Rensselaer Polytechnic Institute, Troy NY USA
Universidade Federal de Viçosa, MG, Brazil

RPI UFV

Fast analysis of upstream features on spatial networks

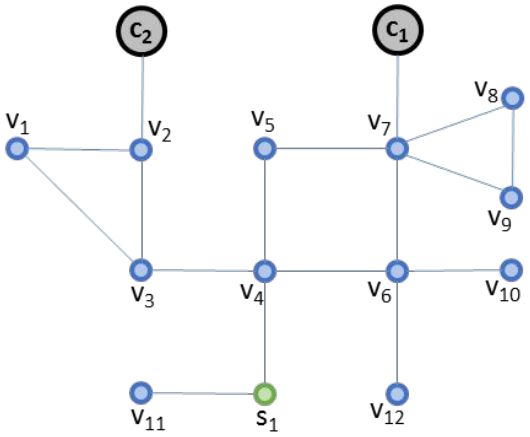
(winner, 1st place, 2018 GIS Cup)

Salles Viana Gomes Magalhães, UFV/RPI
W. Randolph Franklin, RPI
Ricardo dos Santos Ferreira, UFV

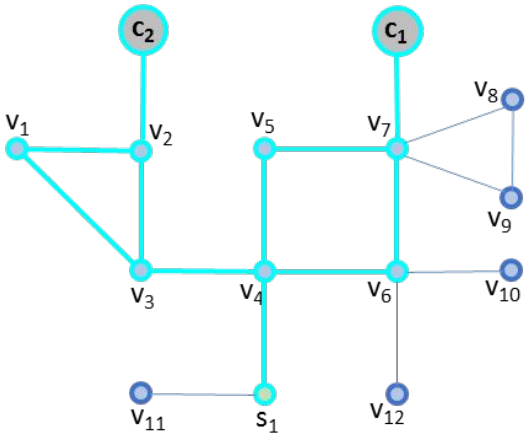
acm SIGSPATIAL

Introduction

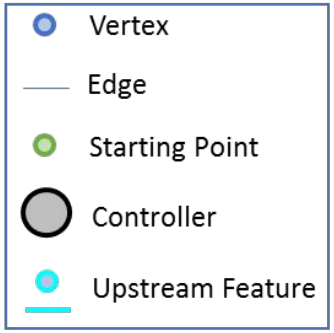
- Input: graph (JSON file), set of **starting points** (vertices/edges) and **controllers** (vertices)
- We call starting points/controllers **important vertices**
- Output: edges and vertices in a simple path between a controller and a starting point.
- **Challenge**: the number of paths between a pair of vértices may be huge (e.g.: exponential in a complete graph) → paths cannot enumerate



Input



Output

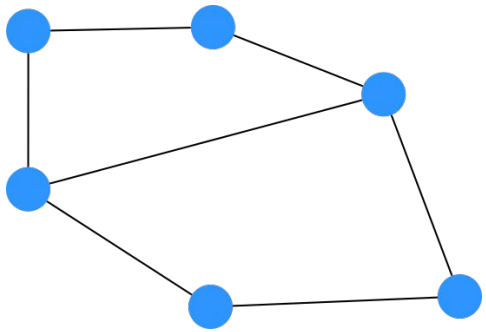


Source: GISCUP 2018

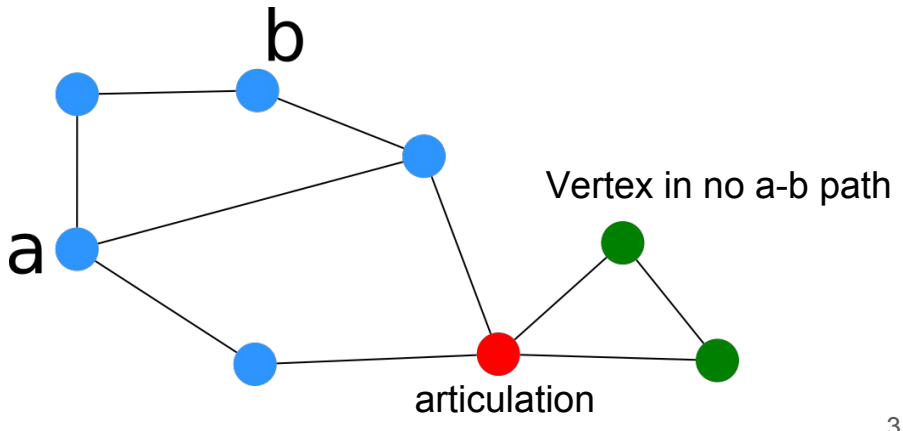
Key ideas

- **Key idea**: avoid enumerating all paths between controllers and starting points.
- **Main concepts**: articulations, biconnected components, Block-Cut trees
- **Articulation**: vertex which, when removed, increases the number of connected components (CCs) in a graph
- **Biconnected graph**: graph without articulations
- **Biconnected components** (or **blocks**): maximal biconnected subgraphs of a graph.
- **Important observation**: given two vertices a,b of a block B, any vertex of B is in an a-b path.

Biconnected graph



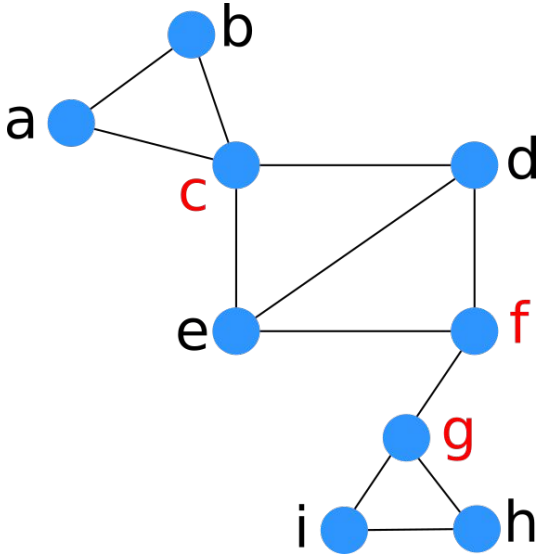
Non biconnected graph



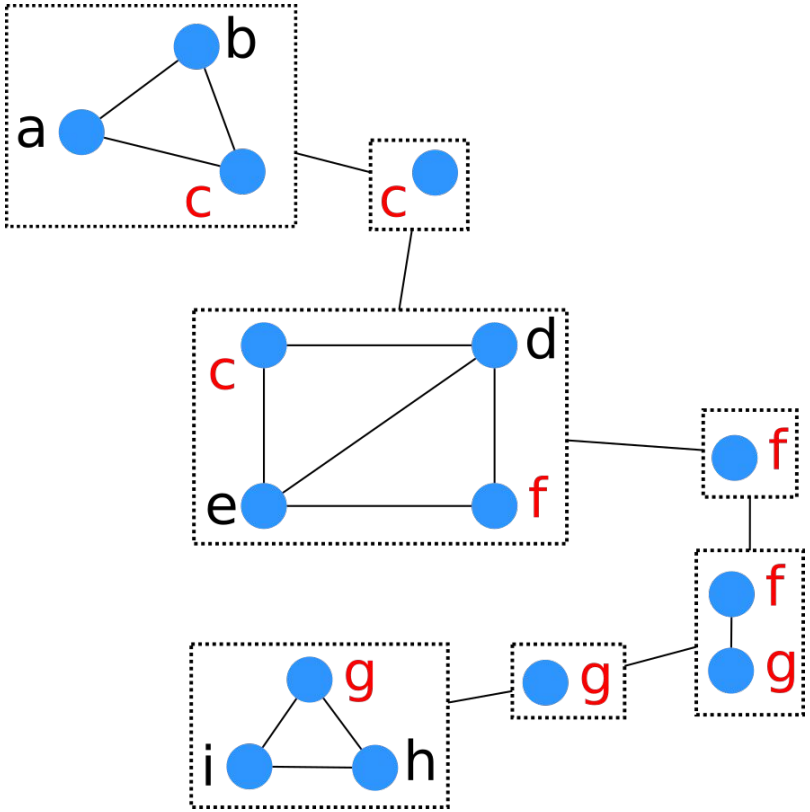
Key ideas

- **Block-cut tree** of a graph G (**BC(G)**): tree where:
  - Vertices are the blocks and articulations of G
  - There is an edge between each block B and the articulations from B.

Graph G (articulations in red)



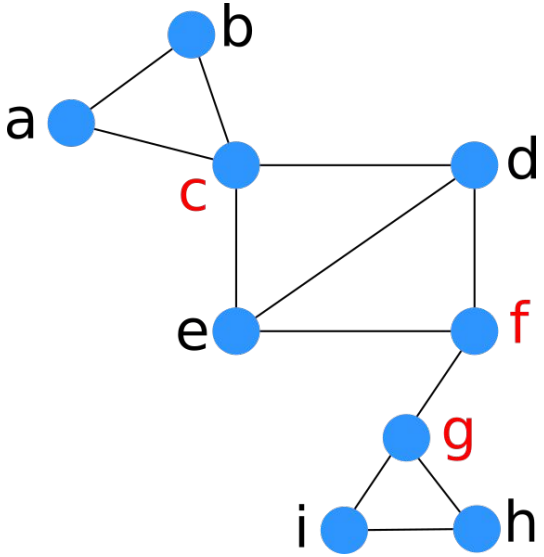
Block-cut tree of G
(rectangles represent the vertices of the BC-Tree)



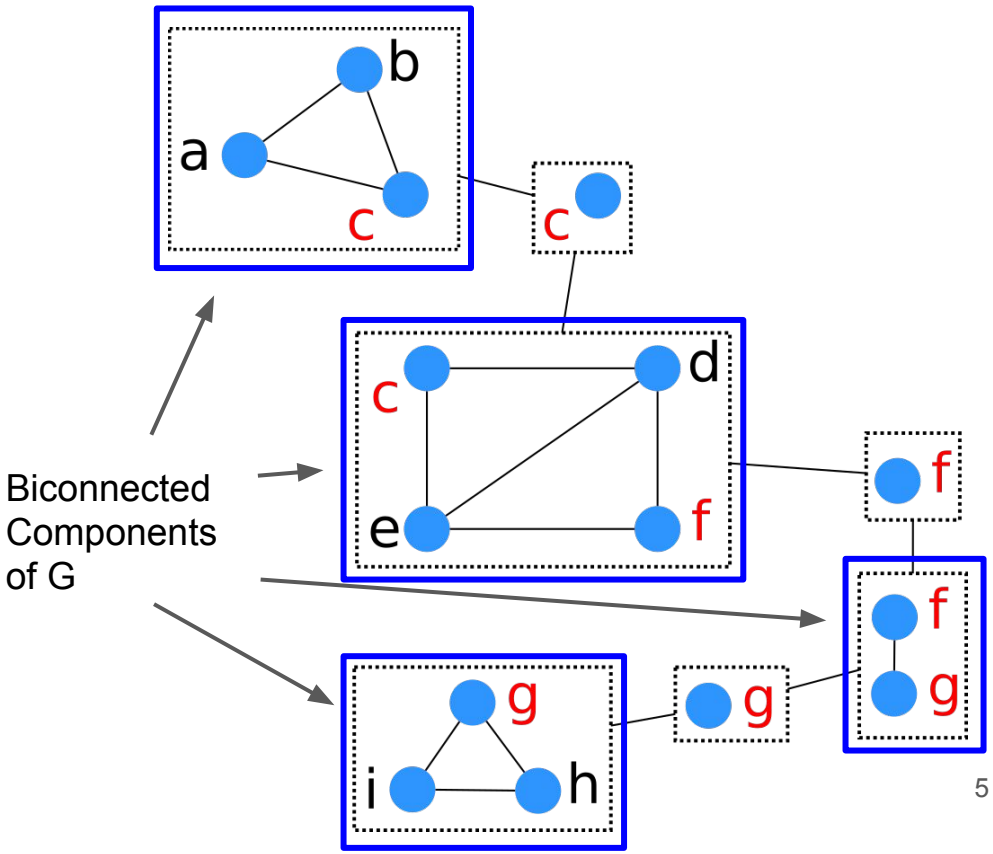
Key ideas

- **Block-cut tree** of a graph G (**BC(G)**): tree where:
  - Vertices are the blocks and articulations of G
  - There is an edge between each block B and the articulations from B.

Graph G (articulations in red)

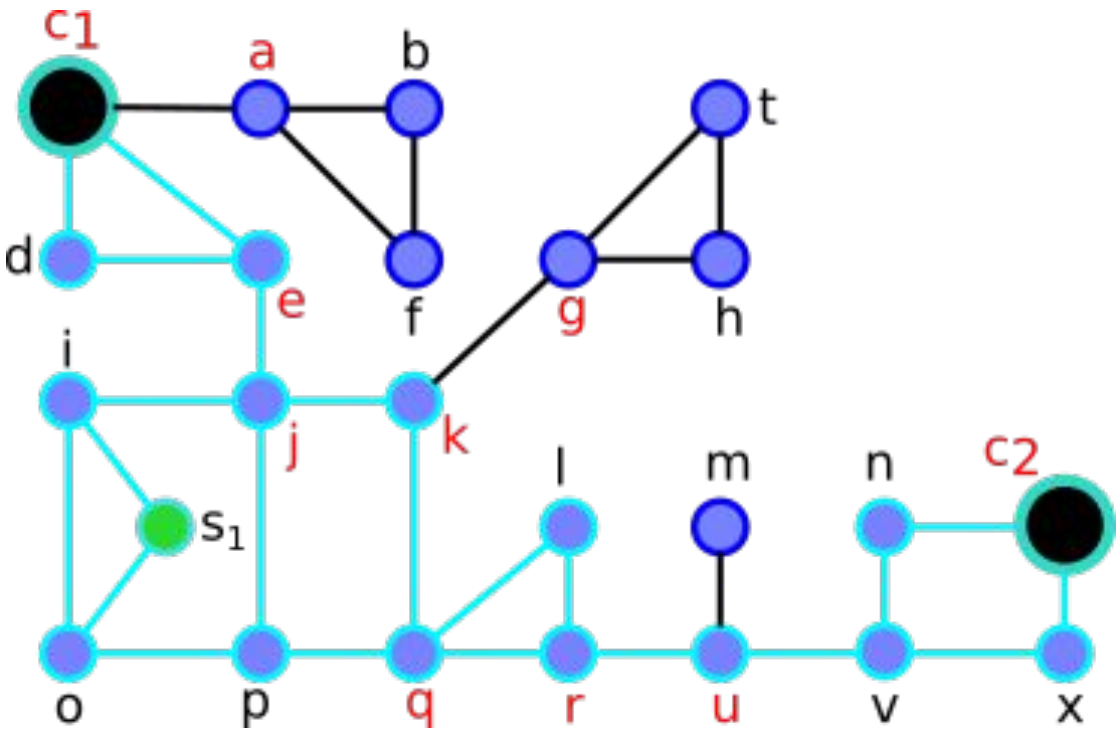


Block-cut tree of G
(rectangles represent the vertices of the BC-Tree)



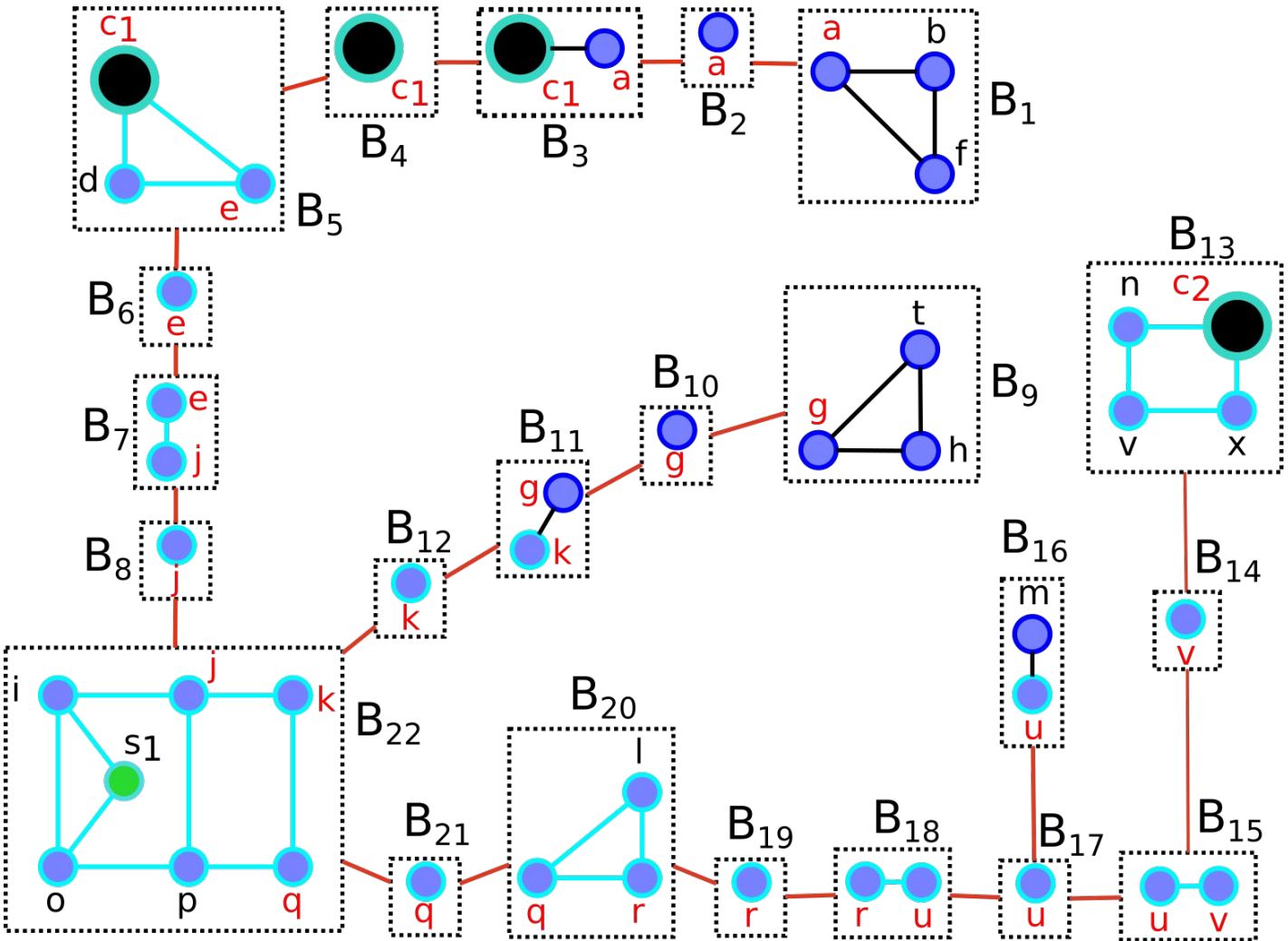
The algorithm

- Consider this Graph with some controllers/starting points.
    - If graph does not contain both → solution is empty
- Graph is connected (otherwise, solve for each connected component)
- Initially, suppose important features can only be vertices.
- Black vertices: controllers
- Green vertex: starting point
- Red labels: articulations
- Features detached in light blue: output features



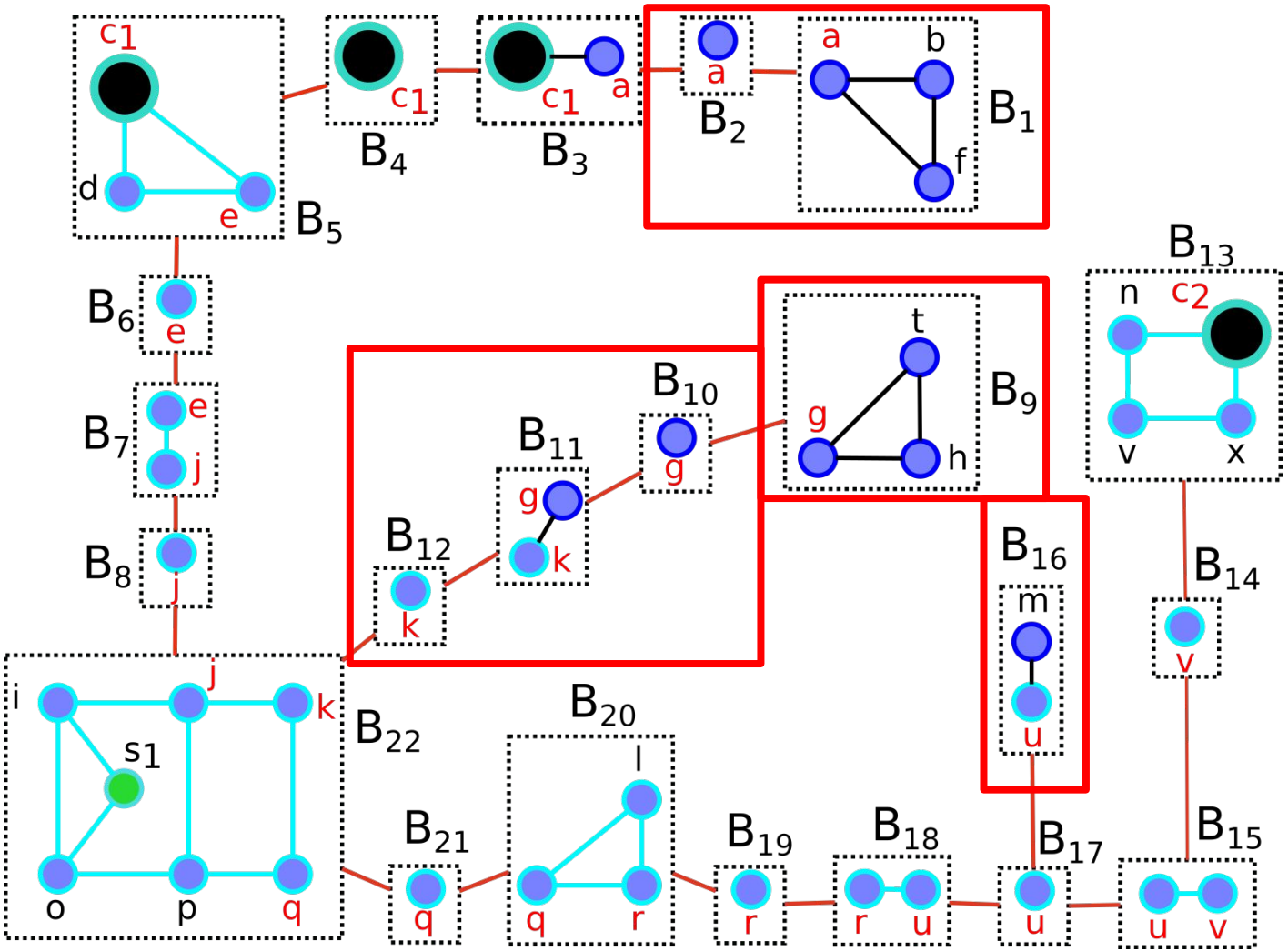
The algorithm

- First step: create block-cut tree



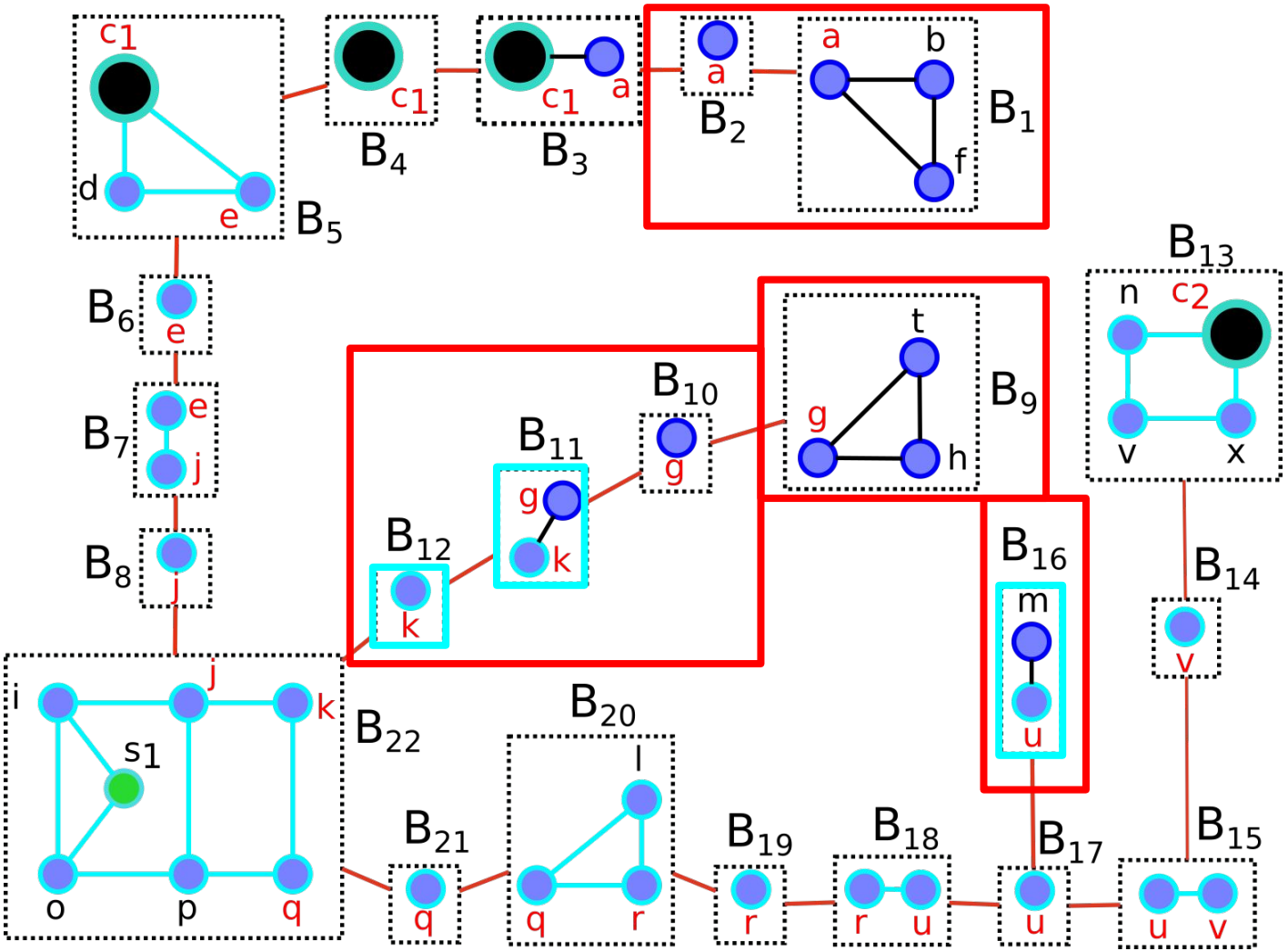
The algorithm

- Leaves without important vértices → are in no path between important vértices → can be iteratively removed



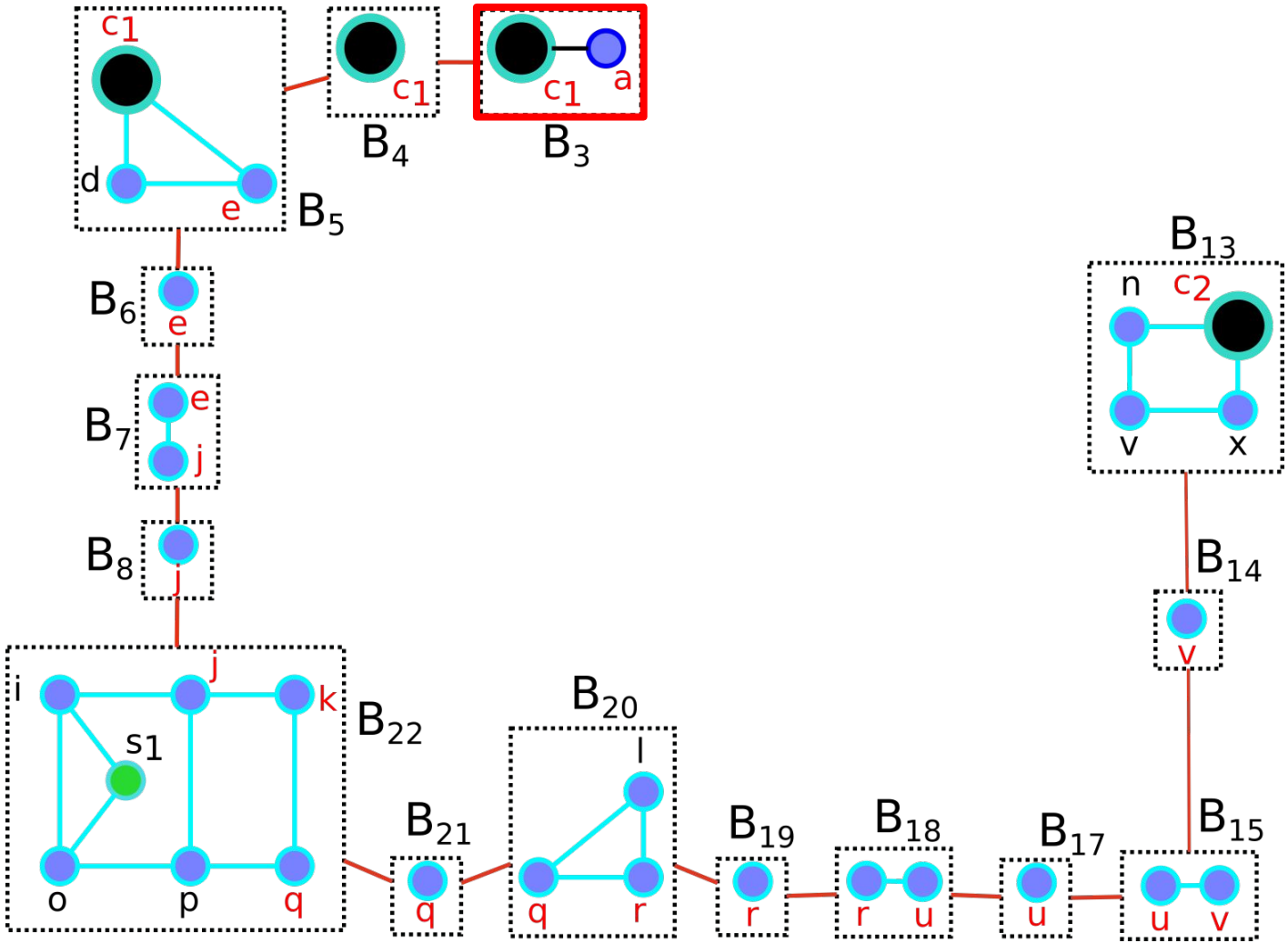
The algorithm

- Some of the removed blocks contain features that should be in the output.
  - This is ok → these vertices are also in neighbor (non-removed) blocks



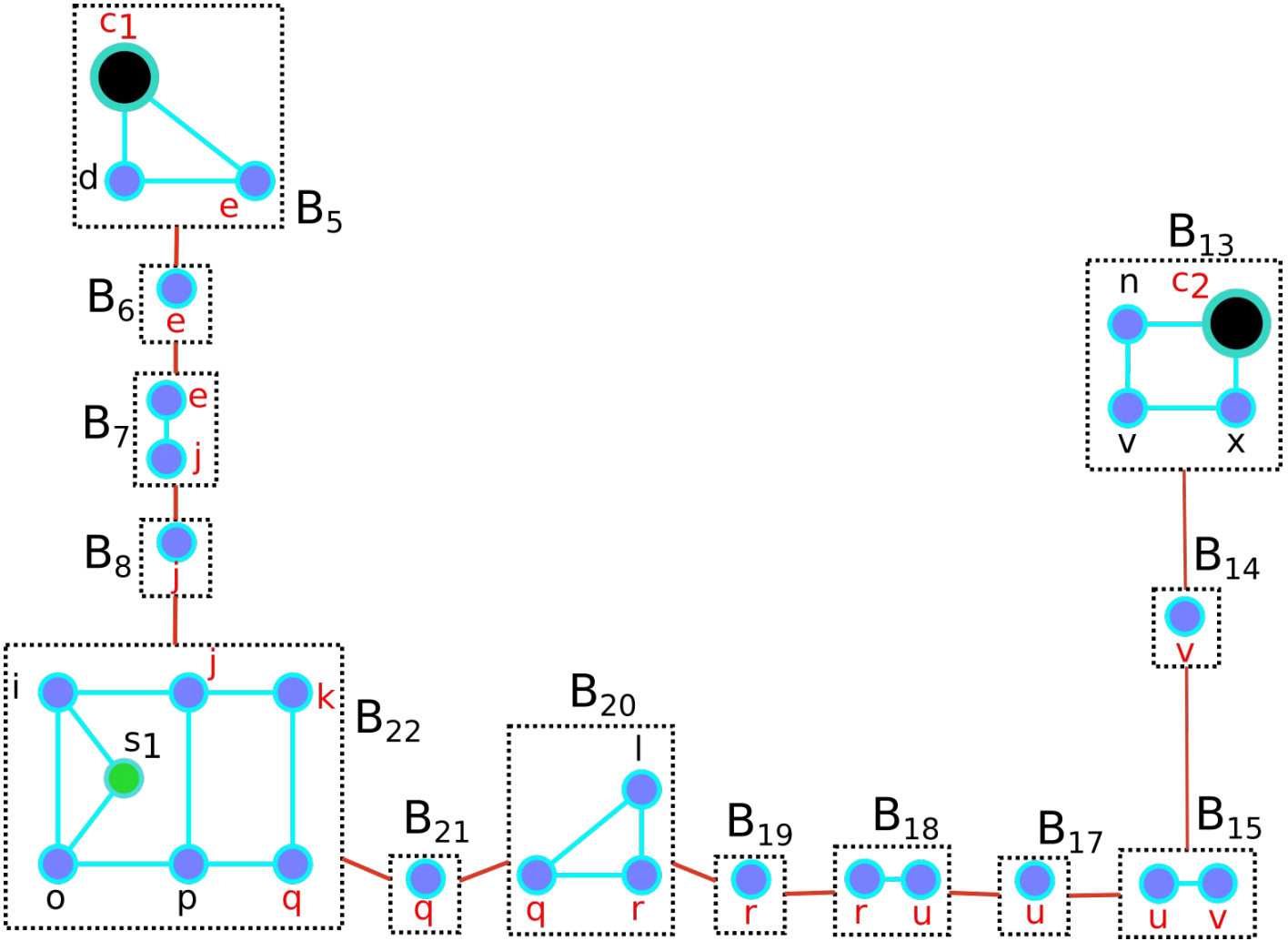
The algorithm

- Now, any vertex of the BC-Tree is in a path between a vertex with a controller and one with a starting point.
- **Special case**: *a* in B_3 won't be in the output → remove leaves where the only important vertex is an articulation still in the tree.



The algorithm

- Now, any vertex of the BC-Tree is in a path between a vertex with a controller and one with a starting point.
- If a block is in a path between a controller and a starting point → all vertices/edges inside this block will be (biconnected comp. property)



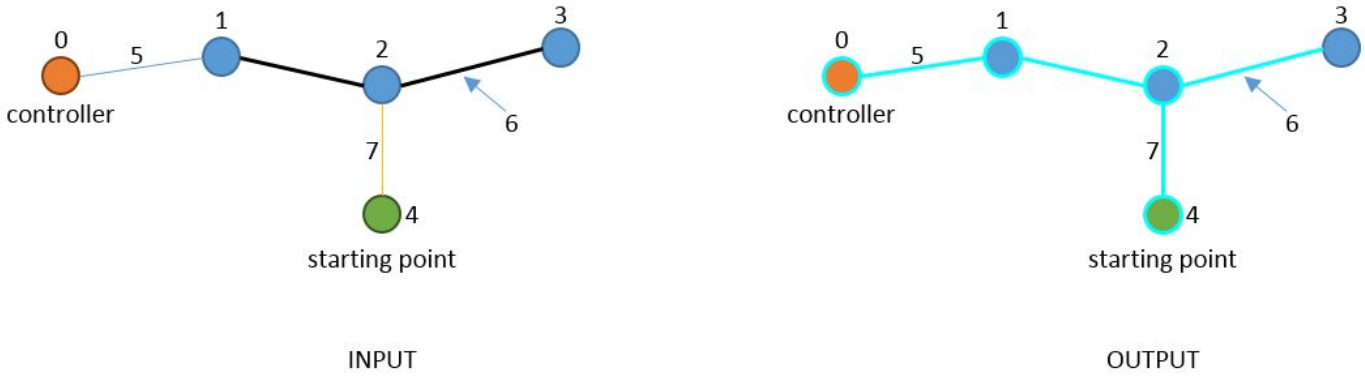
Summary of the algorithm

1. Find blocks and articulations (Tarjan's algorithm: O(V+E))
2. Create a Block-Cut tree (O(|blocks + articulations|) = O(V+E) (worst case))
3. Iteratively remove leaves w/o important vértices (O(|blocks + articulations|))
4. Output features in the remaining leaves (O(V+E))

Total complexity: O(V+E)

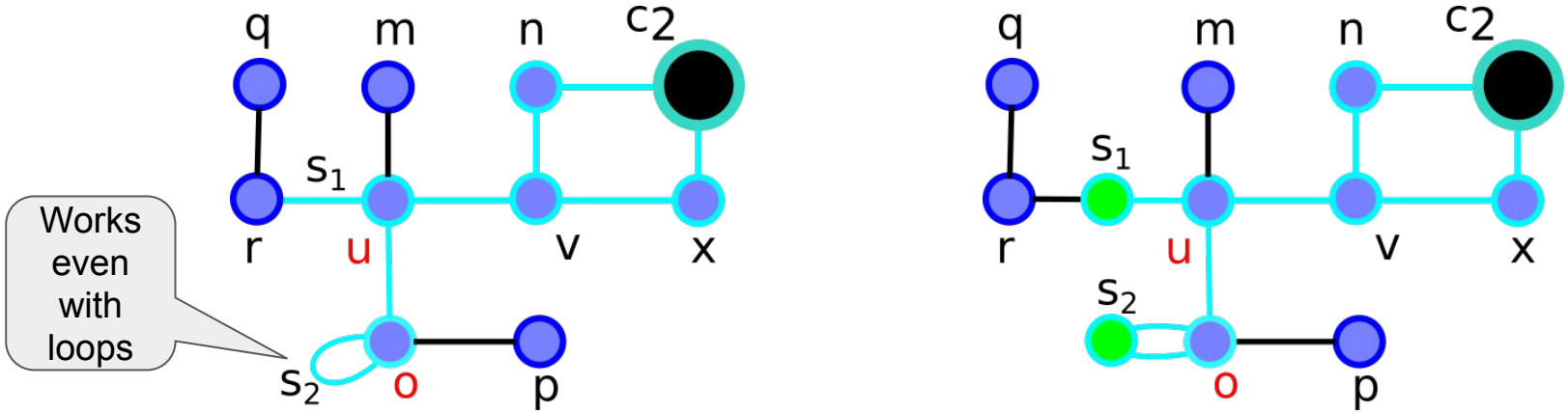
Edge starting points

- Edges may be starting points
- Example (from GISCUP mailing list): vertex 4 and edge 6 are starting points



Source: GISCUP 2018

- Instead of treating this as a special case in the algorithm → modify the input:
  - Starting point edge e=(u,v) → edges e=(u,s), e=(s,v) and s is a starting point vertex.



Implementation details

- According to the GISCUP website, solutions are ranked by efficiency (total elapsed wall clock time) and correctness.
- Several design choices carefully taken based on experiments with varying sized datasets.

- **Parser**: since the total time was evaluated, we created a fast custom parser.
  - Makes some **assumptions** about the input file.
  - According to the organizers, feature (vertex/edges) ids are always the format “{7FC28536-6F4A-4A9A-B439-1D87AE2D8871}” → **we encoded them as two 64-bit integers** → faster graph creation (which requires map lookups) at a small cost of encoding/decoding the ids.
  - Algorithm is loosely coupled → other parsers can be employed if a better (reusable) code is desired.

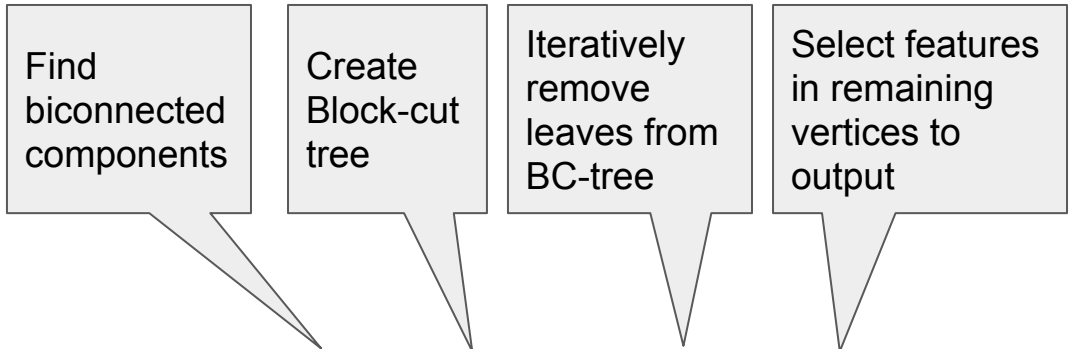
Implementation details

- **Tarjan**'s algorithm for computing biconnected components:
  - We implemented a custom version
  - Since entire graph has to be traversed:
    - Also finds connected components (CCs)
    - Ignores CCs without both controllers and starting points
    - Labels the biconnected component with information about the types of vertices in them (controllers, starting points, regular vertices)
  - Tarjan's algorithm is typically recursive. We implemented it **iteratively** to avoid stack overflows in big graphs.

- **Parallel programming**: we parallelized the main bottlenecks of the algorithm using OpenMP.
  - Graph (adjacency list) is represented using a ragged array and constructed in parallel.
  - Maps with the ids of the vértices (0,1,....V-1) and edges are created in parallel.
  - etc.

Experimental results

- 8-core Xeon Processor, 64 GB of RAM, Linux OS
- First dataset: GISCUP Esri's Naperville Electric Network Dataset
- Other ones: randomly created



	Number of				Time (s)						
Dataset	Vertices	Edges	Biconnected Components	Output features	Graph creation	Tarjan	BC-tree	Remove leaves	Select output	Total	Total w/ IO
EsriNaperville	8465	8302	7859	34	0.001	0.000	0.001	0.000	0.000	0.002	0.016
Graph1	1M	16M	1	17000000	0.780	0.565	0.013	0.000	0.292	1.650	6.368
Graph2	3M	3.3M	1866755	2566493	0.221	0.599	0.209	0.062	0.057	1.148	2.233
Graph3	3M	5M	435857	7128288	0.303	0.769	0.105	0.011	0.091	1.279	2.842
Graph4	5M	8M	843737	11312528	1.227	1.603	0.276	0.023	0.193	3.322	6.160
Graph5	13M	16M	5414204	18171594	0.939	3.047	0.842	0.163	0.449	5.440	10.703
Graph6	16M	16M	15999979	39	1.020	2.937	1.216	0.536	0.117	5.826	10.882
Graph6_100	16M	16M	15999979	1142	1.028	2.939	1.445	0.533	0.112	6.057	11.065
Graph6_1M	16M	16M	16500369	7336103	1.064	3.055	2.658	0.444	0.753	7.974	15.049

Experimental results

- 8-core Xeon Processor, 64 GB of RAM, Linux OS
- First dataset: GISCUP Esri's Naperville Electric Network Dataset
- Other ones: randomly created

- Bottleneck: I/O and Tarjan's algorithm (DFS, data dependency, non parallelizable)
- Other steps are more parallelizable:
  - 5x speedup on Graph Creation (largest test case)
  - 2x speedup on the creation of the BC-Tree (largest test case)

	Number of				Time (s)						
Dataset	Vertices	Edges	Biconnected Components	Output features	Graph creation	Tarjan	BC-tree	Remove leaves	Select output	Total	Total w/ IO
EsriNaperville	8465	8302	7859	34	0.001	0.000	0.001	0.000	0.000	0.002	0.016
Graph1	1M	16M	1	17000000	0.780	0.565	0.013	0.000	0.292	1.650	6.368
Graph2	3M	3.3M	1866755	2566493	0.221	0.599	0.209	0.062	0.057	1.148	2.233
Graph3	3M	5M	435857	7128288	0.303	0.769	0.105	0.011	0.091	1.279	2.842
Graph4	5M	8M	843737	11312528	1.227	1.603	0.276	0.023	0.193	3.322	6.160
Graph5	13M	16M	5414204	18171594	0.939	3.047	0.842	0.163	0.449	5.440	10.703
Graph6	16M	16M	15999979	39	1.020	2.937	1.216	0.536	0.117	5.826	10.882
Graph6_100	16M	16M	15999979	1142	1.028	2.939	1.445	0.533	0.112	6.057	11.065
Graph6_1M	16M	16M	16500369	7336103	1.064	3.055	2.658	0.444	0.753	7.974	15.049

Experimental results

- 8-core Xeon Processor, 64 GB of RAM, Linux OS
- First dataset: GISCUP Esri's Naperville Electric Network Dataset
- Other ones: randomly created

- Little variation on graphs with different amounts of starting points/controllers.
  - Detached: 1 starting point, 1 controller

	Number of				Time (s)						
Dataset	Vertices	Edges	Biconnected Components	Output features	Graph creation	Tarjan	BC-tree	Remove leaves	Select output	Total	Total w/ IO
EsriNaperville	8465	8302	7859	34	0.001	0.000	0.001	0.000	0.000	0.002	0.016
Graph1	1M	16M	1	17000000	0.780	0.565	0.013	0.000	0.292	1.650	6.368
Graph2	3M	3.3M	1866755	2566493	0.221	0.599	0.209	0.062	0.057	1.148	2.233
Graph3	3M	5M	435857	7128288	0.303	0.769	0.105	0.011	0.091	1.279	2.842
Graph4	5M	8M	843737	11312528	1.227	1.603	0.276	0.023	0.193	3.322	6.160
Graph5	13M	16M	5414204	18171594	0.939	3.047	0.842	0.163	0.449	5.440	10.703
Graph6	16M	16M	15999979	39	1.020	2.937	1.216	0.536	0.117	5.826	10.882
Graph6_100	16M	16M	15999979	1142	1.028	2.939	1.445	0.533	0.112	6.057	11.065
Graph6_1M	16M	16M	16500369	7336103	1.064	3.055	2.658	0.444	0.753	7.974	15.049

Experimental results

- 8-core Xeon Processor, 64 GB of RAM, Linux OS
- First dataset: GISCUP Esri's Naperville Electric Network Dataset
- Other ones: randomly created

- Little variation on graphs with different amounts of starting points/controllers.
  - Detached: 100 starting points 100 controllers

	Number of				Time (s)						
Dataset	Vertices	Edges	Biconnected Components	Output features	Graph creation	Tarjan	BC-tree	Remove leaves	Select output	Total	Total w/ IO
EsriNaperville	8465	8302	7859	34	0.001	0.000	0.001	0.000	0.000	0.002	0.016
Graph1	1M	16M	1	17000000	0.780	0.565	0.013	0.000	0.292	1.650	6.368
Graph2	3M	3.3M	1866755	2566493	0.221	0.599	0.209	0.062	0.057	1.148	2.233
Graph3	3M	5M	435857	7128288	0.303	0.769	0.105	0.011	0.091	1.279	2.842
Graph4	5M	8M	843737	11312528	1.227	1.603	0.276	0.023	0.193	3.322	6.160
Graph5	13M	16M	5414204	18171594	0.939	3.047	0.842	0.163	0.449	5.440	10.703
Graph6	16M	16M	15999979	39	1.020	2.937	1.216	0.536	0.117	5.826	10.882
Graph6_100	16M	16M	15999979	1142	1.028	2.939	1.445	0.533	0.112	6.057	11.065
Graph6_1M	16M	16M	16500369	7336103	1.064	3.055	2.658	0.444	0.753	7.974	15.049

Experimental results

- 8-core Xeon Processor, 64 GB of RAM, Linux OS
- First dataset: GISCUP Esri's Naperville Electric Network Dataset
- Other ones: randomly created

- Little variation on graphs with different amounts of starting points/controllers.
  - Detached: 1M starting points, 1M controllers
  - 37% slower than the dataset with 1 controller (same network)

	Number of				Time (s)						
Dataset	Vertices	Edges	Biconnected Components	Output features	Graph creation	Tarjan	BC-tree	Remove leaves	Select output	Total	Total w/ IO
EsriNaperville	8465	8302	7859	34	0.001	0.000	0.001	0.000	0.000	0.002	0.016
Graph1	1M	16M	1	17000000	0.780	0.565	0.013	0.000	0.292	1.650	6.368
Graph2	3M	3.3M	1866755	2566493	0.221	0.599	0.209	0.062	0.057	1.148	2.233
Graph3	3M	5M	435857	7128288	0.303	0.769	0.105	0.011	0.091	1.279	2.842
Graph4	5M	8M	843737	11312528	1.227	1.603	0.276	0.023	0.193	3.322	6.160
Graph5	13M	16M	5414204	18171594	0.939	3.047	0.842	0.163	0.449	5.440	10.703
Graph6	16M	16M	15999979	39	1.020	2.937	1.216	0.536	0.117	5.826	10.882
Graph6_100	16M	16M	15999979	1142	1.028	2.939	1.445	0.533	0.112	6.057	11.065
Graph6_1M	16M	16M	16500369	7336103	1.064	3.055	2.658	0.444	0.753	7.974	15.049

Conclusions

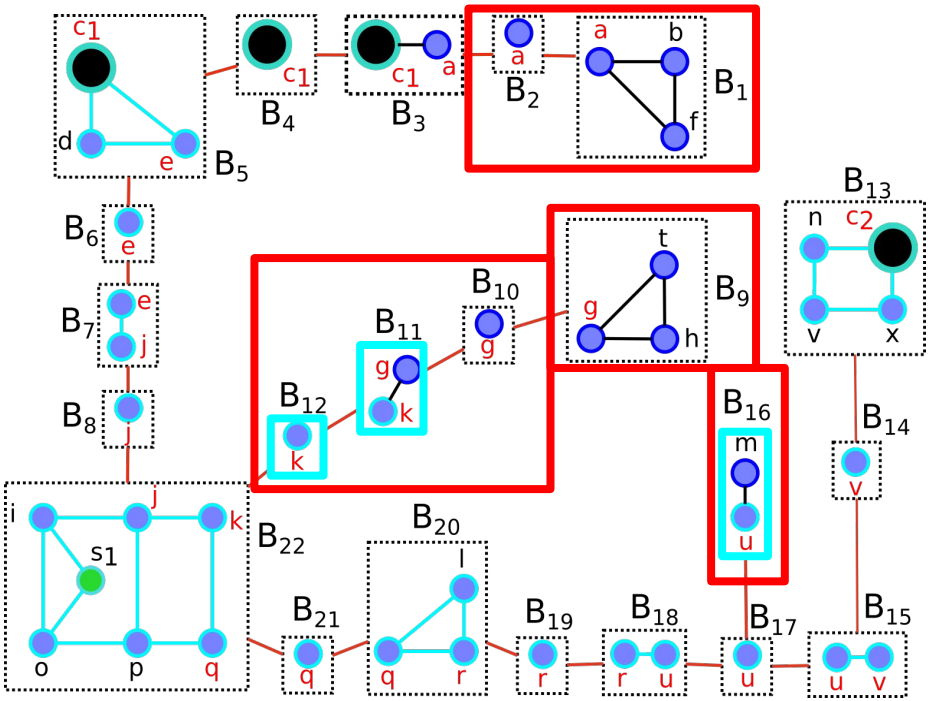
- Linear-time algorithm to find upstream features in utility networks
- Implemented in parallel
- (asymptotic) Time independent of the number of starting points/controllers

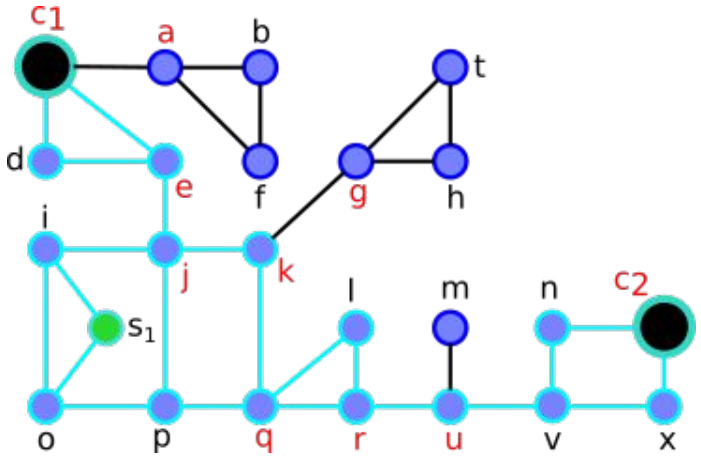
More broadly:

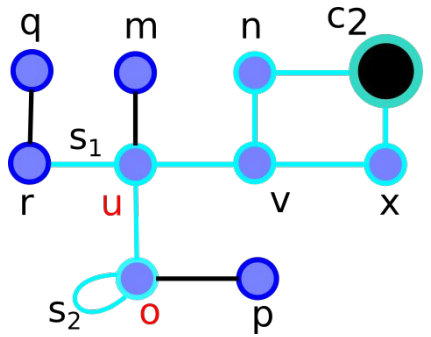
We could process a billion element (a full continent) dataset in under 10 minutes on a $5000 workstation. We would not need

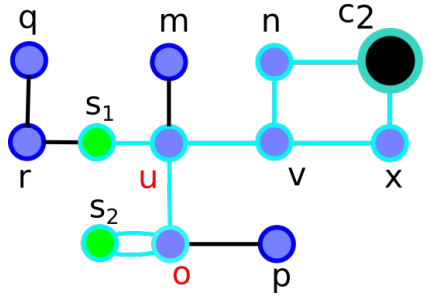
- ~~Cloud computing Hadoop Spark~~
- ~~Supercomputing~~
- ~~MPI~~

Thank you









Acknowledgement:

CAPES

CNPq

FAPEMIG