Fast Algorithms for Connectivity Problems in Networks

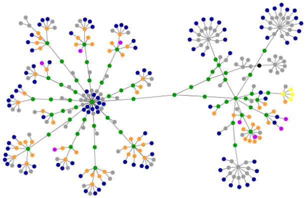
Steiner Cuts, Gomory-Hu Trees, and Edge Splitting

Ramesh Hariharan
Strand Life Sciences

29 May 2008/ MCDES

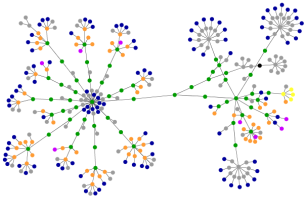
The Setting

- A network G of n nodes and m edges.
- Many nodes, relatively fewer connections.
- Unweighted edges.
- Redundancy requirement (k) much smaller than n.
- Reduce algorithms with time complexity $O(n^2)$ or more to $\tilde{O}(n \text{ poly}(k))$



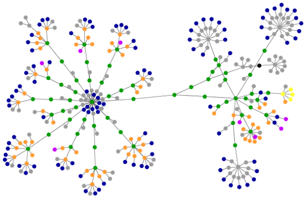
The Setting

- A network G of n nodes and m edges.
- Many nodes, relatively fewer connections.
- Unweighted edges.
- Redundancy requirement (k) much smaller than n.
- Reduce algorithms with time complexity $O(n^2)$ or more to $\tilde{O}(n\,\mathrm{poly}(k))$



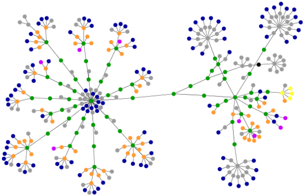
The Setting

- A network G of n nodes and m edges.
- Many nodes, relatively fewer connections.
- Unweighted edges.
- Redundancy requirement (k) much smaller than n.
- Reduce algorithms with time complexity $O(n^2)$ or more to $\tilde{O}(n \text{ poly}(k))$



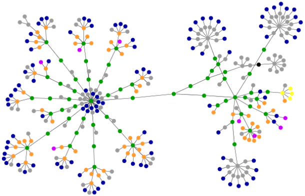
The Setting

- A network G of n nodes and m edges.
- Many nodes, relatively fewer connections.
- Unweighted edges.
- Redundancy requirement (k) much smaller than n.
- Reduce algorithms with time complexity $O(n^2)$ or more to $\tilde{O}(n \text{ poly}(k))$



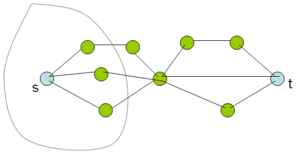
The Setting

- A network G of n nodes and m edges.
- Many nodes, relatively fewer connections.
- Unweighted edges.
- Redundancy requirement (k) much smaller than n.
- Reduce algorithms with time complexity $O(n^2)$ or more to $\tilde{O}(n \text{ poly}(k))$



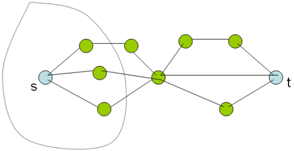
$s - t$ Edge Connectivity/Min-Cut

- The maximum number of edge disjoint paths between two vertices.
- Equals the min-cut separating the 2 vertices
- Can be computed in $O(mk)$ time via Ford-Fulkerson max flows, k is the min-cut size



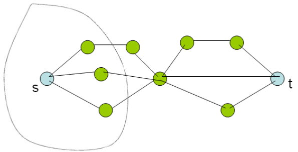
$s - t$ Edge Connectivity/Min-Cut

- The maximum number of edge disjoint paths between two vertices.
- Equals the min-cut separating the 2 vertices
- Can be computed in $O(mk)$ time via Ford-Fulkerson max flows, k is the min-cut size



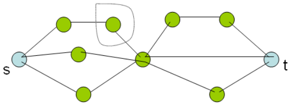
$s - t$ Edge Connectivity/Min-Cut

- The maximum number of edge disjoint paths between two vertices.
- Equals the min-cut separating the 2 vertices
- Can be computed in $O(mk)$ time via Ford-Fulkerson max flows, k is the min-cut size



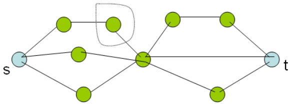
Global Edge Connectivity/Min-Cut

- The minimum edge connectivity over all vertex pairs.
- A global min-cut
- Best Deterministic Algorithm: $O(mk)$ by Gabow where k is the global min-cut size; near linear Monte Carlo algorithm by Karger.



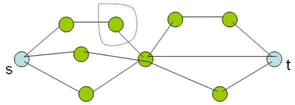
Global Edge Connectivity/Min-Cut

- The minimum edge connectivity over all vertex pairs.
- A global min-cut
- Best Deterministic Algorithm: $O(mk)$ by Gabow where k is the global min-cut size; near linear Monte Carlo algorithm by Karger.



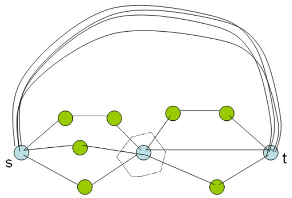
Global Edge Connectivity/Min-Cut

- The minimum edge connectivity over all vertex pairs.
- A global min-cut
- Best Deterministic Algorithm: $O(mk)$ by Gabow where k is the global min-cut size; near linear Monte Carlo algorithm by Karger.



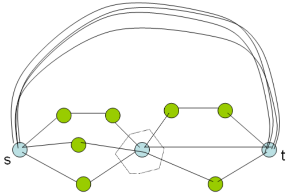
Steiner Edge Connectivity/Min-Cut

- S is a subset of interesting Steiner vertices.
- The minimum edge connectivity over all vertex pairs from S.
- The min-cut separating vertices in S
- Question: Is an $O(mk)$ algorithm possible, where k is the Steiner min-cut?



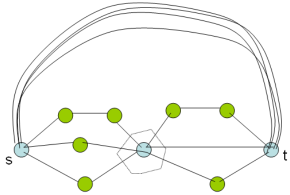
Steiner Edge Connectivity/Min-Cut

- S is a subset of interesting Steiner vertices.
- The minimum edge connectivity over all vertex pairs from S.
- The min-cut separating vertices in S
- Question: Is an $O(mk)$ algorithm possible, where k is the Steiner min-cut?



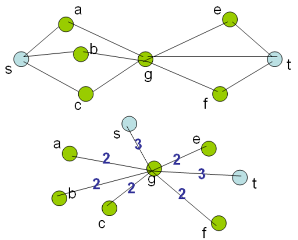
Steiner Edge Connectivity/Min-Cut

- S is a subset of interesting Steiner vertices.
- The minimum edge connectivity over all vertex pairs from S.
- The min-cut separating vertices in S
- Question: Is an $O(mk)$ algorithm possible, where k is the Steiner min-cut?



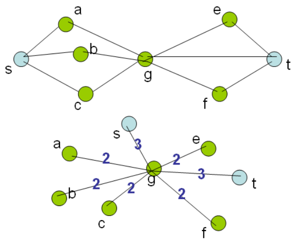
The Gomory-Hu Tree

- A tree which carries min-cut information for all pairs of vertices.
- $n-1$ max-flow/min-cut computations suffice.
- Time taken is $O(n^3)$ or more.
- Question: Is $\tilde{O}(nm)$ time possible?



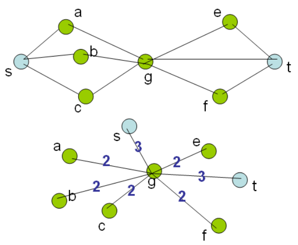
The Gomory-Hu Tree

- A tree which carries min-cut information for all pairs of vertices.
- $n - 1$ max-flow/min-cut computations suffice.
- Time taken is $O(n^3)$ or more.
- Question: Is $\tilde{O}(nm)$ time possible?



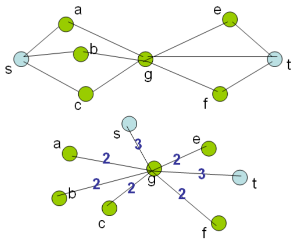
The Gomory-Hu Tree

- A tree which carries min-cut information for all pairs of vertices.
- $n - 1$ max-flow/min-cut computations suffice.
- Time taken is $O(n^3)$ or more.
- Question: Is $\tilde{O}(nm)$ time possible?



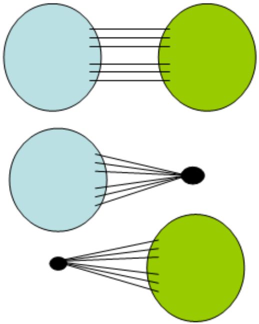
The Gomory-Hu Tree

- A tree which carries min-cut information for all pairs of vertices.
- $n - 1$ max-flow/min-cut computations suffice.
- Time taken is $O(n^3)$ or more.
- Question: Is $\tilde{O}(nm)$ time possible?



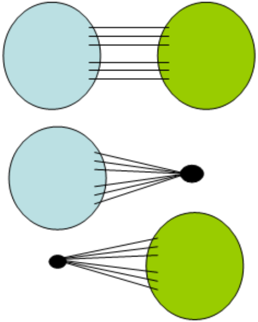
Gomory-Hu Trees & Steiner Min-Cuts

- Compute Min-Cut (say Global)
- Create two recursive sub-problems
- General Recursive Problem; Steiner Min-Cut



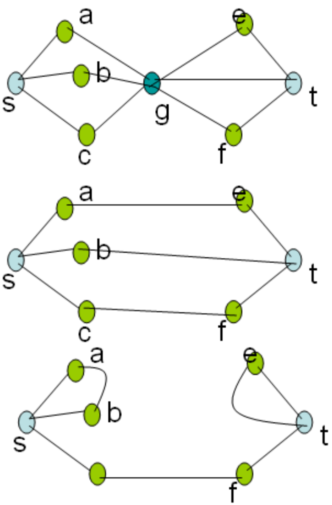
Gomory-Hu Trees & Steiner Min-Cuts

- Compute Min-Cut (say Global)
- Create two recursive sub-problems
- General Recursive Problem; Steiner Min-Cut



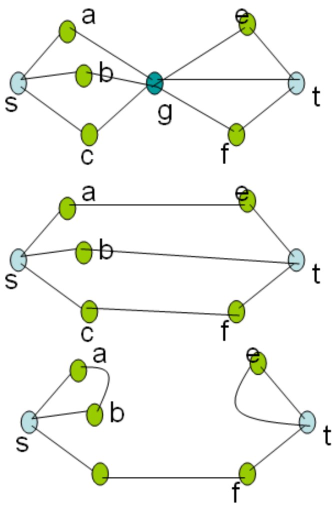
Edge Splitting

- Pair edges incident on a vertex and remove it
- Ensure that pairing conserves the global min-cut of the remaining vertices
- Is this possible at all?
- Lovasz/Mader: Yes, for eulerian directed graphs and for even degree vertices in undirected graphs



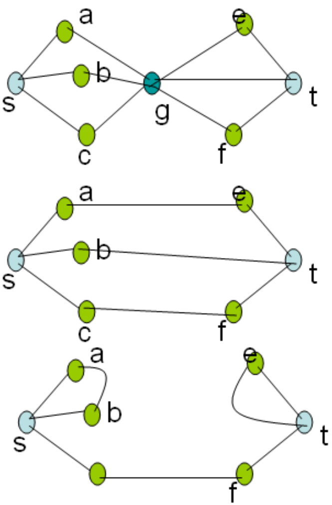
Edge Splitting

- Pair edges incident on a vertex and remove it
- Ensure that pairing conserves the global min-cut of the remaining vertices
- Is this possible at all?
- Lovasz/Mader: Yes, for eulerian directed graphs and for even degree vertices in undirected graphs



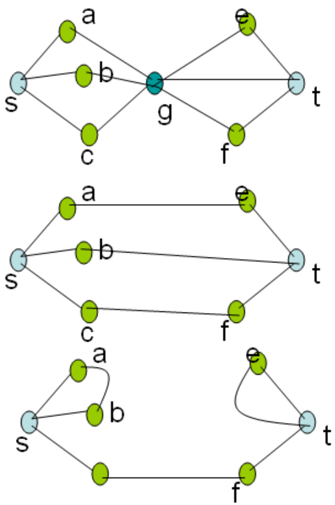
Edge Splitting

- Pair edges incident on a vertex and remove it
- Ensure that pairing conserves the global min-cut of the remaining vertices
- Is this possible at all?
- Lovasz/Mader: Yes, for eulerian directed graphs and for even degree vertices in undirected graphs



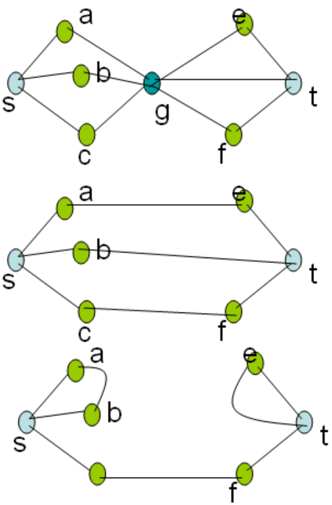
Edge Splitting

- Pair edges incident on a vertex and remove it
- Ensure that pairing conserves the global min-cut of the remaining vertices
- Is this possible at all?
- Lovasz/Mader: Yes, for eulerian directed graphs and for even degree vertices in undirected graphs



Edge Splitting

- Pair edges incident on a vertex and remove it
- Ensure that pairing conserves the global min-cut of the remaining vertices
- Is this possible at all?
- Lovasz/Mader: Yes, for eulerian directed graphs and for even degree vertices in undirected graphs



Edge Splitting and Steiner Min-Cuts

- Split off all non-Steiner vertices
- Compute global min-cut for what remains in $\tilde{O}(mk)$ time.

Edge Splitting and Steiner Min-Cuts

- Split off all non-Steiner vertices
- Compute global min-cut for what remains in $\tilde{O}(mk)$ time.

Edge Splitting Complexity

- How fast can one vertex be split off? $O(n^2)$ time Karger, Benczur and $\tilde{O}(nk^2)$ time Gabow
- How fast can many vertices be split off? Nothing better than plain multiplication known.
- Is $\tilde{O}(n \, \mathrm{poly} k)$ possible?

Edge Splitting Complexity

- How fast can one vertex be split off? $O(n^2)$ time Karger, Benczur and $\tilde{O}(nk^2)$ time Gabow
- How fast can many vertices be split off? Nothing better than plain multiplication known.
- Is $\tilde{O}(n \operatorname{poly} k)$ possible?

Edge Splitting Complexity

- How fast can one vertex be split off? $O(n^2)$ time Karger, Benczur and $\tilde{O}(nk^2)$ time Gabow
- How fast can many vertices be split off? Nothing better than plain multiplication known.
- Is $\tilde{O}(n \operatorname{poly} k)$ possible?

Certifying $s - t$ and global min-cuts

- How can one certify a min-cut?
- $s - t$ min-cut by finding edge-disjoint paths.
- Global min-cut by Edmonds' Arborescences

Certifying $s - t$ and global min-cuts

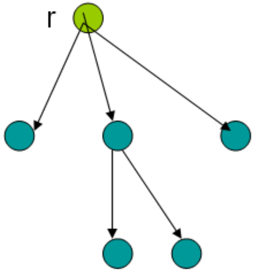
- How can one certify a min-cut?
- $s - t$ min-cut by finding edge-disjoint paths.
- Global min-cut by Edmonds' Arborescences

Certifying $s - t$ and global min-cuts

- How can one certify a min-cut?
- $s - t$ min-cut by finding edge-disjoint paths.
- Global min-cut by Edmonds' Arborescences

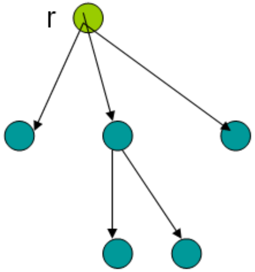
Edmonds' Arborescences

- Direct an undirected graph by orienting each edge in both directions.
- Arborescence: A spanning tree with edges directed away from an arbitrary root
- Build as many edge-disjoint arborescences as possible rooted at some chosen vertex r
- $\tilde{O}(n^2)$ time construction by Gabow.



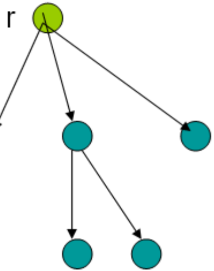
Edmonds' Arborescences

- Direct an undirected graph by orienting each edge in both directions.
- Arborescence: A spanning tree with edges directed away from an arbitrary root
- Build as many edge-disjoint arborescences as possible rooted at some chosen vertex *r*
- $\tilde{O}(n^2)$ time construction by Gabow.



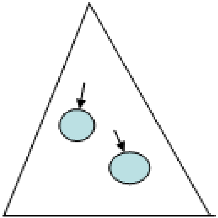
Edmonds' Arborescences

- Direct an undirected graph by orienting each edge in both directions.
- Arborescence: A spanning tree with edges directed away from an arbitrary root
- Build as many edge-disjoint arborescences as possible rooted at some chosen vertex r
- $\tilde{O}(n^2)$ time construction by Gabow.



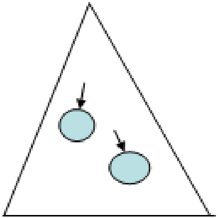
Global Min-Cut Characterizations from Edmonds' Arborescences

- Any subset of vertices not containing the root must have in-degree at least 1 in each tree.
- Existence of k edge-disjoint arborescences implies global min-cut $\geq k$



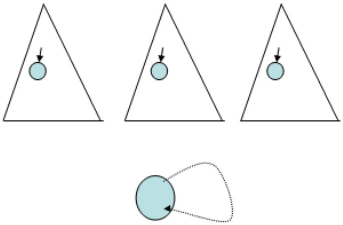
Global Min-Cut Characterizations from Edmonds' Arborescences

- Any subset of vertices not containing the root must have in-degree at least 1 in each tree.
- Existence of k edge-disjoint arborescences implies global min-cut $\geq k$



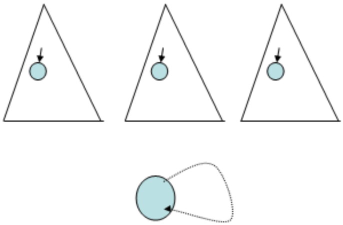
Global Min-Cut Characterizations from Edmonds' Arborescences

- A subset S of vertices s.t. $r \notin S$, S is contiguous in all trees, and all unused edges directed into S have both endpoints in S, is a global min-cut
- Gabow's construction shows that such a set exists when no more edge-disjoint arborescences can be conctructed.



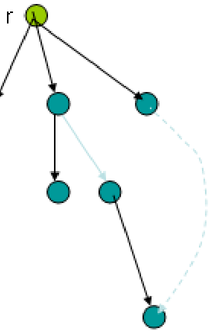
Global Min-Cut Characterizations from Edmonds' Arborescences

- A subset S of vertices s.t. $r \notin S$, S is contiguous in all trees, and all unused edges directed into S have both endpoints in S, is a global min-cut
- Gabow's construction shows that such a set exists when no more edge-disjoint arborescences can be conctructed.



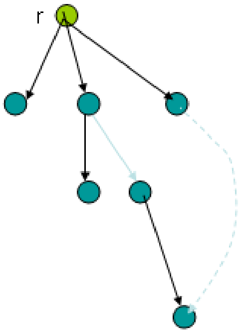
Edmonds' Directionless Trees

- Swaps need to be constrained to maintain the arborescence property.
- Relax the property that edges are directed away from the root. Instead insist that in-degree of a vertex over all directionless trees equals k
- Existence of k edge-disjoint directionless trees implies global min-cut $\geq k$
- $\tilde{O}(mk)$ time construction by Gabow.



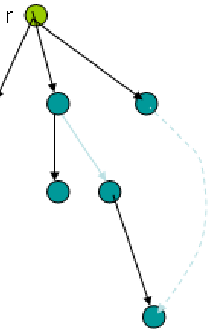
Edmonds' Directionless Trees

- Swaps need to be constrained to maintain the arborescence property.
- Relax the property that edges are directed away from the root. Instead insist that in-degree of a vertex over all directionless trees equals k
- Existence of k edge-disjoint directionless trees implies global min-cut $\geq k$
- $\tilde{O}(mk)$ time construction by Gabow.



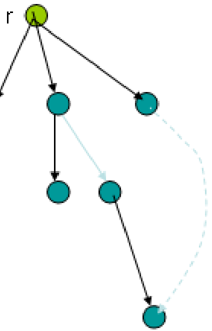
Edmonds' Directionless Trees

- Swaps need to be constrained to maintain the arborescence property.
- Relax the property that edges are directed away from the root. Instead insist that in-degree of a vertex over all directionless trees equals k
- Existence of k edge-disjoint directionless trees implies global min-cut $\geq k$
- $\tilde{O}(mk)$ time construction by Gabow.



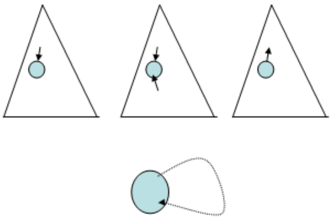
Edmonds' Directionless Trees

- Swaps need to be constrained to maintain the arborescence property.
- Relax the property that edges are directed away from the root. Instead insist that in-degree of a vertex over all directionless trees equals k
- Existence of k edge-disjoint directionless trees implies global min-cut $\geq k$
- $\tilde{O}(mk)$ time construction by Gabow.



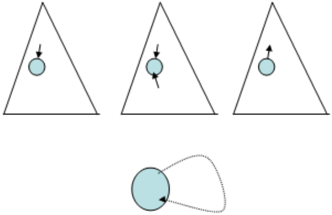
Global Min-Cut Characterizations from Edmonds' Directionless Trees

- A subset S of vertices s.t. $r \notin S$, S is contiguous in all trees, and all unused edges directed into S having both endpoints in S, is a global min-cut
- Gabow's construction shows that such a set exists when no more edge-disjoint directionless trees can be constructed.



Global Min-Cut Characterizations from Edmonds' Directionless Trees

- A subset S of vertices s.t. $r \notin S$, S is contiguous in all trees, and all unused edges directed into S having both endpoints in S, is a global min-cut
- Gabow's construction shows that such a set exists when no more edge-disjoint directionless trees can be constructed.



Edmonds' Directionless Trees Construction Algorithm

- Suppose i trees have been constructed and the $i+1$th tree is a forest.
- For each component in the $i+1$th tree, run a closure algorithm to identify a subset S of vertices not containing the root, contiguous in all trees, and with all unused edges directed into S having both endpoints in S.
- If no such subset can be found then a sequence of swaps results in components in the $i+1$th tree connecting up.
- $\tilde{O}(m)$ time for the $i+1$th tree so $\tilde{O}(mk)$ overall; and m can be made $O(nk)$ via Nagamochi-Ibaraki

Edmonds' Directionless Trees Construction Algorithm

- Suppose i trees have been constructed and the $i+1$th tree is a forest.
- For each component in the $i+1$th tree, run a closure algorithm to identify a subset S of vertices not containing the root, contiguous in all trees, and with all unused edges directed into S having both endpoints in S.
- If no such subset can be found then a sequence of swaps results in components in the $i+1$th tree connecting up.
- $\tilde{O}(m)$ time for the $i+1$th tree so $\tilde{O}(mk)$ overall; and m can be made $O(nk)$ via Nagamochi-Ibaraki

Edmonds' Directionless Trees Construction Algorithm

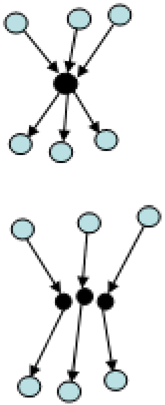
- Suppose i trees have been constructed and the $i+1$th tree is a forest.
- For each component in the $i+1$th tree, run a closure algorithm to identify a subset S of vertices not containing the root, contiguous in all trees, and with all unused edges directed into S having both endpoints in S.
- If no such subset can be found then a sequence of swaps results in components in the $i+1$th tree connecting up.
- $\tilde{O}(m)$ time for the $i+1$th tree so $\tilde{O}(mk)$ overall; and m can be made $O(nk)$ via Nagamochi-Ibaraki

Edmonds' Directionless Trees Construction Algorithm

- Suppose i trees have been constructed and the $i+1$th tree is a forest.
- For each component in the $i+1$th tree, run a closure algorithm to identify a subset S of vertices not containing the root, contiguous in all trees, and with all unused edges directed into S having both endpoints in S.
- If no such subset can be found then a sequence of swaps results in components in the $i+1$th tree connecting up.
- $\tilde{O}(m)$ time for the $i+1$th tree so $\tilde{O}(mk)$ overall; and m can be made $O(nk)$ via Nagamochi-Ibaraki

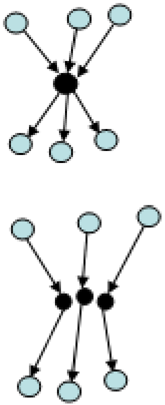
Steiner Min-Cut Algorithm

- Split-off non-Steiner (black) vertices via arbitrary pairing of edges (the global min-cut of the remaining white Steiner vertices could drop in the process)
- Each edge is actually a path with internal blacks
- Run Gabow's construction of Edmonds' directionless trees on this new graph comprising only whites (ignore blacks)



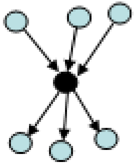
Steiner Min-Cut Algorithm

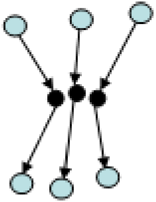
- Split-off non-Steiner (black) vertices via arbitrary pairing of edges (the global min-cut of the remaining white Steiner vertices could drop in the process)
- Each edge is actually a path with internal blacks
- Run Gabow's construction of Edmonds' directionless trees on this new graph comprising only whites (ignore blacks)



Steiner Min-Cut Algorithm

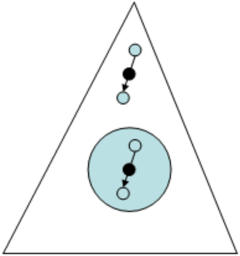
- Split-off non-Steiner (black) vertices via arbitrary pairing of edges (the global min-cut of the remaining white Steiner vertices could drop in the process)
- Each edge is actually a path with internal blacks
- Run Gabow's construction of Edmonds' directionless trees on this new graph comprising only whites (ignore blacks)





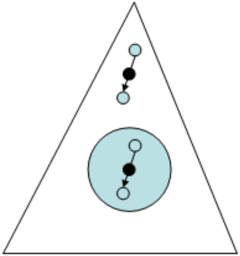
Steiner Min-Cut Algorithm

- Suppose this culminates in finding a set S of vertices not containing the root, contiguous in all trees, and with all unused edges directed into S having both endpoints in S;
- S need not be a Steiner min-cut because it need not be contiguous when considering blacks



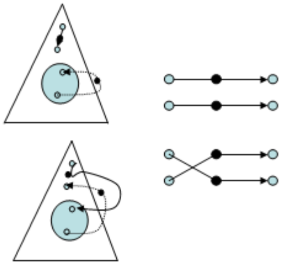
Steiner Min-Cut Algorithm

- Suppose this culminates in finding a set S of vertices not containing the root, contiguous in all trees, and with all unused edges directed into S having both endpoints in S;
- S need not be a Steiner min-cut because it need not be contiguous when considering blacks



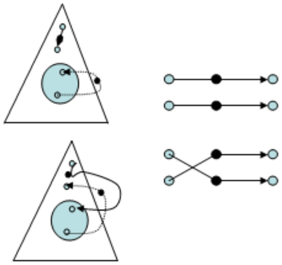
Steiner Min-Cut Algorithm Contd

- We introduce a mating operation (i.e., revision of pairing) whenever contiguity on blacks is violated
- This allows closure computation to continue
- And if black contiguity holds, then S is indeed a Steiner min-cut.
- And we show how mates can be performed efficiently so the total time stays $\tilde{O}(nk^2)$, where k is the Steiner min-cut
- Several technicalities.



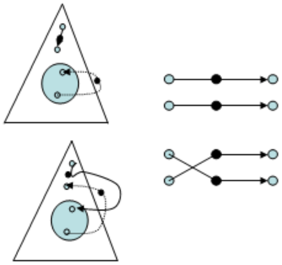
Steiner Min-Cut Algorithm Contd

- We introduce a mating operation (i.e., revision of pairing) whenever contiguity on blacks is violated
- This allows closure computation to continue
- And if black contiguity holds, then S is indeed a Steiner min-cut.
- And we show how mates can be performed efficiently so the total time stays $\tilde{O}(nk^2)$, where k is the Steiner min-cut
- Several technicalities.



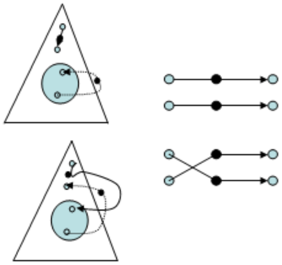
Steiner Min-Cut Algorithm Contd

- We introduce a mating operation (i.e., revision of pairing) whenever contiguity on blacks is violated
- This allows closure computation to continue
- And if black contiguity holds, then S is indeed a Steiner min-cut.
- And we show how mates can be performed efficiently so the total time stays $\tilde{O}(nk^2)$, where k is the Steiner min-cut
- Several technicalities.



Steiner Min-Cut Algorithm Contd

- We introduce a mating operation (i.e., revision of pairing) whenever contiguity on blacks is violated
- This allows closure computation to continue
- And if black contiguity holds, then S is indeed a Steiner min-cut.
- And we show how mates can be performed efficiently so the total time stays $\tilde{O}(nk^2)$, where k is the Steiner min-cut
- Several technicalities.



Splitting-off Vertices in Undirected Graphs

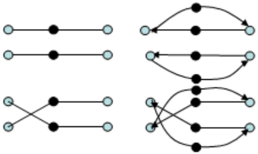
- Goal: Split-off a specified subset of vertices so global min-cut of the remaining vertices is preserved.
- Algorithm as above, split-off with arbitrary pairings and then revise pairing via mating

Splitting-off Vertices in Undirected Graphs

- Goal: Split-off a specified subset of vertices so global min-cut of the remaining vertices is preserved.
- Algorithm as above, split-off with arbitrary pairings and then revise pairing via mating

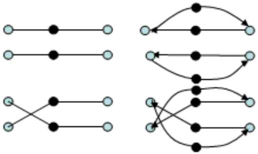
Splitting-off Vertices in Undirected Graphs

- Complication: this is done on the directed version of the undirected graph; for true undirected splitting-off whenever two directed edges are paired, there reverse edges must be paired as well
- Coupled Matings required and can be done in $\tilde{O}(nk^2)$ time.



Splitting-off Vertices in Undirected Graphs

- Complication: this is done on the directed version of the undirected graph; for true undirected splitting-off whenever two directed edges are paired, there reverse edges must be paired as well
- Coupled Matings required and can be done in $\tilde{O}(nk^2)$ time.



Faster Edmonds' Arborescence Construction

- Goal: Construct k arborescences where k is the global min-cut
- Algo: Split-off a constant fraction of the vertices, recursively build arborescences for the rest, then put back the split-off vertices
- Putting back requires that the vertices split-off are independent
- By Turan's theorem, there is a n/k sized independent set
- So time taken is $\tilde{O}(nk^3)$.

Faster Edmonds' Arborescence Construction

- Goal: Construct k arborescences where k is the global min-cut
- Algo: Split-off a constant fraction of the vertices, recursively build arborescences for the rest, then put back the split-off vertices
- Putting back requires that the vertices split-off are independent
- By Turan's theorem, there is a n/k sized independent set
- So time taken is $\tilde{O}(nk^3)$.

Faster Edmonds' Arborescence Construction

- Goal: Construct k arborescences where k is the global min-cut
- Algo: Split-off a constant fraction of the vertices, recursively build arborescences for the rest, then put back the split-off vertices
- Putting back requires that the vertices split-off are independent
- By Turan's theorem, there is a n/k sized independent set
- So time taken is $\tilde{O}(nk^3)$.

Faster Edmonds' Arborescence Construction

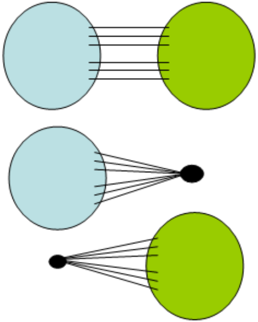
- Goal: Construct k arborescences where k is the global min-cut
- Algo: Split-off a constant fraction of the vertices, recursively build arborescences for the rest, then put back the split-off vertices
- Putting back requires that the vertices split-off are independent
- By Turan's theorem, there is a n/k sized independent set
- So time taken is $\tilde{O}(nk^3)$.

Faster Edmonds' Arborescence Construction

- Goal: Construct k arborescences where k is the global min-cut
- Algo: Split-off a constant fraction of the vertices, recursively build arborescences for the rest, then put back the split-off vertices
- Putting back requires that the vertices split-off are independent
- By Turan's theorem, there is a n/k sized independent set
- So time taken is $\tilde{O}(nk^3)$.

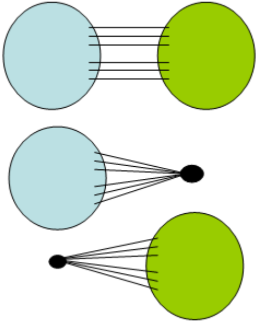
Gomory-Hu Tree Construction

- Find global min-cut
- Create two Steiner min-cut sub-problems
- Consider computation tree



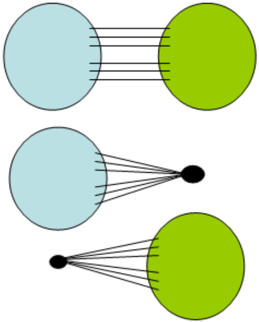
Gomory-Hu Tree Construction

- Find global min-cut
- Create two Steiner min-cut sub-problems
- Consider computation tree



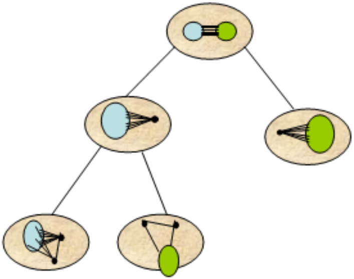
Gomory-Hu Tree Construction

- Find global min-cut
- Create two Steiner min-cut sub-problems
- Consider computation tree



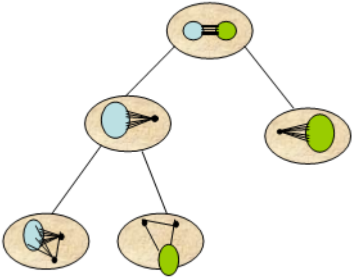
Gomory-Hu Tree Construction

- Consider a leftgoing path in the computation tree
- New small cuts are identified
- These are shrunk into single black vertices and split-off
- So split vertices as they are discovered; vertices to be split are not available in advance



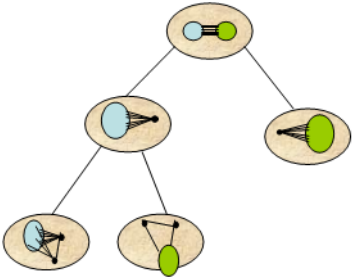
Gomory-Hu Tree Construction

- Consider a leftgoing path in the computation tree
- New small cuts are identified
- These are shrunk into single black vertices and split-off
- So split vertices as they are discovered; vertices to be split are not available in advance



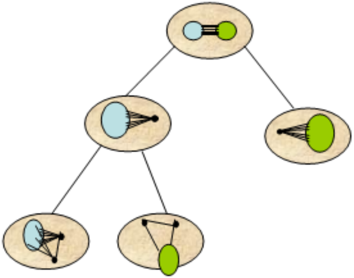
Gomory-Hu Tree Construction

- Consider a leftgoing path in the computation tree
- New small cuts are identified
- These are shrunk into single black vertices and split-off
- So split vertices as they are discovered; vertices to be split are not available in advance



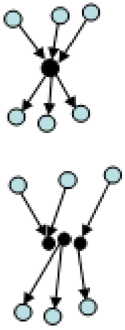
Gomory-Hu Tree Construction

- Consider a leftgoing path in the computation tree
- New small cuts are identified
- These are shrunk into single black vertices and split-off
- So split vertices as they are discovered; vertices to be split are not available in advance



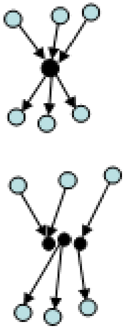
Gomory-Hu Tree Construction

- When a new small cut (black vertex) is discovered, edges incident on it must be paired up for splitting-off
- Doing so may disconnect existing trees
- Solution: Directionless Splitting
- So one leftgoing path takes $O(mn)$ time



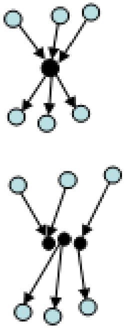
Gomory-Hu Tree Construction

- When a new small cut (black vertex) is discovered, edges incident on it must be paired up for splitting-off
- Doing so may disconnect existing trees
- Solution: Directionless Splitting
- So one leftgoing path takes $O(mn)$ time



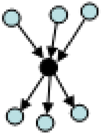
Gomory-Hu Tree Construction

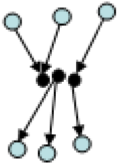
- When a new small cut (black vertex) is discovered, edges incident on it must be paired up for splitting-off
- Doing so may disconnect existing trees
- Solution: Directionless Splitting
- So one leftgoing path takes $O(mn)$ time



Gomory-Hu Tree Construction

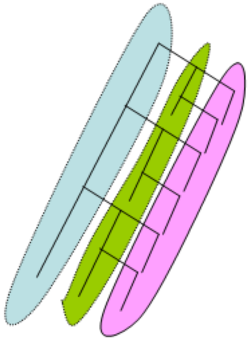
- When a new small cut (black vertex) is discovered, edges incident on it must be paired up for splitting-off
- Doing so may disconnect existing trees
- Solution: Directionless Splitting
- So one leftgoing path takes $O(mn)$ time





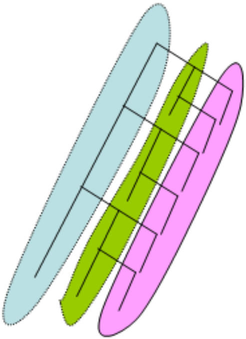
Gomory-Hu Tree Construction

- What is the time overall?
- Partition computation tree into layers
- The total number of edges in a layer is $O(m)$ (a bit tricky)
- How many layers are there?



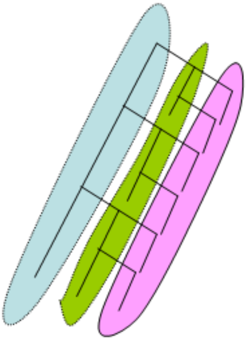
Gomory-Hu Tree Construction

- What is the time overall?
- Partition computation tree into layers
- The total number of edges in a layer is $O(m)$ (a bit tricky)
- How many layers are there?



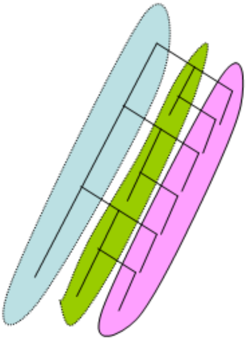
Gomory-Hu Tree Construction

- What is the time overall?
- Partition computation tree into layers
- The total number of edges in a layer is $O(m)$ (a bit tricky)
- How many layers are there?



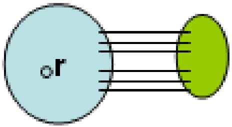
Gomory-Hu Tree Construction

- What is the time overall?
- Partition computation tree into layers
- The total number of edges in a layer is $O(m)$ (a bit tricky)
- How many layers are there?



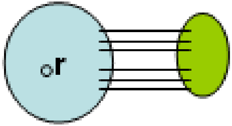
Gomory-Hu Trees Construction

- Choose the root for tree construction randomly
- $O(\log n)$ layers
- The Gomory-Hu tree can be constructed in $\tilde{O}(nm)$ time



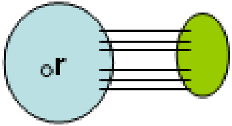
Gomory-Hu Trees Construction

- Choose the root for tree construction randomly
- $O(\log n)$ layers
- The Gomory-Hu tree can be constructed in $\tilde{O}(nm)$ time



Gomory-Hu Trees Construction

- Choose the root for tree construction randomly
- $O(\log n)$ layers
- The Gomory-Hu tree can be constructed in $\tilde{O}(nm)$ time



The Edge Orientation Problem

- Given a graph with global min-cut 2k
- Orient the edges so the resulting directed graph has global min-cut k
- Challenge: Odd degree vertices cannot be split-off time
- We show that a minimal graph has sufficiently many even degree vertices
- $\tilde{O}(npoly(k))$ time algorithm.

The Edge Orientation Problem

- Given a graph with global min-cut 2k
- Orient the edges so the resulting directed graph has global min-cut k
- Challenge: Odd degree vertices cannot be split-off time
- We show that a minimal graph has sufficiently many even degree vertices
- $\tilde{O}(npoly(k))$ time algorithm.

The Edge Orientation Problem

- Given a graph with global min-cut 2k
- Orient the edges so the resulting directed graph has global min-cut k
- Challenge: Odd degree vertices cannot be split-off time
- We show that a minimal graph has sufficiently many even degree vertices
- $\tilde{O}(npoly(k))$ time algorithm.

The Edge Orientation Problem

- Given a graph with global min-cut 2k
- Orient the edges so the resulting directed graph has global min-cut k
- Challenge: Odd degree vertices cannot be split-off time
- We show that a minimal graph has sufficiently many even degree vertices
- $\tilde{O}(npoly(k))$ time algorithm.

The Edge Orientation Problem

- Given a graph with global min-cut 2k
- Orient the edges so the resulting directed graph has global min-cut k
- Challenge: Odd degree vertices cannot be split-off time
- We show that a minimal graph has sufficiently many even degree vertices
- $\tilde{O}(npoly(k))$ time algorithm.

The Survivable Network Design Problem

- Given a weighted graph
- Each vertex v has a demand d; v must be connected to every other vertex with at least d edge-disjoint paths.
- Problem: Find the least weight solution
- The first sub-quadratic time implementation of the Williamson, Goemans, Mihail, Vazirani algorithm

The Survivable Network Design Problem

- Given a weighted graph
- Each vertex v has a demand d; v must be connected to every other vertex with at least d edge-disjoint paths.
- Problem: Find the least weight solution
- The first sub-quadratic time implementation of the Williamson, Goemans, Mihail, Vazirani algorithm

The Survivable Network Design Problem

- Given a weighted graph
- Each vertex v has a demand d; v must be connected to every other vertex with at least d edge-disjoint paths.
- Problem: Find the least weight solution
- The first sub-quadratic time implementation of the Williamson, Goemans, Mihail, Vazirani algorithm

The Survivable Network Design Problem

- Given a weighted graph
- Each vertex v has a demand d; v must be connected to every other vertex with at least d edge-disjoint paths.
- Problem: Find the least weight solution
- The first sub-quadratic time implementation of the Williamson, Goemans, Mihail, Vazirani algorithm

Open Problems

- Combine splitting-off and sampling (a la Karger) to obtain a $\tilde{O}(nk)$ time algorithm for exact/approximate Steiner min-cut
- Las-Vegas $\tilde{O}(nk)$ time algorithm for global Min-Cut

Open Problems

- Combine splitting-off and sampling (a la Karger) to obtain a $\tilde{O}(nk)$ time algorithm for exact/approximate Steiner min-cut
- Las-Vegas $\tilde{O}(nk)$ time algorithm for global Min-Cut

Collaborators and References

- Series of joint works with Richard Cole, Anand Balghat, Kavitha T, Debmalya Panigrahy
- STOC 2003, 2007, SODA 2007, SODA 2008

Collaborators and References

- Series of joint works with Richard Cole, Anand Balghat, Kavitha T, Debmalya Panigrahy
- STOC 2003, 2007, SODA 2007, SODA 2008

And Finally....

- THANK YOU