

15-853: Algorithms in the Real World

Separators

- Introduction
- Applications

15-853

Page1

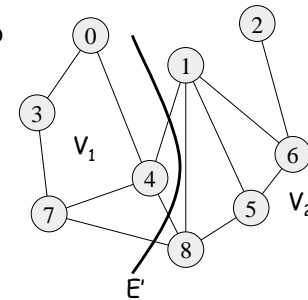
Edge Separators

An edge separator:

a set of edges $E' \subset E$
which partitions V into
 V_1 and V_2

Criteria:

- $|V_1|, |V_2|$ balanced
- $|E'|$ is small



15-853

Page2

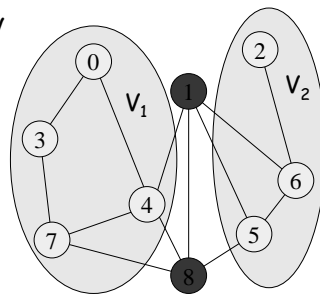
Vertex Separators

An vertex separator:

a set of vertices $V' \subset V$
which partitions V into
 V_1 and V_2

Criteria:

- $|V_1|, |V_2|$ balanced
- $|V'|$ is small



15-853

Page3

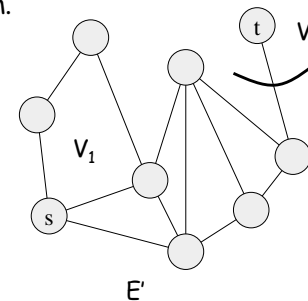
Compared with Min-cut

Min-cut: as in the min-cut, max-flow theorem.

Min-cut has no balance criteria.

Min-cut typically has a source (s) and sink (t).

Will tend to find unbalanced cuts.



15-853

Page4

Other names

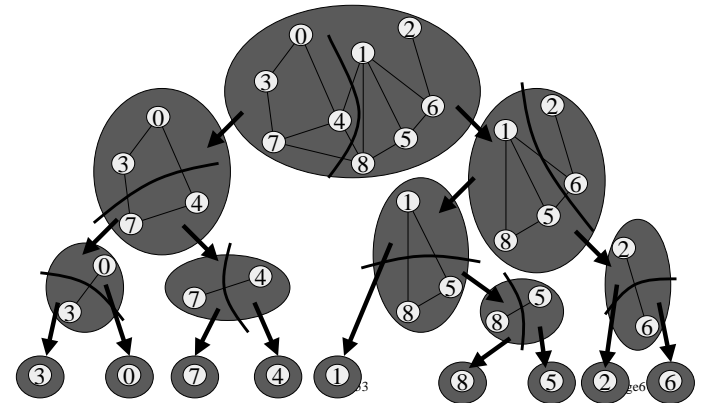
Sometimes referred to as

- **graph partitioning** (probably more common than "graph separators")
- graph bisectors
- graph bifurcators
- balanced or normalized graph cuts

15-853

Page5

Recursive Separation



What graphs have small separators

Planar graphs: $O(n^{1/2})$ vertex separators

2d meshes, constant genus, excluded minors

Almost planar graphs:

the internet, power networks, road networks

Circuits

need to be laid out without too many crossings

Social network graphs:

phone-call graphs, link structure of the web,
citation graphs, "friends graphs"

3d-grids and meshes: $O(n^{1/2})$

15-853

Page7

What graphs don't have small separators

Expanders:

Can someone give a quick proof that they don't have small separators

Hypercubes:

$O(n)$ edge separators

$O(n/(\log n)^{1/2})$ vertex separators

Butterfly networks:

$O(n/\log n)$ separators ?

It is exactly the fact that they don't have small separators that make them useful.

15-853

Page8

Applications of Separators

Circuit Layout (dates back to the 60s)
VLSI layout
Solving linear systems (nested dissection)
 $n^{3/2}$ time for planar graphs
Partitioning for parallel algorithms
Approximations to certain NP hard problems
 TSP, maximum-independent-set
Clustering and machine learning
Machine vision

15-853

Page9

More Applications of Separators

Out of core algorithms
Register allocation
Shortest Paths
Graph compression
Graph embeddings

15-853

Page10

Available Software

METIS: U. Minnesota
PARTY: University of Paderborn
CHACO: Sandia national labs
JOSTLE: U. Geenwich
SCOTCH: U. Bordeaux
GNU: Popinet

Benchmarks:

- [Graph Partitioning Archive](#)

15-853

Page11

Different Balance Criteria

Bisectors: 50/50
Constant fraction cuts: e.g. 1/3, 2/3
Trading off cut size for balance:

$$\text{min cut criteria: } \min_{V' \subset V} \left(\frac{|V'|}{|V_1| + |V_2|} \right)$$

$$\text{min quotient separator: } \min_{V' \subset V} \left(\frac{|V'|}{\min(|V_1|, |V_2|)} \right)$$

All versions are NP-hard

15-853

Page12

Other Variants of Separators

k-Partitioning:

Might be done with recursive partitioning, but direct solution can give better answers.

Weighted:

Weights on edges (cut size), vertices (balance)

Hypergraphs:

Each edge can have more than 2 end points
common in VLSI circuits

Multiconstraint:

Trying to balance different values at the same time.

15-853

Page13

Asymptotics

If S is a class of graphs closed under the subgraph relation, then

Definition: S satisfies a $f(n)$ vertex-separator theorem if there are constants $\alpha < 1$ and $\beta > 0$ so that for every $G \in S$ there exists a cut set $C \subset V$, with

1. $|C| \leq \beta f(|G|)$ cut size
2. $|A| \leq \alpha |G|, |B| \leq \alpha |G|$ balance

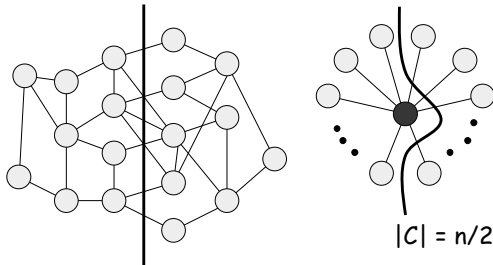
Similar definition for edge separators.

15-853

Page14

Edge vs. Vertex separators

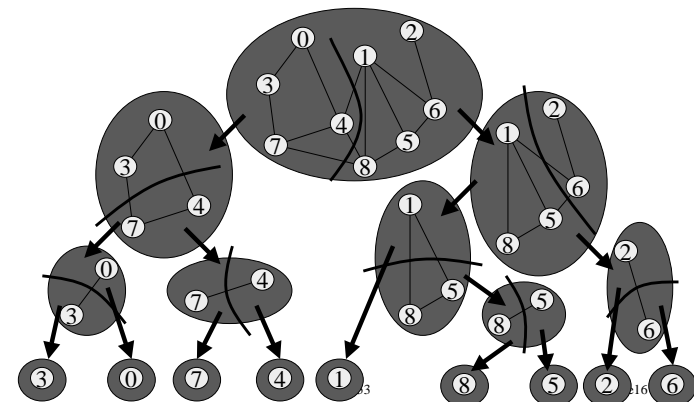
If a class of graphs satisfies an $f(n)$ edge-separator theorem then it satisfies an $f(n)$ vertex-separator.
The other way is not true (unless degree is bounded)



15-853

Page15

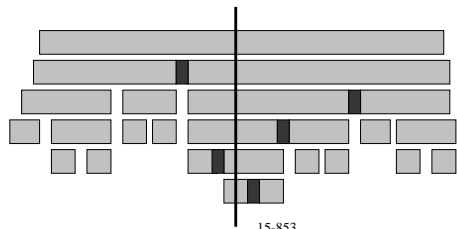
Separator Trees



Separator Trees

Theorem: For S satisfying an $(\alpha, \beta) n^{1-\epsilon}$ edge separator theorem, we can generate a perfectly balanced separator tree with separator size $|C| \leq k \beta f(|G|)$.

Proof: by picture $|C| = \beta n^{1-\epsilon}(1 + \alpha + \alpha^2 + \dots) = \beta n^{1-\epsilon}(1/1-\alpha)$



15-853

Page17

Algorithms

All are either heuristics or approximations

- Kernighan-Lin (heuristic)
- Planar graph separators (finds $O(n^{1/2})$ separators)
- Geometric separators (finds $O(n^{(d-1)/d})$ separators)
- Spectral (finds $O(n^{(d-1)/d})$ separators)
- Flow techniques (give $\log(n)$ approximations)
- Multilevel recursive bisection (heuristic, currently most practical)

15-853

Page18

Kernighan-Lin Heuristic

Local heuristic for edge-separators based on "hill climbing". Will most likely end in a local-minima.

Two versions:

Original: takes n^2 times per step

Fiduccia-Mattheyses: takes n times per step

15-853

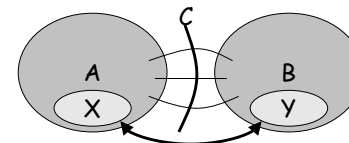
Page19

High-level description for both

Start with an initial cut that partitions the vertices into two equal size sets V_1 and V_2

Want to swap two equal sized sets

$X \subset A$ and $Y \subset B$ to reduce the cut size.



Note that finding the optimal subsets X and Y solves the optimal separator problem, so it is NP hard.

We want some heuristic that might help.

15-853

Page20

Some Terminology

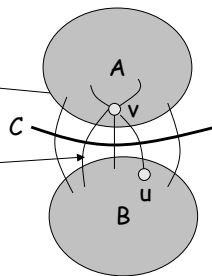
$C(A,B)$: the weighted cut between A and B

$I(v)$: the number of edges incident on v that stay within the partition

$E(v)$: the number of edges incident on v that go to the other partition

$D(v)$: $E(v) - I(v)$

$D(u,v)$: $D(u) + D(v) - 2w(u,v)$
the gain for swapping u and v



15-853

Page21

Kernighan-Lin improvement step

$KL(G, A_0, B_0)$

$\forall u \in A_0, v \in B_0$

put (u,v) in a PQ based on $D(u,v)$

for $k = 1$ to $|V|/2$

$(u,v) = \max(PQ)$

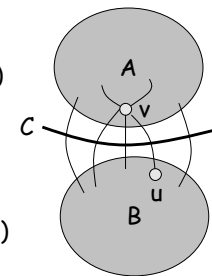
$(A_k, B_k) = (A_{k-1}, B_{k-1})$ swap (u,v)

delete u and v entries from PQ

update D on neighbors (and PQ)

select A_k, B_k with best C_k

Note that can take backward steps
($D(u,v)$ can be negative).



15-853

Page22

Fiduccia-Mattheyses improvement step

$FM(G, A_0, B_0)$

$\forall u \in A_0$ put u in PQ_A based on $D(u)$

$\forall v \in B_0$ put v in PQ_B based on $D(v)$

for $k = 1$ to $|V|/2$

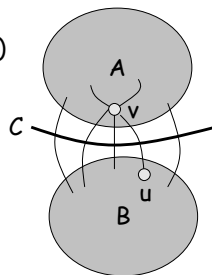
$u = \max(PQ_A)$

put u on B side and update D

$v = \max(PQ_B)$

put v on A side and update D

select A_k, B_k with best C_k

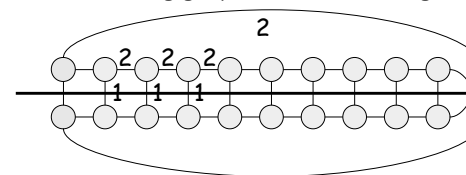


15-853

Page23

A Bad Example for KL or FM

Consider following graph with initial cut given in red.



KL (or FM) will start on one side of the grid (e.g. the blue pair) and flip pairs over moving across the grid until the whole thing is flipped.

After one round the graph will look identical?

15-853

Page24

Boundary Kernighan-Lin (or FM)

Instead of putting all pairs (u,v) in Q (or all u and v in Q for FM), just consider the boundary vertices (i.e. vertices adjacent to a vertex in the other partition).

Note that vertices might not originally be boundaries but become boundaries.

In practice for reasonable initial cuts this can speed up KL by a **large** factor, but won't necessarily find the same solution as KL.

15-853

Page25

Performance in Practice

In general the algorithms do very well at smoothing a cut that is approximately correct.

Works best for graphs with reasonably high degree.

Used by most separator packages either

1. to smooth final results
2. to smooth partial results during the algorithm

15-853

Page26

Separators Outline

Introduction:

Algorithms:

- Kernighan Lin
- BFS and PFS
- Multilevel
- Spectral
- Lipton Tarjan

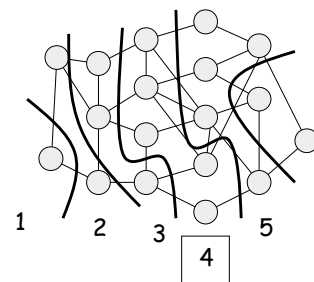
Applications:

- Graph Compression
- Nested Dissection (solving linear systems)

15-853

Page27

Breadth-First Search Separators



Run BFS and as soon as you have included half the vertices return that as the partition.

Won't necessarily be 50/50, but can arbitrarily split vertices in middle level.

Used as substep in Lipton-Tarjan planar separators.

In practice does not work well on its own.

15-853

Page28

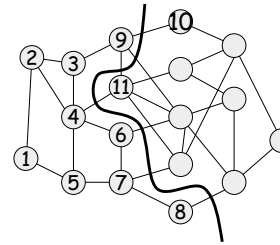
Picking the Start Vertex

1. Try a few random starts and select best partition found
2. Start at an "extreme" point. Do an initial DFS starting at any point and select a vertex from the last level to start with.
3. If multiple extreme points, try a few of them.

15-853

Page29

Priority-First Search Separators



Prioritize the vertices based on their gain (as defined in KL) with the current set.
Search until you have half the vertices.

15-853

Page30

Multilevel Graph Partitioning

Suggested by many researchers around the same time (early 1990s).

Packages that use it:

- METIS
- Jostle
- TSL (GNU)
- Chaco

Best packages in practice (for now), but not yet properly analyzed in terms of theory.

Mostly applied to edge separators.

15-853

Page31

High-Level Algorithm Outline

MultilevelPartition(G)

If G is small, do something brute force

Else

Coarsen the graph into G' (Coarsen)

$A', B' = \text{MultilevelPartition}(G')$

Expand graph back to G and project the partitions A' and B' onto A and B

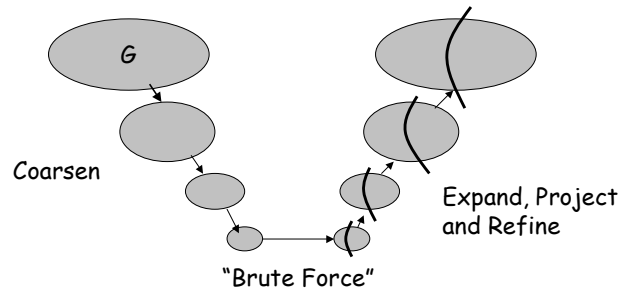
Refine the partition A, B and return result

Many choices on how to do underlined parts

15-853

Page32

MGP as Bubble Diagram



15-853

Page33

How to Coarsen

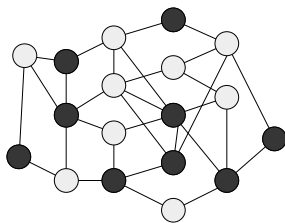
Goal is to pick a sample G' such that when we find its partition it will help us find the partition of G .

Possibilities?

15-853

Page34

Random Sampling

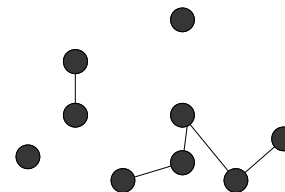


Pick a random subset of the vertices.
Remove the unchosen vertices and their incident edges

15-853

Page35

Random Sampling



Pick a random subset of the vertices.
Remove the unchosen vertices and their incident edges
Graph falls apart if it is not dense enough.

15-853

Page36

Maximal Matchings

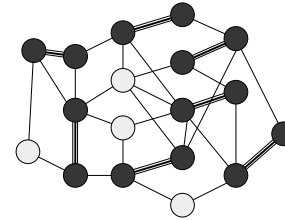
A maximal matching is a pairing of neighbors so that no unpaired vertex can be paired with an unpaired neighbor.

The idea is to contract pairs into a single vertex.

15-853

Page37

A Maximal Matching

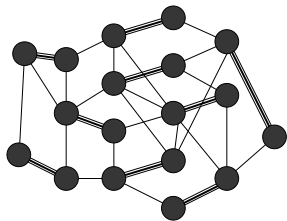


Can be found in linear time greedily.

15-853

Page38

A side note

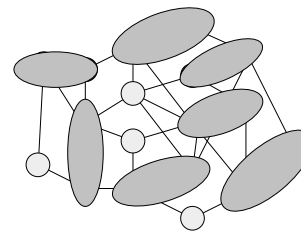


Compared to a **maximum** matching: a pairing such that the number of covered nodes is maximum

15-853

Page39

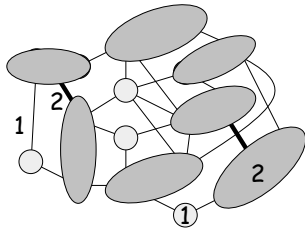
Coarsening



15-853

Page40

Colapsing and Weights



Why care about weights?

New vertices become weighted by sum of weights of their pair.

New edges (u,v) become weighted by sum of weights of multiple edges (u,v)

We therefore have solve the weighted problem.

15-853

Page41

Heuristics for finding the Matching

Random : randomly select edges.

Prioritized: the edges are prioritized by weight.

Visit vertices in random order, but pick highest priority edge first.

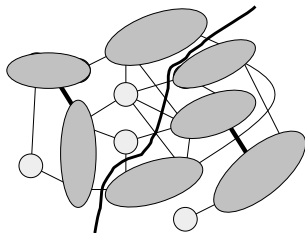
- **Heaviest first**: Why might this be a good heuristic?
- **Lightest first**: Why might this be a good heuristic?

Highly connected components: (or heavy clique matching). Looks not only at two vertices but the connectivity of their own structure.

15-853

Page42

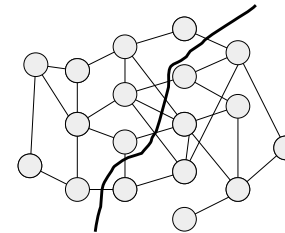
Finding the Cut on the Coarsened Graph



15-853

Page43

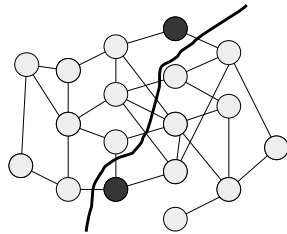
Exanding and "Projecting"



15-853

Page44

Refining

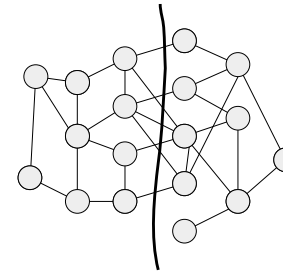


e.g. by using
Kernighan-Lin

15-853

Page45

After Refinement



15-853

Page46

METIS

Coarsening: "Heavy Edge" maximal matching.

Base case: Priority-first search based on gain.

Randomly select 4 starting points and pick best cut.

Smoothing: Boundary Kernighan-Lin

Has many other options. e.g. Multiway separators.

15-853

Page47

Separators Outline

Introduction:

Algorithms:

- Kernighan Lin
- BFS and PFS
- Multilevel
- Spectral
- Lipton Tarjan

Applications:

- Graph Compression
- Nested Dissection (solving linear systems)

15-853

Page48

Spectral Separators

Based on the second eigenvector of the "Laplacian" matrix for the graph.

Let **A** be the adjacency matrix for G .

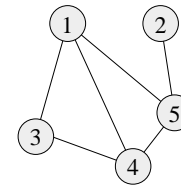
Let **D** be a diagonal matrix with degree of each vertex.

The **Laplacian** matrix is defined as $L = D - A$

15-853

Page49

Laplacian Matrix: Example



$$L = \begin{pmatrix} 3 & 0 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 2 & -1 & 0 \\ -1 & 0 & -1 & 3 & -1 \\ -1 & -1 & 0 & -1 & 3 \end{pmatrix}$$

Note that each row sums to 0.

15-853

Page50

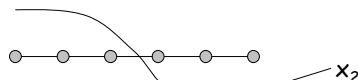
Fiedler Vectors

Find eigenvector corresponding to the second smallest eigenvalue: $Lx = \lambda x$

This is called the **Fiedler** vector.

What is true about the first eigenvector?

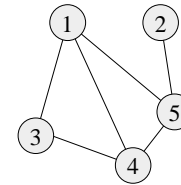
Fiedler vector can be thought of as lowest frequency "mode" of vibration.



15-853

Page51

Fiedler Vector: Example



$$L = \begin{pmatrix} 3 & 0 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 2 & -1 & 0 \\ -1 & 0 & -1 & 3 & -1 \\ -1 & -1 & 0 & -1 & 3 \end{pmatrix} x_2 = \begin{pmatrix} -.26 \\ .81 \\ -.44 \\ -.26 \\ .13 \end{pmatrix}$$

$$Lx_2 = .83x_2$$

Note that each row sums to 0.

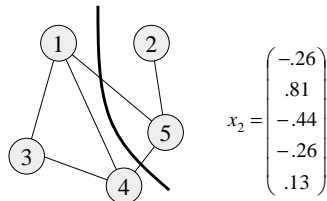
If graph is not connected, what is the second eigenvalue?

15-853

Page52

Finding the Separator

Sort Fiedler vector by value, and split in half.



sorted vertices: [3, 1, 4, 5, 2]

15-853

Page53

Power Method

Iterative method for finding first few eigenvectors.
Every vector is a linear combination of its eigenvectors

e_1, e_2, \dots

Consider: $p_0 = a_1 e_1 + a_2 e_2 + \dots$

Iterating $p_{i+1} = A p_i$ until it settles will give the principal eigenvector (largest magnitude eigenvalue) since

$$p_i = \lambda_1^i a_1 e_1 + \lambda_2^i a_2 e_2 + \dots$$

(Assuming all a_i are about the same magnitude)

The more spread in first two eigenvalues, the faster it will settle (related to the rapid mixing of expander graphs)

15-853

Page54

The second eigenvector

Assuming we have the principal eigenvector, after each iteration remove the component that is aligned with the principal eigenvector.

$$n_i = A p_{i-1}$$

$$p_i = n_i - (e_1 \cdot n_i) e_1 \quad (\text{assuming } e_1 \text{ is normalized})$$

Now

$$p_i = \lambda_2^i a_2 e_2 + \lambda_3^i a_3 e_3 + \dots$$

Can use random vector for initial p_0

15-853

Page55

Power method for Laplacian

To apply the power method we have to shift the eigenvalues, since we are interested in eigenvector with eigenvalue closest to zero.

How do we shift eigenvalues by a constant amount?

Lancos algorithm is faster in practice if starting from scratch, but if you have an approximate solution, the power method works very well.

15-853

Page56

Multilevel Spectral

MultilevelFiedler(G)

If G is small, do **something brute force**

Else

Coarsen the graph into G'

$e'_2 = \text{MultilevelFiedler}(G')$

Expand graph back to G and **project** e'_2 onto e_2

Refine e_2 using power method and return

To project, you can just copy the values in location i of e'_2 into both vertices i expands into.

This idea is used by Chaco.