Dynamic Graph Algorithms

Giuseppe F. Italiano

University of Rome "Tor Vergata"

giuseppe.italiano@uniroma2.it

http://www.disp.uniroma2.it/users/italiano

Outline

Dynamic Graph Problems – Quick Intro

Lecture 1. (Undirected Graphs)
  Dynamic Connectivity

Lecture 2. (Undirected/Directed Graphs)
  Dynamic Shortest Paths

Lecture 3. (Non-dynamic?)
  2-Connectivity in Directed Graphs

Outline

Dynamic Graph Problems – Quick Intro

Lecture 1. (Undirected Graphs)
  Dynamic Connectivity

Lecture 2. (Undirected/Directed Graphs)
  Dynamic Shortest Paths

Lecture 3. (Non-dynamic?)
  2-Connectivity in Directed Graphs

Today's Outline

1. 2-Connectivity on directed graphs
2. Algorithms for strong articulation points and strong bridges
3. Experiments
4. Open Problems

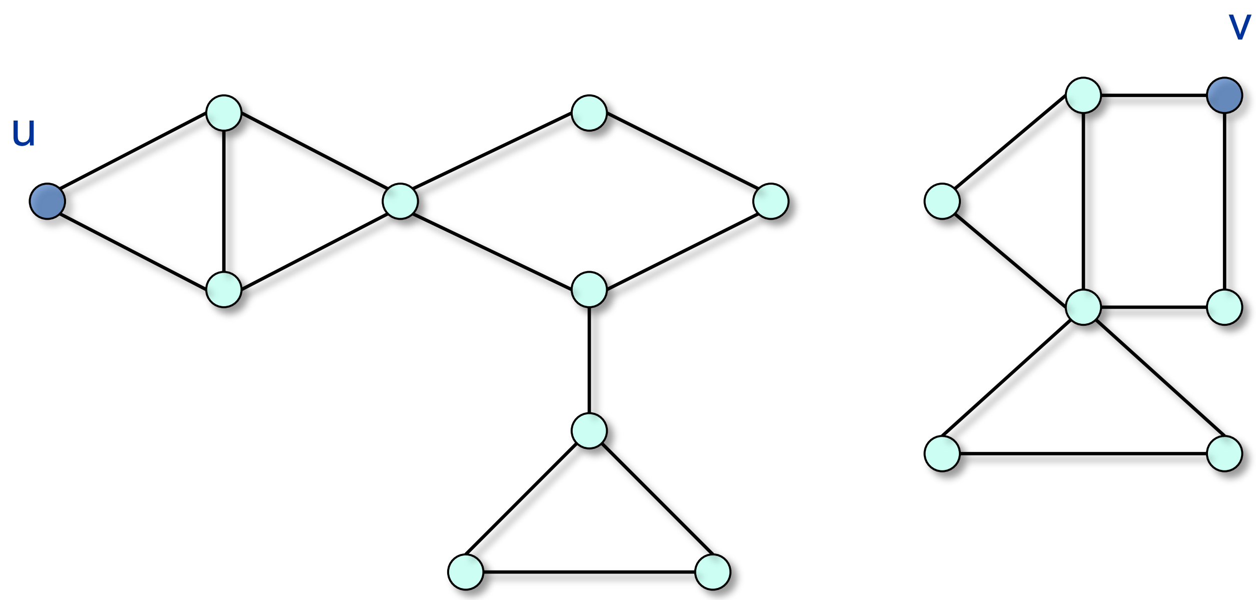
Today's Outline

1. **2-Connectivity on directed graphs**
2. Algorithms for strong articulation points and strong bridges
3. Experiments
4. Open Problems

Graph Connectivity

- Fundamental concept in Graph Theory.

- Numerous practical applications, e.g. :
  → Reliable and secure communication
  → Routing
  → Navigation
  → …



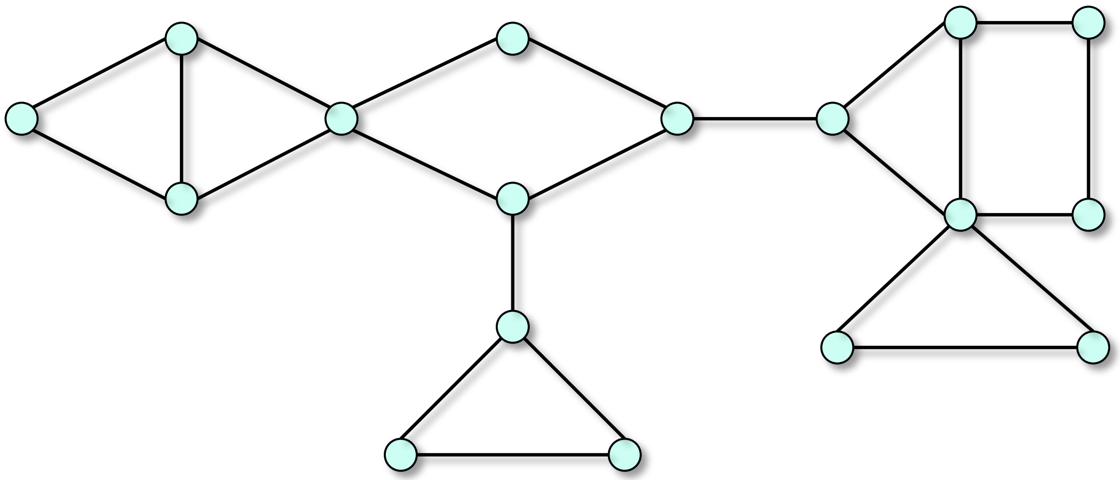
2-Edge Connectivity

Let $G = (V,E)$ be an **undirected** connected graph, with m edges and n vertices.

An edge $e \in E$ is a **bridge** if its removal increases the number of connected components of G.

Graph G is **2-edge-connected** if it has no bridges.

The **2-edge-connected components** of G are its maximal 2-edge-connected subgraphs.



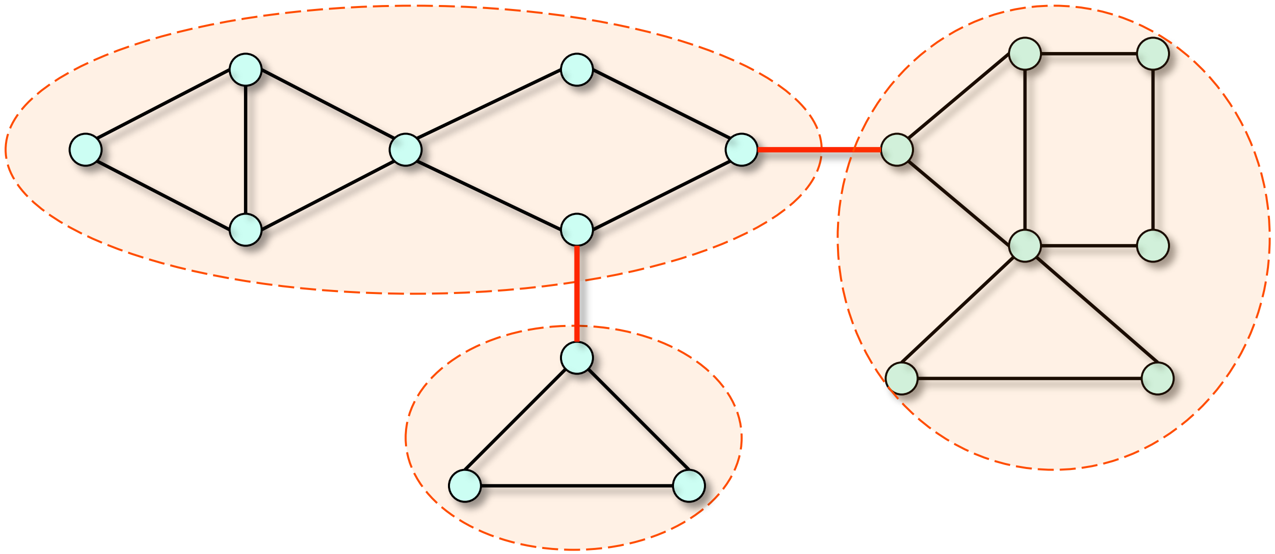
2-Edge Connectivity

Let $G = (V,E)$ be an **undirected** connected graph, with m edges and n vertices.

An edge $e \in E$ is a **bridge** if its removal increases the number of connected components of G.

Graph G is **2-edge-connected** if it has no bridges.

The **2-edge-connected components** of G are its maximal 2-edge-connected subgraphs.

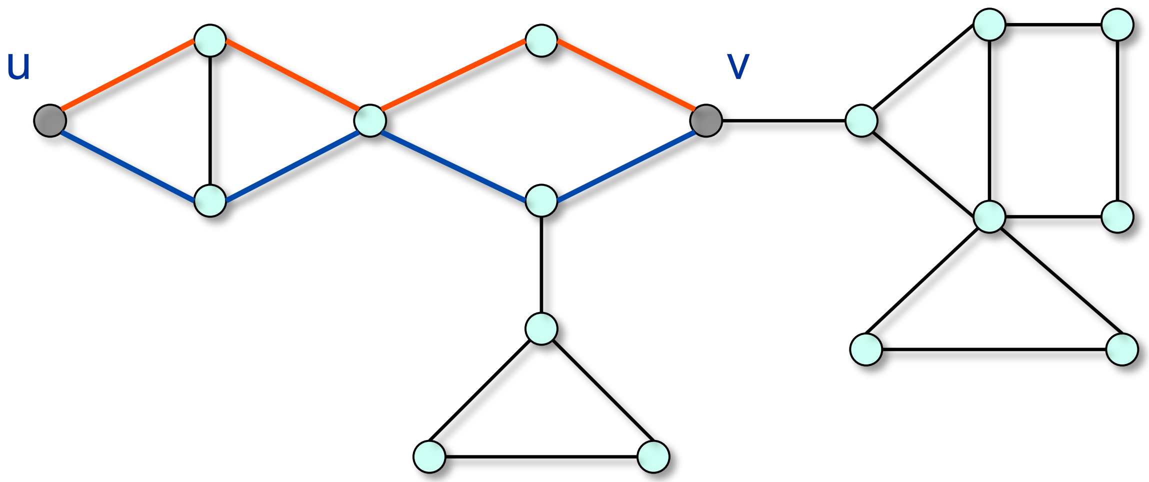


(Point2Point) 2-Edge Connectivity

Vertices u and v are **2-edge-connected** if if there are two edge-disjoint paths between u and v

By Menger's Theorem, vertices u and v are 2-edge-connected if and only removal of any edge leaves them in same connected component.

Can define a **2-edge-connected block** of G as a maximal subset $B \subseteq V$ s. t. u and v are 2-edge-connected for all $u, v \in B$.

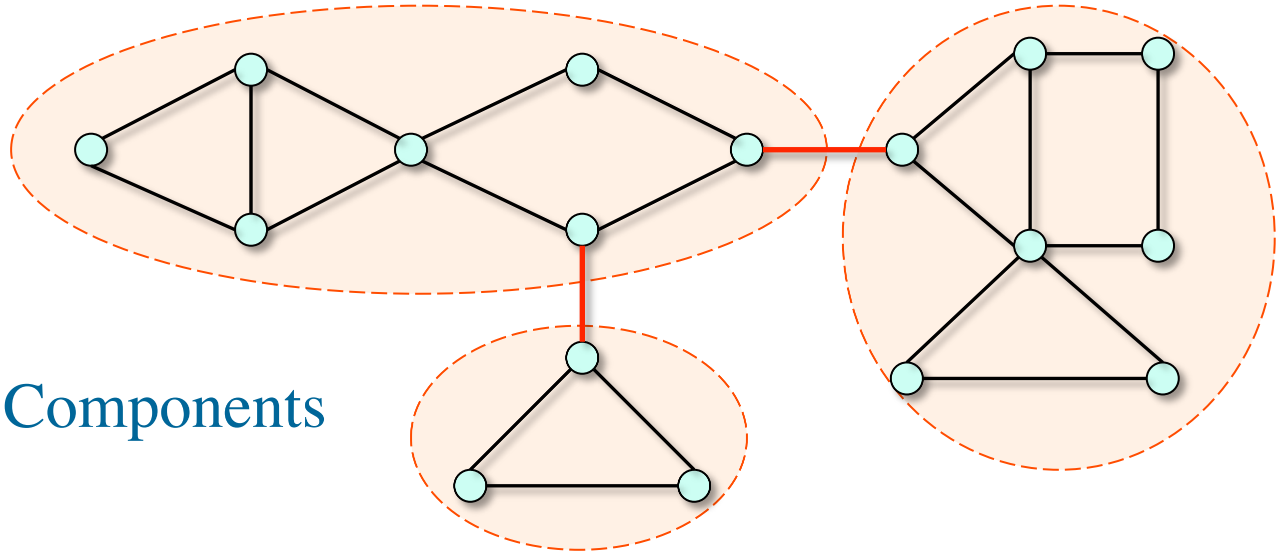


(Point2Point) 2-Edge Connectivity

Vertices u and v are **2-edge-connected** if there are two edge-disjoint paths between u and v

By Menger's Theorem, vertices u and v are 2-edge-connected if and only removal of any edge leaves them in same connected component.

Can define a **2-edge-connected block** of G as a maximal subset $B \subseteq V$ s. t. u and v are 2-edge-connected for all $u, v \in B$.



Blocks = Components

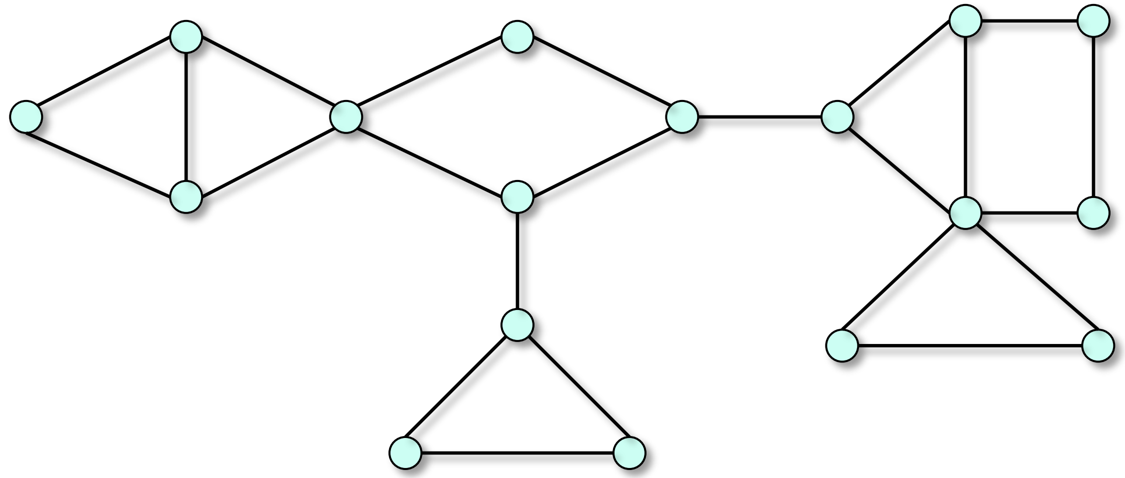
2-Vertex Connectivity

Let $G = (V,E)$ be an **undirected** connected graph, with m edges and n vertices.

A vertex $v \in V$ is an **articulation point** if its removal increases the number of connected components of G.

Graph G is **2-vertex-connected** if it has at least 3 vertices (don't allow for degenerate components) and no articulation points.

The **2-vertex-connected components** of G are its maximal 2-vertex-connected subgraphs.



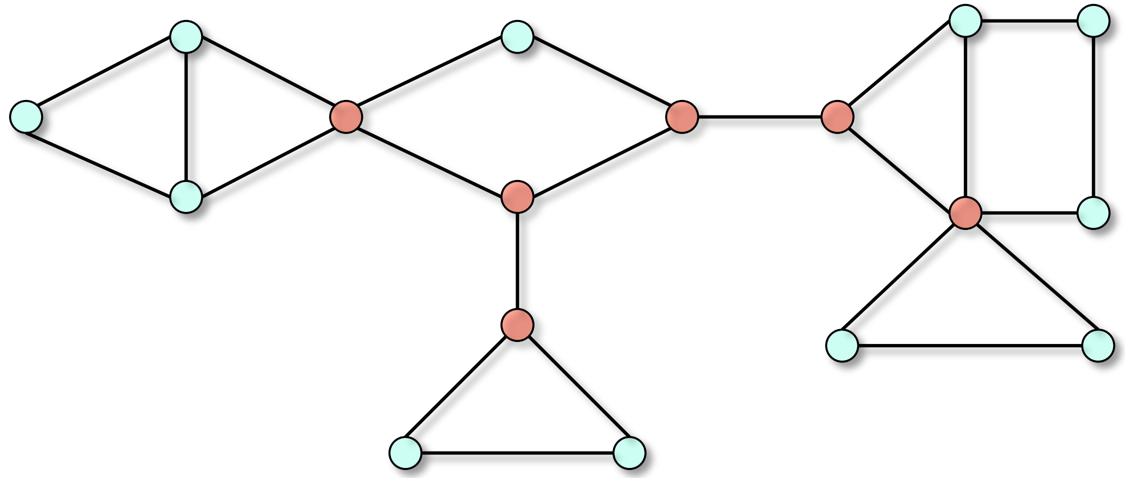
2-Vertex Connectivity

Let $G = (V,E)$ be an **undirected** connected graph, with m edges and n vertices.

A vertex $v \in V$ is an **articulation point** if its removal increases the number of connected components of G.

Graph G is **2-vertex-connected** if it has at least 3 vertices (don't allow for degenerate components) and no articulation points.

The **2-vertex-connected components** of G are its maximal 2-vertex-connected subgraphs.



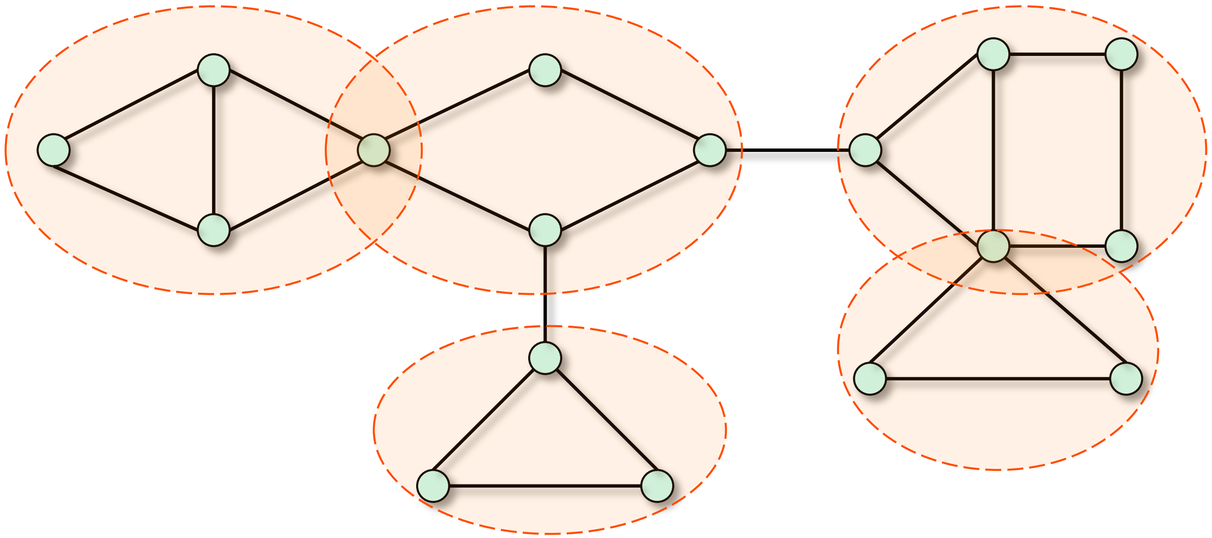
2-Vertex Connectivity

Let $G = (V,E)$ be an **undirected** connected graph, with m edges and n vertices.

A vertex $v \in V$ is an **articulation point** if its removal increases the number of connected components of G.

Graph G is **2-vertex-connected** if it has at least 3 vertices (don't allow for degenerate components) and no articulation points.

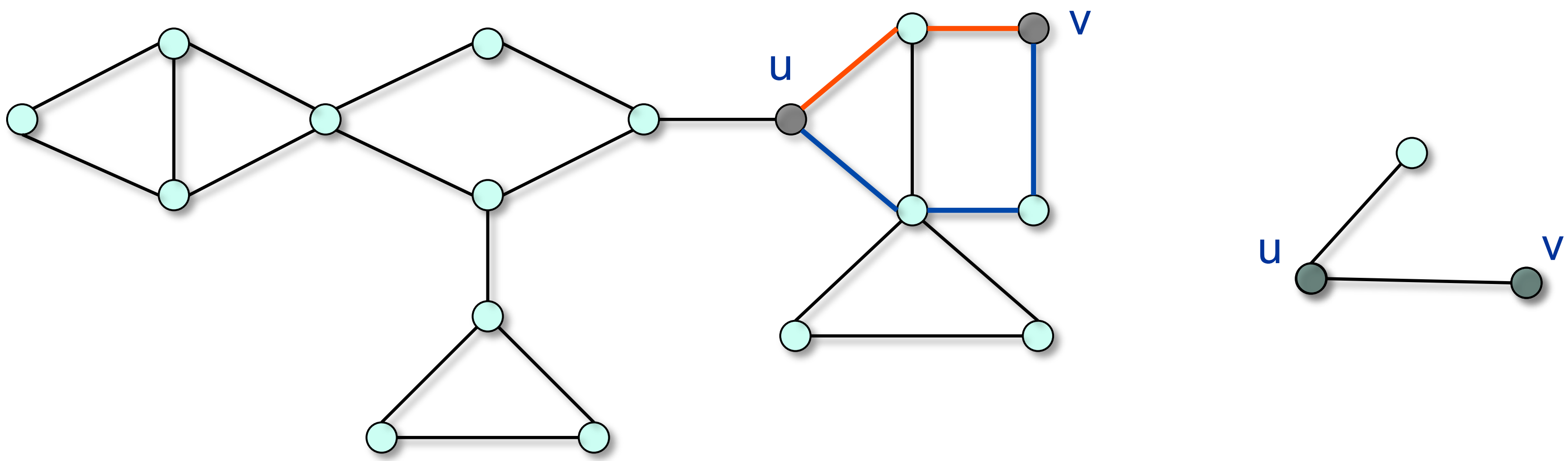
The **2-vertex-connected components** of G are its maximal 2-vertex-connected subgraphs.



(P2P) 2-Vertex Connectivity

Vertices u and v are **2-vertex-connected** if there are two (internally) vertex-disjoint paths between u and v.

By Menger's Theorem, if vertices u and v are 2-vertex-connected, then removal of any vertex ($\neq u, v$) leaves them in same connected component.



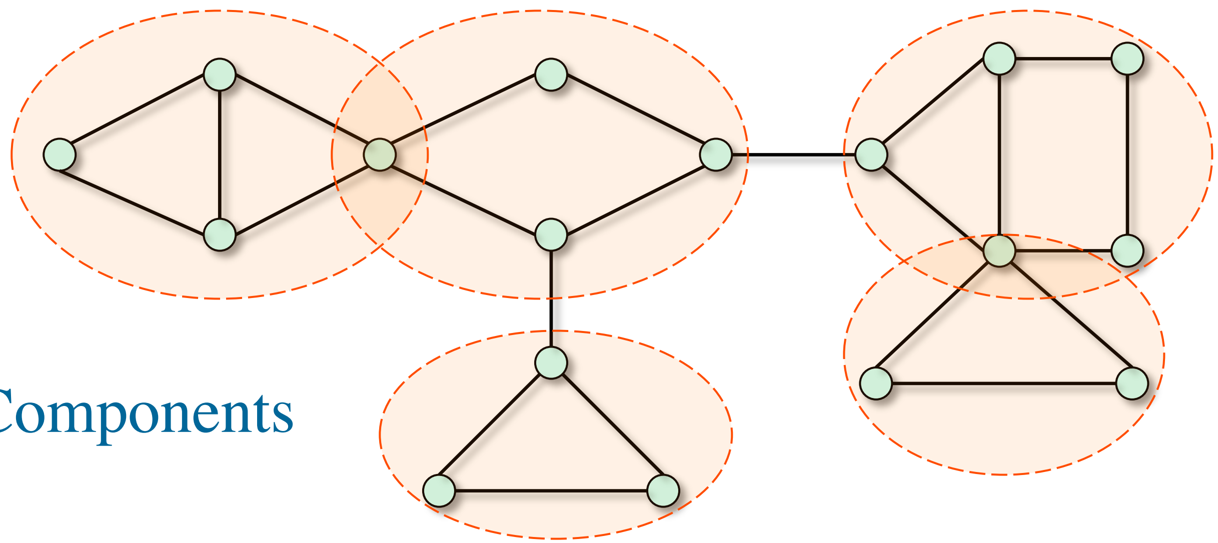
Can define a **2-vertex-connected block** of G as a maximal subset $B \subseteq V$ s. t. u and v are 2-vertex-connected for all $u, v \in B$.

(P2P) 2-Vertex Connectivity

Vertices u and v are **2-vertex-connected** if there are two (internally) vertex-disjoint paths between u and v.

By Menger's Theorem, if vertices u and v are 2-vertex-connected, then removal of any vertex ($\neq u, v$) leaves them in same connected component.

Can define a **2-vertex-connected block** of G as a maximal subset $B \subseteq V$ s. t. u and v are 2-vertex-connected for all $u, v \in B$.



Blocks = Components

Bounds for Undirected *G*

Q1: Find whether *G* is 2-vertex-connected (2-edge-connected).
I.e., find **one** connectivity cut (if any) — ***O(m+n)***

Q2: Find **all** connectivity cuts (articulation points, bridges) in *G* — ***O(m+n)***

Q3: Find the **2-connectivity** (2-vertex-, 2-edge- connected) **blocks** of *G* — ***O(m+n)***

Q4: Find the **2-connectivity** (2-vertex-, 2-edge-connected) **components** of *G* — ***O(m+n)***

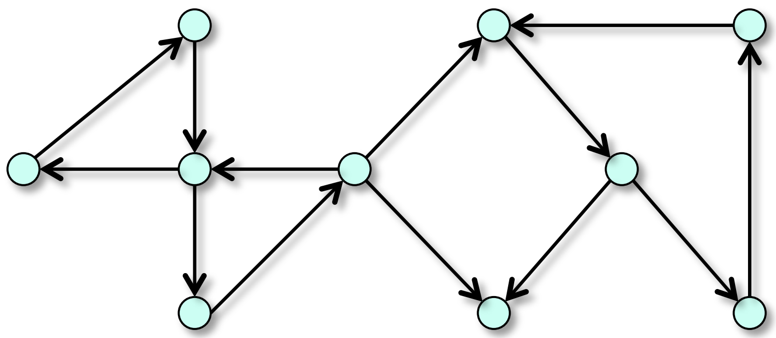
[R.E.Tarjan, SIAM Journal on Computing 1972]

Directed Graphs

Let $G = (V,E)$ be a **directed** graph, with m edges and n vertices.

G is **strongly connected** if there is a directed path from each vertex to every other vertex in G.

The **strongly connected components** (SCCs) of G are its maximal **strongly** connected subgraphs.

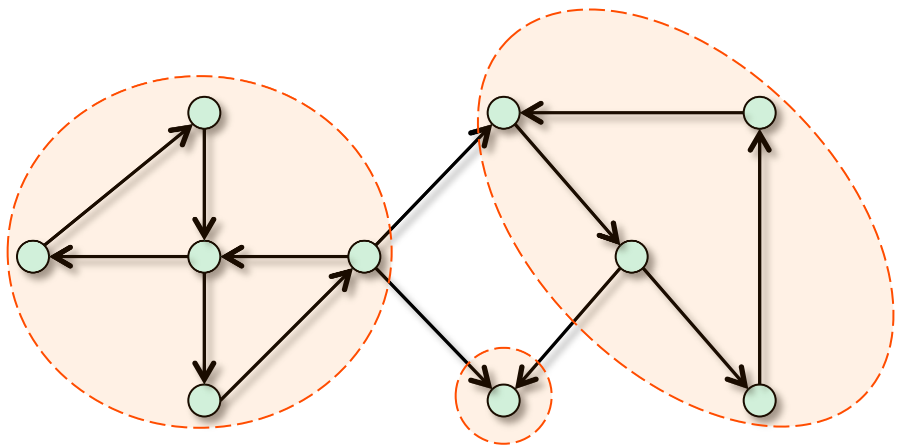


Directed Graphs

Let $G = (V,E)$ be a **directed** graph, with m edges and n vertices.

G is **strongly connected** if there is a directed path from each vertex to every other vertex in G.

The **strongly connected components** (SCCs) of G are its maximal **strongly** connected subgraphs.



Undirected: 2-Edge Connectivity

Let $G = (V,E)$ be a connected graph, with m edges and n vertices.

An edge $(u,v) \in E$ is a **bridge** if its removal increases the number of connected components of G

Graph G is **2-edge-connected** if it has no bridges.

The **2-edge-connected components** of G are its maximal 2-edge-connected subgraphs

Directed: 2-Edge Connectivity

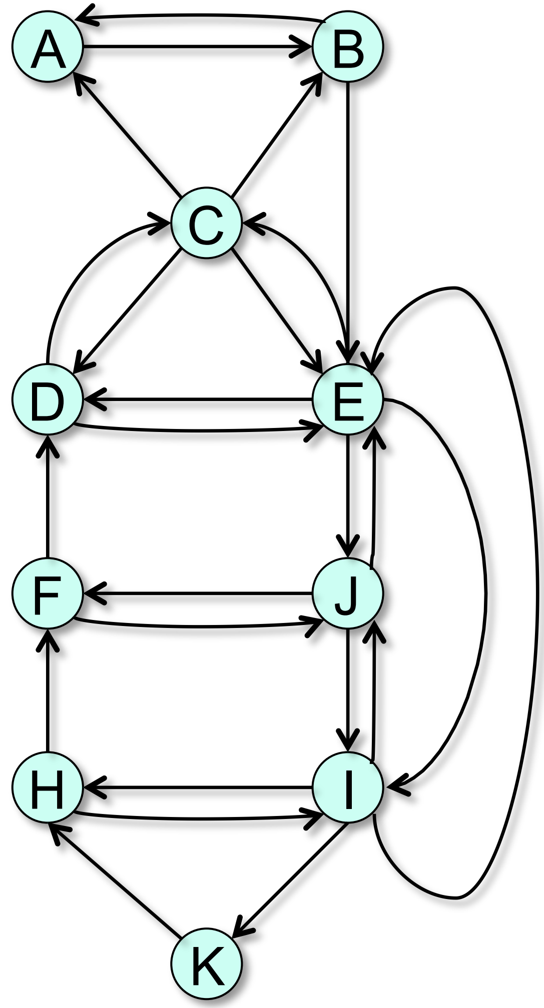
Let $G = (V,E)$ be a ***directed strongly*** connected graph, with m edges and n vertices.

An edge $(u,v) \in E$ is a ***strong*** **bridge** if its removal increases the number of ***strongly*** connected components of G

Directed: 2-Edge Connectivity

Let $G = (V,E)$ be a ***directed strongly*** connected graph, with m edges and n vertices.

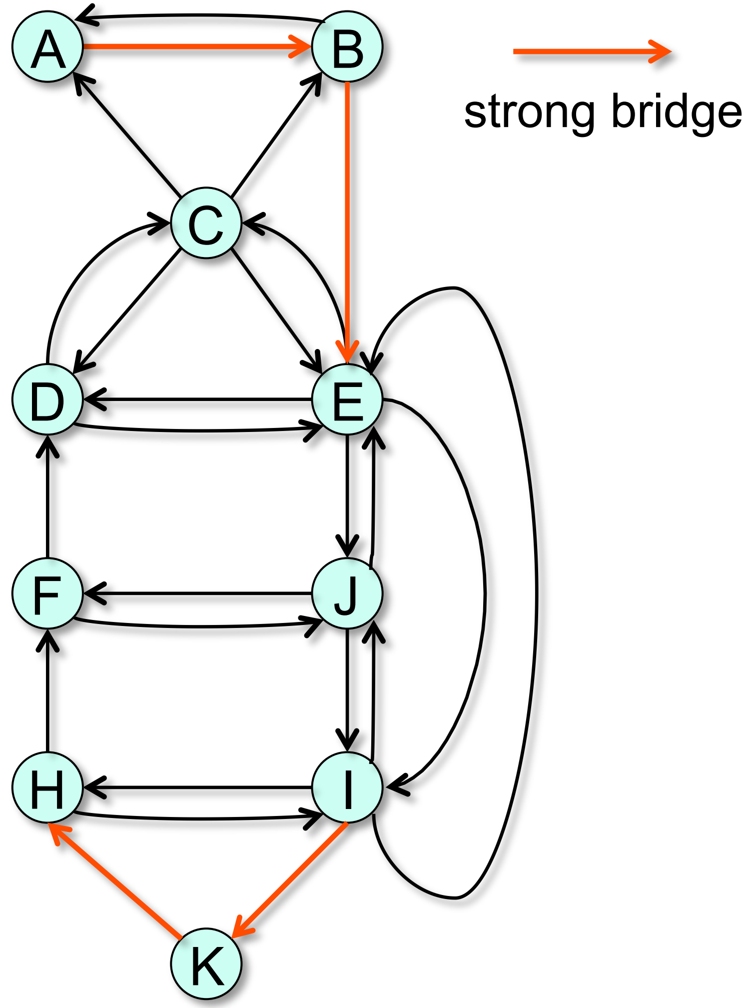
An edge $(u,v) \in E$ is a ***strong*** **bridge** if its removal increases the number of ***strongly*** connected components of G



Directed: 2-Edge Connectivity

Let $G = (V,E)$ be a ***directed strongly*** connected graph, with m edges and n vertices.

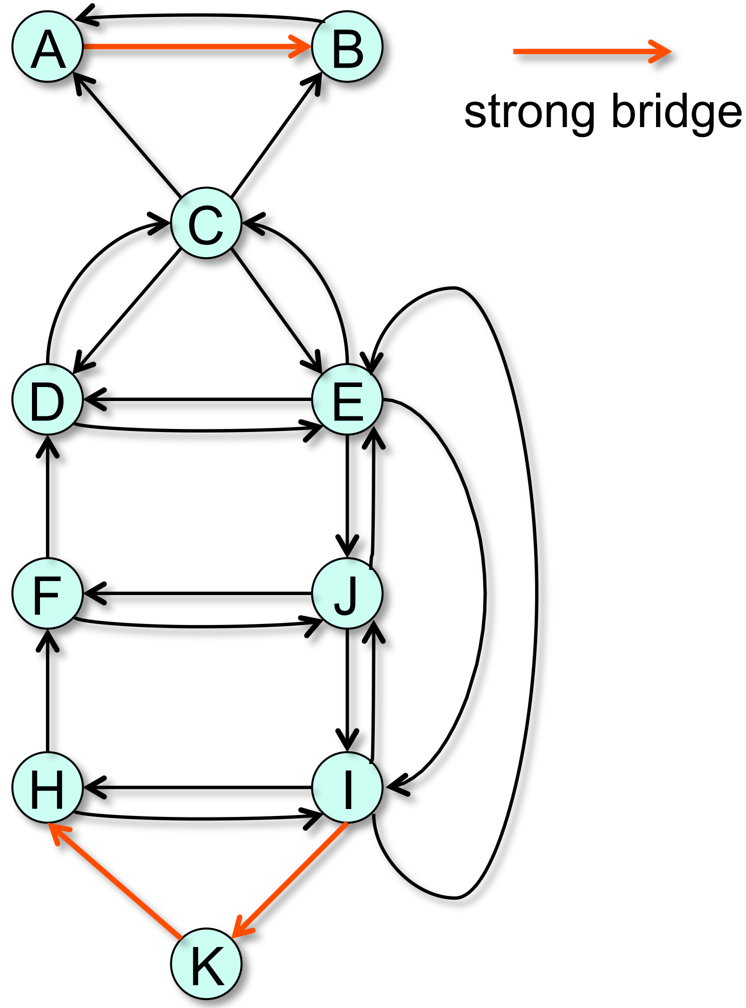
An edge $(u,v) \in E$ is a ***strong*** **bridge** if its removal increases the number of ***strongly*** connected components of G



Directed: 2-Edge Connectivity

Let $G = (V,E)$ be a ***directed strongly*** connected graph, with m edges and n vertices.

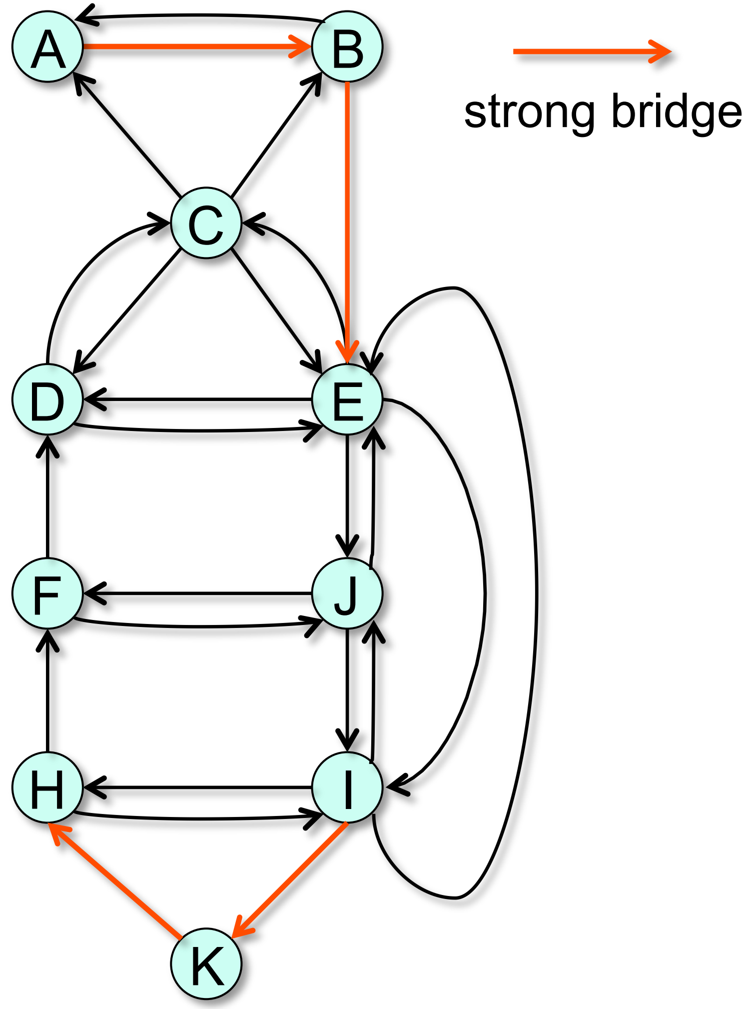
An edge $(u,v) \in E$ is a ***strong*** **bridge** if its removal increases the number of ***strongly*** connected components of G



Directed: 2-Edge Connectivity

Let $G = (V,E)$ be a ***directed strongly*** connected graph, with m edges and n vertices.

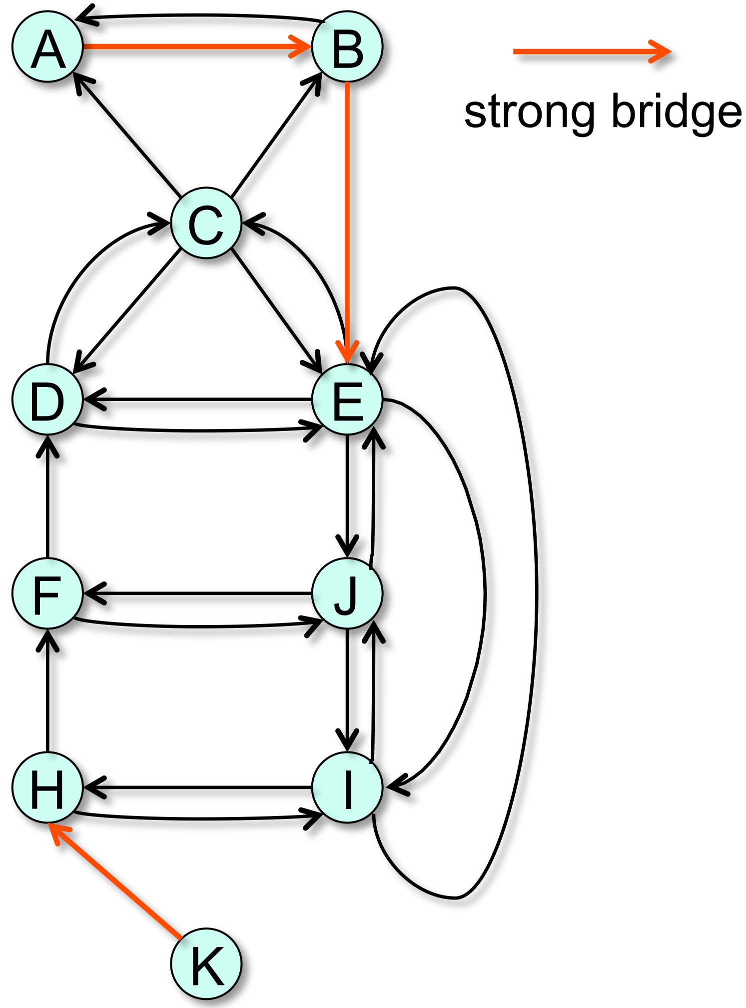
An edge $(u,v) \in E$ is a ***strong*** **bridge** if its removal increases the number of ***strongly*** connected components of G



Directed: 2-Edge Connectivity

Let $G = (V,E)$ be a ***directed strongly*** connected graph, with m edges and n vertices.

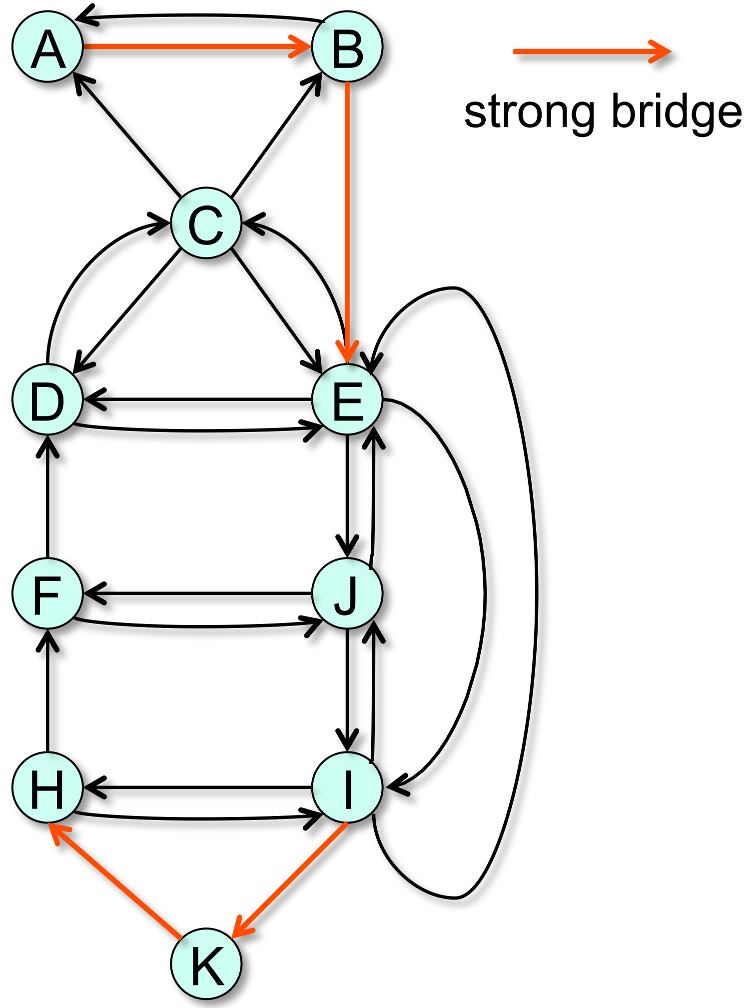
An edge $(u,v) \in E$ is a ***strong* bridge** if its removal increases the number of ***strongly*** connected components of G



Directed: 2-Edge Connectivity

Let $G = (V,E)$ be a ***directed strongly*** connected graph, with m edges and n vertices.

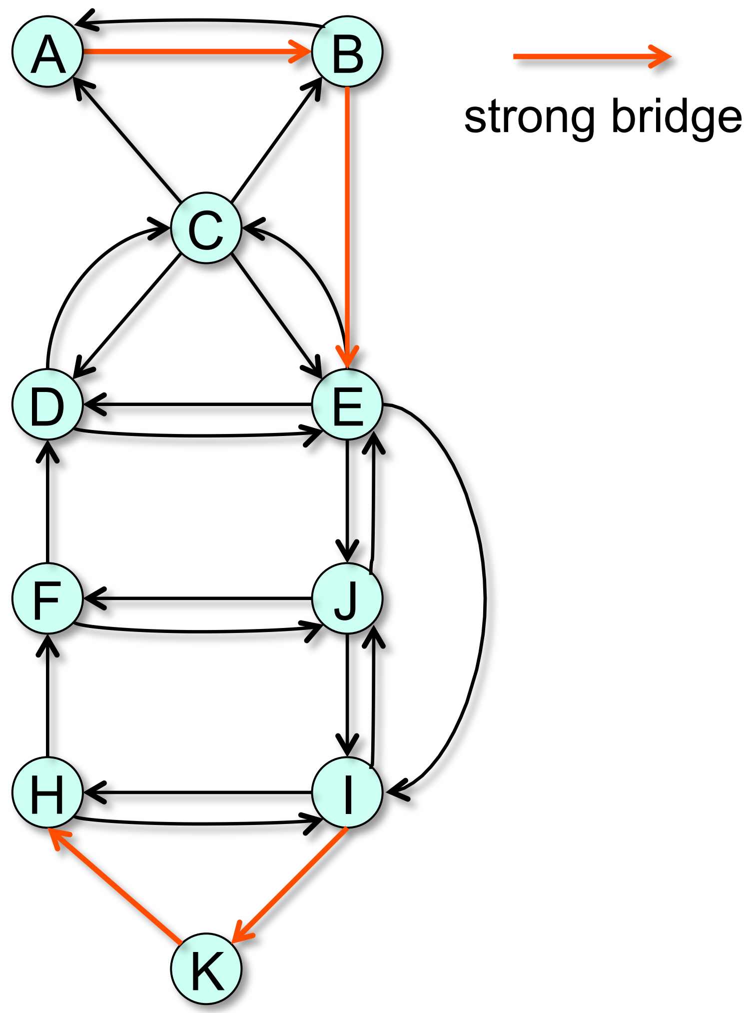
An edge $(u,v) \in E$ is a ***strong*** **bridge** if its removal increases the number of ***strongly*** connected components of G



Directed: 2-Edge Connectivity

Let $G = (V,E)$ be a ***directed strongly*** connected graph, with m edges and n vertices.

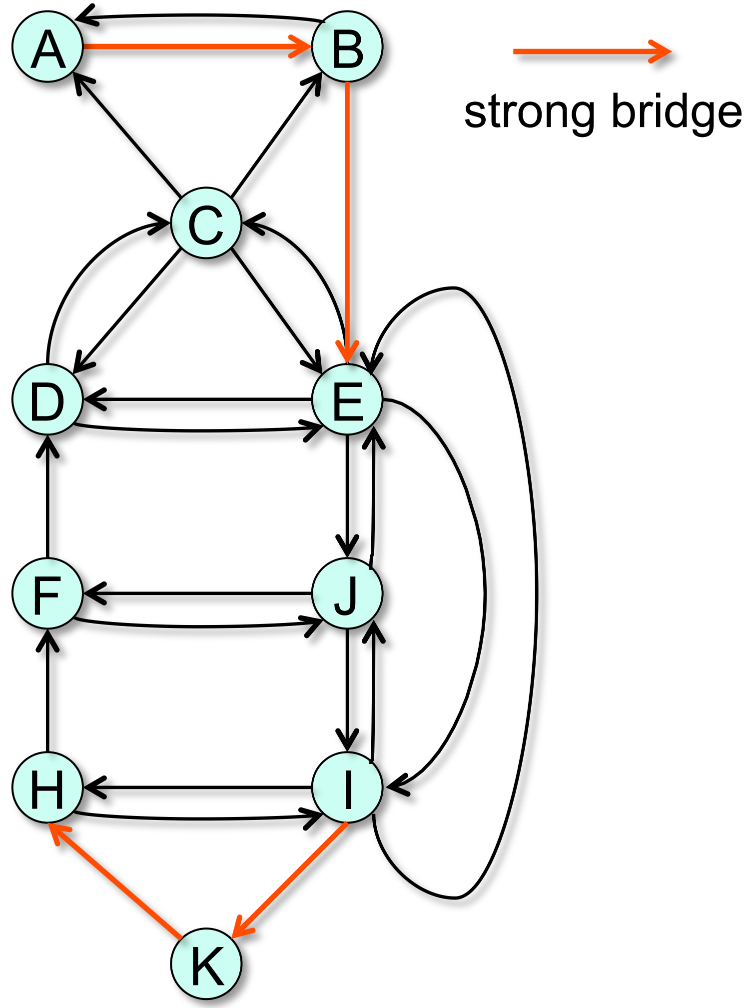
An edge $(u,v) \in E$ is a ***strong* bridge** if its removal increases the number of ***strongly*** connected components of G



Directed: 2-Edge Connectivity

Let $G = (V,E)$ be a ***directed strongly*** connected graph, with m edges and n vertices.

An edge $(u,v) \in E$ is a ***strong*** **bridge** if its removal increases the number of ***strongly*** connected components of G



Directed: 2-Edge Connectivity

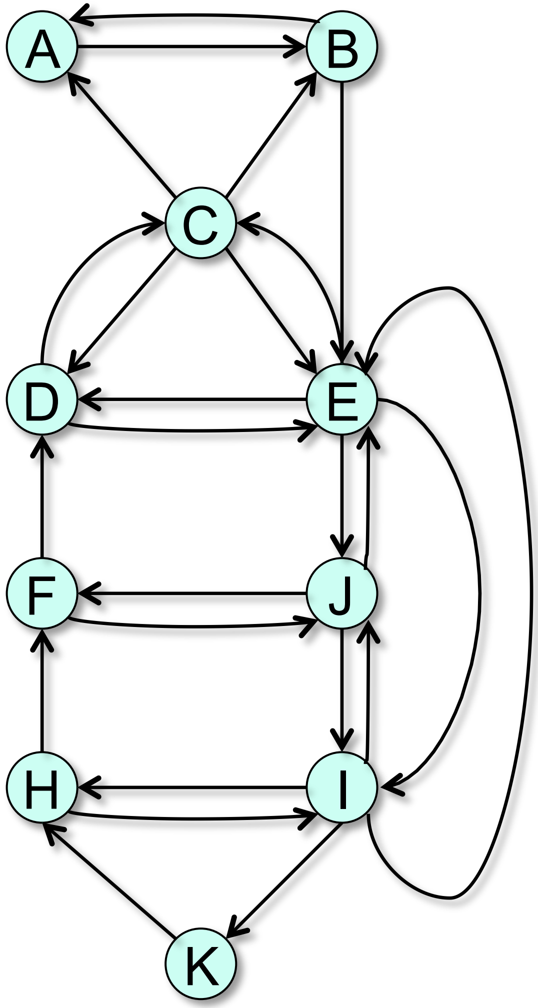
Let $G = (V,E)$ be a ***directed strongly*** connected graph, with m edges and n vertices.

An edge $(u,v) \in E$ is a ***strong* bridge** if its removal increases the number of ***strongly*** connected components of G

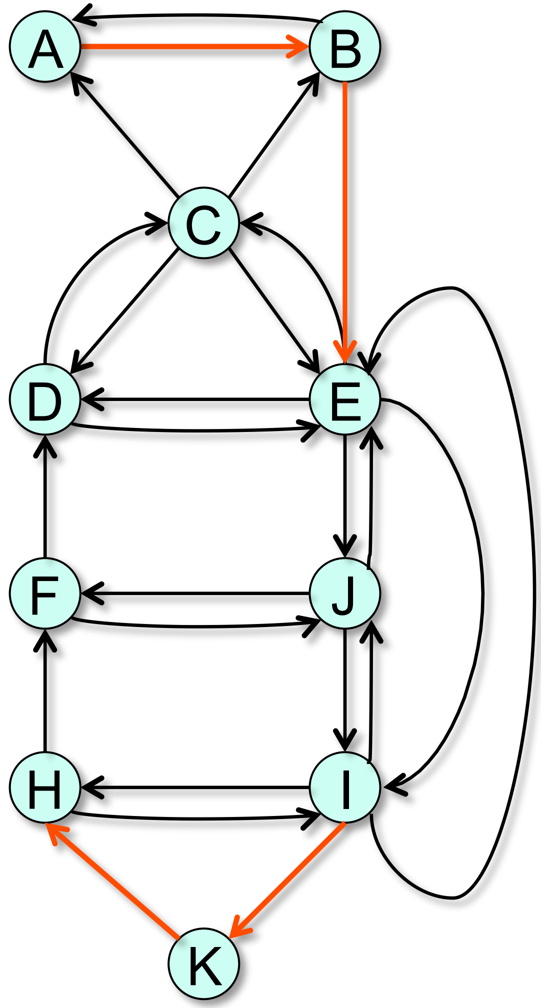
Graph G is **2-edge-connected** if it has no ***strong*** bridges.

The **2-edge-connected components** of G are its maximal 2-edge-connected subgraphs

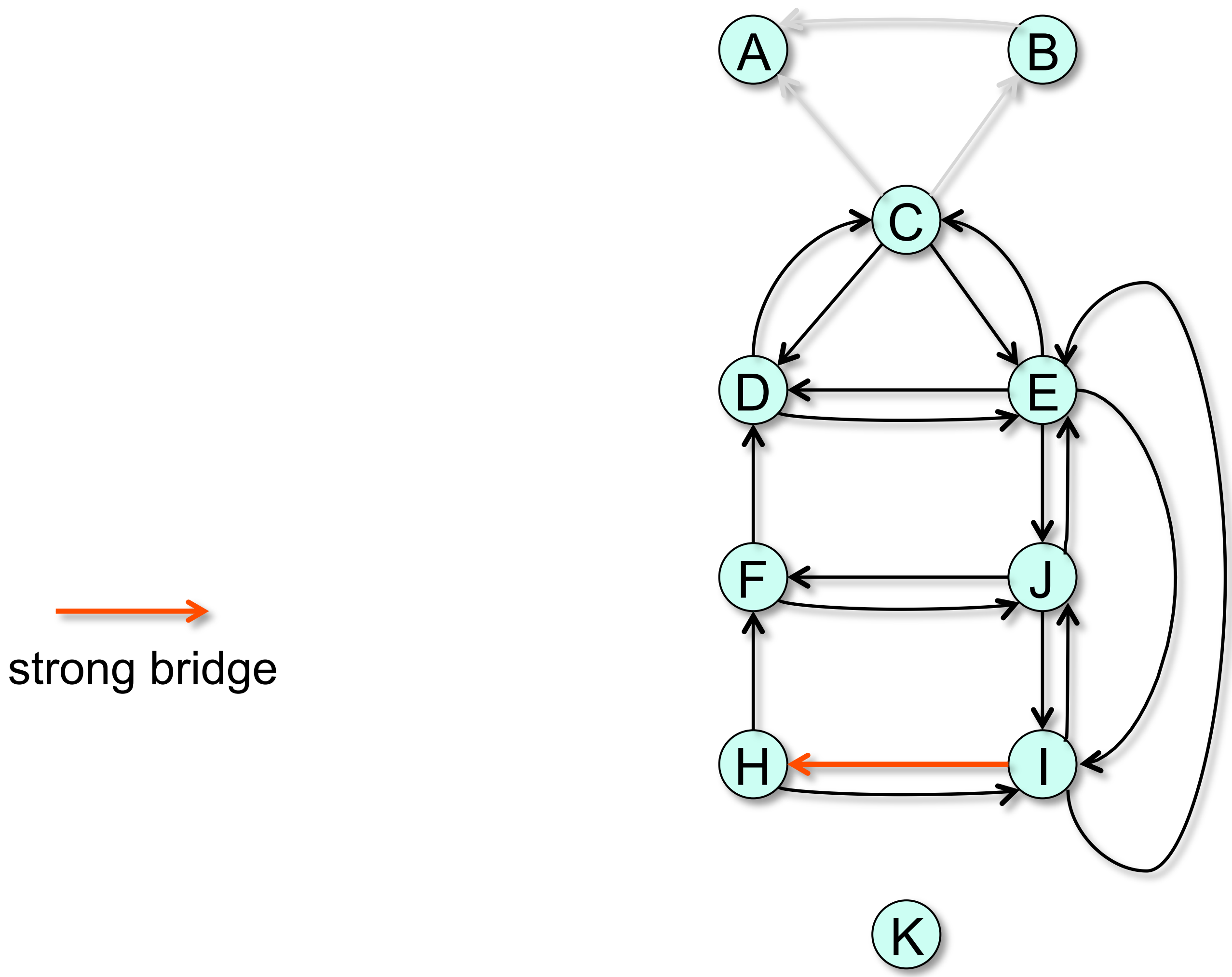
2-Edge Connected Components



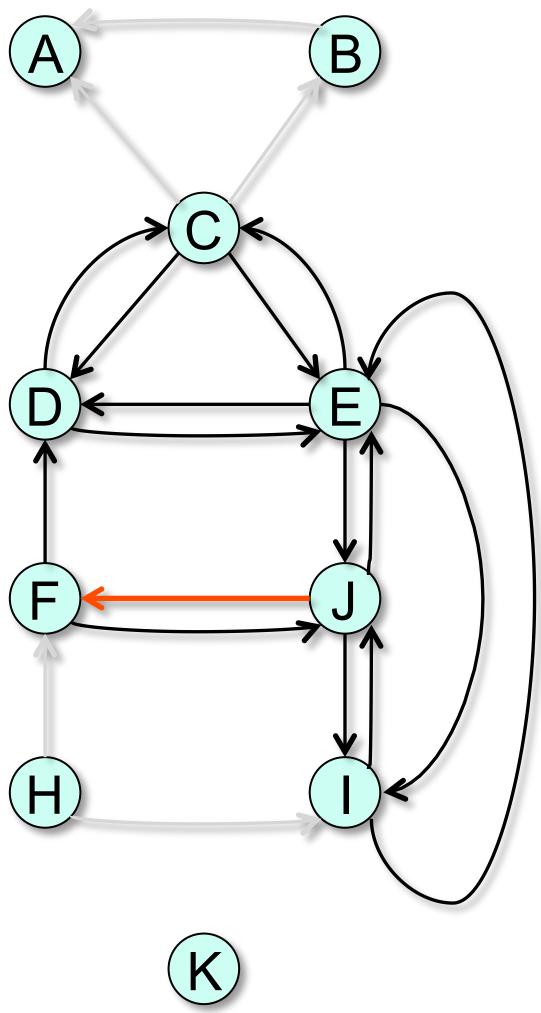
2-Edge Connected Components



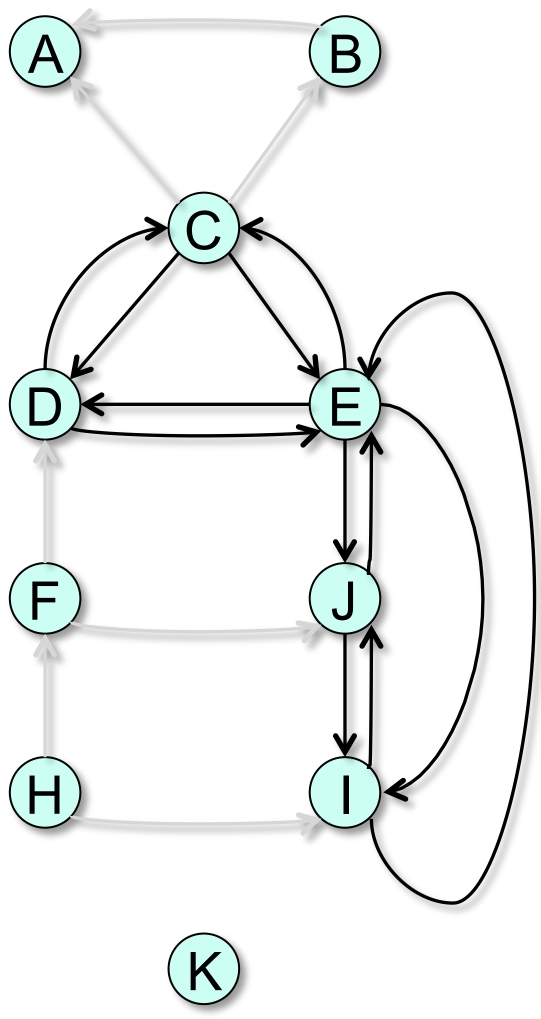
2-Edge Connected Components



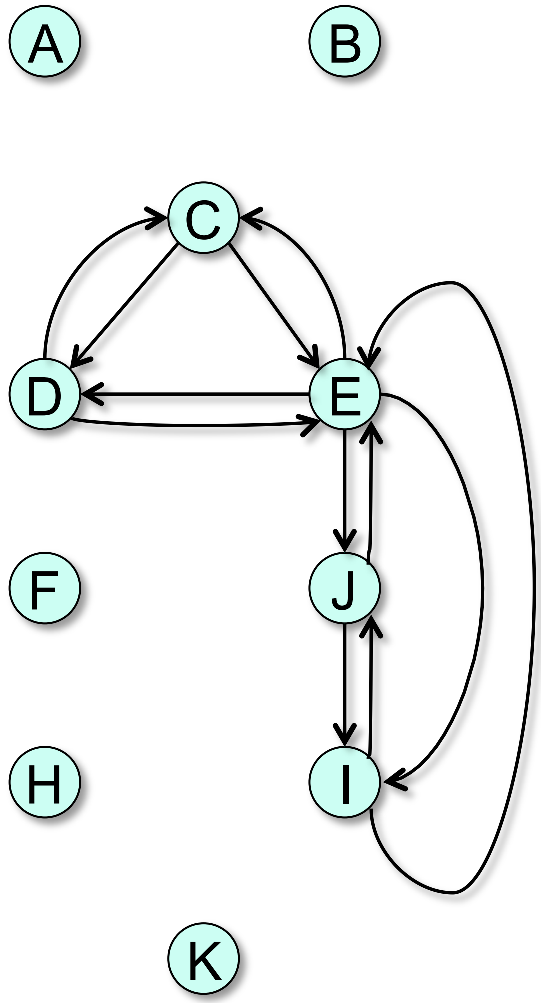
2-Edge Connected Components



2-Edge Connected Components



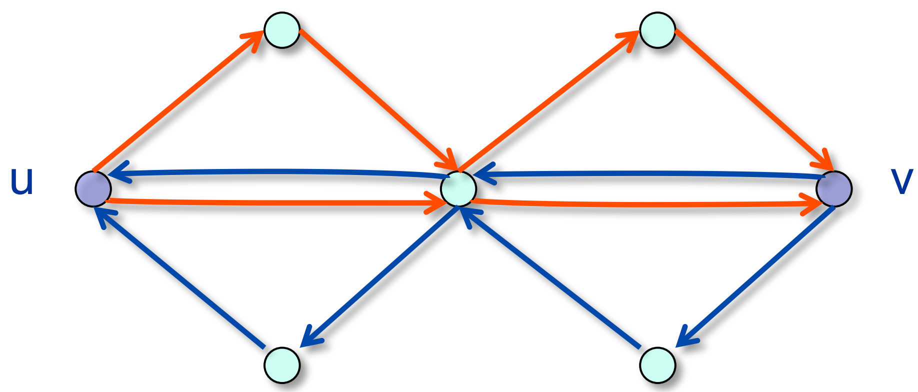
2-Edge Connected Components



P2P 2-Edge Connectivity

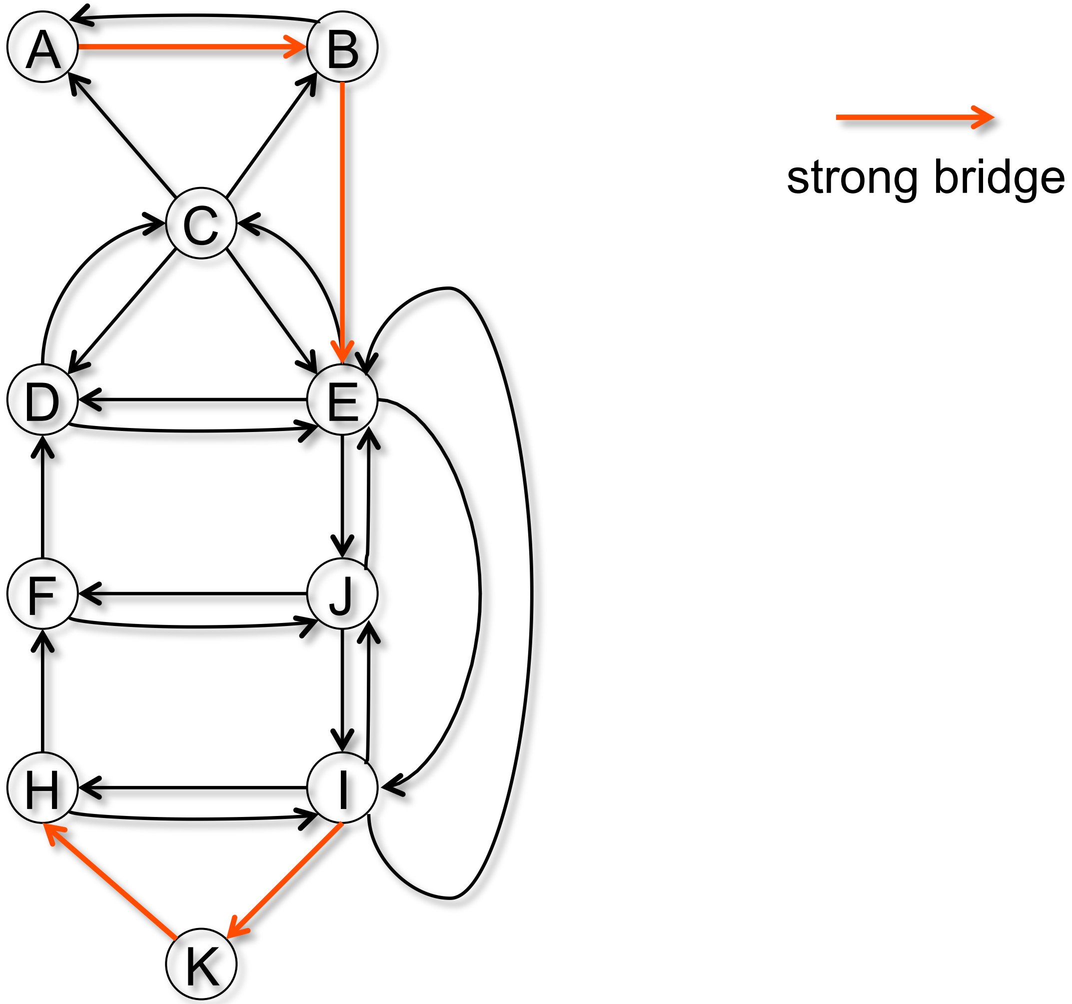
Vertices u and v are **2-edge-connected** if there are two edge-disjoint paths **from** u **to** v **and two edge-disjoint paths from v to u**.

By **Menger's Theorem,** vertices u and v are **2-edge-connected** if and only if the removal of any edge leaves them in same **strongly** connected component.

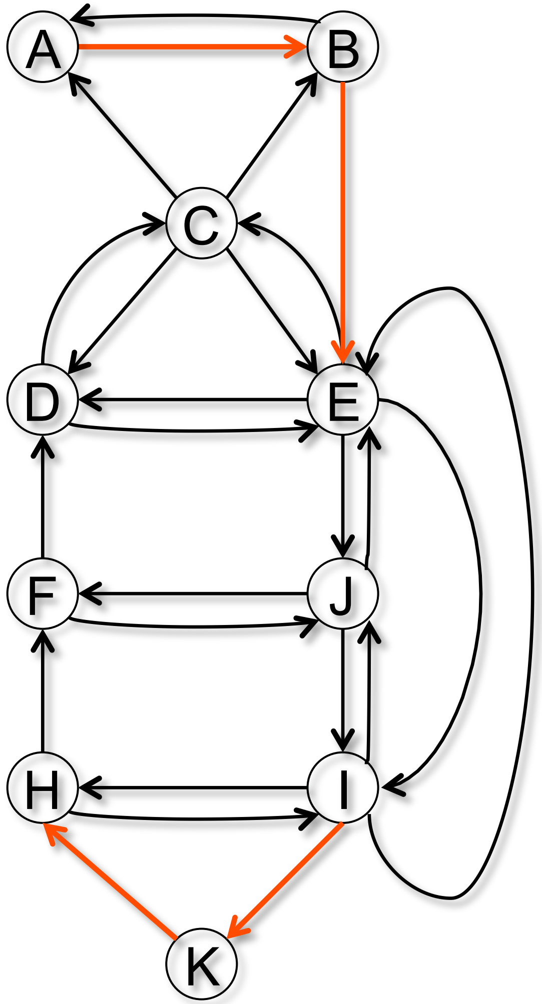


Can define a **2-edge-connected block** of G as a maximal subset $B \subseteq V$ s. t. u and v are **2-edge-connected** for all $u, v \in B$.

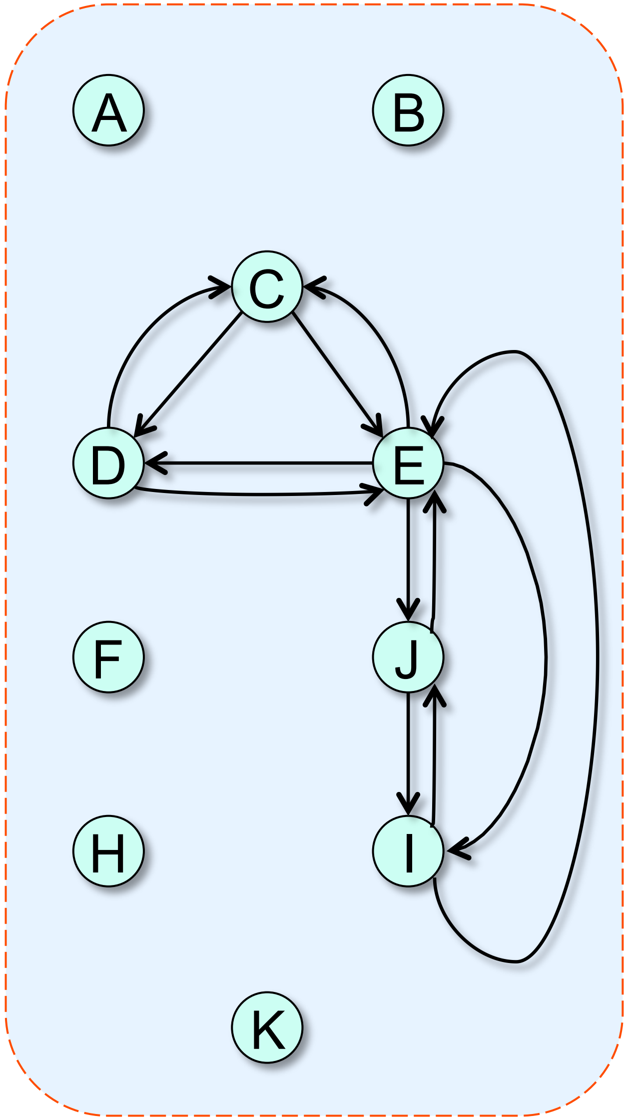
2-Edge-Connected Comp's ≠ Blocks



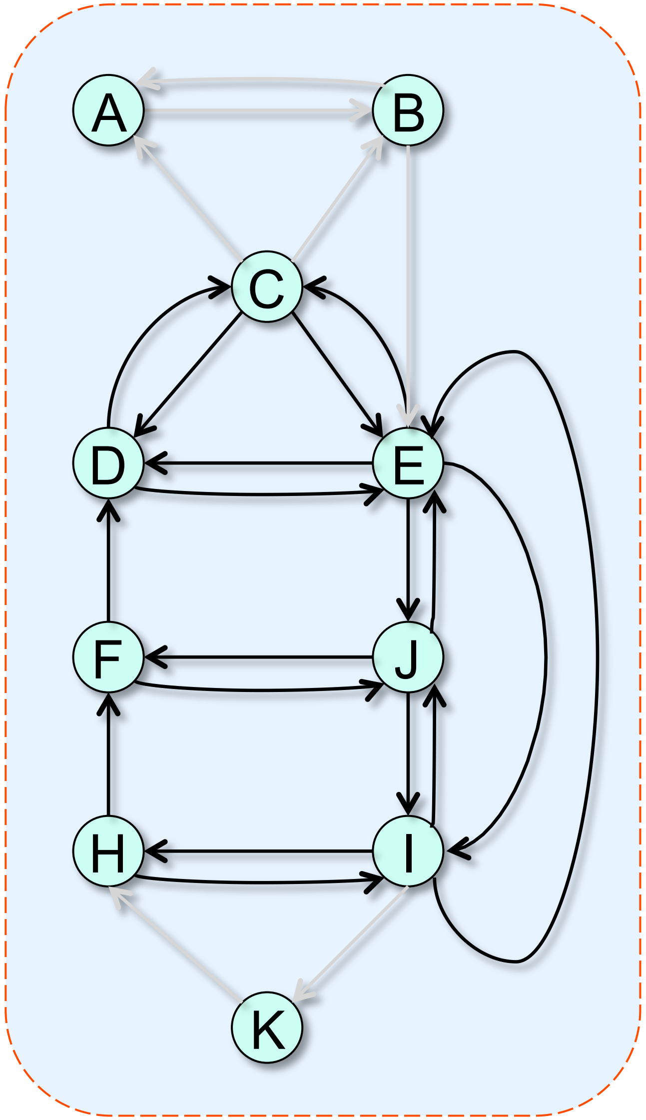
2-Edge-Connected Comp's ≠ Blocks



2-edge-connected components



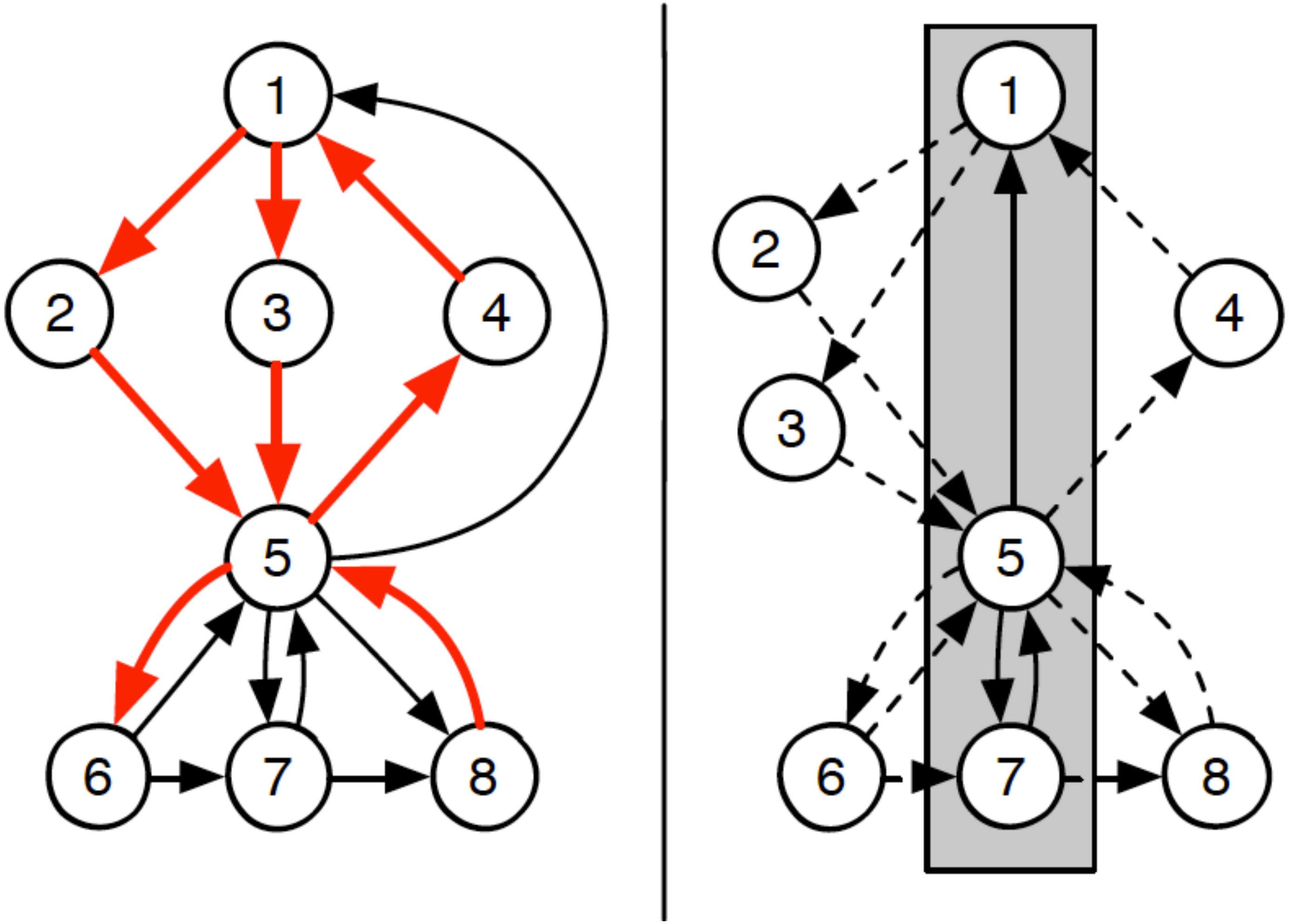
2-edge-connected blocks



2-Edge-Connected Blocks

How easy is it to compute 2-edge-connected blocks?

Can we just remove strong bridges?



Undirected: 2-Vertex Connectivity

Let $G = (V,E)$ be an **undirected** connected graph, with m edges and n vertices.

A vertex $v \in V$ is an **articulation point** if its removal increases the number of connected components of G.

Graph G is **2-vertex-connected** if it has at least 3 vertices (don't allow for degenerate components) and no articulation points.

The **2-vertex-connected components** of G are its maximal 2-vertex-connected subgraphs.

Directed: 2-Vertex Connectivity

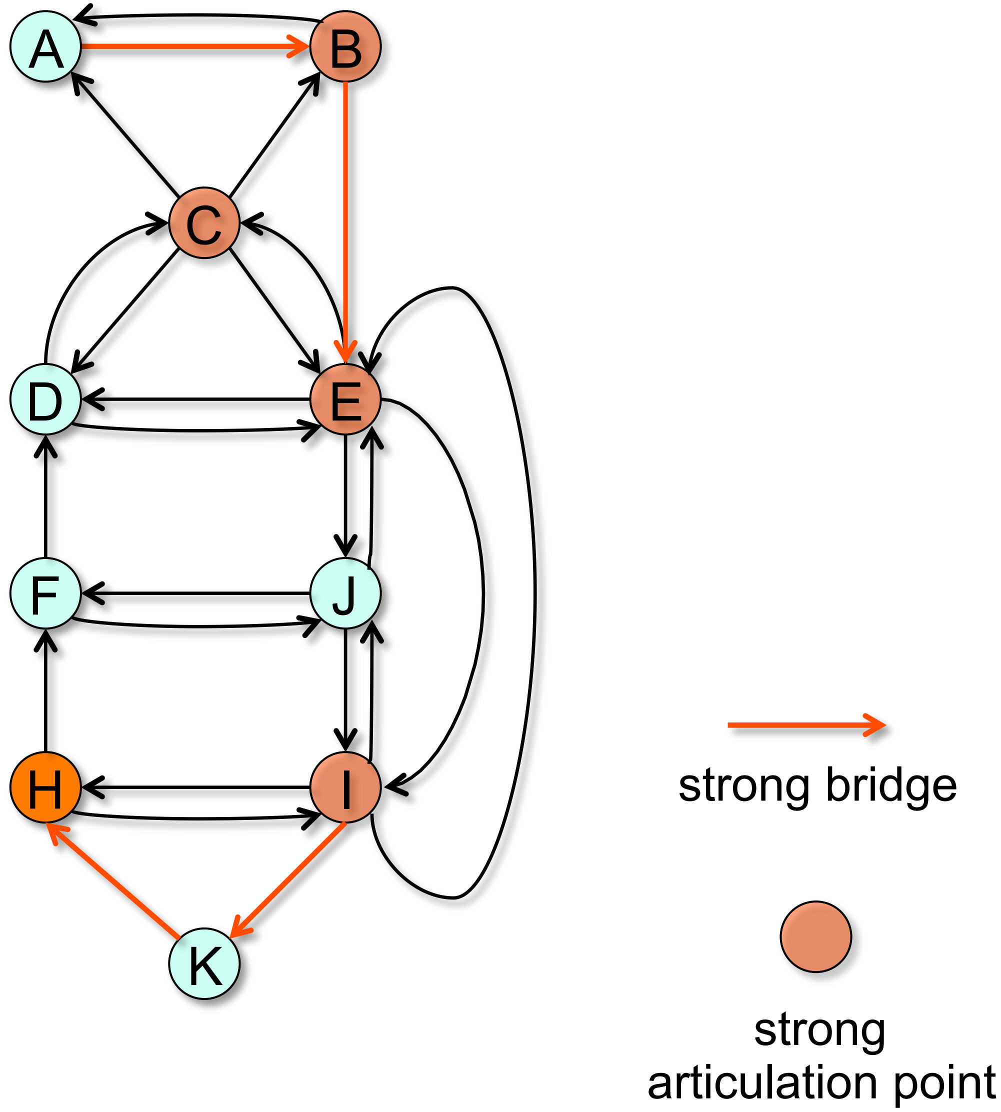
Let $G = (V,E)$ be a **directed strongly** connected graph, with m edges and n vertices.

A vertex $v \in V$ is a **strong articulation point** if its removal, increases the number of **strongly** connected components of G.

Directed: 2-Vertex Connectivity

Let $G = (V,E)$ be a **directed strongly** connected graph, with m edges and n vertices.

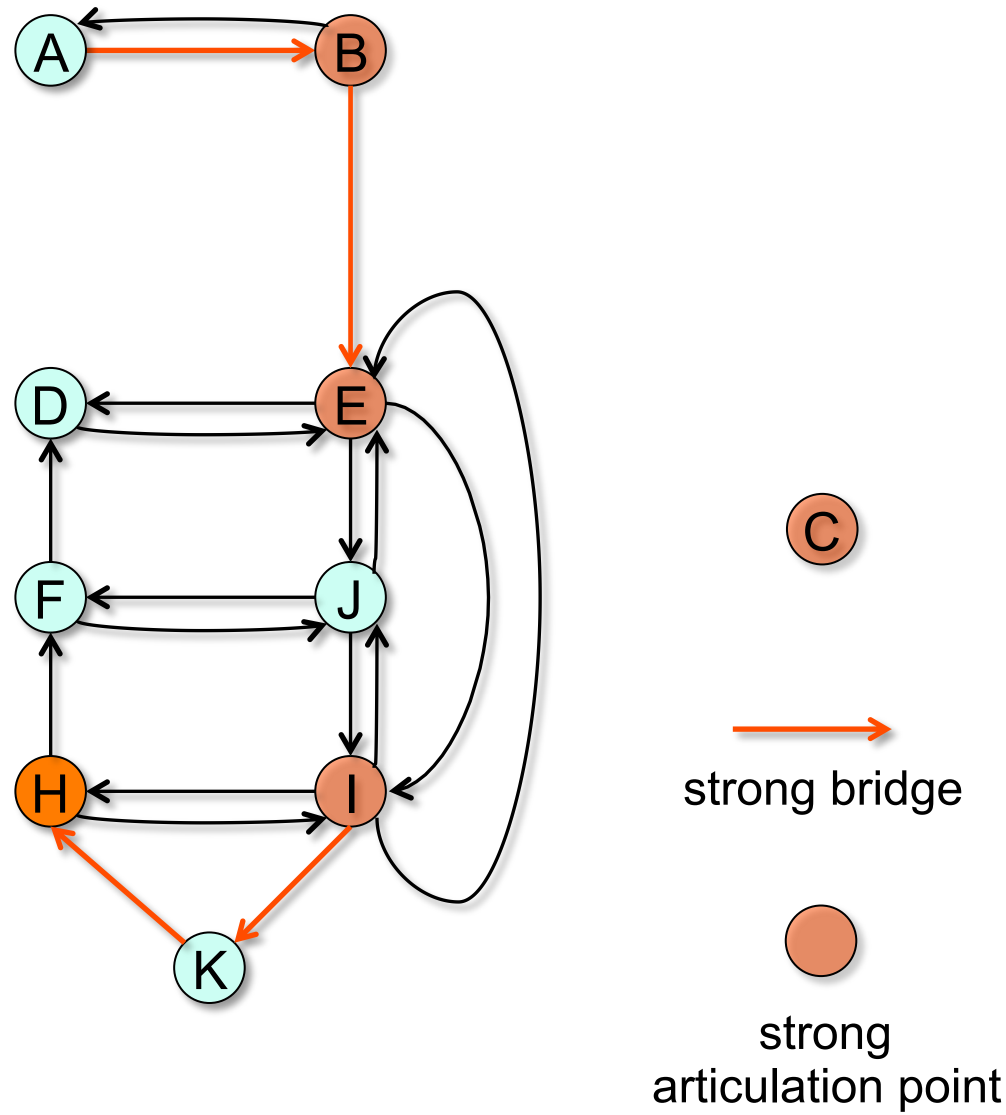
A vertex $v \in V$ is a **strong articulation point** if its removal, increases the number of **strongly** connected components of G.



Directed: 2-Vertex Connectivity

Let $G = (V,E)$ be a **directed strongly** connected graph, with m edges and n vertices.

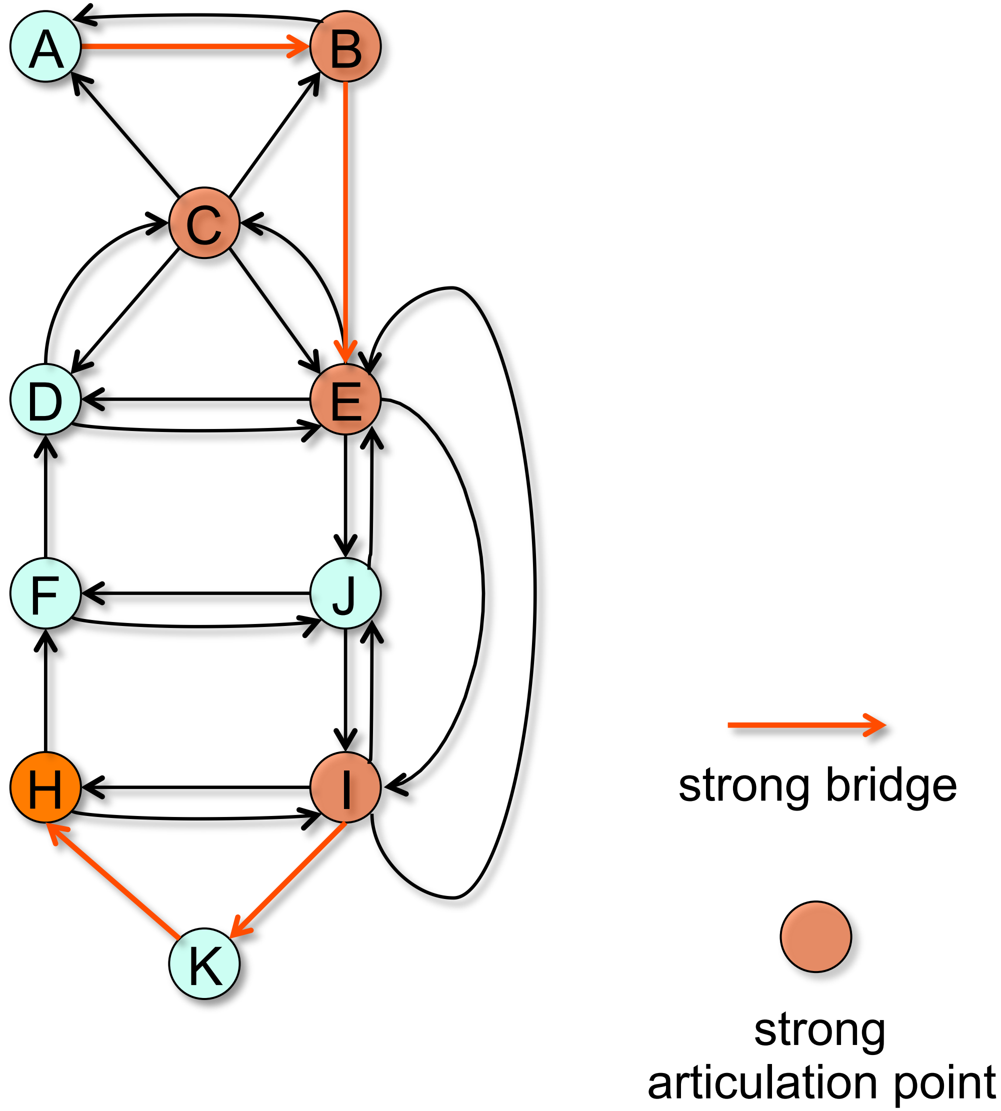
A vertex $v \in V$ is a **strong articulation point** if its removal, increases the number of **strongly** connected components of G.



Directed: 2-Vertex Connectivity

Let $G = (V,E)$ be a **directed strongly** connected graph, with m edges and n vertices.

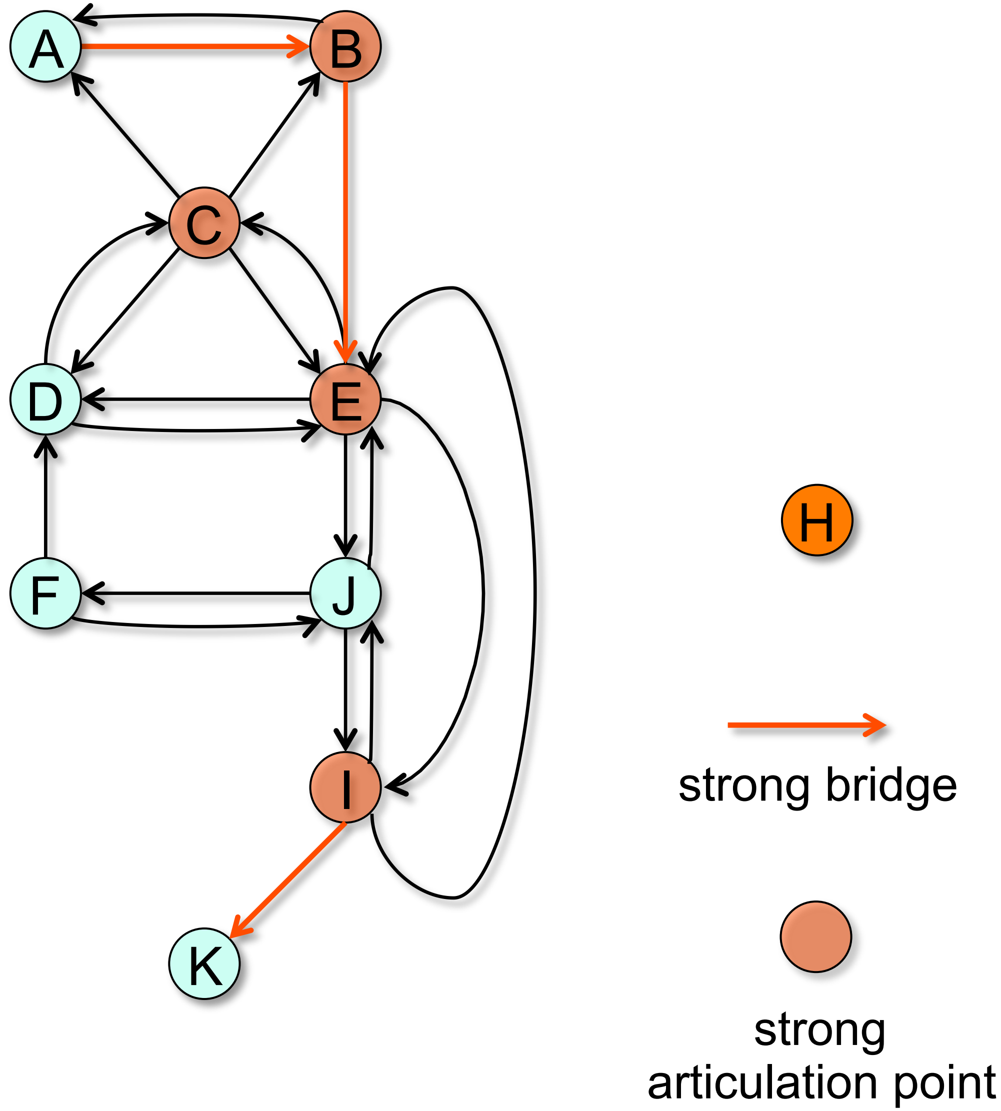
A vertex $v \in V$ is a **strong articulation point** if its removal, increases the number of **strongly** connected components of G.



Directed: 2-Vertex Connectivity

Let $G = (V,E)$ be a **directed strongly** connected graph, with m edges and n vertices.

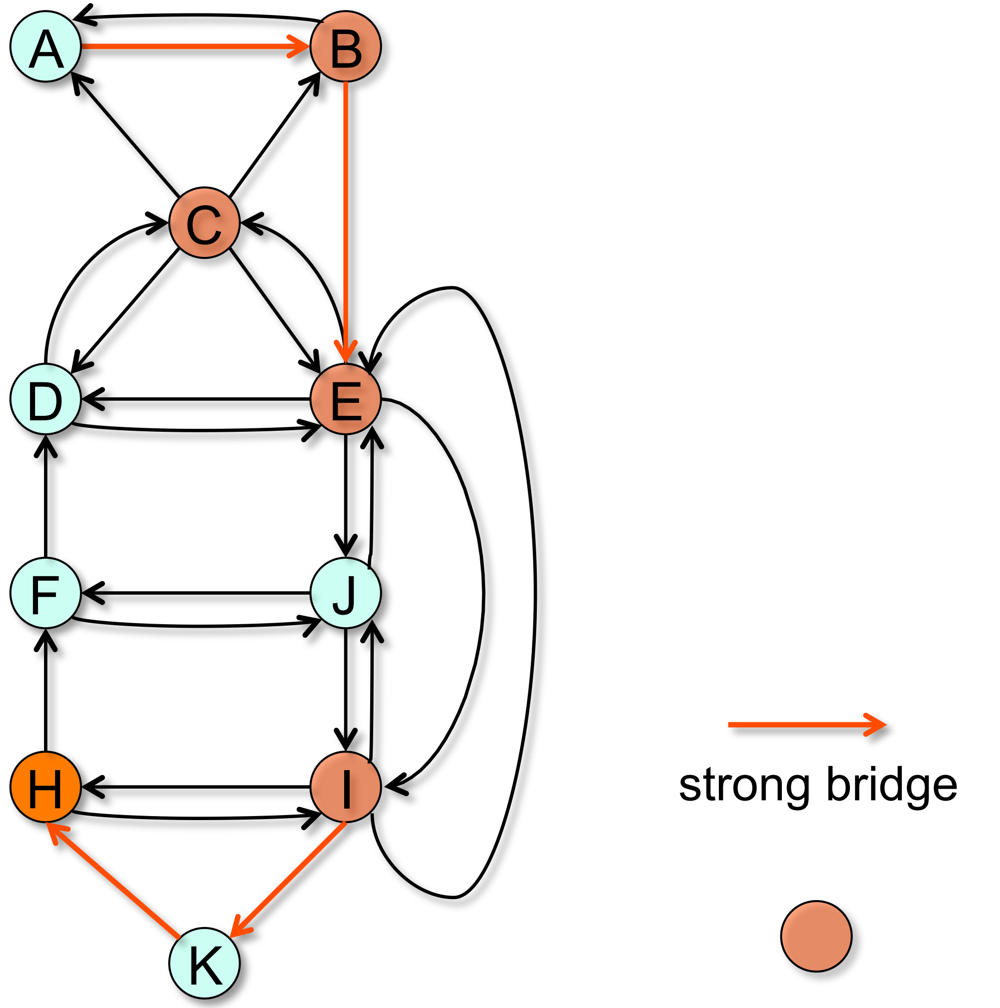
A vertex $v \in V$ is a **strong articulation point** if its removal, increases the number of **strongly** connected components of G.



Directed: 2-Vertex Connectivity

Let $G = (V,E)$ be a **directed strongly** connected graph, with m edges and n vertices.

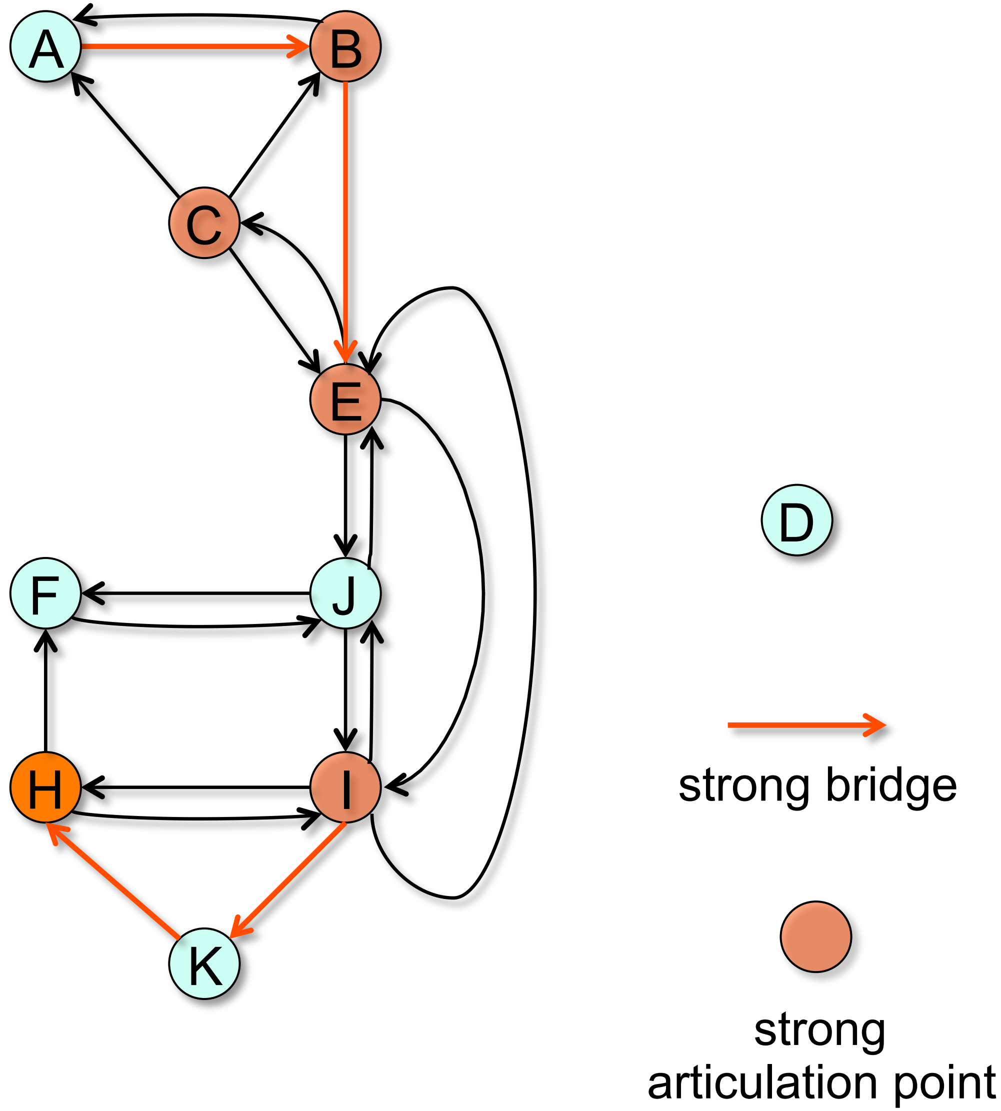
A vertex $v \in V$ is a **strong articulation point** if its removal, increases the number of **strongly** connected components of G.



Directed: 2-Vertex Connectivity

Let $G = (V,E)$ be a **directed strongly** connected graph, with m edges and n vertices.

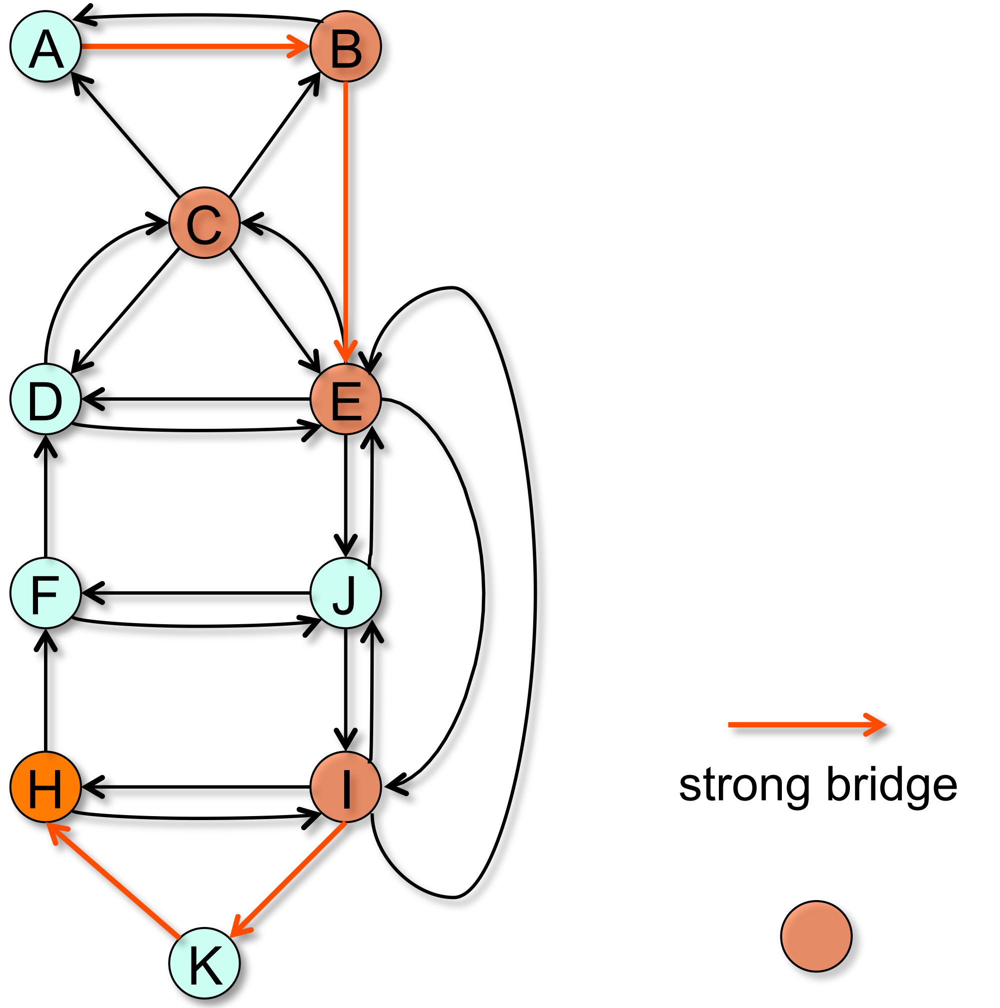
A vertex $v \in V$ is a **strong articulation point** if its removal, increases the number of **strongly** connected components of G.



Directed: 2-Vertex Connectivity

Let $G = (V,E)$ be a **directed strongly** connected graph, with m edges and n vertices.

A vertex $v \in V$ is a **strong articulation point** if its removal, increases the number of **strongly** connected components of G.



Directed: 2-Vertex Connectivity

Let $G = (V,E)$ be a **directed strongly** connected graph, with m edges and n vertices.

A vertex $v \in V$ is a **strong articulation point** if its removal, increases the number of **strongly** connected components of G.

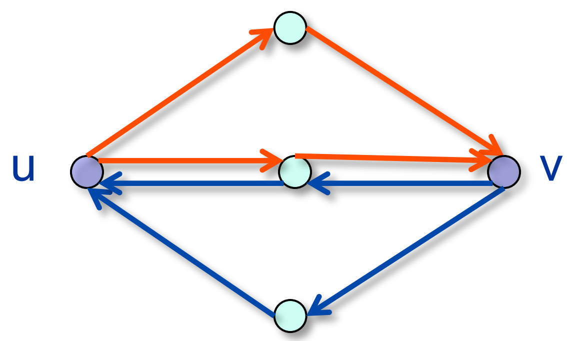
Graph G is **2-vertex-connected** if it has at least 3 vertices (don't allow for degenerate components) and no **strong** articulation points.

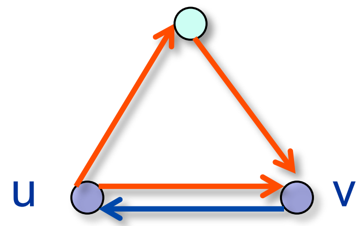
The **2-vertex-connected components** of G are its maximal 2-vertex-connected subgraphs.

P2P 2-Vertex Connectivity

Vertices u and v are **2-vertex-connected** if there are two (internally) vertex-disjoint paths **from** u **to** v **and two (internally) vertex-disjoint paths from v to u**.

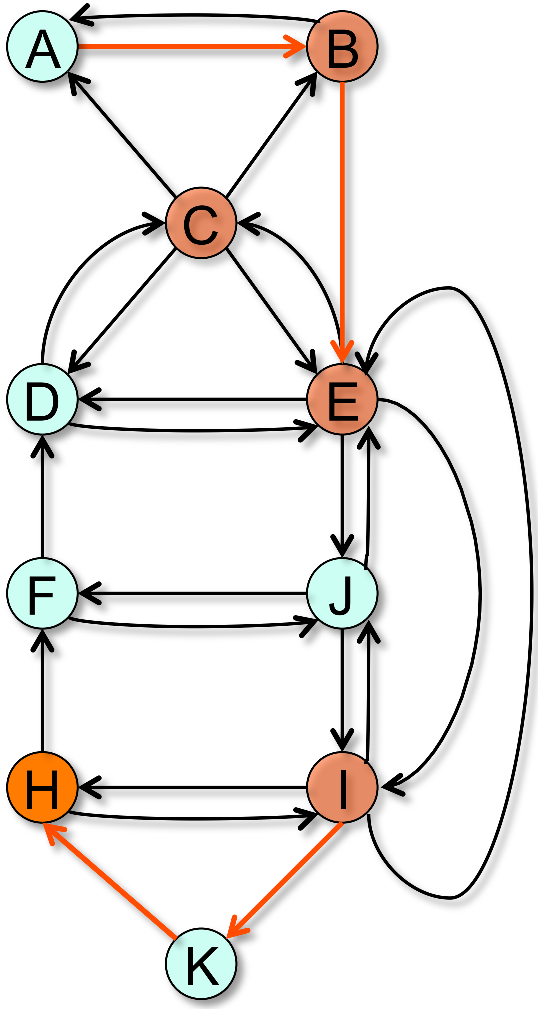
By **Menger's Theorem, if** vertices u and v are **2-vertex-connected** then the removal of any vertex ($\neq u, v$) leaves them in same **strongly** connected component.





Can define a **2-vertex-connected block** of G as a maximal subset B ⊆ V s. t. u and v are **2-vertex-connected** for all u, v ∈ B.

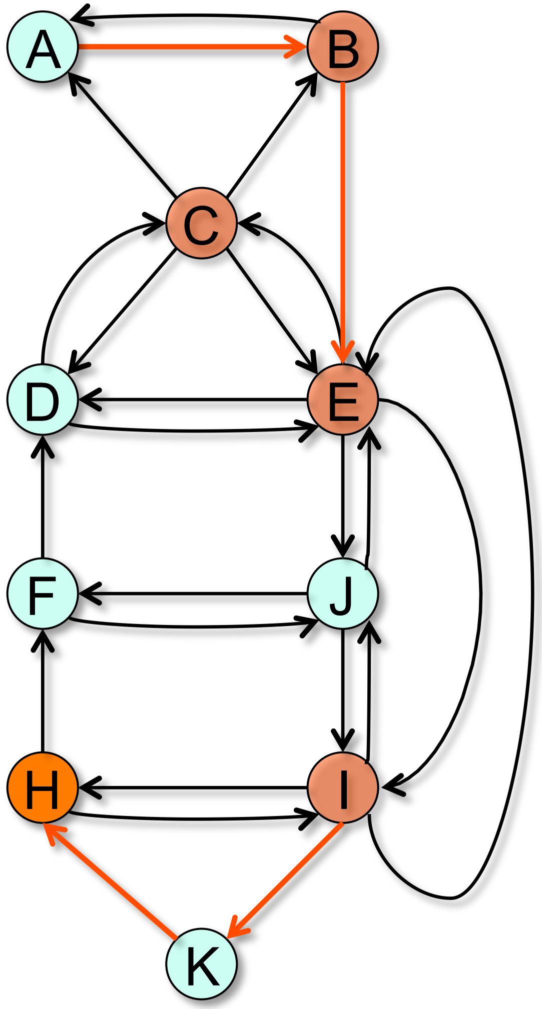
2-Vertex-Connected Comp's ≠ Blocks



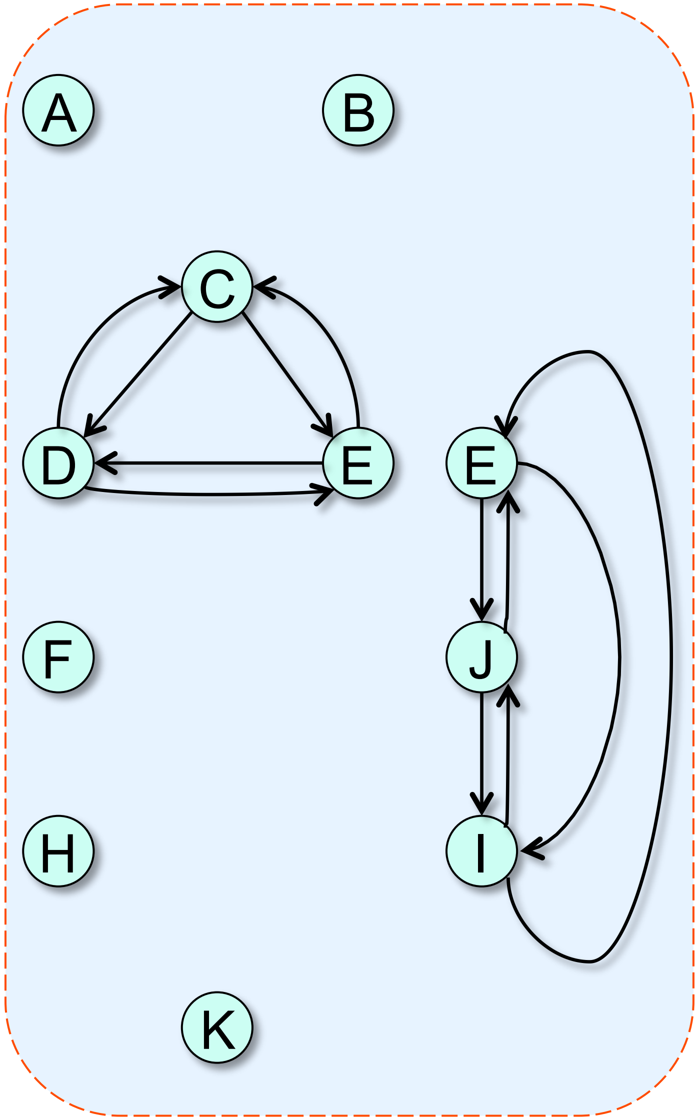
strong bridge

strong
articulation point

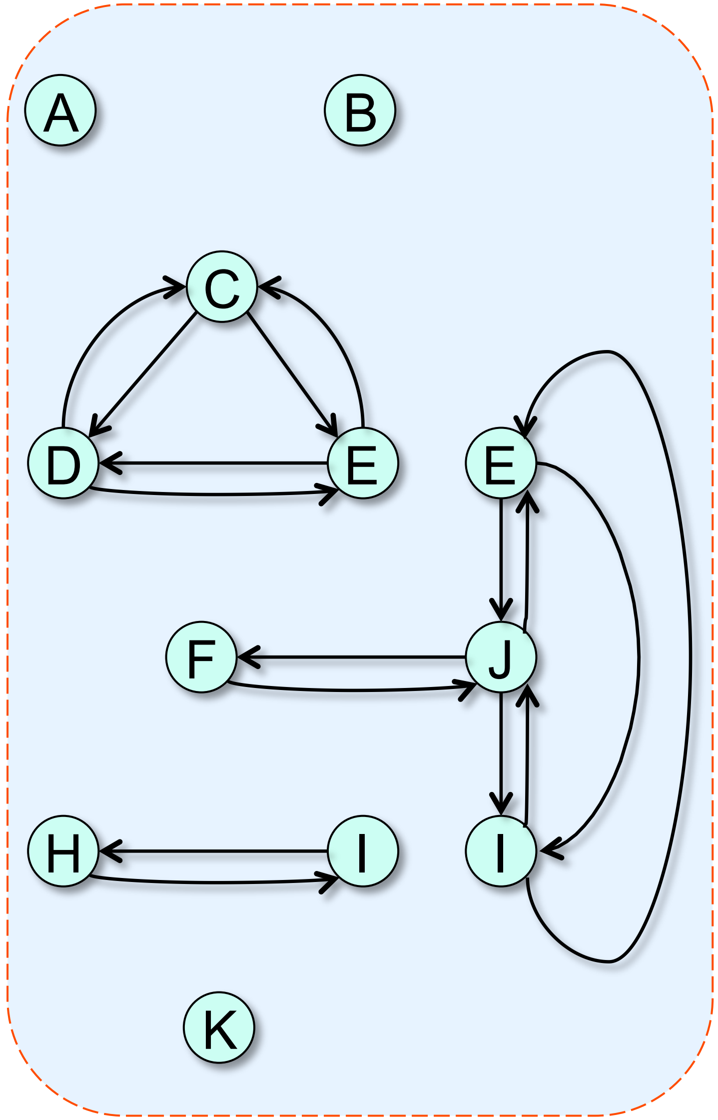
2-Vertex-Connected Comp's ≠ Blocks



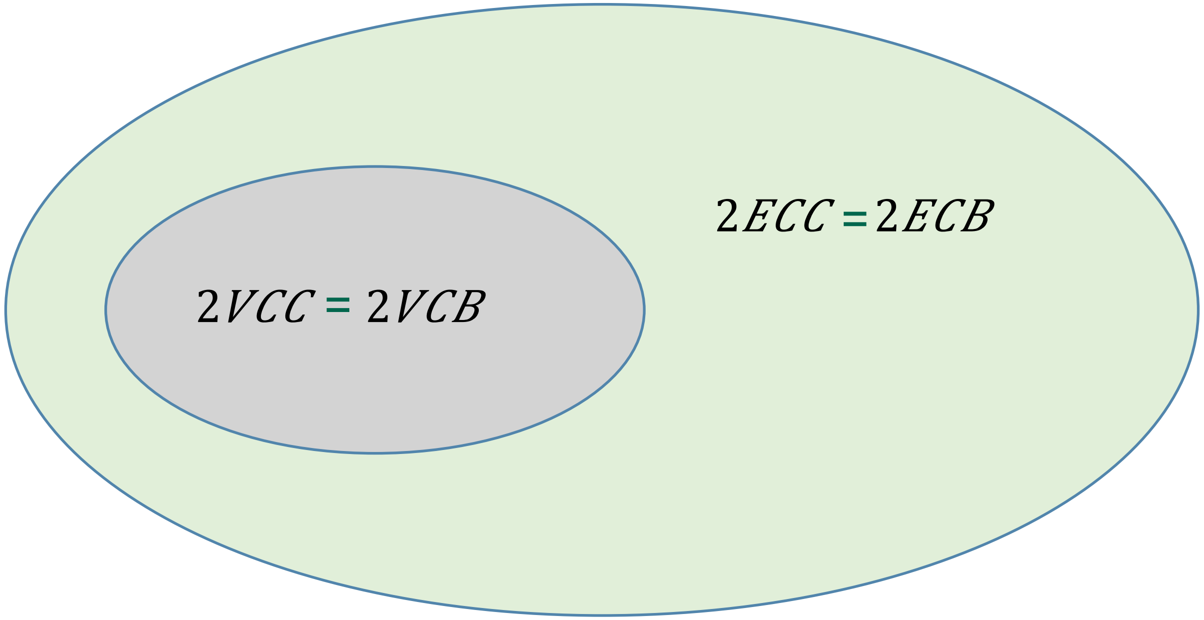
2-vertex-connected components



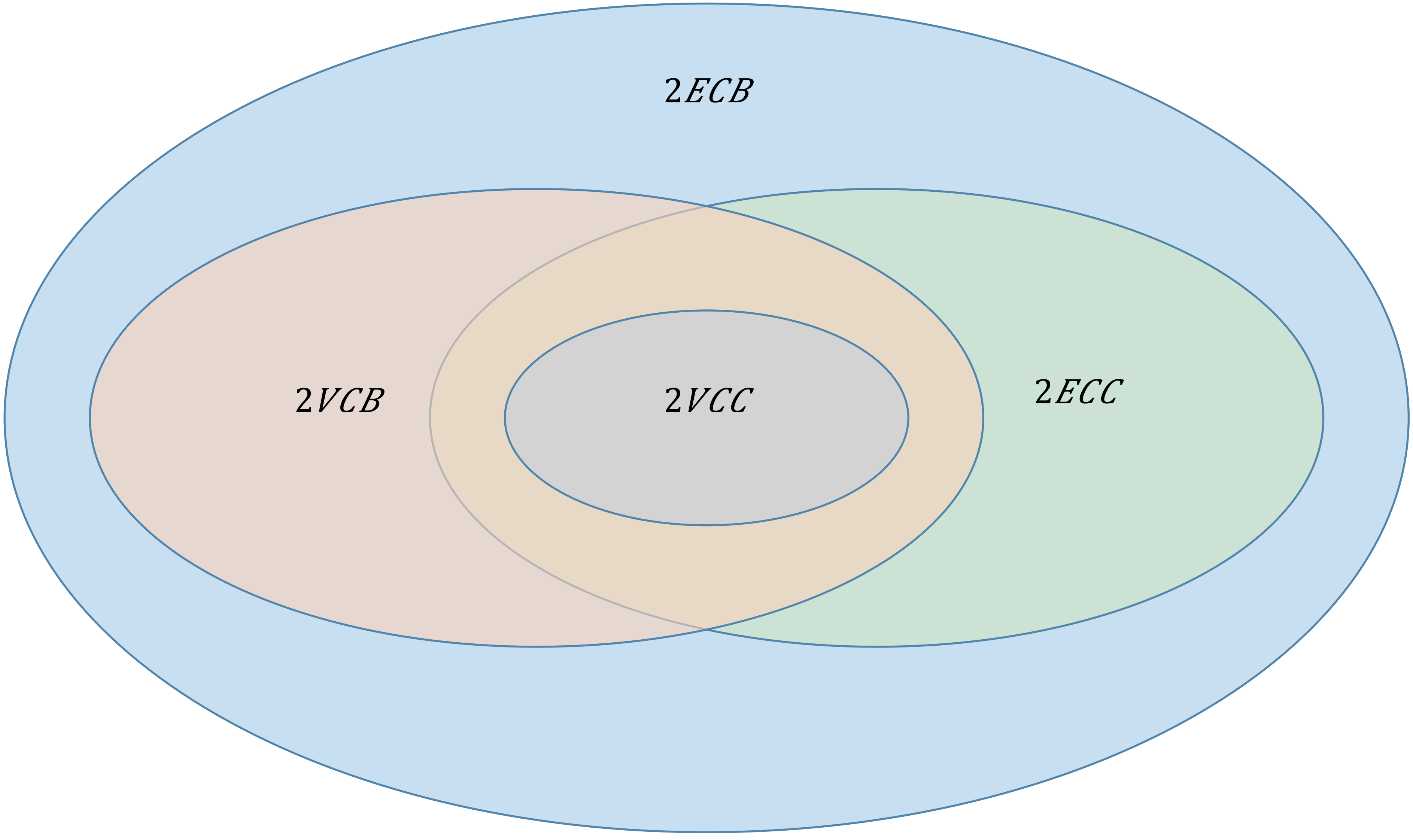
2-vertex-connected blocks



Big Picture (Undirected)



Big Picture (Directed)



Bounds for Directed *G*

Q1: Find whether *G* is 2-vertex-connected (2-edge-connected). I.e., find **one** connectivity cut (if any)

O(m+n)

[Tarjan 76] + [Gabow & Tarjan 83]

[Georgiadis 10]

Q2: Find **all** 2-connectivity cuts (articulation points, bridges) in *G*

O(m+n)

[Italiano et al 10]

Q3: Find the **2-connectivity** (2-vertex-, 2-edge- connected) **blocks** of *G*

O(m+n)

[Georgiadis et al 15]

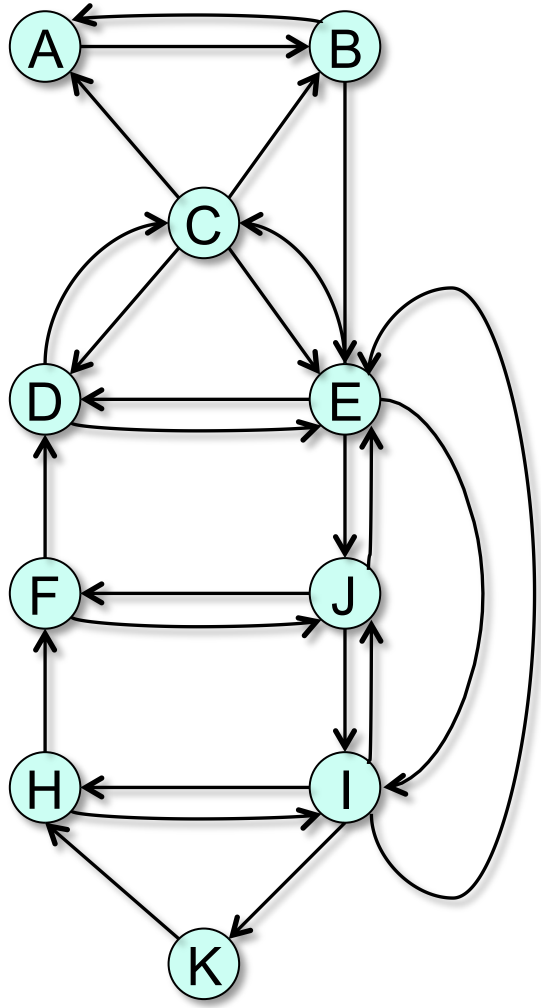
Q4: Find the **2-connectivity** (2-vertex-, 2-edge-connected) **components** of *G*

O(mn)

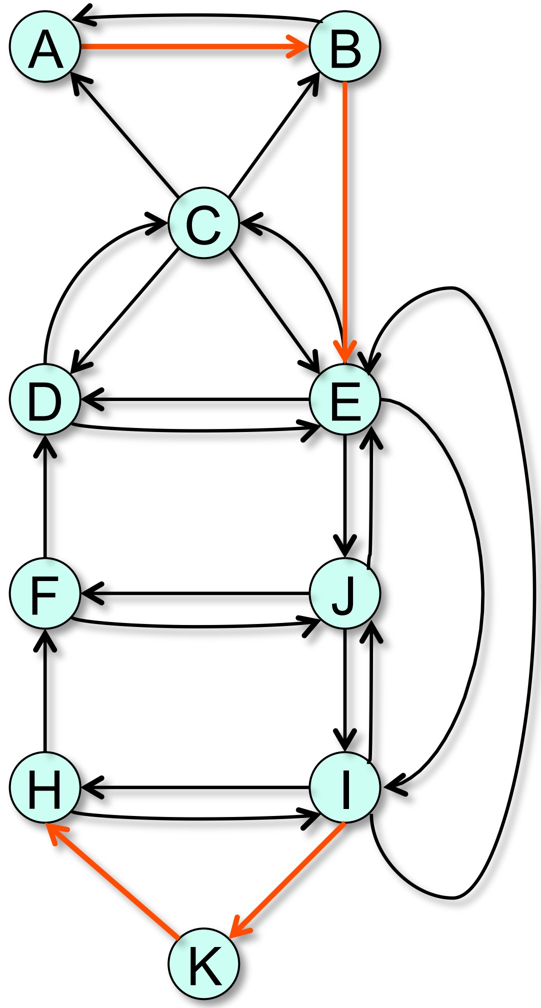
[Jaberi 14]

Can we do better?

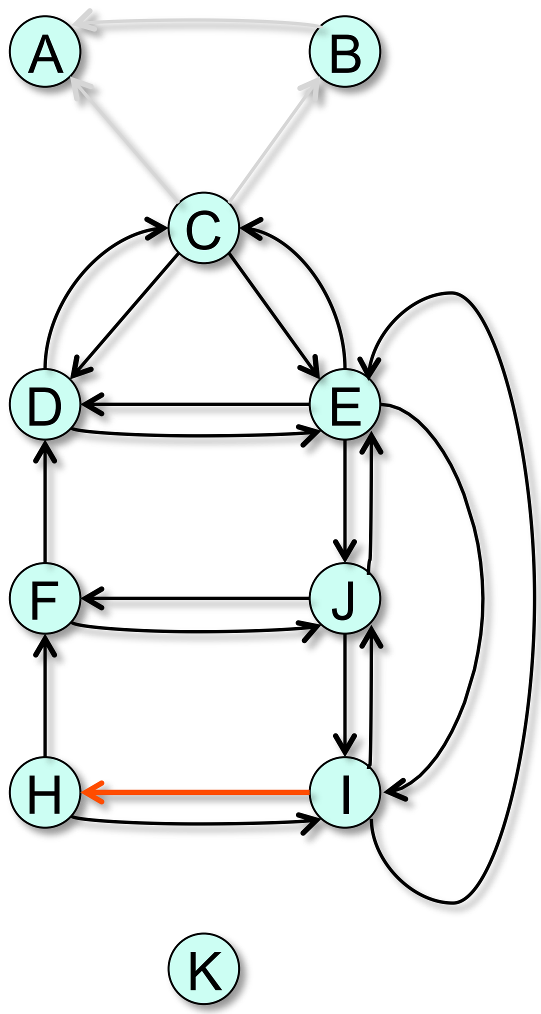
2-Edge Connected Components



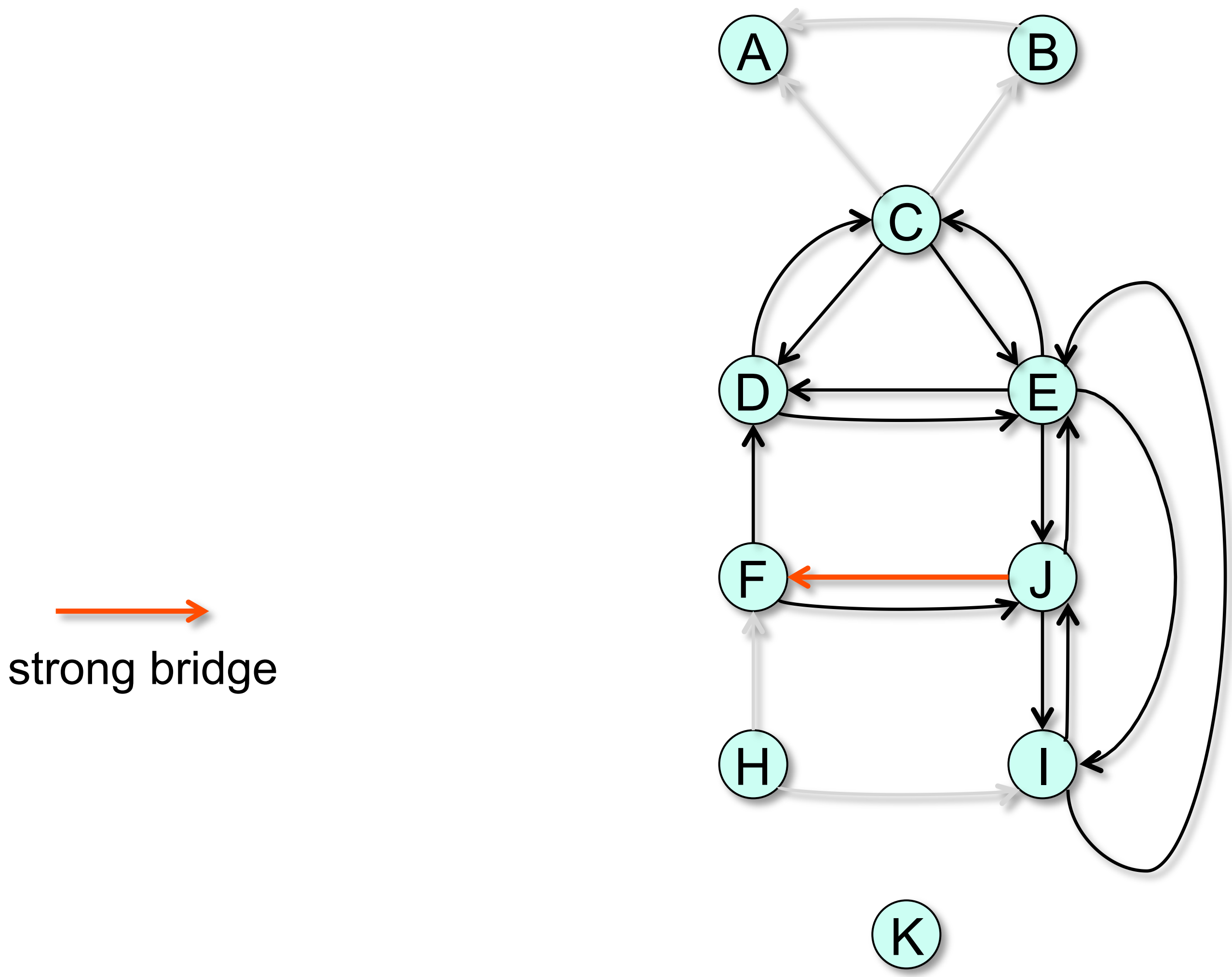
2-Edge Connected Components



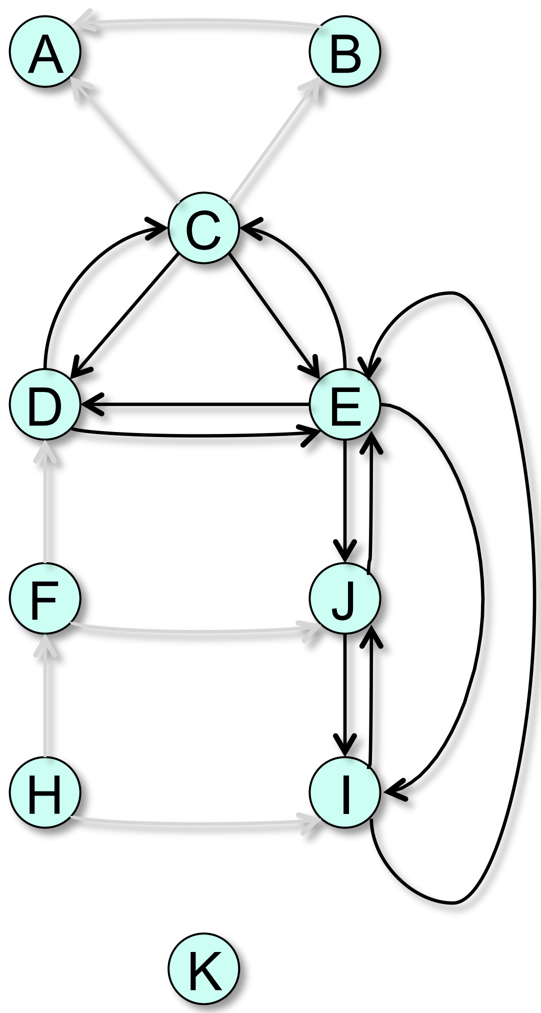
2-Edge Connected Components



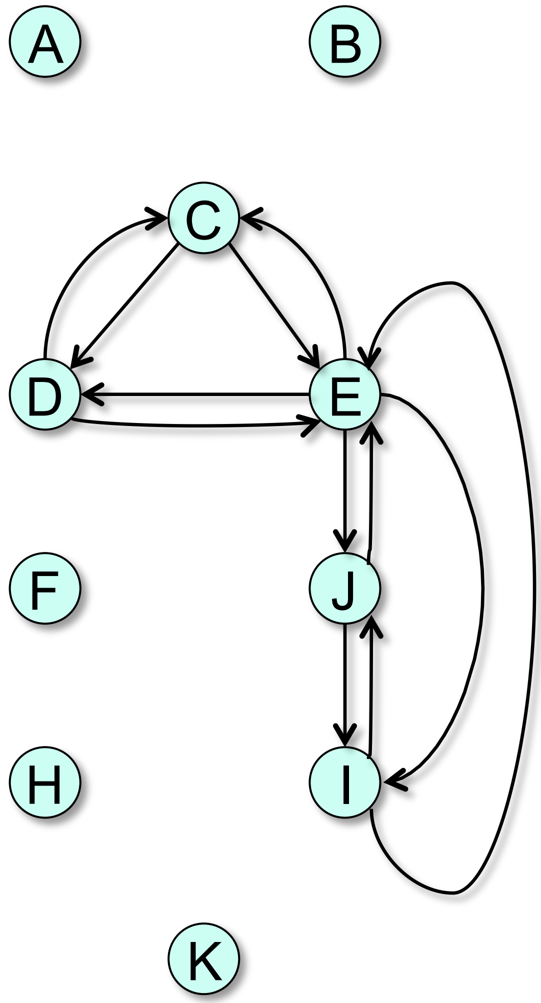
2-Edge Connected Components



2-Edge Connected Components



2-Edge Connected Components



Why should we care?

Theoretically interesting problem

It also has several intriguing applications

Bounds for Directed *G*

Q1: Find whether *G* is 2-vertex-connected (2-edge-connected).
I.e., find **one** connectivity cut (if any)

O(m+n)

[Tarjan 76] + [Gabow & Tarjan 83]

[Georgiadis 10]

Q2: Find **all** 2-connectivity cuts (articulation points, bridges) in *G*

O(m+n)

[Italiano et al 10]

Q3: Find the **2-connectivity** (2-vertex-, 2-edge- connected) **blocks** of *G*

O(m+n)

[Georgiadis et al 15]

Q4: Find the **2-connectivity** (2-vertex-, 2-edge-connected) **components** of *G*

O(mn)

[Jaberi 14]

Today's Outline

1. 2-Connectivity on directed graphs
2. **Algorithms for strong articulation points and strong bridges**
3. Experiments
4. Open Problems

Naive Algorithms

Check whether vertex v is strong articulation point in G :

Compute strongly connected components of $G/\{v\}$

$O(n(m+n))$ for computing all strong articulation points

Check whether edge e is strong bridge in G :

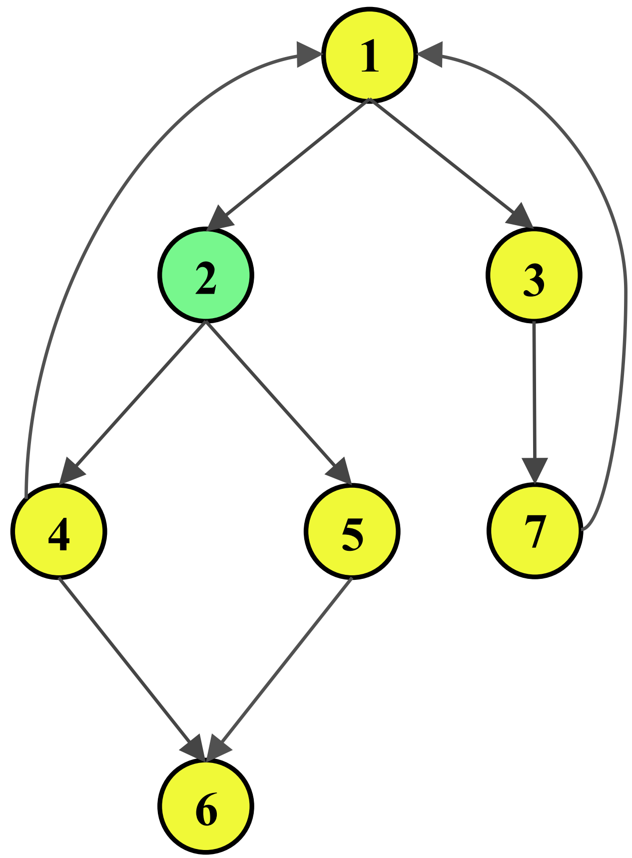
Compute strongly connected components of $G/\{e\}$

$O(m(m+n))$ for computing all strong bridges

Not difficult to get $O(n(m+n))$ algorithm

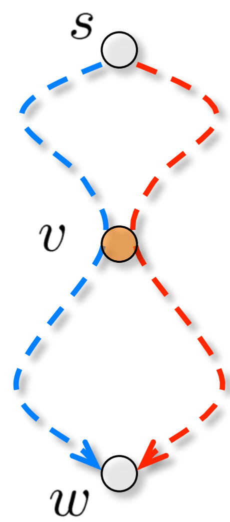
Flow graphs and Dominators

A *flow graph $G(s) = (V,E,s)$* is a directed graph with a start vertex s in V such that every vertex in V reachable from s



Flow graphs and Dominators

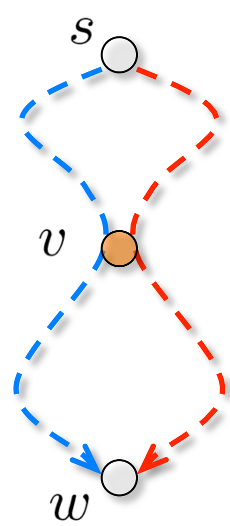
A *flow graph G(s) = (V,E,s)* is a directed graph with a start vertex s in V such that every vertex in V reachable from s



Given a flow graph *G(s)=(V,E,s)*, can define a *dominance relation*: vertex *v* *dominates* vertex *w* if every path from *s* to *w* includes *v*

Flow graphs and Dominators

A *flow graph G(s) = (V,E,s)* is a directed graph with a start vertex s in V such that every vertex in V reachable from s



Given a flow graph *G(s)=(V,E,s)*, can define a *dominance relation*: vertex *v* *dominates* vertex *w* if every path from *s* to *w* includes *v*

Let *dom(w)* be set of vertices that dominate *w*. For any $w \neq s$ we have that $\{s,w\} \subseteq dom(w)$: *s* and *w* are the *trivial dominators* of *w*

Dominator Trees

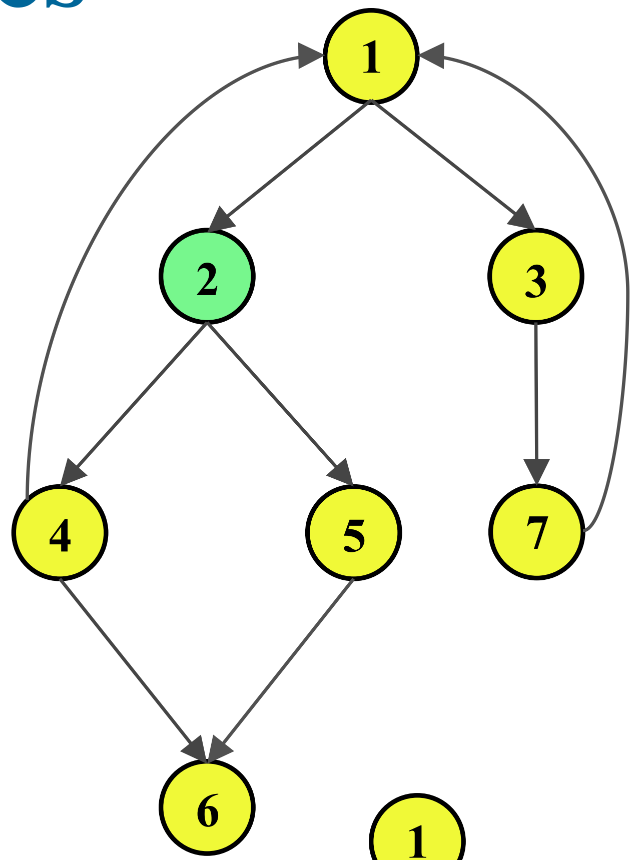
Dominance relation is transitive and its transitive reduction is referred to as the *dominator tree DT(s)*.

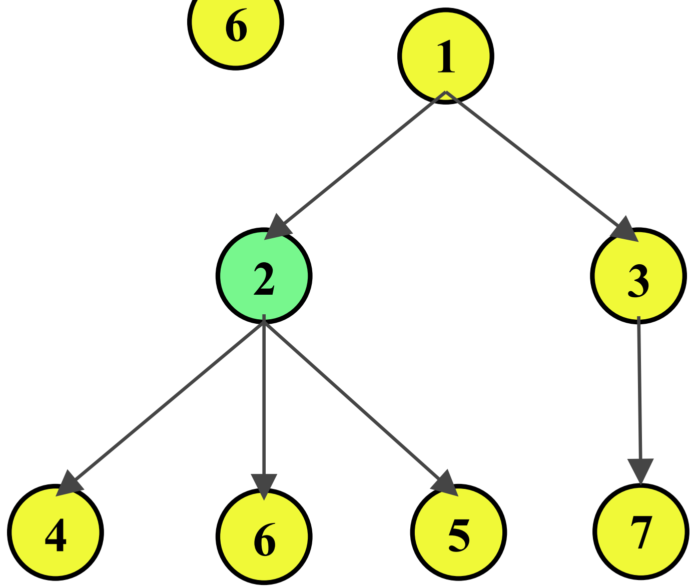
DT(s) rooted at *s*.

v dominates *w* if and only if *v* is ancestor of *w* in dominator tree *DT(s)*.

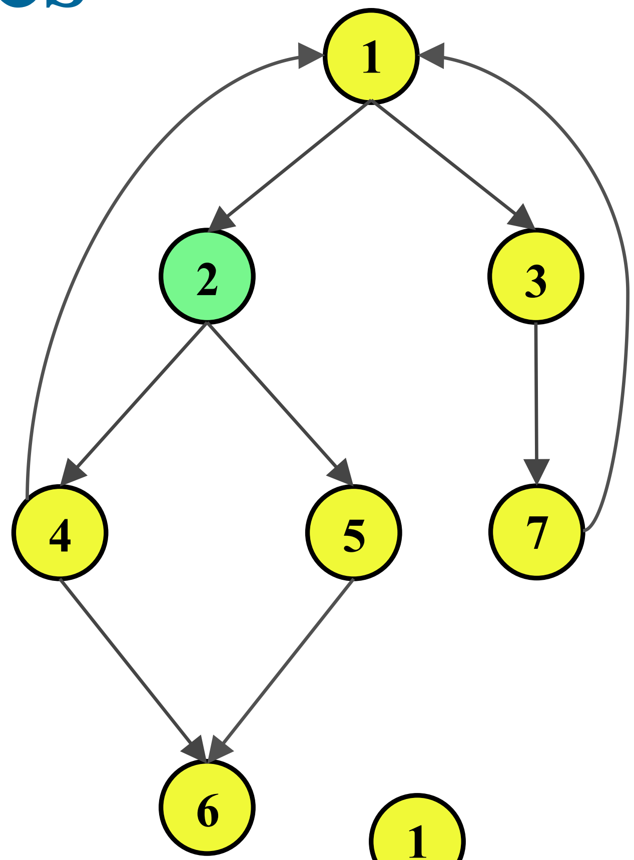
If *v* dominates *w*, and every other non-trivial dominator of *v* also dominates *w*, *v* is an *immediate dominator* of *w*.

If *v* has any non-trivial dominators, then *v* has a unique immediate dominator: the immediate dominator of *v* is the parent of *v* in the dominator tree *DT(s)*.





Dominator Trees

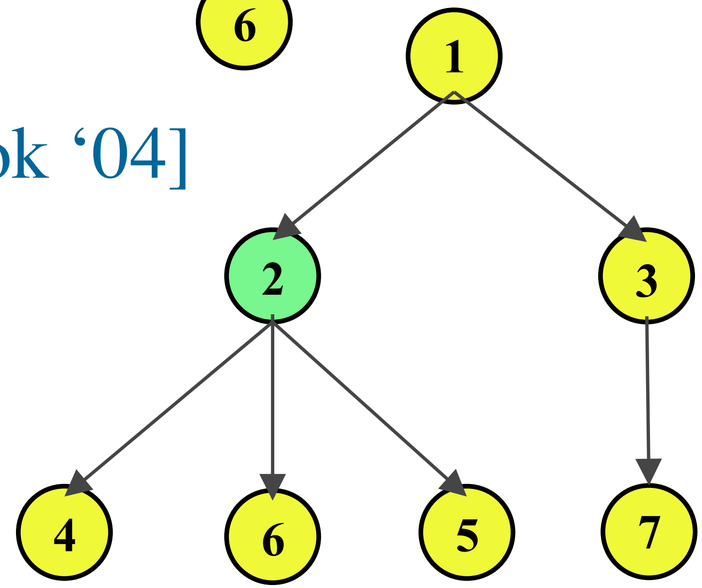


Dominators (and dominator tree) can be computed in $O(m+n)$ time:

[Alstrup, Harel, Lauridsen &Thorup '97]

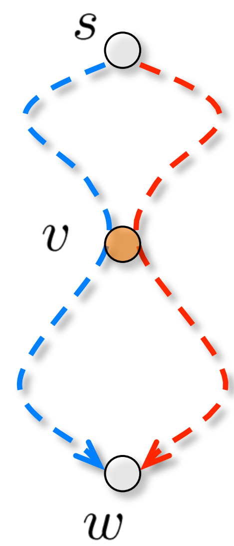
[Buchsbaum, Kaplan, Rogers & Westbrook '04]

[Georgiadis & Tarjan '04]



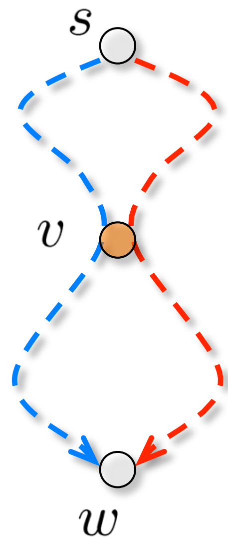
Vertex Dominators and SAP

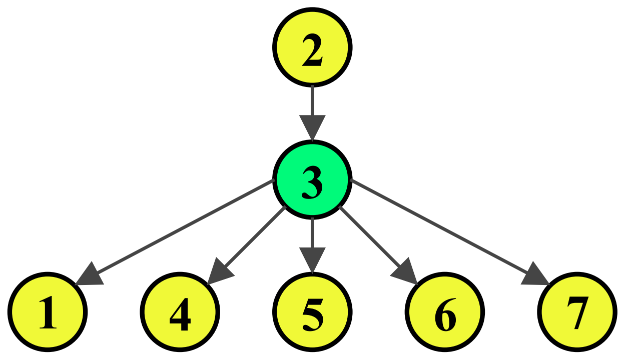
Lemma 1 *Let $G = (V,E)$ be a strongly connected graph, and let s be any vertex in G. Let $G(s) = (V,E,s)$ be the flow graph with start vertex s. If v is a non-trivial dominator of a vertex w in $G(s)$, then v is a strong articulation point in G.*



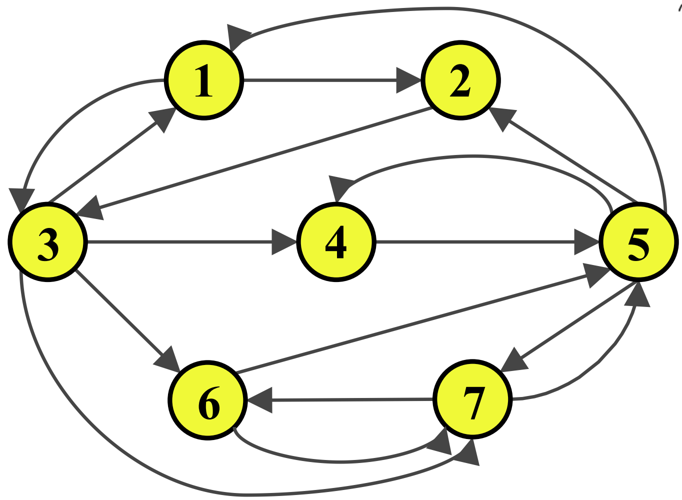
Vertex Dominators and SAP

Lemma 1 *Let G = (V,E) be a strongly connected graph, and let s be any vertex in G. Let G(s) = (V,E,s) be the flow graph with start vertex s. If v is a non-trivial dominator of a vertex w in G(s), then v is a strong articulation point in G.*





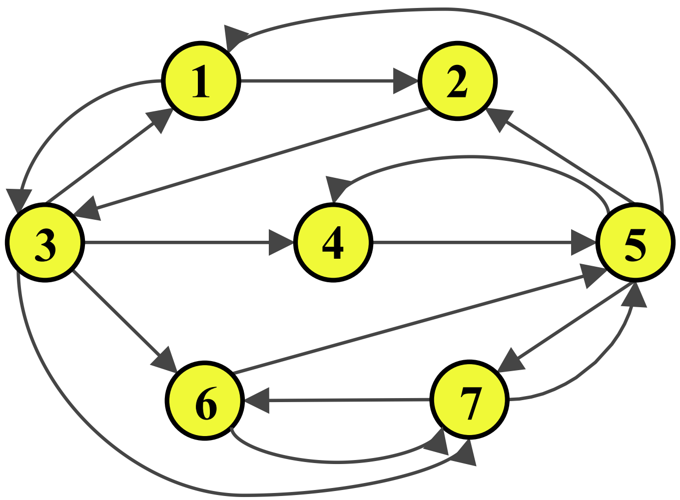
Vertex 3 is a non-trivial dominator in G(2)

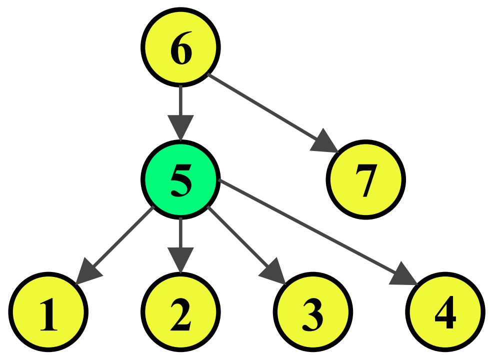


Vertex 3 is strong articulation point in G

Vertex Dominators and SAP

Lemma 2 *Let G = (V,E) be a strongly connected graph. If v is a strong articulation point in G, then there must be a vertex s ∈ V such that v is a non-trivial dominator of a vertex w in the flow graph G(s) = (V,E,s).*





Vertex 5 is strong articulation point in G

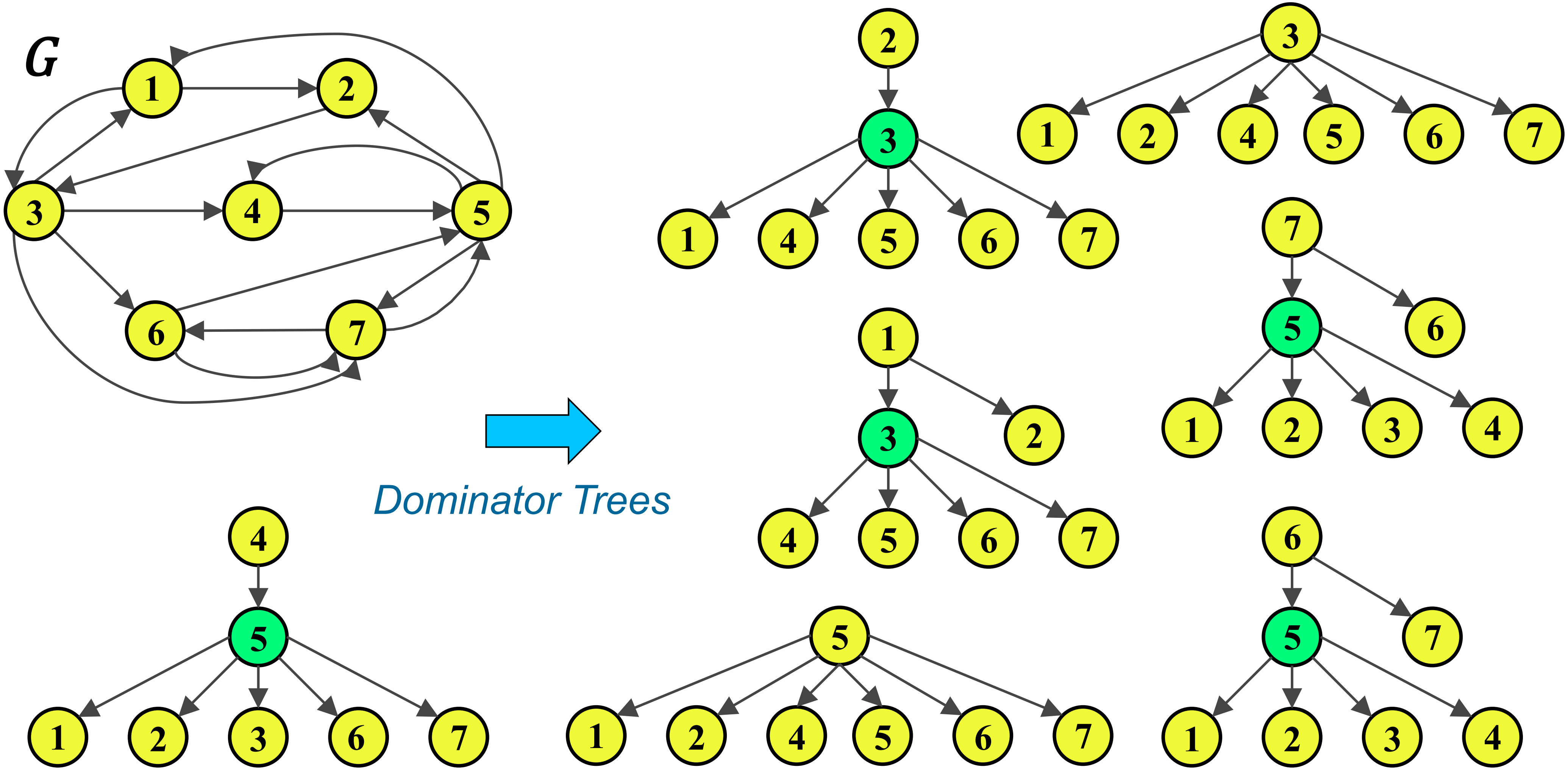
Vertex 5 must be non-trivial dominator in some G(s). Here s=6.

Still Not Efficient

Corollary *Let $G = (V,E)$ be a strongly connected graph. Vertex u is a strong articulation point in G if and only there is a vertex $s \in V$ such that u is a non-trivial dominator of a vertex v in the flow graph $G(s) = (V,E,s)$.*

Must compute dominator trees for all flow graphs G(v), for each vertex v in V, and output all non-trivial dominators found.

Dominator Trees



Still Not Efficient

Corollary *Let $G = (V,E)$ be a strongly connected graph. Vertex u is a strong articulation point in G if and only then there is a vertex $s \in V$ such that u is a non-trivial dominator of a vertex v in the flow graph $G(s) = (V,E,s)$.*

Must compute dominator trees for all flow graphs G(v), for each vertex v in V, and output all non-trivial dominators found.

Takes $O(n(m+n))$ time

Like trivial algorithm

Only more complicated...

Reversal Graph

Reversal Graph $G^R = (V,E^R)$: reverse all edges in G.
If (u,v) in G then (v,u) in G^R.

Observation. *Let $G = (V,E)$ be a strongly connected graph and $G^R = (V,E^R)$ be its reversal graph. Then G^R is strongly connected. Furthermore, vertex v is a strong articulation point in G if and only if v is a strong articulation point in G^R.*

Exploit Dominators

Given a strongly connected graph $G=(V,E)$, let

- $G(s) = (V,E,s)$ be the flow graph with start vertex s
- $D(s)$ the set of non-trivial dominators in $G(s)$
- $G^R(s) = (V,E^R,s)$ be the flow graph with start vertex s
- $D^R(s)$ the set of non-trivial dominators in $G^R(s)$

Theorem. *Let $G = (V,E)$ be a strongly connected graph, and let $s \in V$ be any vertex in G. Then vertex $v \neq s$ is a strong articulation point in G if and only if $v \in D(s) \cup D^R(s)$.*

Strong Articulation Points

Theorem. *Let $G = (V,E)$ be a strongly connected graph, and let $s \in V$ be any vertex in G. Then vertex $v \neq s$ is a strong articulation point in G if and only if $v \in D(s) \cup D^R(s)$.*

Proof:

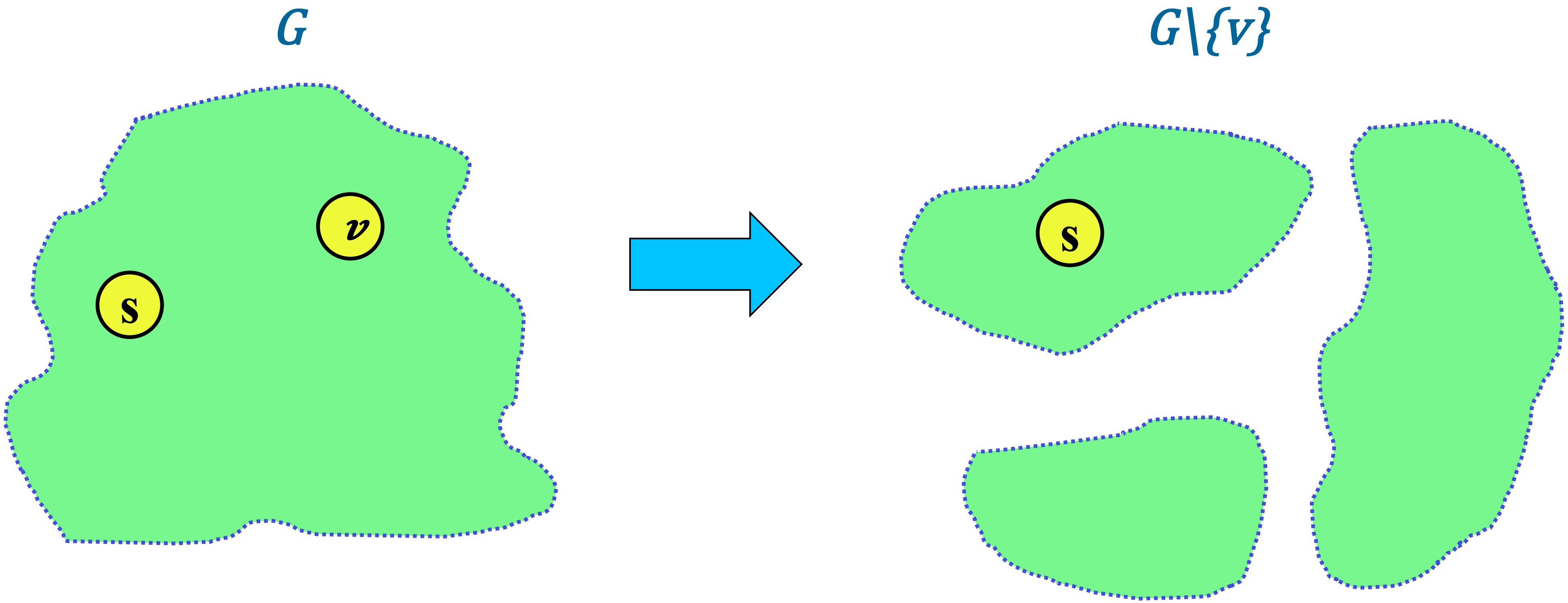
If $v \in D(s) \cup D^R(s)$ we know from previous lemmas that v must be an articulation point.

So, we need to prove only one direction.

Strong Articulation Points

Theorem. *Let $G = (V,E)$ be a strongly connected graph, and let $s \in V$ be any vertex in G. Then vertex $v \neq s$ is a strong articulation point in G if and only if $v \in D(s) \cup D^R(s)$.*

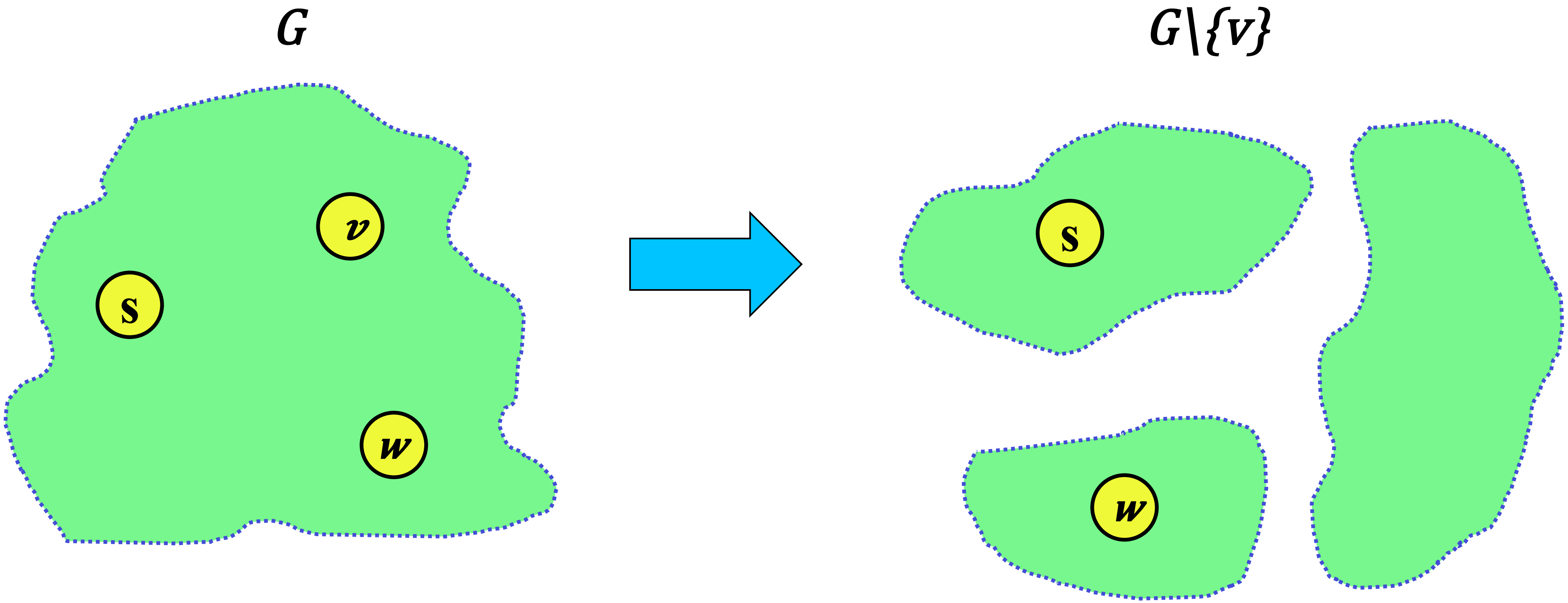
Proof**:** Let v be a strong articulation point



Strong Articulation Points

Theorem. *Let $G = (V,E)$ be a strongly connected graph, and let $s \in V$ be any vertex in G. Then vertex $v \neq s$ is a strong articulation point in G if and only if $v \in D(s) \cup D^R(s)$.*

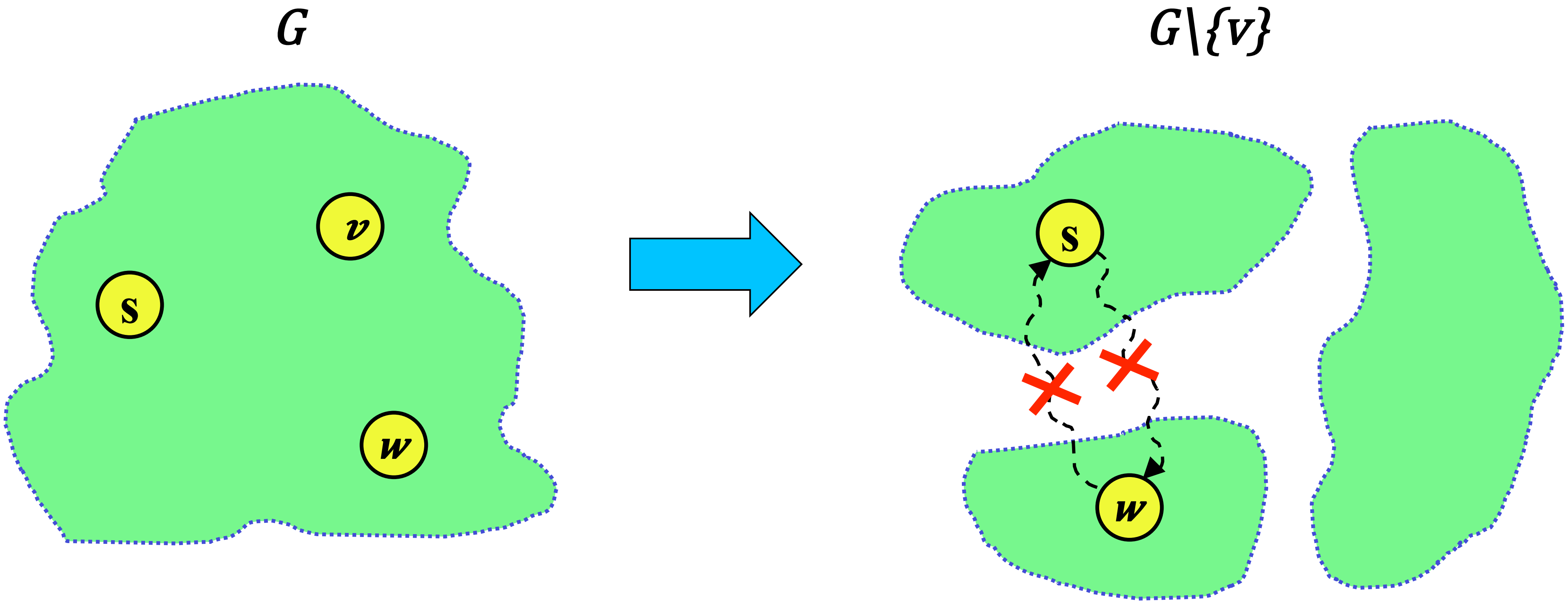
Proof: Let v be a strong articulation point



Strong Articulation Points

Theorem. *Let $G = (V,E)$ be a strongly connected graph, and let $s \in V$ be any vertex in G. Then vertex $v \neq s$ is a strong articulation point in G if and only if $v \in D(s) \cup D^R(s)$.*

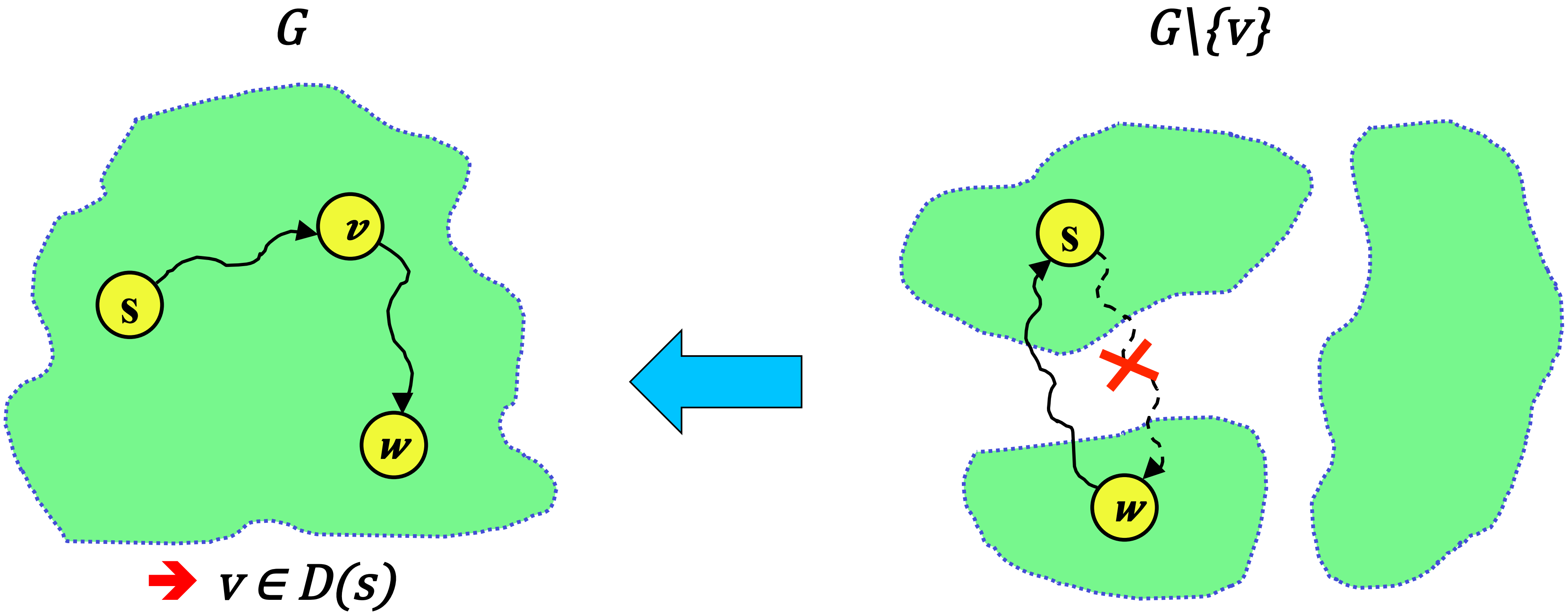
Proof: Let v be a strong articulation point



Strong Articulation Points

Theorem. *Let $G = (V,E)$ be a strongly connected graph, and let $s \in V$ be any vertex in G. Then vertex $v \neq s$ is a strong articulation point in G if and only if $v \in D(s) \cup D^R(s)$.*

Proof: Let v be a strong articulation point

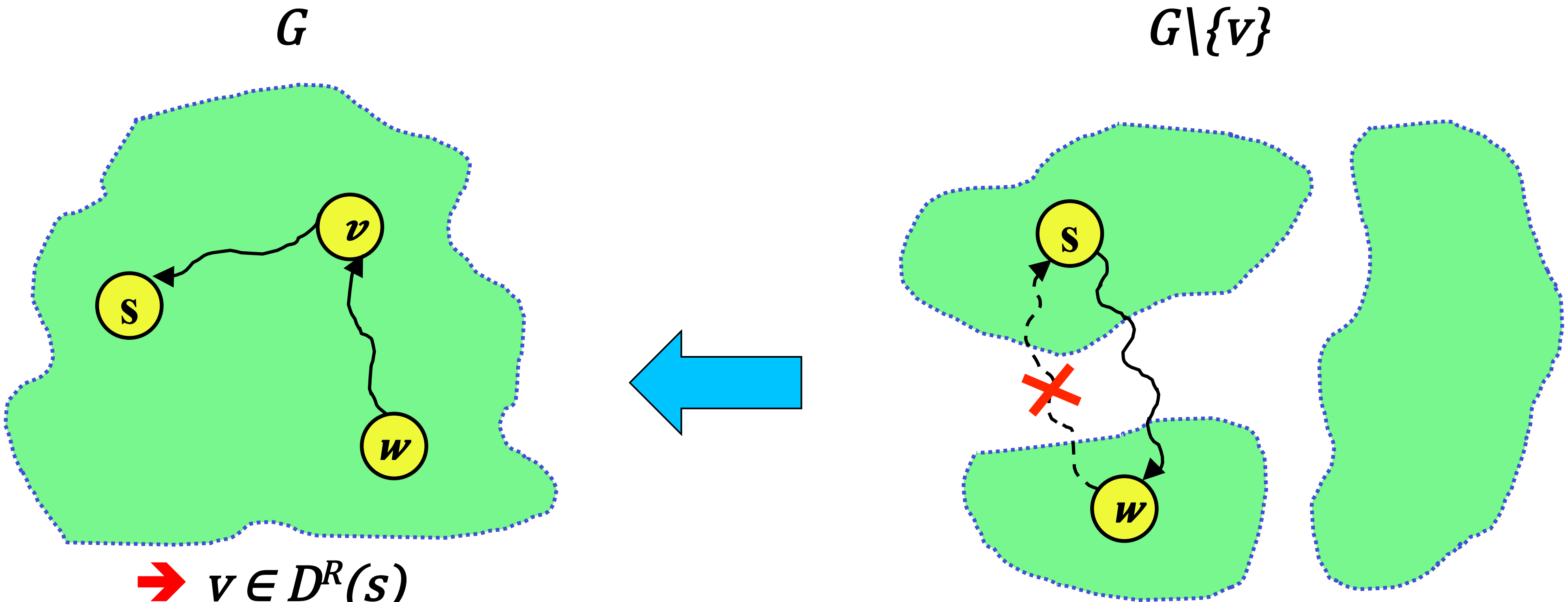


➔ $v \in D(s)$

Strong Articulation Points

Theorem. *Let $G = (V,E)$ be a strongly connected graph, and let $s \in V$ be any vertex in G. Then vertex $v \neq s$ is a strong articulation point in G if and only if $v \in D(s) \cup D^R(s)$.*

Proof: Let v be a strong articulation point



➔ $v \in D^R(s)$

Linear-Time Algorithm

Input: A strongly connected graph $G = (V, E)$, with n vertices and m edges.

Output: The strong articulation points of G.

1. Choose arbitrarily a vertex $s \in V$ in G, and test whether s is a strong articulation point in G. If s is a strong articulation point, then output s.

2. Compute and output $D(s)$, the set of non-trivial dominators in the flow graph $G(s) = (V,E,s)$.

3. Compute the reversal graph $G^R = (V,E^R)$.

4. Compute and output $D^R(s)$, the set of non-trivial dominators in the flow graph $G^R(s) = (V,E^R,s)$.

Total time is $O(m+n)$

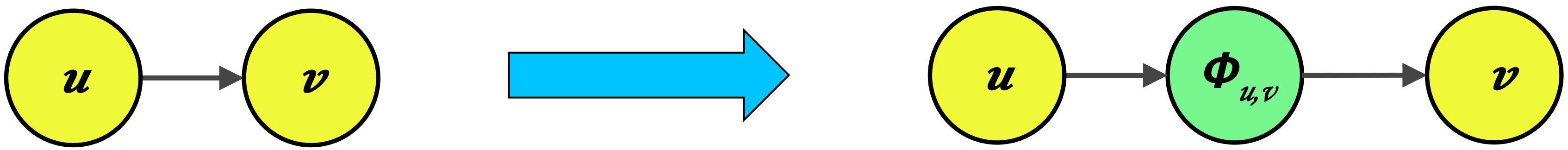
Strong Bridges

Strong Bridges

1. Reduction:

Lemma. *If there is an algorithm to compute the strong articulation points of a strongly connected graph in time $T(m,n)$, then there is algorithm to compute the strong bridges of a strongly connected graph in time $O(m + n + T(2m, n + m))$.*

"Proof" :



Mainly of theoretical interest (# vertices blows up)

Strong Bridges

2. Edge Dominators

Edge (u,v) *dominates* vertex w if every path from s to v contains edge (u,v)

If edge (u,v) dominates vertex w, and every other edge dominator of u dominates w, we say that (u,v) is an *immediate edge dominator* of vertex w.

If a vertex has an edge dominator, then it has a *unique* immediate edge dominator.

With some care, able to extend all the theory from (vertex) dominators to edge dominators.

Given a flow graph $G(s) = (V,E,s)$, edge dominators can be computed in time $O(m+n)$. But you need to re-implement code for dominators.

Edge Dominators in Practice

Lemma. [Tarjan 1974] *Let $G = (V,E,s)$ be a flow graph and let T be a DFS tree of G with start vertex s. Edge (v,w) is an edge dominator in flow graph G if and only if all of the following conditions are met:*
- (v,w) is a tree edge,
- w has no entering forward edge or cross edge, and
- there is no back edge (x,w) such that w does not dominate x.

Need to (1) compute dominator tree $DT(s)$ and (2) check whether w ancestor of x in $DT(s)$ for back edge (x,w).

Given a flow graph $G(s) = (V,E,s)$, edge dominators can be computed in time $O(m+n)$. Reuse code for (vertex) dominators. More efficient in practice. But still slightly slower than (vertex) dominators.

Computing All Strong Bridges

Given a strongly connected graph $G=(V,E)$, let

- $G(s) = (V,E,s)$ be the flow graph with start vertex s
- $ED(s)$ the set of *edge dominators* in $G(s)$
- $G^R(s) = (V,E^R,s)$ be the flow graph with start vertex s
- $ED^R(s)$ the set of *edge dominators* in $G^R(s)$

Theorem. *Let $G = (V,E)$ be a strongly connected graph, and let $s \in V$ be any vertex in G. Then edge (u,v) is a strong bridge in G if and only if $(u,v) \in ED(s) \cup ED^R(s)$.*

Incidentally, this proves also that can be at most $2n-2$ strong bridges in a directed graph.

Today's Outline

1. 2-Connectivity on directed graphs
2. Algorithms for strong articulation points and strong bridges
3. **Experiments (very rough, still ongoing)**
4. Open Problems

2-Connectivity

Can 2-connectivity be useful to understand the (macroscopic) structure of social networks / web graphs / other networks?

Do social networks / web graphs / other networks have different 2-connectivity properties?

Nodes / links which act more as "information" gateways (tweets / diseases / etc…) in the network?

2-Vertex Cores

Delete recursively all the strong articulation points of a directed graph *G*, as follows

While there are strong articulation points in G do:

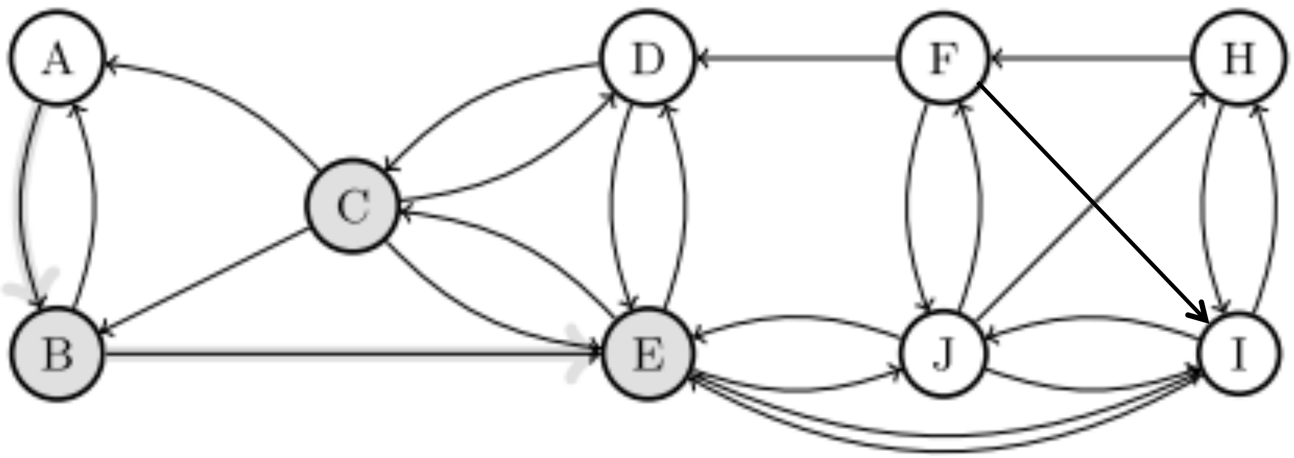
1. Let G' be the graph defined by all the s.c.c.'s of G;
2. Set G to be the graph obtained by deleting all the strong articulation points in G' together with their incident edges.

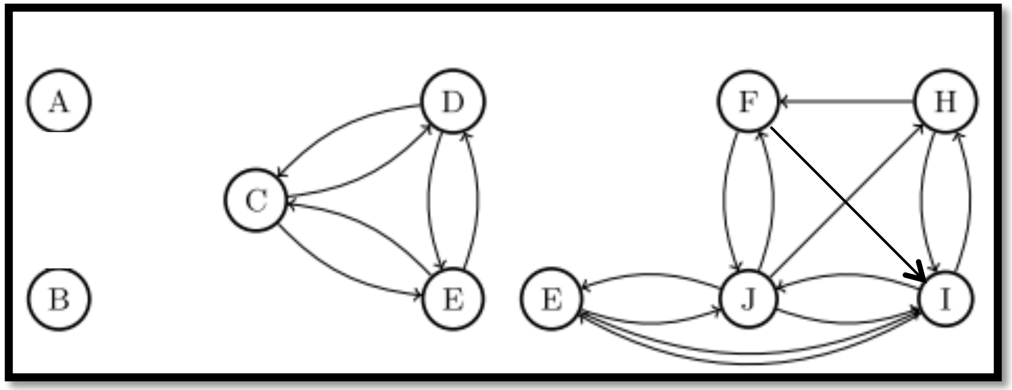
Let G_f be the final graph obtained at the end of this process.

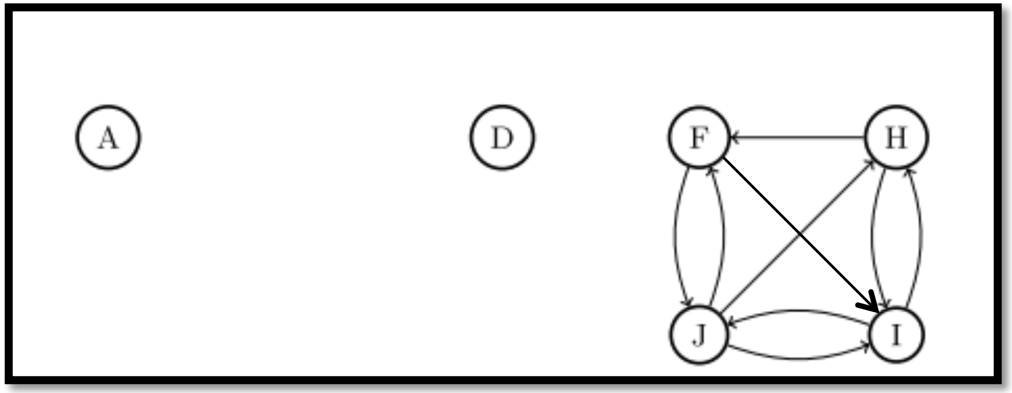
We call the s.c.c.'s of G_f the ***2-vertex connectivity cores*** of the original graph G

2-vertex connectivity cores are subsets of 2-vertex-connected components

2-Vertex-Connected Components and 2-Vertex Cores







2-Edge Cores

Delete recursively all the strong bridges of directed graph *G*, as follows

While there are strong bridges in G:

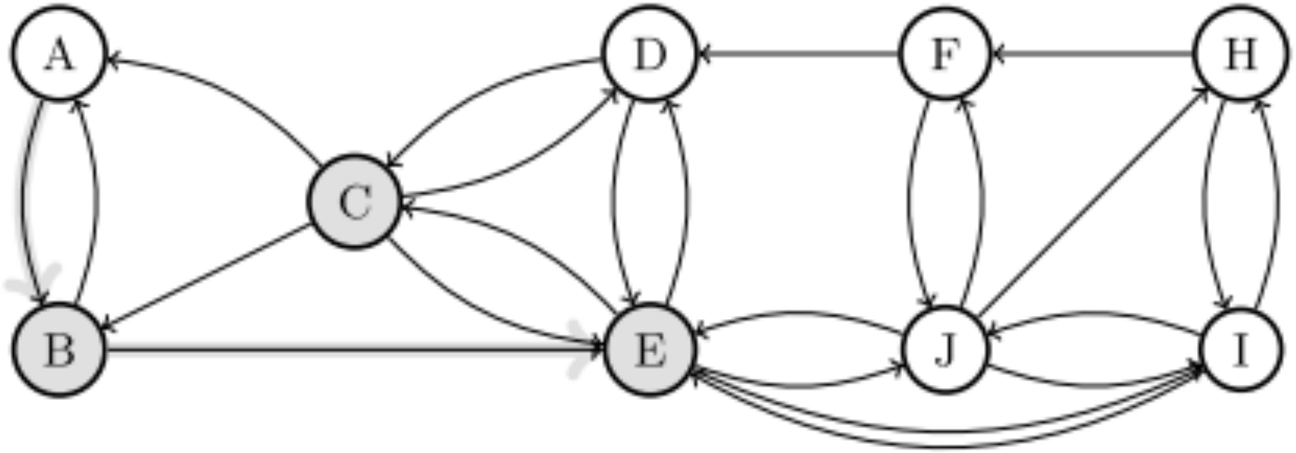
1. Let G' be the graph defined by all the s.c.c.'s of G;
2. Set G to be the graph obtained by deleting all the strong bridges in G'.

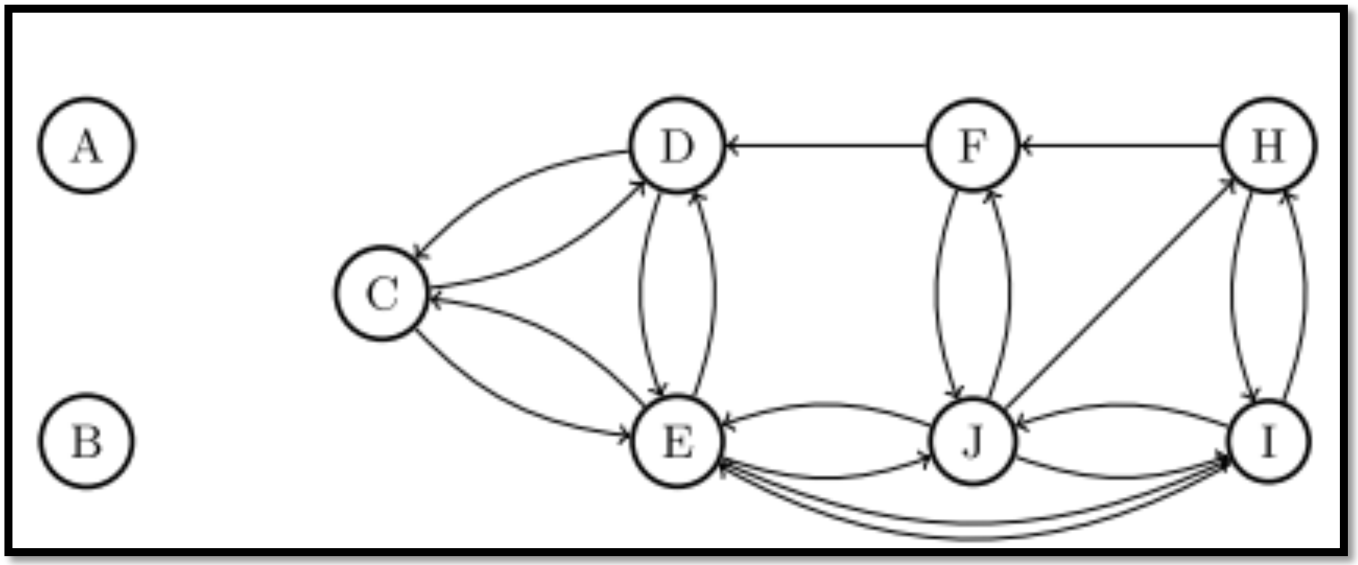
Let G_f be the final graph obtained at the end of this process.

The s.c.c.'s of G_f are the ***2-edge connectivity cores*** of the original graph G

2-edge connectivity cores are exactly 2-edge-connected components

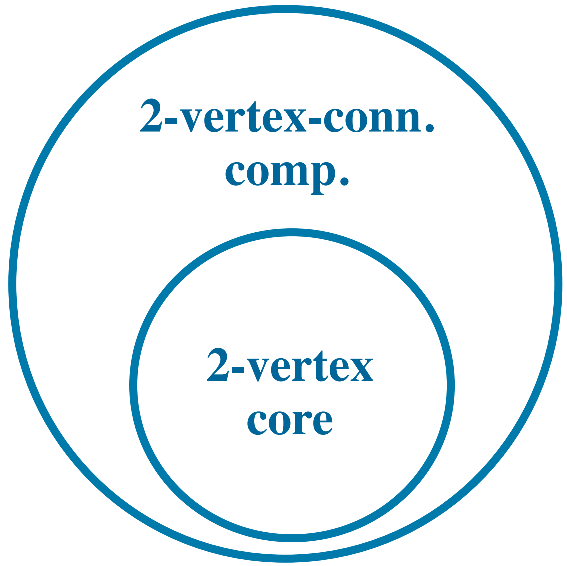
2-Edge-Connected Components = 2-Edge Cores





A Hierarchy of 2-Components

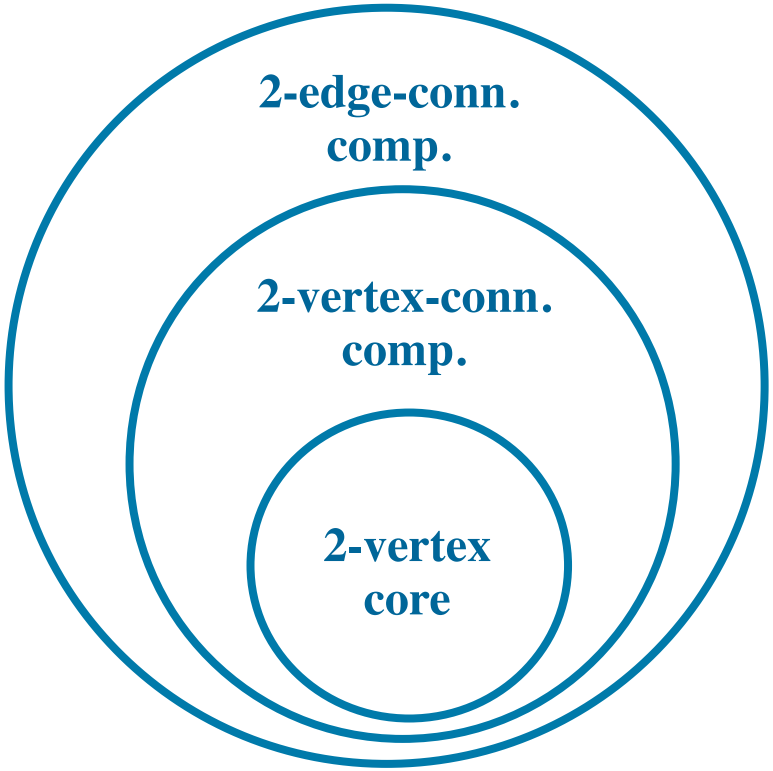
2-vertex core subset of 2-vertex-connected component (modulo degenerate components)



A Hierarchy of 2-Components

2-vertex core subset of 2-vertex-connected component (modulo degenerate components)

2-vertex-connected component subset of 2-edge-connected component



A Hierarchy of 2-Components

2-vertex core subset of 2-vertex-connected component (modulo degenerate components)

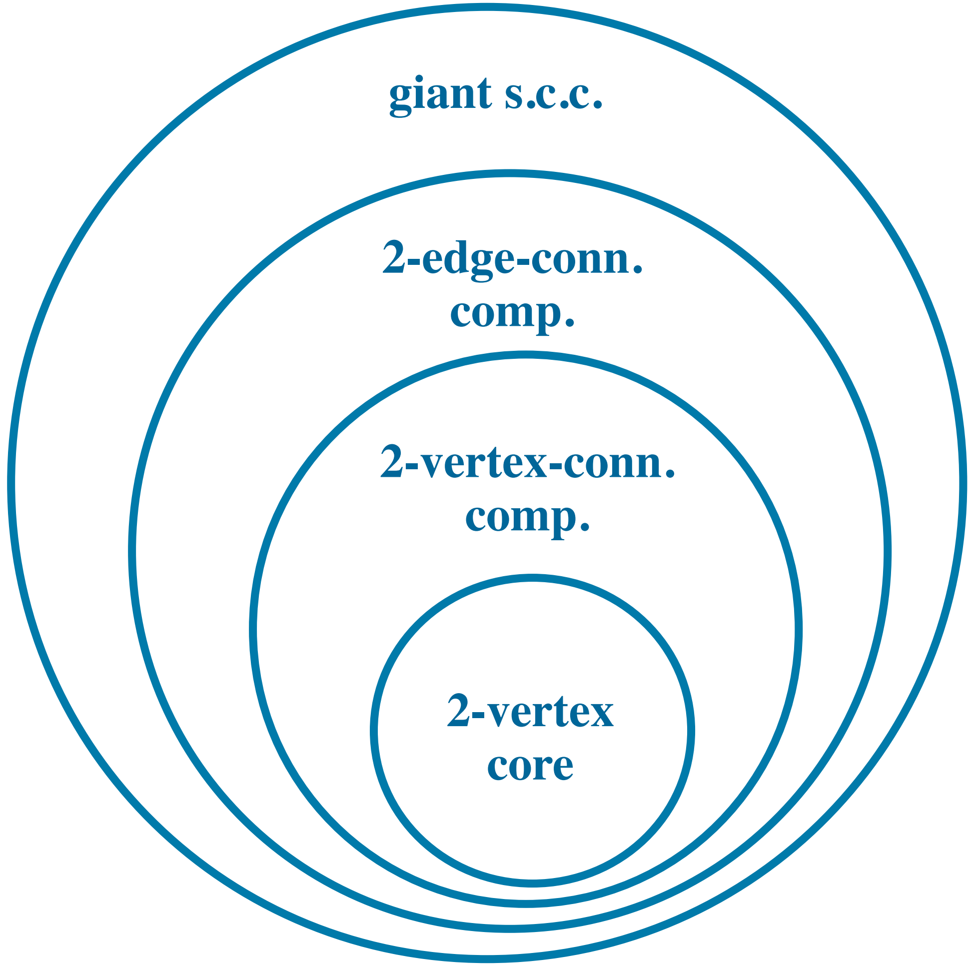
2-vertex-connected component subset of 2-edge-connected component

2-edge-connected component subset of strongly connected component

Roughly speaking:

2VC $\subseteq$ 2VCC $\subseteq$ 2ECC $\subseteq$ SCC

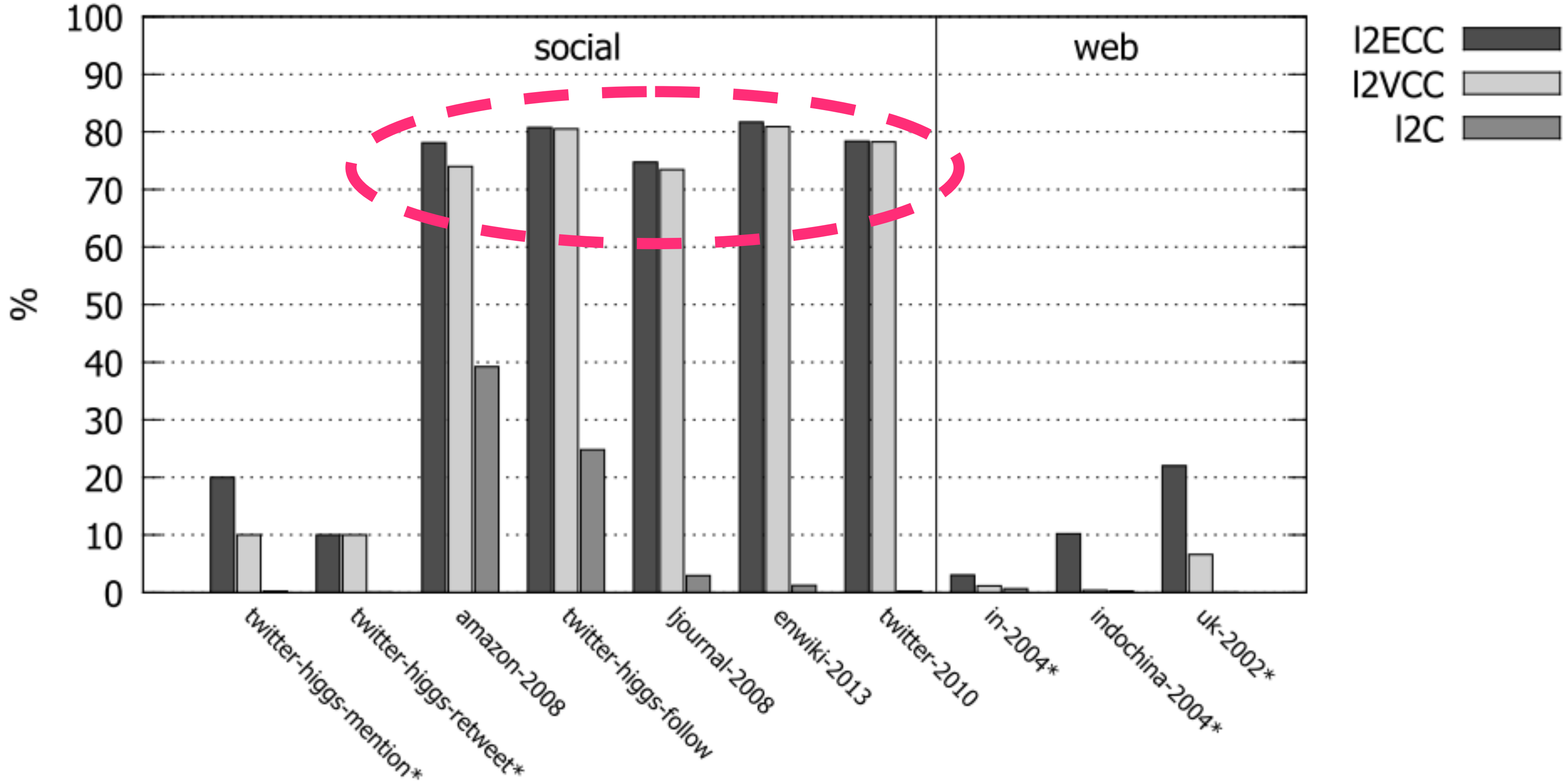
(Will integrate blocks in our experiments soon)



Data Set

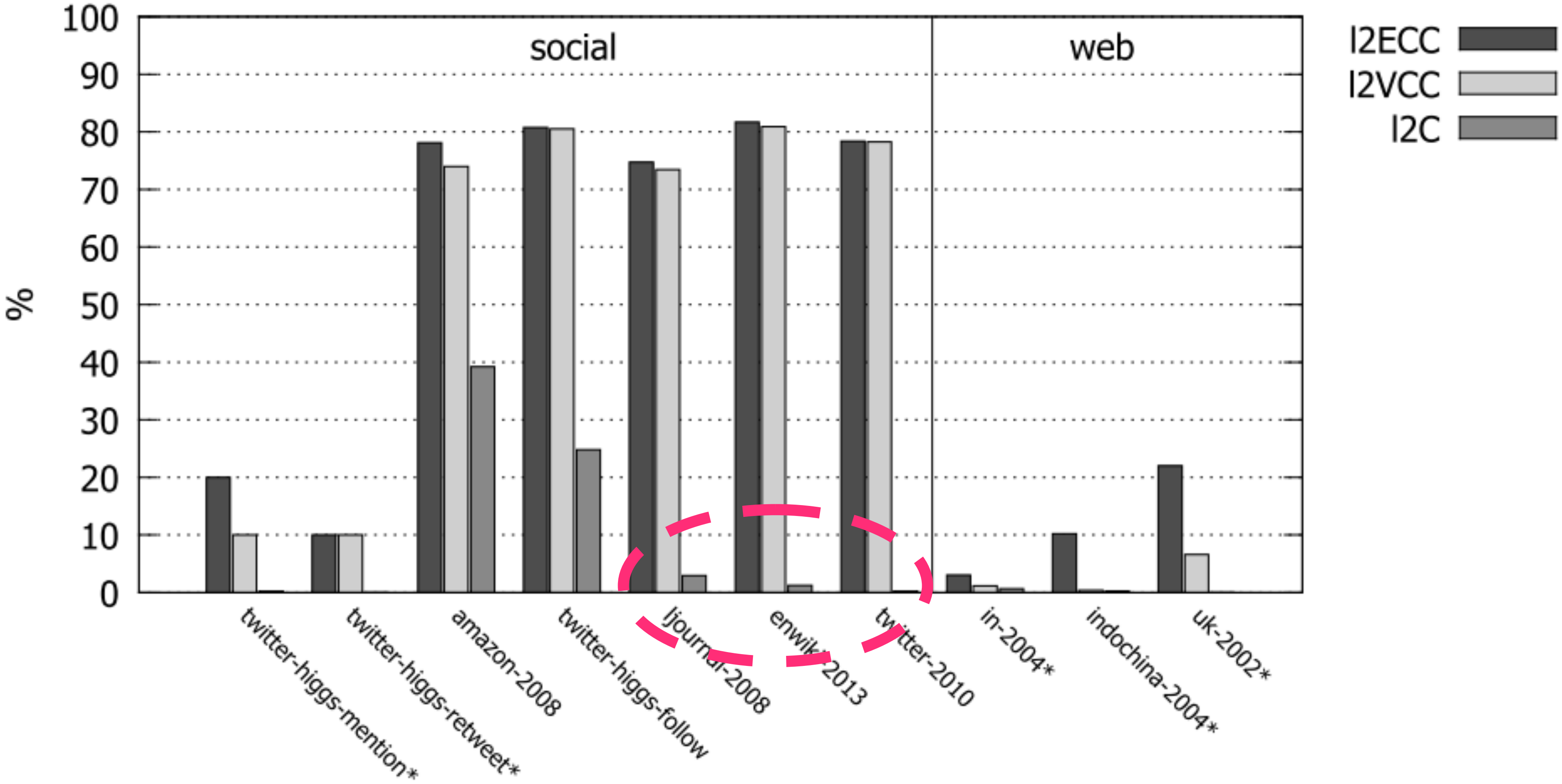
Graph	Type	Repository	n	m	lSCC
twitter-higgs-mention	Social	SNAP	303 K	448.8 K	3%
twitter-higgs-retweet	Social	SNAP	427 K	733.6 K	2%
amazon-2008	Social	WebGraph	735 K	5.1 M	85%
twitter-higgs-follow	Social	SNAP	456 K	14.8 M	79%
ljournal-2008	Social	WebGraph	5.4 M	79 M	78%
enwiki-2013	Social	WebGraph	4.3 M	101.3 M	89%
twitter-2010	Social	WebGraph	41.6 M	1.5 G	80%
in-2004	Web	WebGraph	1.4 M	17 M	43%
indochina-2004	Web	WebGraph	7.4 M	194.1 M	51%
uk-2002	Web	WebGraph	18.5 M	298.1 M	65%

Social: Bigger 2-Components…



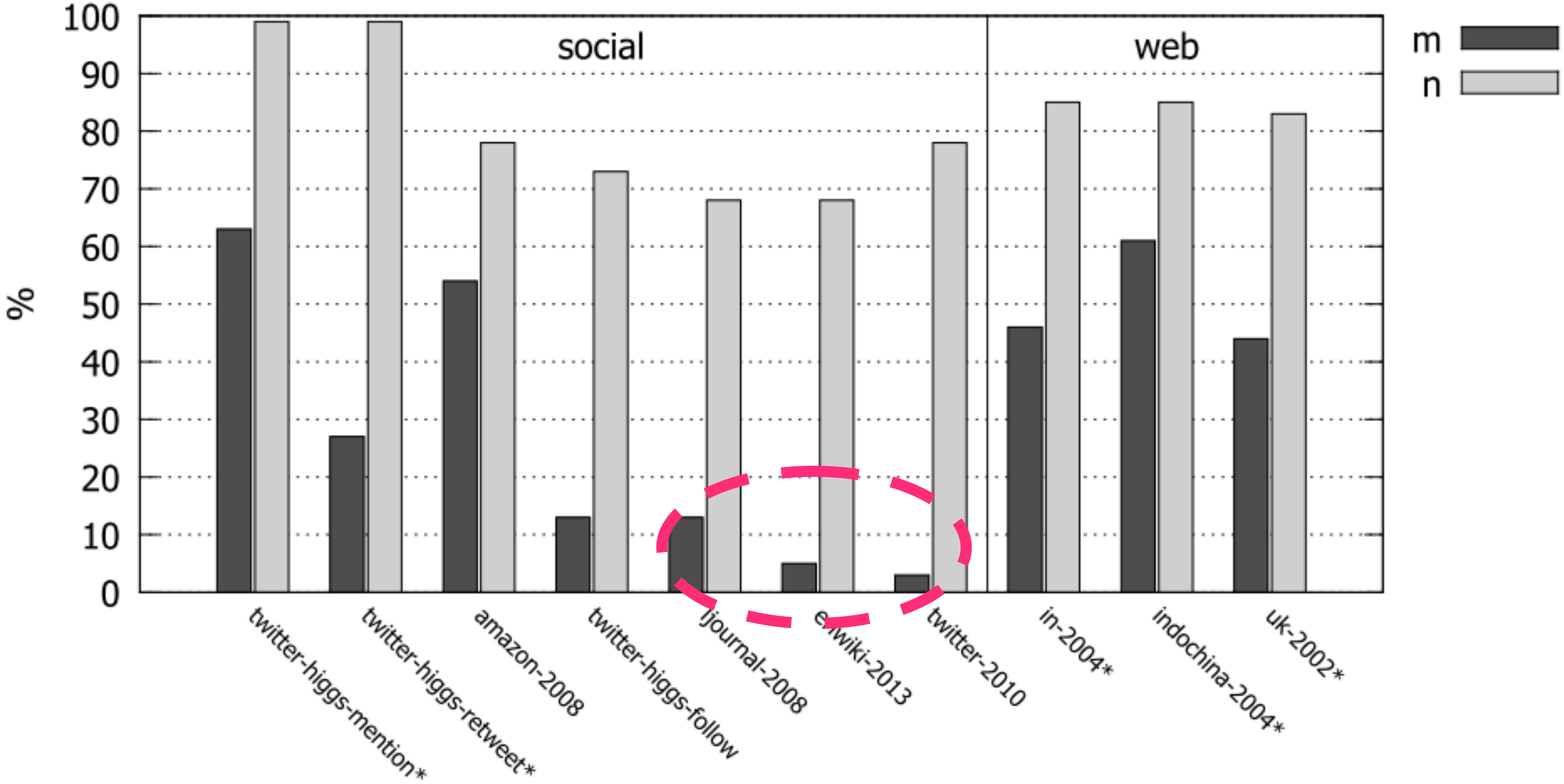
Size of the giant 2-vertex- and 2-edge-connected component (I2VCC and I2ECC) and in the largest 2-vertex-connected core (I2C).
(Expressed as % of vertices in the ISCC)

…some have small cores…



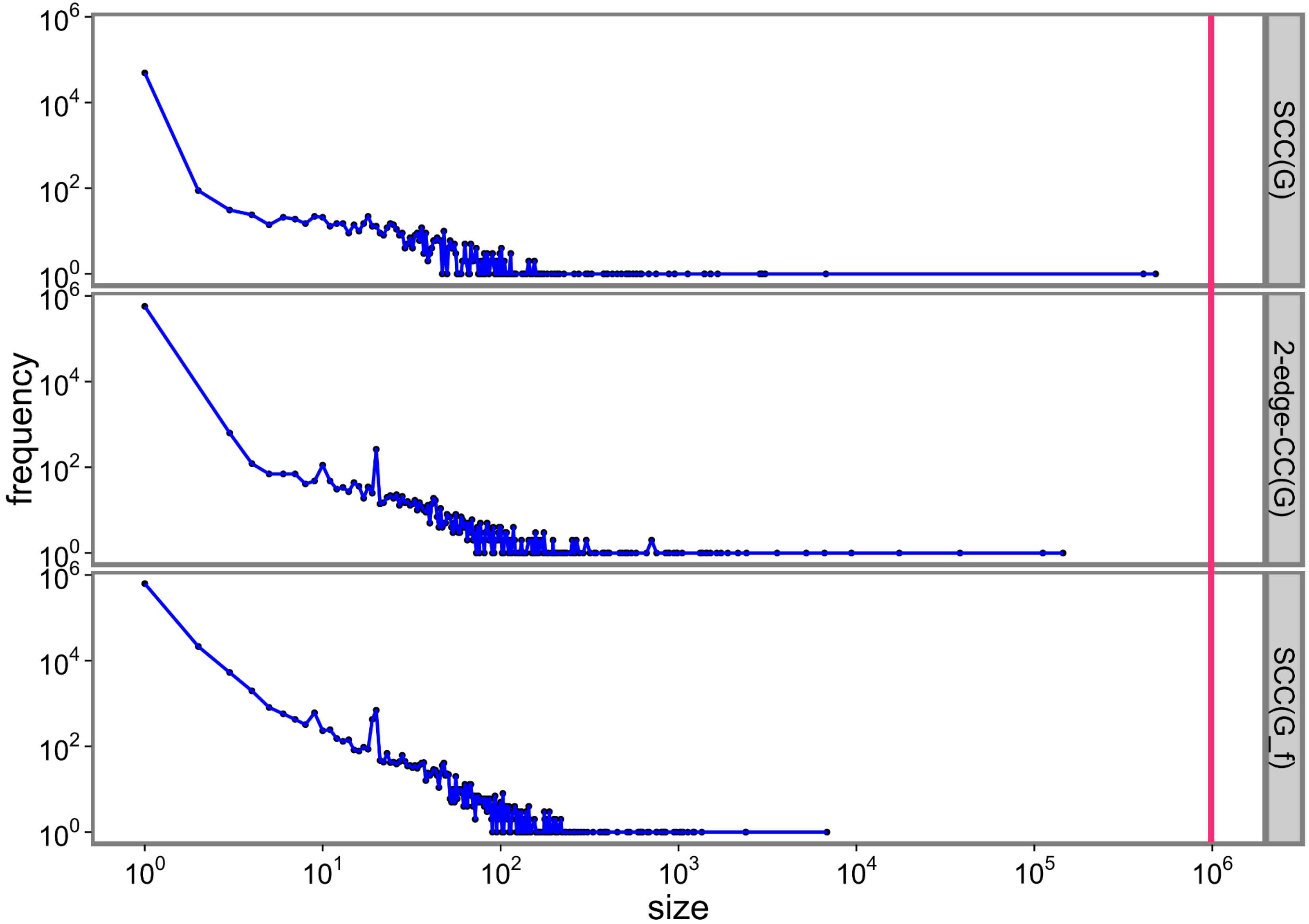
Size of the giant 2-vertex- and 2-edge-connected component (I2VCC and I2ECC) and in the largest 2-vertex-connected core (I2C).
(Expressed as % of vertices in the ISCC)

…but are buried deep inside



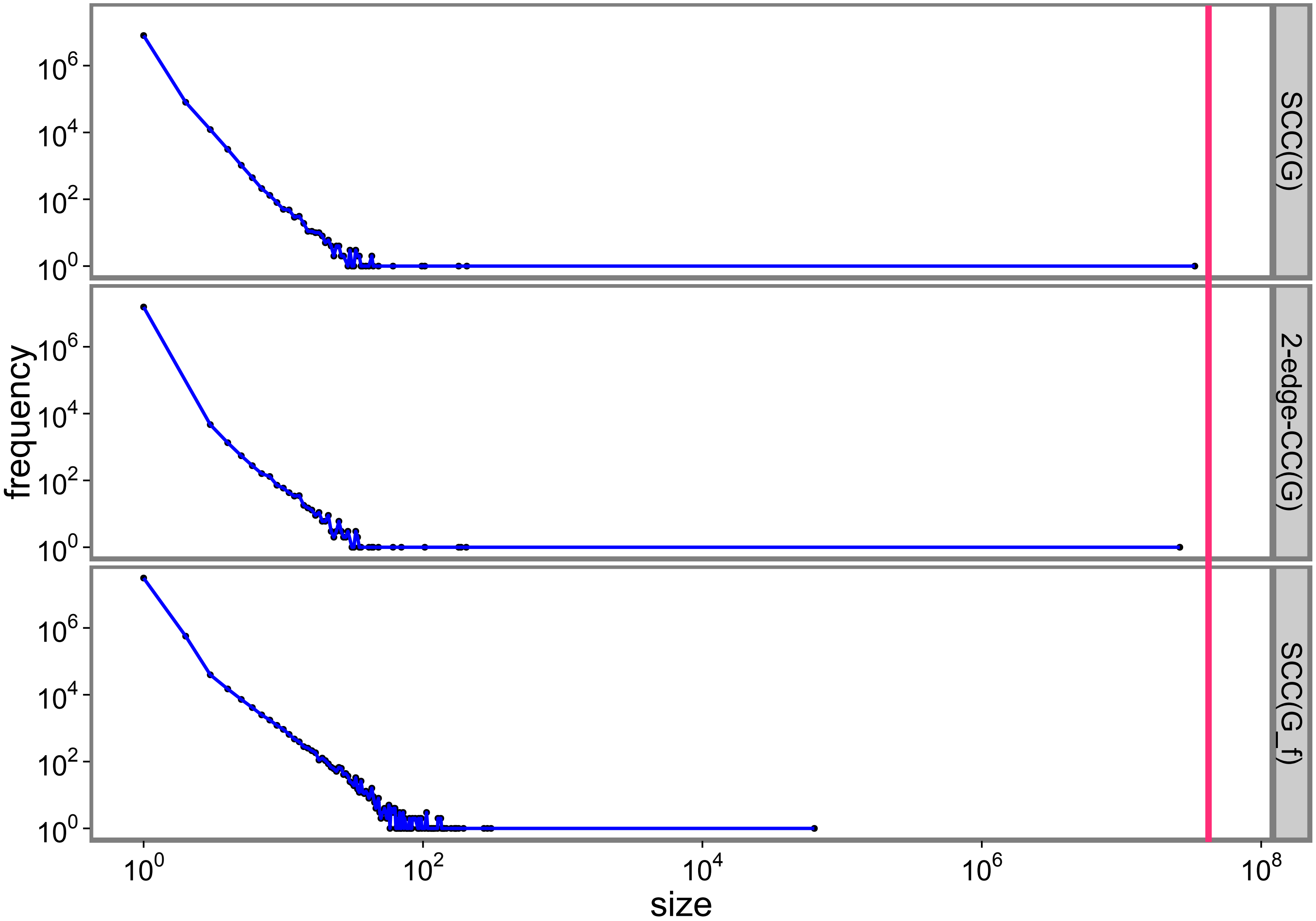
Number of vertices and edges in the final graph G_f obtained after removing recursively all strong articulation points.
(Expressed as % of n and m)

Biconnectivity: Web Graphs



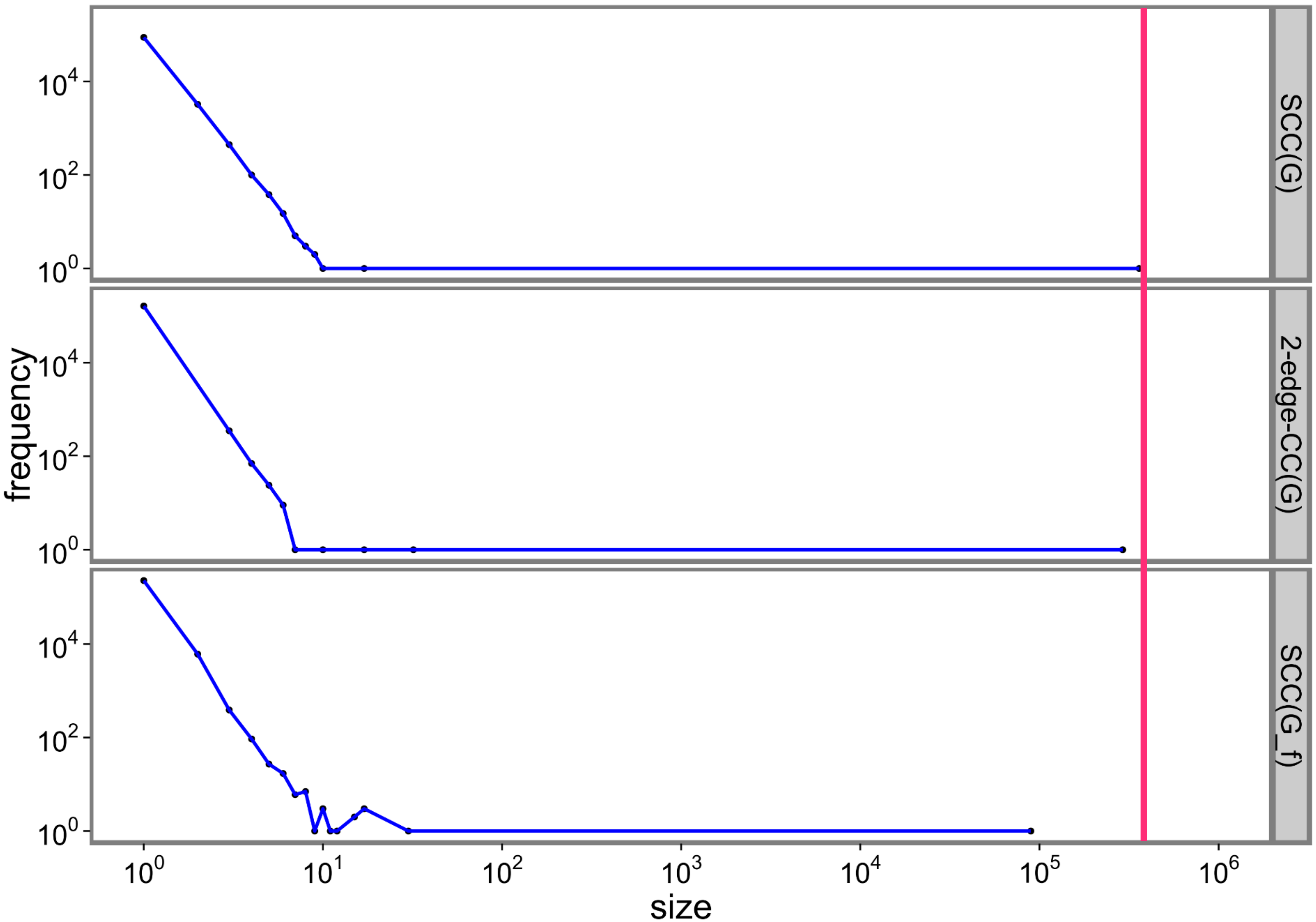
uk-2002@1M (n = 1M, m = 41.2M)

Biconnectivity: Social Networks



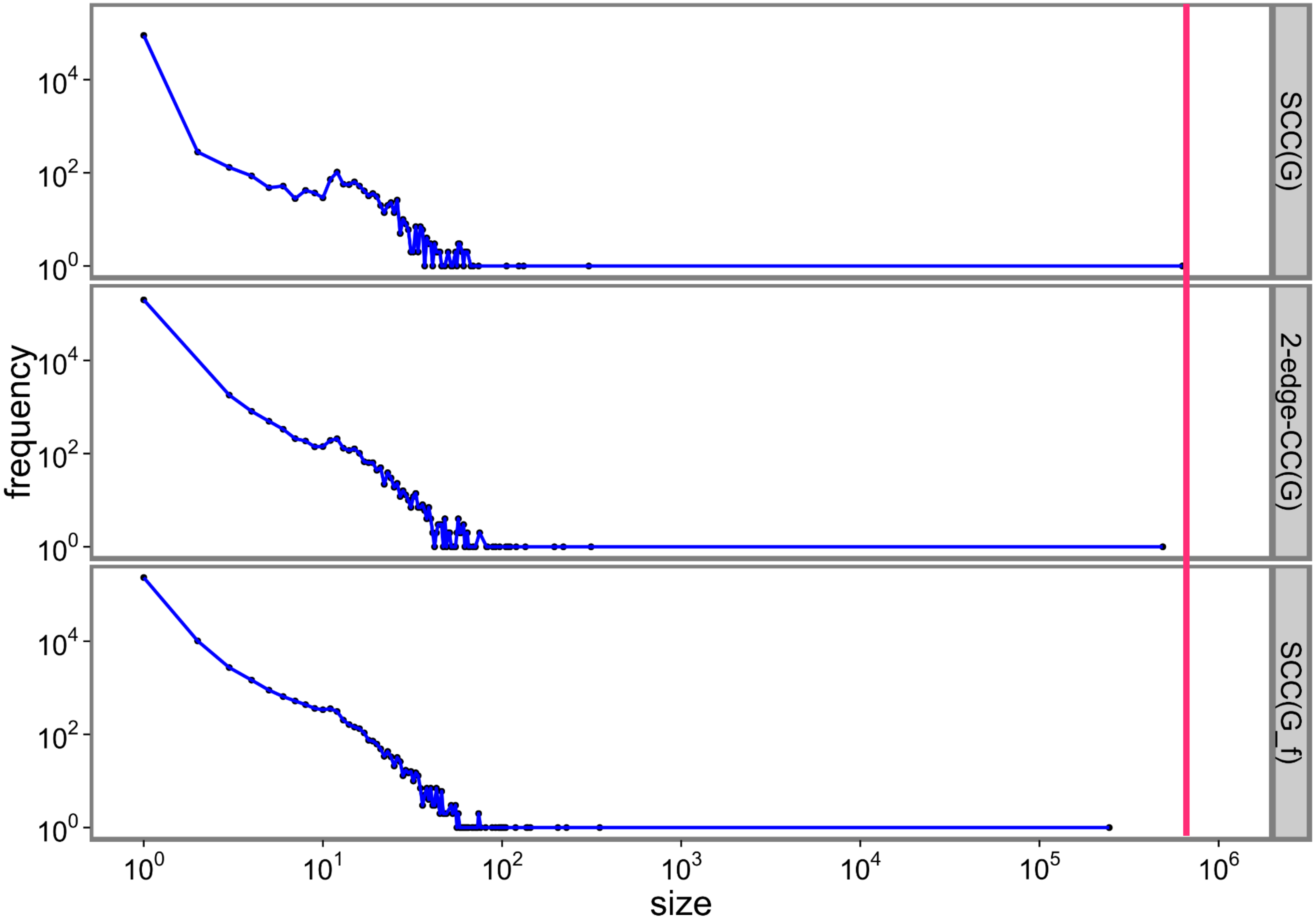
twitter-2010 (n = 41.6M, m = 1,468.3M)

Biconnectivity: Social (sub)networks



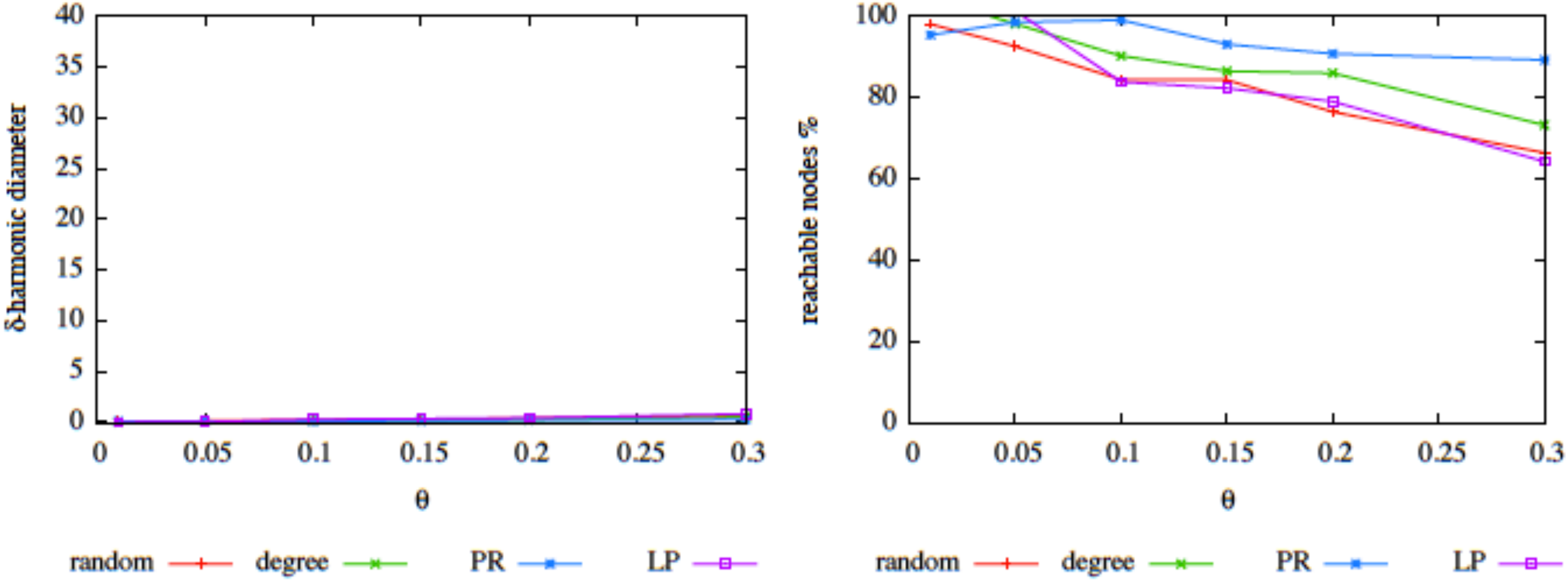
twitter-higgs-follow (n = 456K, m = 14.8M)

Biconnectivity: Co-Purchase

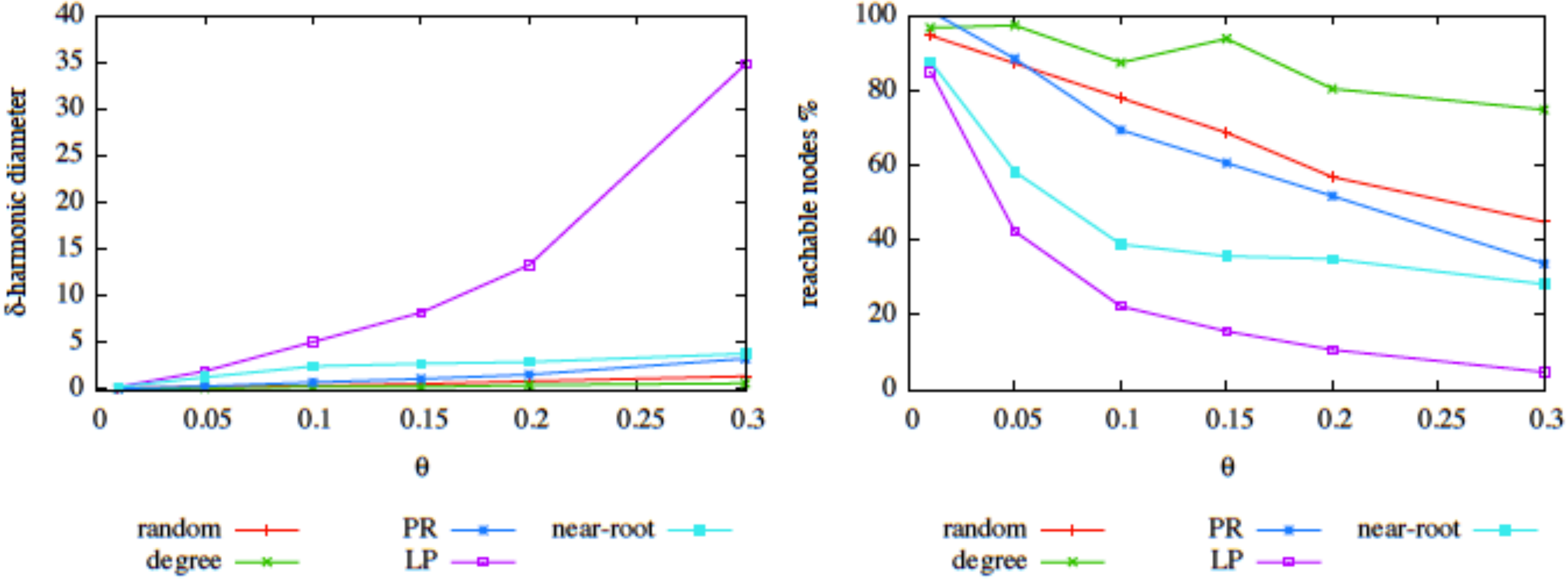


amazon-2008 (n = 735K, m = 5.1M)

[Boldi, Rosa & Vigna 2013]

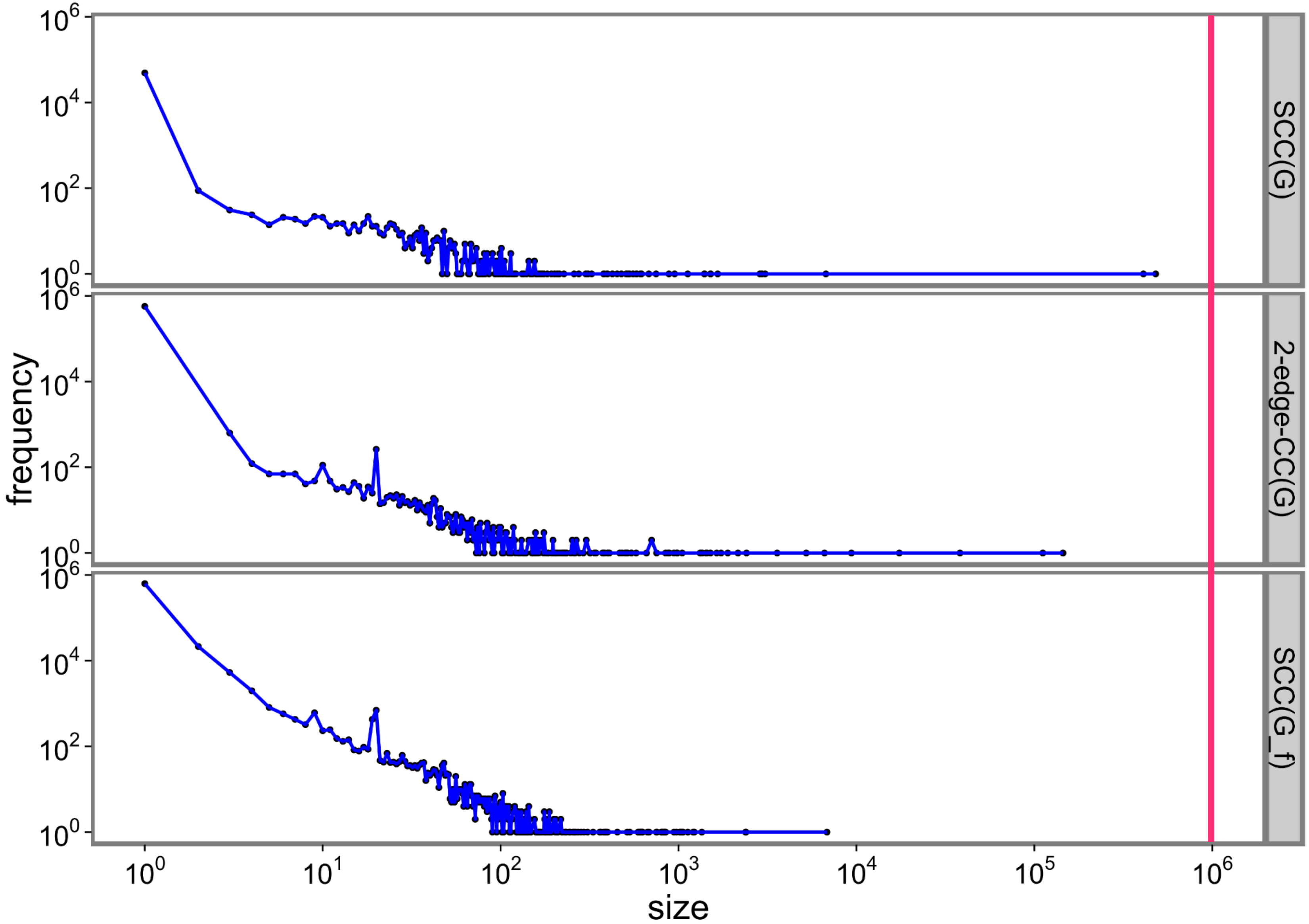


Amazon



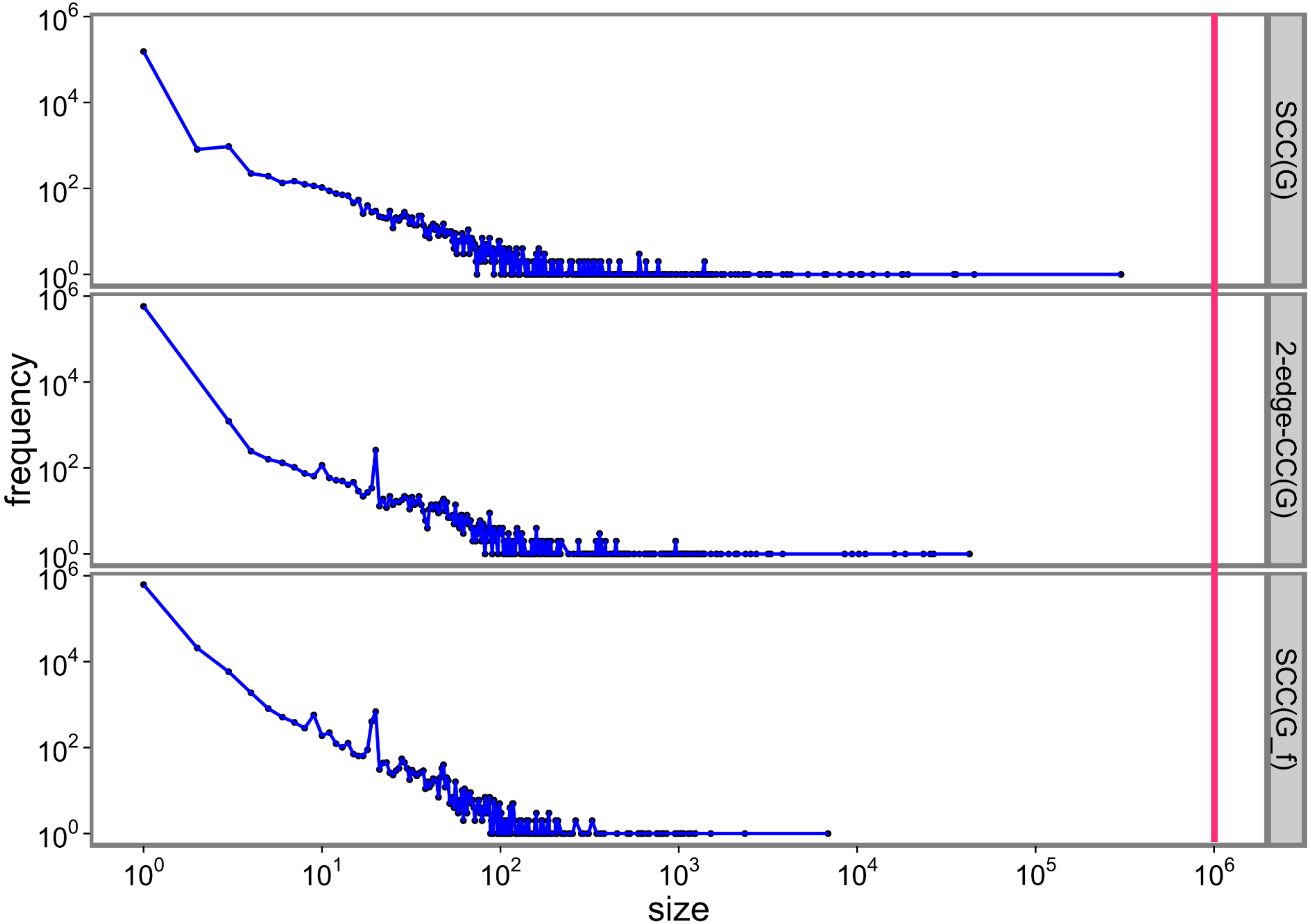
.it

30% LP on Web Graphs



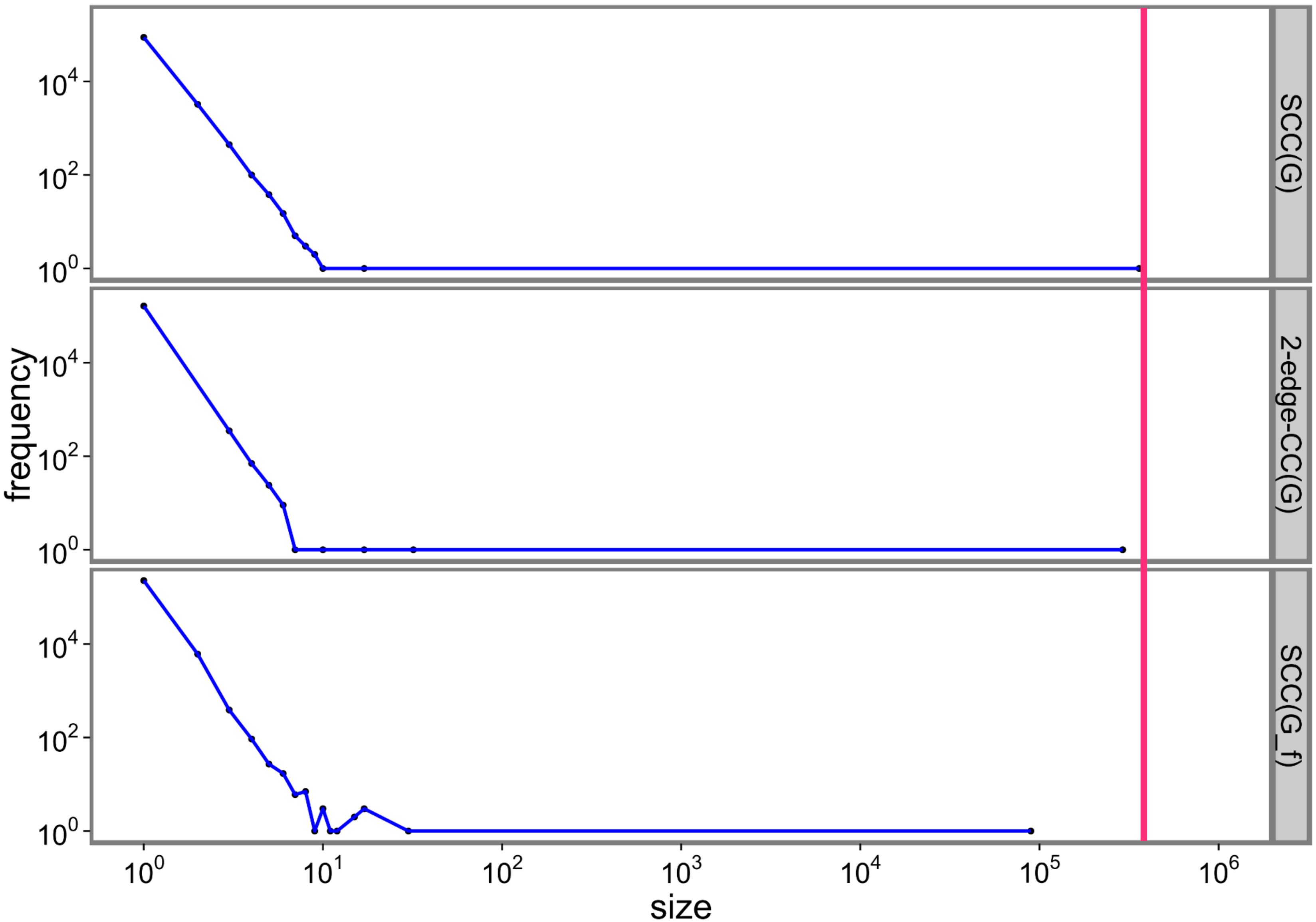
uk-2002@1M (n = 1M, m = 41.2M)

30% LP on Web Graphs



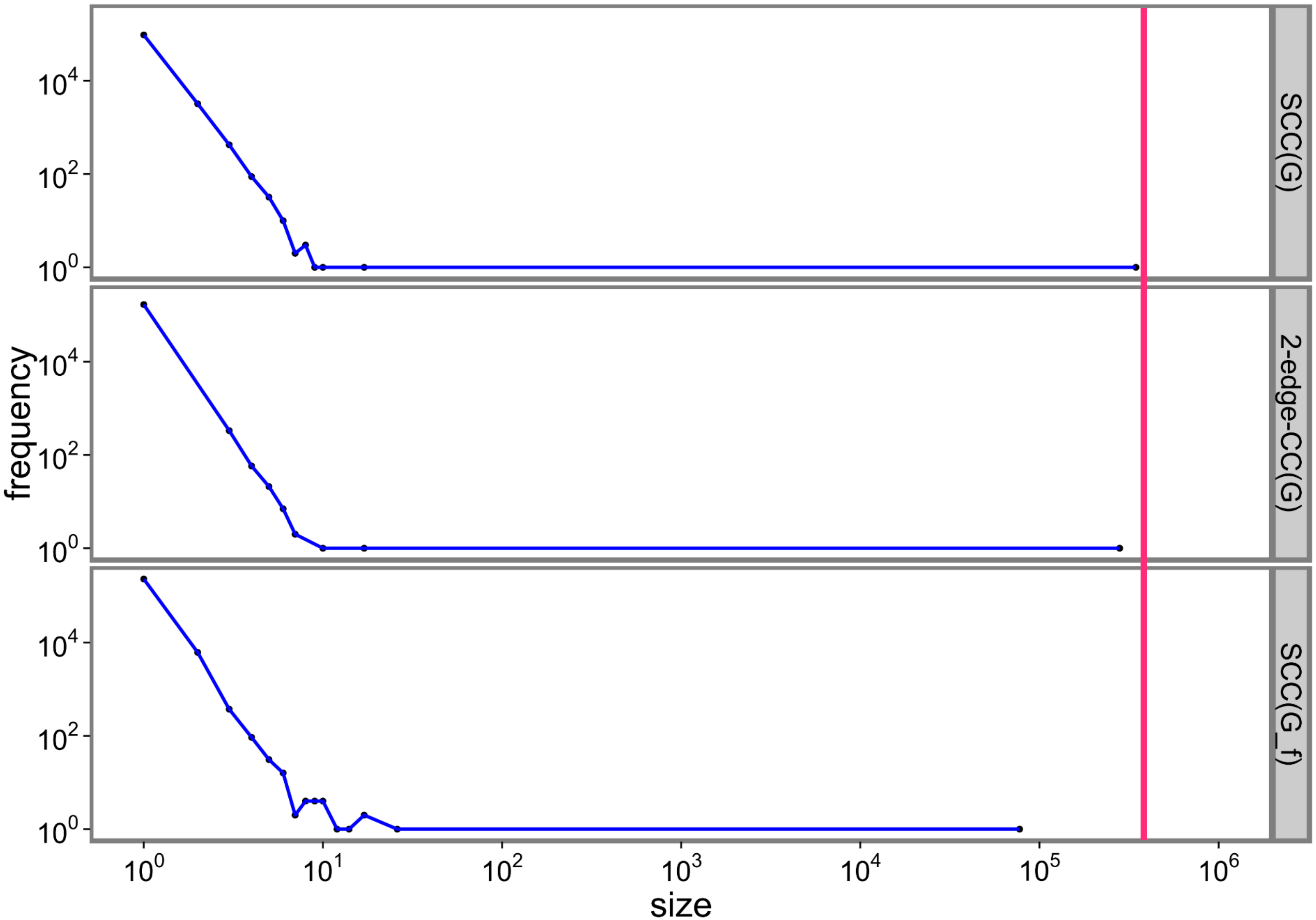
uk-2002@1M (n = 1M, m = 41.2M)

30% LP on Social Networks



twitter-higgs-follow (n = 456K, m = 14.8M)

30% LP on Social Networks



twitter-higgs-follow (n = 456K, m = 14.8M)

Today's Outline

1. 2-Connectivity on directed graphs
2. Algorithms for strong articulation points and strong bridges
3. Experiments
4. **Open Problems**

Open Problems

Can the 2-vertex and 2-edge-connected components of a directed graph be computed in linear time?

> Best known time is $O(n(m+n))$ by repeatedly deleting strong articulation points / strong bridges.
>
> Can a dynamic algorithm help you?

Higher connectivity cuts in strongly connected graphs in linear time? (e.g., separation pairs: vertex and edge cuts of cardinality 2)

Open Problems / Future Work

Would like to understand more the structure and the properties of 2-connectivity components / blocks (especially in real-world graphs)

References

Ya. M. Erusalimskii and G. G. Svetlov. Bijoin points, bibridges, and biblocks of directed graphs. Cybernetics, 16(1):41-44, 1980.

W. Di Luigi, L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis 2-Connectivity in Directed Graphs: An Experimental Study. 17th SIAM Meeting on Algorithm Engineering and Experimentation (ALENEX 2015).

D. Firmani, G. F. Italiano, L. Laura, A. Orlandi, and F. Santaroni. Computing strong articulation points and strong bridges in large scale graphs. In Proc. 10th Int'l. Symp. on Experimental Algorithms (SEA), 195-207, 2012.

L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis. 2-edge connectivity in directed graphs. In Proc. 26th ACM-SIAM Symp. on Discrete Algorithms (SODA), 2015. To appear.

L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis. 2-vertex connectivity in directed graphs. CoRR, abs/1409.6277, 2014.

References

L. Georgiadis, L. Laura, N. Parotsidis, and R. E. Tarjan. Dominator certication and independent spanning trees: An experimental study. In Proc. 12th Int'l. Symp. on Experimental Algorithms (SEA), 284-295, 2013.

L. Georgiadis, L. Laura, N. Parotsidis, and R. E. Tarjan. Loop nesting forests, dominators, and applications. In Proc. 13th Int'l. Symp. on Experimental Algorithms (SEA), 174-186, 2014.

G. F. Italiano, L. Laura, and F. Santaroni. Finding strong bridges and strong articulation points in linear time. Theoretical Computer Science, 447(0):74-84, 2012.

R. Jaberi. Computing the 2-blocks of directed graphs. CoRR, abs/1407.6178, 2014.

R. Jaberi. On computing the 2-vertex-connected components of directed graphs. CoRR, abs/1401.6000, 2014.