15-853: Algorithms in the Real World

Lecture 6, September 30, 2002

Graph Separators
Part II

Lecturer: *Prof. Guy Blelloch*
Scribe: *Flavio Lerda*

1. Separator theorems

At the end of last class we introduced the concept of a separator theorem: a separator theorem proves that it is possible to obtain a good separator for a certain class of graphs.

A good separator was defined as having a cut of size smaller than a fixed constant times a function of the size of the graph, and such that the size of the bigger of the two sub-graphs generated by the cut is smaller than a fixed fraction of the size of the original graph.

It is important to notice that it does not make sense to have a separator theorem for a single graph because it is always possible to find a big enough bound such that any separator is good enough. What we need is a class of graphs where the size of the graphs can vary with a parameter *n*.

We defined a class of graphs so that each sub-graph of a graph that belongs to the class also belongs to that class. This is useful for developing the theory, but for some application this is not always the case: however, in many cases, the results are still applicable in practice.

For instance, if we consider the routing graph of the Internet, it has a good separator, but there are some sub-graphs that are highly connected and they do not have a good separator. The algorithms based on graph separators still work well in practice in cases like this, as long as the highly connected sub-graphs are small enough.

We will start refreshing some of the definitions.

Definition 1.1: A class of graphs is a set S of graphs that is closed under the sub-graph relation.

We can define a vertex-separator theorem as follows:

Definition 1.2: A class of graphs S satisfies a $f(n)$-vertex-separator theorem if there are constants $\alpha < 1$ and $\beta > 0$ such that for every graph $G = (V, E)$ in the class S there exists a cut set $C \subseteq V$ which partitions the graph G in two sub-graphs A and B such that $|C| \leq \beta f(|G|)$, $|A| \leq \alpha |G|$ and $|B| \leq \alpha |G|$.

And analogously an edge-separator theorem:

Definition 1.3: A class of graphs S satisfies a $f(n)$-edge-separator theorem if there are constants $\alpha < 1$ and $\beta > 0$ such that for every graph $G = (V, E)$ in the class S there exists a cut set $C \subseteq E$ which partitions the graph G in two sub-graphs A and B such that $|C| \leq \beta f(|G|)$, $|A| \leq \alpha |G|$ and $|B| \leq \alpha |G|$.

As we showed in the previous class, every good edge-separator can be turned into a good vertex-separator. Therefore if a class of graphs has an $f(n)$-edge-separator theorem, it also has an $f(n)$-vertex-separator theorem.

However the other way around is not always true. For instance, planar graphs (as we will see below) have a $f(n)$-vertex-separator theorem: however, if we consider the planar graph with n vertices obtained connecting one vertex to all the others (to form a star), it has a good vertex-separator (the cut which contains only the center of the star is a vertex-separator), but it does not have any good edge separator: in fact, in order to divide the vertices in half it is necessary to have a cut that contains $n/2$ edges.

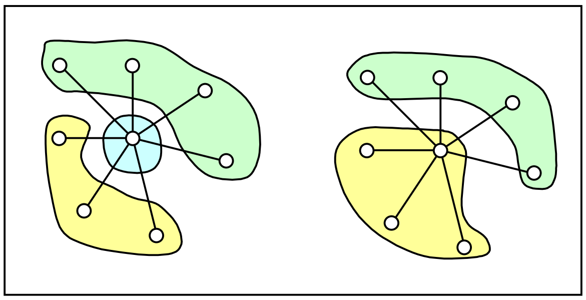

Figure 1.1 – Planar graph with a good vertex separator but not a good edge separator.

We will show that planar graphs satisfy a $\sqrt{n}$-vertex-separator theorem. Also a particular class of d-dimensional meshes satisfies a $n^{(d-1)/d}$-vertex-separator of which planar graphs represent the case $d = 2$.

Theorem 1.4: Any graph from a class with a $n^{1-\varepsilon}$-separator theorem with $\varepsilon > 0$ has $O(n)$ edges.

This means that if a graph is from a class that has a less than linear separator theorem the average degree of the graph is constant.

The proof is left as an exercise.

2. Separator Trees

A separator tree is the tree induced by recursively finding separators until you are left with single vertices. The root of the tree contains the original graph; each node of the tree is either a leaf if the node contains only a vertex, or it has two children: the children contain the partitions of the graph in the parent node defined by a separator (either edge- or vertex-separator).

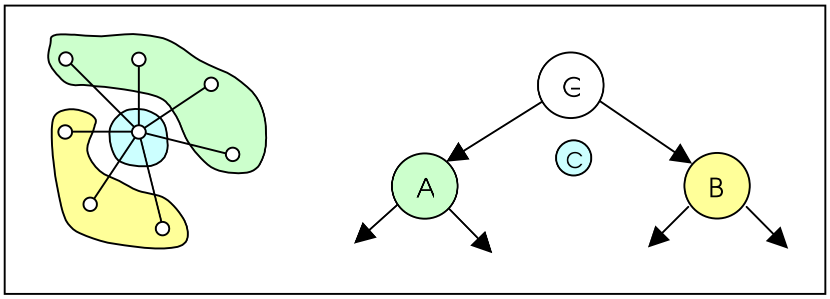

Figure 2.1 – A separator tree.

Sometimes the vertices in the cut are carried to both children, sometimes only to one of them, sometimes to neither.

A separator tree is fully balanced if the two children are equal sized (within one vertex difference).

Theorem 2.1: For a class of graphs S satisfying an (α, β) $n^{1-\varepsilon}$-edge-separator theorem, we can generate a perfectly balanced separator tree with separator size $|C| \le k\beta f(|G|)$.

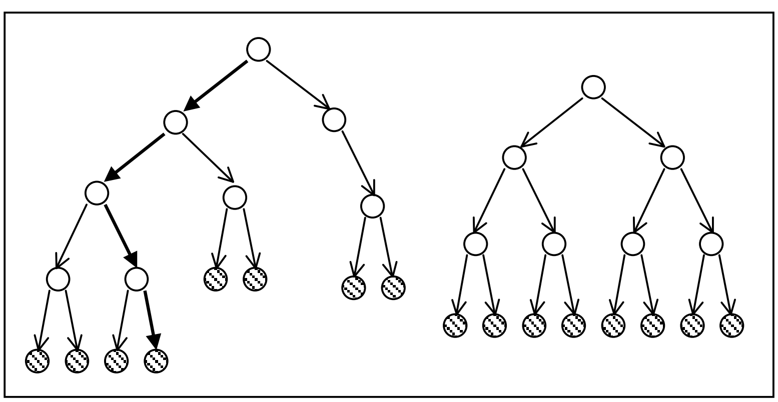
Figure 2.1 – Unbalanced and balanced separator tree.

Proof: The separator tree obtained by the (α, β) $n^{1-\varepsilon}$-edge-separator has a linear order (n) of leafs, which correspond to the vertices of the graph. First find a path in the separator tree from the root to the middle leaf ($n/2$). Consider cutting the tree so that all the nodes on the left of the middle node are on one side and the rest are on the other side. It is possible to separate these two halves by cutting all edges in the nodes on the selected path: the maximum number of edges cut in the nodes, starting at the top, is:

$$\beta n^{1-\varepsilon} + \beta(\alpha n)^{1-\varepsilon} + \beta(\alpha^2 n)^{1-\varepsilon} + ... = \beta n^{1-\varepsilon}(1+\alpha+\alpha^2+...)^{1-\varepsilon}$$

The term $(1+\alpha+\alpha^2+...)^{1-\varepsilon}$ is constant, assuming that α and ε are constants and since there are at most n levels in the tree. So we can define:

$$k = (1+\alpha+\alpha^2+...)^{1-\varepsilon} \leq \left(\frac{1}{1+\alpha}\right)^{1-\varepsilon}$$

and we have:

$$|C| \leq k\beta n^{1-\varepsilon}$$

and since $|G| = n$ and $f(n) = n^{1-\varepsilon}$:

$$|C| \leq k\beta f(|G|)$$

W

What this theorem means is that if the size of the cut is sub-linear to the size of the size of the graph, given a (α, β) $f(n)$-separator theorem we can convert it into a $(1/2, k\beta)$ $f(n)$-separator theorem with $k > 1$ (and therefore allowing a bigger cut size).

3. Planar Separator Theorem

We are going to formulate now a separator theorem for planar graphs. First we need to define the class of planar graphs formally.

Definition 3.1: The set of planar graphs is the set of graphs that can be embedded in a plane or in a sphere so that no two edges cross.

It is easy to see that this is a class of graphs since if a graph can be embedded in a plane (or a sphere) without any edges crossing, any sub-graph of this graph can be embedded in the same plane (or sphere) without any edges crossing as well. Therefore the set of planar graphs is closed under the sub-graph relation and forms a class of graphs.

Also note that if a graph is planar, an embedding in a plane (or in a sphere) can be found in linear time.

Theorem 3.2 (Planar Separators – Lipton-Tarjan, 1977): The class of planar graphs obeys a $(2/3, 4)$ $\sqrt{n}$-vertex-separator theorem.

Proof:

The proof of this theorem is going to be constructive, in which we show that there exists such a separator by giving an algorithm constructing it.

The algorithm, which is linear-time, is described in the following and a pseudo-code for it is given in Figure 3.1.

First of all we need to find an embedding of the planar graph in the plane: then we perform a breadth first search starting from a random vertex. The starting vertex can be any vertex, even if in practice it is better to choose a peripheral vertex, since this improves the performance of the algorithm.

If the number of levels that are necessary to visit the whole graph is less than $\sqrt{n}$ (Figure 3.1), then we call a sub-procedure called CUTSHALLOW, which gives a separator which satisfies the constraints – we will describe this procedure later on.

This represents the case where the graph is not very deep.

If there are more than $\sqrt{n}$ levels, we will look for the level j in the BFS which contains the $n/2^{th}$ vertex. If this level contains less than $\sqrt{n}$ vertices (Figure 3.2), then the vertices in the level j are a separator for the graph because they split the space in two halves, one before level j and one after: since level j contains the $n/2$ vertex, there are at most $n/2-1$ vertices in the levels before (and after) level j. These two sets will be sets A and B in which the graph is partitioned whose cardinality is at most $n/2-1$. This would be enough to satisfy the bounds imposed on the size of the sub-graphs.

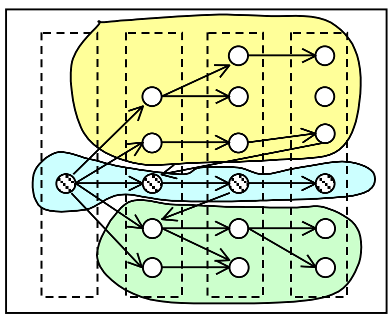
Figure 3.1 – A Shallow graph.

This represents the case where the graph is not very thick, where at each level of the graph there are not many vertices.

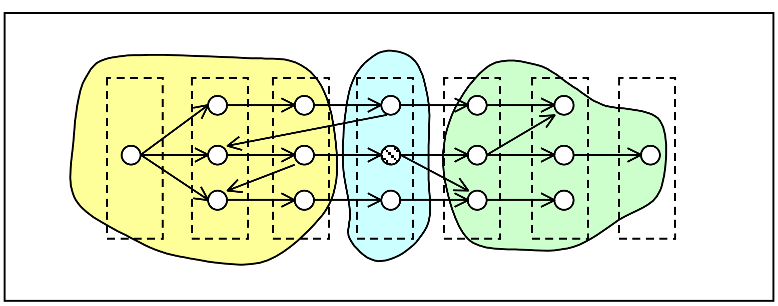
Figure 3.2 – A thin graph.

What we still have to take care of are those graphs that are too deep to fall in the first case but have some levels around the middle of the graph that contain too many vertices to be used to partition the graph (Figure 3.3).

If there are more than $\sqrt{n}$ vertices in level j we will look for levels i and k, such that $i<j<k$, $k-i<\sqrt{n}$, and there are less than $\sqrt{n}$ vertices in levels i and k.

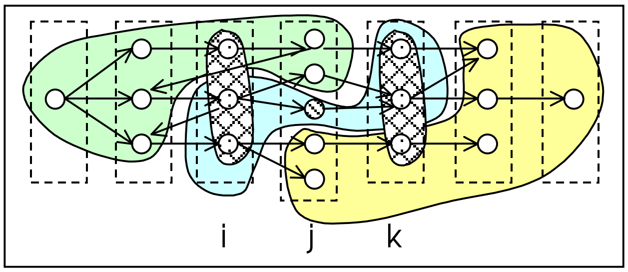

Figure 3.3 – Levels i, j, and k.

By a simple counting argument it is possible to show that such level must exist. If there were no such i and k, then for every value of i and k we would have that there are more than $\sqrt{n}$ vertices in both levels, which means that for each level l such that $i<l<k$ there are at least $\sqrt{n}$ vertices on each level. Since i and k can be at most $\sqrt{n}$ levels away, there must be $\sqrt{n}$ levels with more than $\sqrt{n}$ vertices in each of them, which means that there are more than $\sqrt{n}\sqrt{n}=n$ vertices in the graph. But the graph has exactly n vertices so that's a contradiction and levels i and k which satisfy the given properties must exist.

If the number of vertices before level i is less than $n/3$, then we can use level i as a separator (the same applies if the number of vertices after level k is less than $n/3$).

Otherwise we can extract the graph between levels i and k and substitute level i with a single root vertex: calling CUTSHALLOW on this graph will return a cut of this sub-graph. We can piece together the two parts of the graph obtained with CUTSHALLOW with the two parts obtained removing levels i through j to form a $2/3-1/3$ partition of the graph, adding the bigger of the left and right components to the bigger of the two partitions.

Let R be the number of vertices in the levels before i; let S be the number of vertices in the levels after k. Then the number of nodes between level i and k is $n-R-S$: the procedure CUTSHALLOW computes a partitioning with is in the worst-case $1/3-2/3$. If we assume, without loss of generality, that $R<S$ we have that:

$$|A| = R + \frac{2}{3}(n-R-S)$$

$$|B| = S + \frac{1}{3}(n-R-S)$$

since we assumed $R<S$ and from before $R<n/3$ and $S<n/3$, we have:

$$|A| = R + \frac{2}{3}(n-R-S) = R + \frac{2n}{3} - \frac{2R}{3} - \frac{2S}{3} = \frac{2n}{3} + \frac{R-2S}{3}$$

and:

$$R-2S<0$$

because $R<S$, so:

$$|A| < \frac{2n}{3} + \frac{R-2S}{3} < \frac{2n}{3}$$

similarly for $|B|$:

$$|B| = S + \frac{1}{3}(n-R-S) = S + \frac{n}{3} - \frac{R}{3} - \frac{S}{3} = \frac{n}{3} + \frac{2S-R}{3}$$

but:

$$\frac{2S-R}{3} < \frac{2S}{3} < \frac{2\frac{n}{3}}{3} = \frac{2n}{9} < \frac{n}{3}$$

because $R<S$ and $S<n/3$, so:

$$|B| < \frac{n}{3} + \frac{n}{3} = \frac{2n}{3}$$

W

```
SEPARATOR(G)
  Find an embedding of G in the plane.
  Perform BFS.
```

```
If number of levels < √n Then
  C = CUTSHALLOW(G).
  Done.
Fi.

Find level j containing n/2 vertex.
If |L_j| < √n Then
  C = L_j .
  Done.
Fi.

Find i and j such that:
  i < j < k ;
  k − i < √n ;
  |L_i| < √n ;
  |L_k| < √n .
If the number of vertices before
 level i is greater than n/3 Then
  C = L_i .
  Done.
Fi.

If the number of vertices after level k
 is greater than n/3 Then
  C = L_k .
  Done.
Fi.

Extract the part of the graph between
 levels i and k.
Replace level i with a single root
 vertex.
Call CUTSHALLOW.
Add the bigger of the left and right
 components to the smaller of the two
 partitions and vice versa.

Done.
```

Figure 3.4 – Lipton-Tarjan algorithm.

The subroutine CUTSHALLOW assumes that the DFS of the graph has d levels and returns a $2/3-1/3$-vertex-separator of size at most $2d+1$.

Here's the intuition behind the CUTSHALLOW procedure: let's consider the BFS tree: this tree is embedded in the planar graph, and since the graph is planar, it is embedded in a plane (or sphere).

We can therefore find a path from the root to one of the vertices including at most one vertex from each level: therefore this path will have at most $d+1$ vertices, including the root. Taking another such path we split the tree (and the graph) in two and we have a cut of size at most $2d+1$, because they share the root.

It is interesting to notice that a version of this theorem for slightly looser bounds, a $\sqrt{n}\log n$-vertex-separator theorem, dates back to Ungar in 1951.

4. Kernighan-Lin Heuristic

Even if the planar graph separator theorem by Lipton and Tarjan proposes an algorithm to find such a separator, this is not the currently most used algorithm for this purpose.

Kernighan and Lin proposed a heuristic to improve a given edge separator based on "hill climbing" which is most used in practice. The initial partitioning can be obtained using a different algorithm (for example the CUTSHALLOW presented before), or by randomly partitioning the graph.

This algorithm was developed in the context of circuit design for solving the problem of routing connection between boards: a system would contain different boards that need to be interconnected. If we can find an edge separator (with a small amount of crossing edges) we can reduce the number of interconnections that are necessary among the boards.

This algorithm allows for weighted edges, which means that each edge is assigned a weight and the problem we want to solve is to find an edge separator with a small weight, where the weight of the separator is the sum of the weight of the edges that belong to the cut.

The algorithm assumes we have an initial cut, with two equal sized partitions A and B. We want then to swap a subset X of A with a subset Y of B, where X and Y have the same size: the hope is that this switch will reduce the cut size.

Finding the optimal subset would solve the optimal separator program itself, therefore it must be NP-hard.

Instead, the Kernighan and Lin propose a heuristic which swaps a single pair of vertices at a time, swapping that pair that at the time most decreases the cut size, or less increases it.

The idea is to improve the cut size when possible with a greedy scheme. When this is not impossible, instead of stopping, the algorithm allows a swap to increase the cut size: this is useful when we reached a local minimum, since it might take us out of it being able to reach a better result later on.

Here are some definitions that we will use during the explanation of the algorithm:

$w(u,v)$	the weight of the edge between u and v.
$C(A,B)$	the weighted cut between A and B.
$I(v)$	the sum of the weights of the edges incident in v which stay within the same partition.
$E(v)$	the sum of the weights of the edges incident in v that go to the other partition.

$G(v) = E(v) - I(v)$

the single vertex gain, the reduction of the cut size achieved by putting v on the other side.

$G(u,v) = G(u) + G(v) - 2w(u,v)$

the pair wise gain, the reduction of the cut size achieve by swapping u and v.

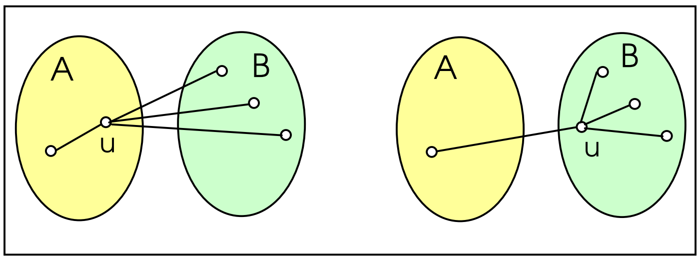

Figure 4.1 – I(u)=1, E(u)=3, G(u)=2 on a graph with unit weights.

Given a graph $G=(V,E)$ and a partitioning (A,B) of its vertices in two equal size subsets, the Kernighan-Lin algorithm puts every pair (u,v) where $u \in A$ and $v \in B$ in a priority queue Q based on the priority $G(u,v)$, the pair wise gain.

Then it repeats $|V|/2$ times the following steps: it extract from Q the pair (u,v) with the highest priority, removes from Q every pair involving either u or v; updates the priorities of the neighbors of u and v, updating their position in the queue if necessary: if w is a neighbor of u, its priority is increased by $2w(u,w)$ if w is on the same side as u before changing the partitions, decreases it by the same quantity otherwise; swaps the vertices u and v.

At the end, the partition obtained corresponds to the original one where the subsets A and B have been swapped. We select the best intermediate cut C_k. This requires to be able to reconstruct the best cut, either by undoing some of the swaps, redo the sequence of swaps that led to such a cut, or saving the best cut during processing.

We can estimate the running time as $O(n^2)$ for each iteration of the loop, so the total running time would be $O(n^3)$.

Fiduccia-Mattheyses is a variation of Kernighan-Lin where we consider individual vertices instead of pairs.

First we build two queues Q_a and Q_b which contain the vertices in A and B respectively, based on the priority $G(u)$.

Then at each step, that we repeat again $|V|/2$ times, we select the two vertices at the top of the two queues, and we swap them. We don't need to remove any other vertices, but we still need to update the priorities for the neighbors of the two chosen vertices.

Again, at the end, we have to select the best cut among the intermediate ones.

This version of the algorithm has a running time that is $O(|V|+|E|)$ using appropriate data structures: remember that each queue now has a size that is linear in the number of vertices, while before it was quadratic.