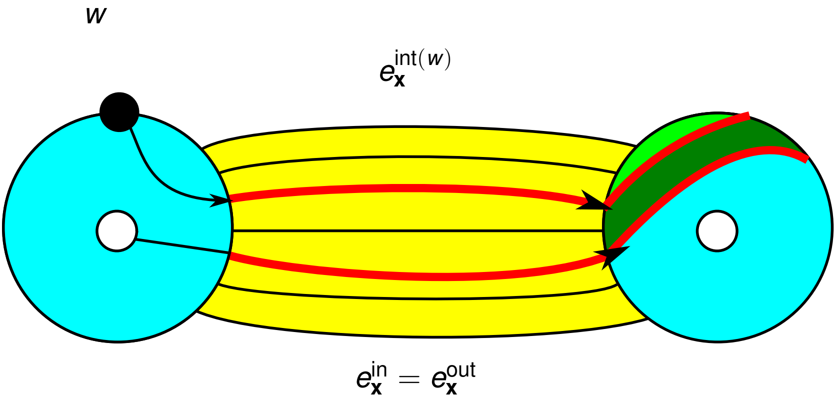
$$e_{\mathbf{x}}^{\text{in}} = e_{\mathbf{x}}^{\text{out}}$$

Structure of Nesting Patterns



$e_{\mathbf{x}}^{\text{in}} = e_{\mathbf{x}}^{\text{out}}$

Structure of Nesting Patterns

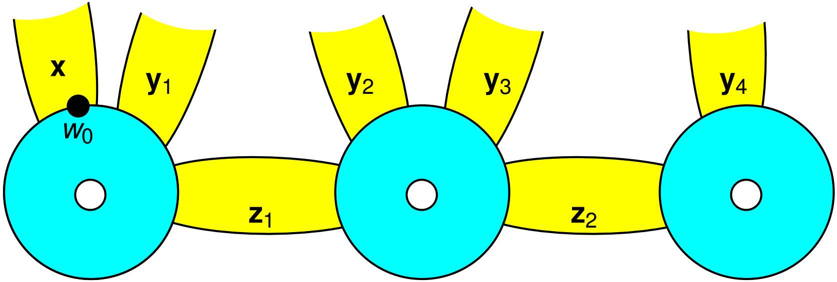


Adjacency Certificates

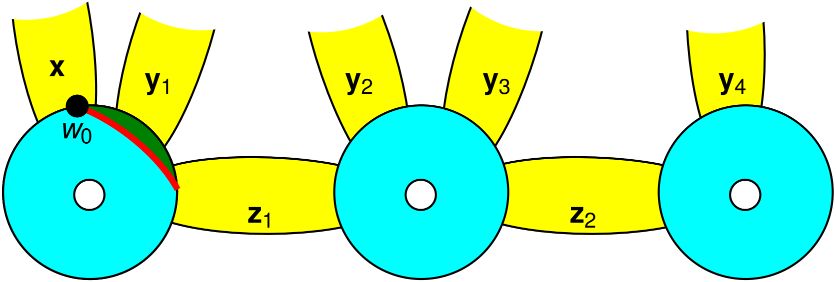
For two pattern descriptions $\mathbf{x}$, $\mathbf{y}$, place an edge $\mathbf{x} \to \mathbf{y}$ if and only if there is an **adjacency certificate** $\mathbf{z}_1, \ldots, \mathbf{z}_k$ of nesting patterns so that

1. Define $w_0 = \text{Head}(e_{\mathbf{x}}^{\text{out}})$ and $w_{j+1} = \text{Head}(e_{\mathbf{z}_{j+1}}^{\text{int}(w_j)})$.
2. For all $j \in \{0, \ldots, k-1\}$, the vertex w_j cannot reach $e_{\mathbf{z}_{j+1}}^{\text{in}}$ via local paths.
3. The vertex w_k can reach $e_{\mathbf{y}}^{\text{in}}$ via a local path.

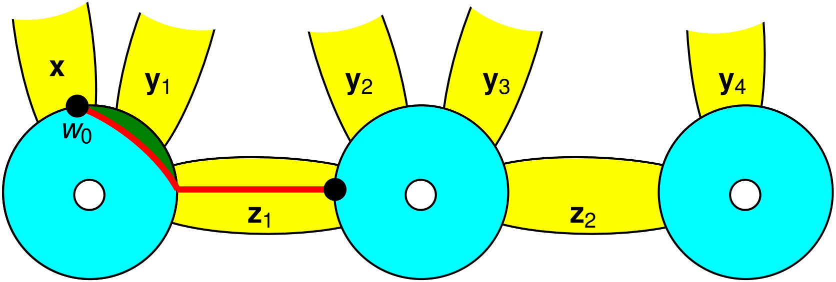
Adjacency Certificates



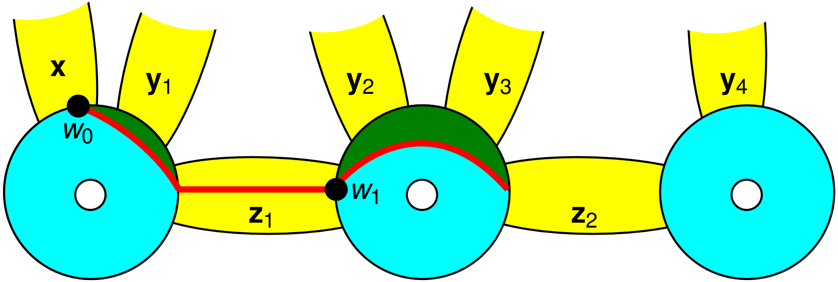
Adjacency Certificates



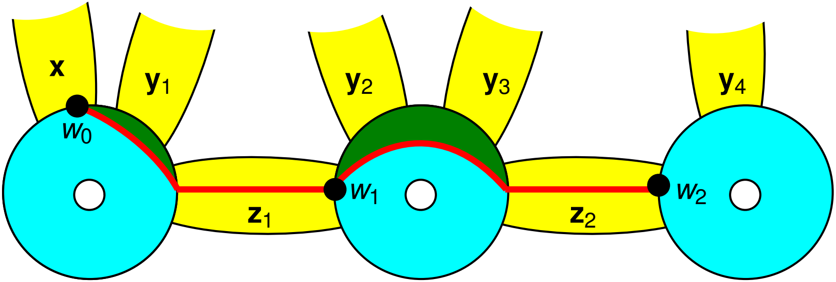
Adjacency Certificates



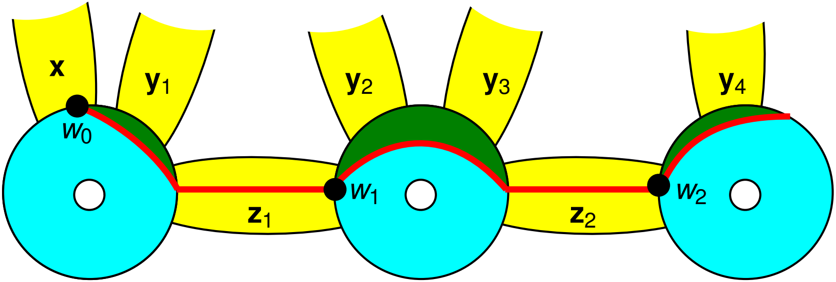
Adjacency Certificates



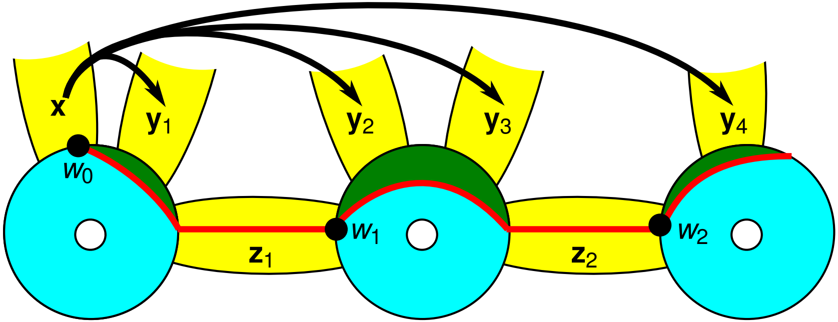
Adjacency Certificates



Adjacency Certificates



Adjacency Certificates



Special Vertices

Two special vertices: u' and v'.

1. $u' \to \mathbf{x}$ if and only if $\mathbf{x}$ is a pattern description over an equivalence class incident to T_u *and* starts on T_u.
2. $\mathbf{x} \to v'$ if and only if $\mathbf{x}$ is a pattern description over an equivalence class incident to T_v *and* ends on T_v.

Now, there is a path from u to v in G **if and only if** there is a path from u' to v' in G'!

Main Theorem

Theorem (Stolee, Vinodchandran, '12) Given a graph $G \in \mathcal{G}(m, g)$ and $u, v \in V(G)$, we can compute in **log-space** a graph G' with vertices u', v' so that

1. There is a path from u to v in G **if and only if** there is a path from u' to v' in G'.
2. G' has $O(m + g)$ vertices.

Future Directions

Extend this construction:

Future Directions

Extend this construction:

- Length of paths from u to v in G'?

Future Directions

Extend this construction:

- Length of paths from u to v in G'?
- "Smart" forest decomposition?

Future Directions

Extend this construction:

- Length of paths from u to v in G'?
- “Smart” forest decomposition?
- Further compression of G'?

Future Directions

Extend this construction:

- Length of paths from u to v in G'?
- “Smart” forest decomposition?
- Further compression of G'?

Completely new directions?

Space-efficient algorithms for reachability in surface-embedded graphs

Derrick Stolee* N. V. Vinodchandran

University of Nebraska–Lincoln
s-dstolee1@math.unl.edu
http://www.math.unl.edu/∼s-dstolee1/

CCC 2012
June 29, 2012

Supported by NSF grants DMS-0354008, DMS-0914815, and CCF-0916525, and a CCC Student Travel Grant.