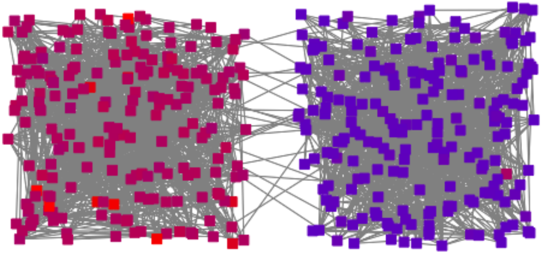
An Algorithm for Improving Graph Partitions

Reid Andersen [Microsoft Research]
–joint work with–
Kevin Lang [Yahoo! Research]

Graph Partitioning

Divide vertex set into groups, with few edges between groups.

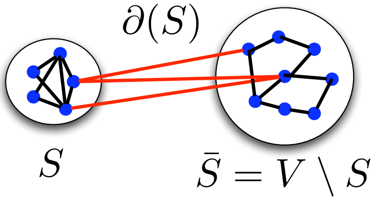


Applications:

- Divide-and-conquer algorithms for graphs
- Parallel computing, VLSI layout
- Image processing, finding communities

The Minimum Quotient Cut Problem

Find a set $S \subseteq V$ minimizing the *quotient cost* $Q(S)$.



Quotient cost

$$Q(S) = \frac{\partial(S)}{\min\left(\pi(S), \pi(\bar{S})\right)} = \frac{\text{\# edges between } S \text{ and } \bar{S}}{\text{weight of vertices in } S}$$

Vertex weights

Let each vertex have a nonnegative integer weight $\pi(v)$.
$\pi(S)$ is the total weight of vertices in S.

Algorithms for finding quotient cuts

The Minimum Quotient Cut problem is NP-hard.

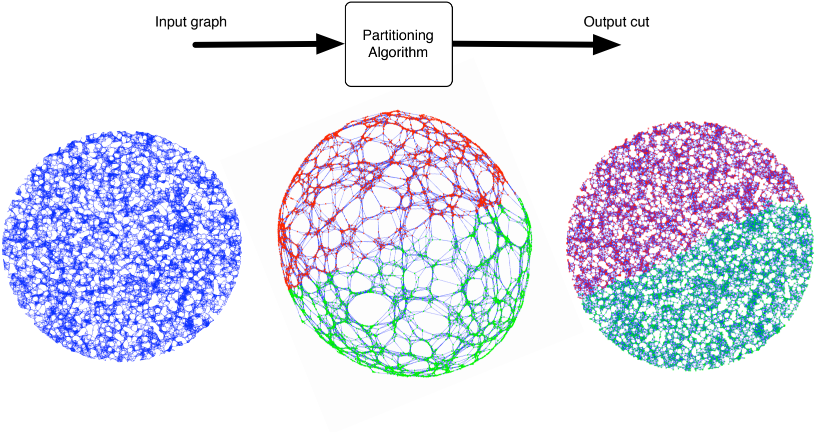
Approximation algorithms

- Spectral Partitioning [Fiedler 72]
- Leighton Rao [LR 88]
- Arora Rao Vazirani [ARV 04]
  (Best approximation algorithm known, with ratio $O(\sqrt{\log n})$)

Effective heuristics

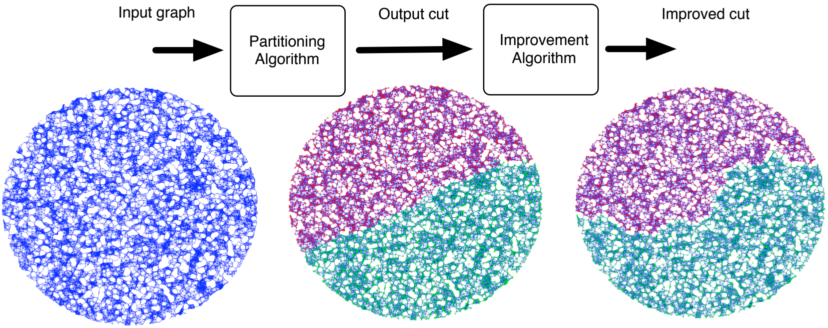
- Local search [Kernighan,Lin 70] [Fidducia,Mattheyses 82]
- Multilevel methods (METIS) [Karypis, Kumar 98]

A partitioning algorithm



Improving the proposed cut

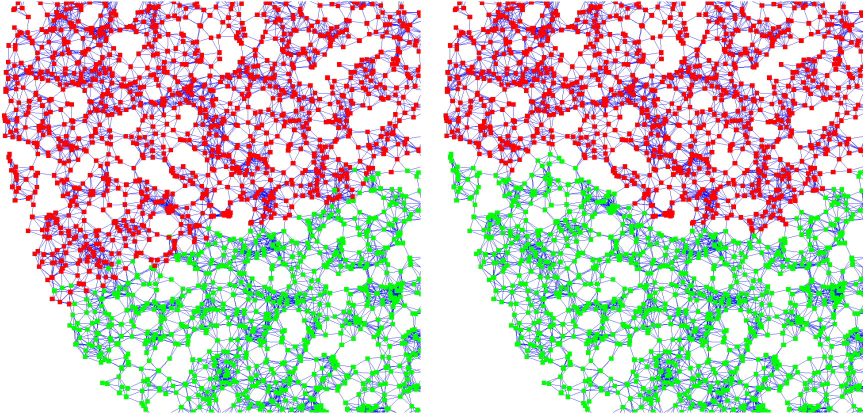
This talk is about an algorithm called `Improve`.
We apply it to the cut proposed by the partitioning algorithm to obtain a cut with better quotient score.



Q(Proposed cut) $= 0.108400 =$ (542 edges / 5000 nodes)

Q(Improved cut) $= 0.040085 =$ (189 edges / 4715 nodes)

Detail of proposed cut and improved cut



Previous work: methods for improving partitions

Swapping vertices

- greedy hill-climbing, simulated annealing
- local search: Kernighan-Lin, Fidduccia-Mattheyses
- Flow algorithms can find a block of vertices to swap for a big greedy improvement.

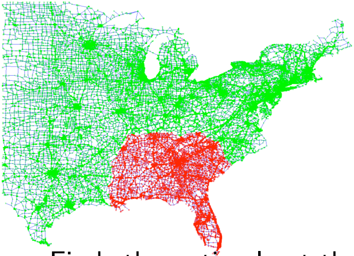
There is a powerful known method for improving the quotient cost, based on parametric flow ...

- Described in [Gallo Grigoriadis Tarjan 89].
- Key tool in the polylog approximation algorithm for minimum bisection [Feige Krauthgamer 01].
- Lang and Rao [LR 04] demonstrated its practical utility.

Previous work: parametric flow improvement algorithm

PFI algorithm

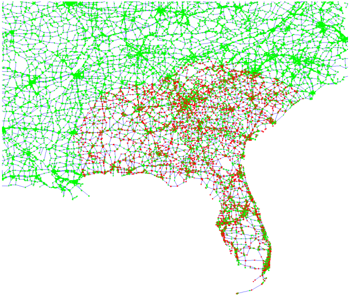
Input : a proposed set A with weight $\pi(A) \leq \pi(\bar{A})$.
Output : the subset of A with the smallest quotient cost.



- Finds the optimal cut that can be obtained by *removing* vertices from the proposed cut.
- Works by solving a sequence of $s - t$ min cut problems, or one parametric flow problem. [Gallo Grigoriadis Tarjan 89]

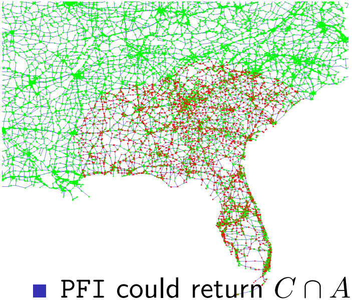
Motivation for the `Improve` algorithm

What if there exists a great cut C,
and the proposed cut A contains a significant fraction of C,
but not all of it?



Motivation for the `Improve` algorithm

What if there exists a great cut C,
and the proposed cut A contains a significant fraction of C,
but not all of it?

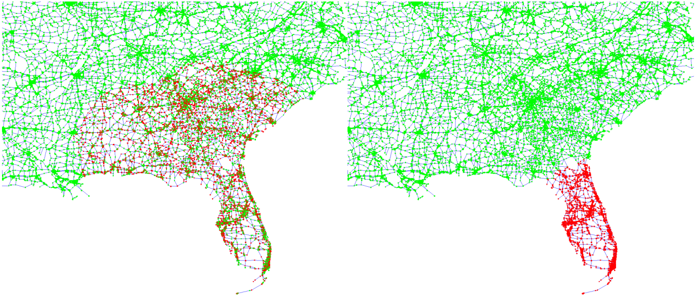


- `PFI` could return $C \cap A$, but that cut might be terrible.
- `PFI` cannot "fill in the holes" by adding nodes from outside A.

The `Improve` algorithm

`Improve`

Input : a proposed set A with $\pi(A) \leq \pi(\bar{A})$
Output : a set S

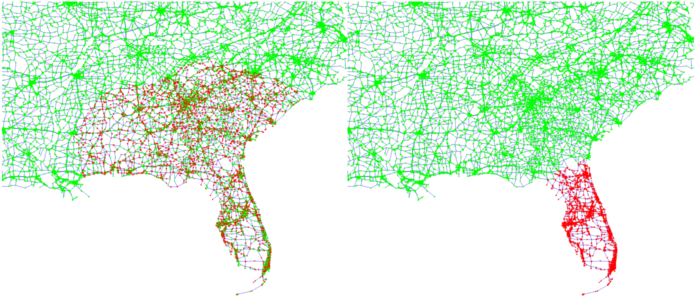


- Adds and removes vertices from the proposed cut.
- Works by solving a sequence of $s - t$ min cut problems, that cannot be cast as a parametric flow problem.

The `Improve` algorithm

`Improve`

Input : a proposed set A with $\pi(A) \leq \pi(\bar{A})$
Output : a set S



- Main theorem: to find a cut nearly as good as C, the proposed set just needs to have a larger intersection with C than a randomly chosen set.

Main theorem about `Improve`

`Improve`

Input : a proposed set A with $\pi(A) \leq \pi(\bar{A})$
Output : a set S

Theorem (part 1)

- `Improve` *runs in polynomial time.*
- *The set returned by* `Improve` *has quotient cost at least as small as the best subset of* A*,*

$$Q(S) \leq \min_{C \subseteq A} Q(C).$$

Main theorem about `Improve`

Theorem (part 2)

Let C be any set whose intersection with the proposed set A satisfies

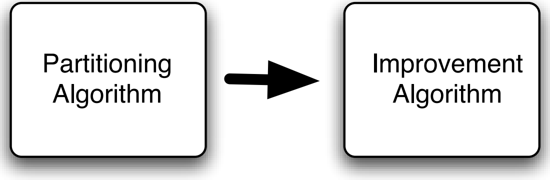
$$\frac{\pi(A \cap C)}{\pi(C)} \geq \frac{\pi(A)}{\pi(V)} + \epsilon.$$

Then the set S output by `Improve` *has quotient cost almost as small as C,*

$$Q(S) \leq \frac{1}{\epsilon} Q(C).$$

- If the fraction of C contained in the proposed set A is larger than you would expect if A were chosen randomly, then `Improve` will return a cut almost as good as C.

The role of partitioning and improvement algorithms



The role of partitioning and improvement algorithms

Partitioning Algorithm → Improvement Algorithm

Techniques

Modified quotient cost

We define a modified quotient cost relative to A that penalizes sets for including vertices outside of A.

Iterated minimum cut computations

`Improve` computes and outputs a set minimizing the modified quotient cost.

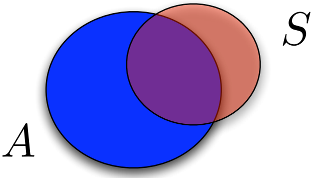
It does this by constructing and solving a sequence of s-t minimum cut problems in an augmented graph.

Weight function relative to the proposed cut

This modified weight function gives you credit for including vertices in A, and penalizes you for including vertices outside of A.

Given a proposed set $A \subseteq V$, define

$$D_A(S) = \pi(S \cap A) - \pi(S \cap \bar{A}) \left(\pi(A)/\pi(\bar{A})\right) .$$



Remarks

- The modified weight $D_A(S)$ is at most the true weight $\pi(S)$.
- It can be zero or negative.

Quotient cost relative to the proposed cut

The modified quotient cost uses the weight relative to the proposed cut as the denominator, instead of the true weight.

Quotient cost relative to A

Given a set $A \subseteq V$ that satisfies $\pi(A) \leq \pi(\bar{A})$, define

$$\tilde{Q}_A(S) = \begin{cases} \partial(S)/D_A(S) & \text{if } D_A(S) > 0. \\ +\infty & \text{if } D_A(S) \leq 0. \end{cases}$$

Remark:

- Defined for all subsets of V.

Relating the relative quotient cost to the true quotient cost

Lemma (Relative quotient cost is bigger than true quotient cost)

Assume $\pi(A) \leq \pi(\bar{A})$.

1. *For any set* $S \subseteq V$, *we have* $\tilde{Q}_A(S) \geq Q(S)$.

Lemma (Upper bounds on the relative quotient cost)

1. *If* $C \subseteq A$, *then* $\tilde{Q}_A(S) = Q(S)$.
2. *If* C *is a set for which the intersection of* A *with* C *satisfies*

$$\frac{\pi(A \cap C)}{\pi(C)} \geq \frac{\pi(A)}{\pi(V)} + \epsilon \frac{\pi(\bar{A})}{\pi(V)},$$

then $\tilde{Q}_A(C) \leq \frac{1}{\epsilon} Q(C)$.

Sketch of the main theorem

Proof sketch of Theorem 1.

$\texttt{Improve}(A)$ outputs a set S that minimizes $\tilde{Q}_A$.

If C and A have a large intersection, say $\pi(C \cap A) \geq (\frac{1}{2} + \epsilon)\pi(C)$, then

$$\tilde{Q}_A(C) \leq \frac{1}{\epsilon} Q(C).$$

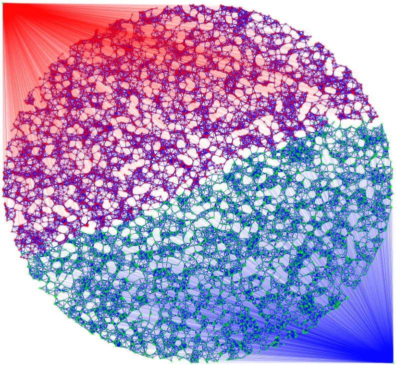
That gives us an upper bound on the true quotient cost of the set output by $\texttt{Improve}$,

$$Q(S) \leq \tilde{Q}_A(S) \leq \tilde{Q}_A(C) \leq \frac{1}{\epsilon} Q(C).$$

□

Augmented graph construction

The augmented graph $G_A(\alpha)$ depends on the input graph G, on the proposed set A, and on a parameter $\alpha \in [0, \infty)$



- Keep the nodes and edges in G with their original weights.
- Add a source node $\mathtt{s}$ and sink node $\mathtt{t}$.
- Add edges from s to each node v in A, with weight $\alpha\pi(v)$.
- Add edges from t to each node v in $\bar{A}$, with weight $\alpha\pi(v)f(A)$, where $f(A) = \pi(A)/\pi(\bar{A})$.

How to use the augmented graph cuts

Let $\mathrm{cost}_{A,\alpha}(S)$ be the cutsize of $\{\mathrm{s}\} \cup S$ in the augmented graph.

By construction,

$$\mathrm{cost}_{A,\alpha}(S) = \alpha\pi(A) + (\partial(S) - \alpha D_A(S)) \,.$$

By computing the minimum cut, you answer the following question:

Is there a set of vertices in the graph for which $\dfrac{\partial(S)}{D_A(S)} < \alpha$?

Improve pseudocode

This is a simple iterative procedure for finding the set that minimizes $\tilde{Q}_A$.

- **Input.** a set $A \subseteq V$ satisfying $\pi(A) \leq \pi(\bar{A})$.
- **Initialization.** Let $S = A$ and let $\alpha = Q(A)$.
- **Main loop.**
  1. Compute the minimum $\mathtt{s} - \mathtt{t}$ cut in the graph $G_A(\alpha)$.
  2. Let S' be the set of vertices on the source side.
  3. Let $\alpha' = \tilde{Q}_A(S')$ be the new value of α.
  4. If $\alpha' < \alpha$, continue the loop. Otherwise, halt and output S.

Number of iterations required

To bound the number of iterations, we show that $\tilde{Q}_A(S_i)$, $D_A(S_i)$, and $\partial(S_i)$ strictly decrease at each step.

Corollary

- *If the vertex weights $\pi(v)$ are integers, the algorithm halts after at most $\pi(V)^2$ iterations.*
- *If the edges of the graph are unweighted, the algorithm halts after at most m iterations, where m is the number of edges in the graph.*

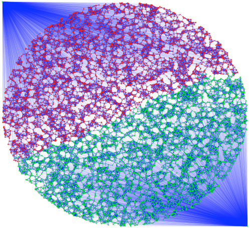
In our experiments. . .

- The average number of flow computations required was 4.
- No instance required more than 10 flow computations.

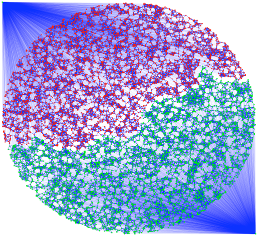
Details from each iteration

iteration	cutsize	numnodes	swap-ins	denom	$Q(S)$	$\tilde{Q}_A(S)$
0	542	5000	0	5000	0.108400	0.108400
1	198	4908	133	4642	0.040342	0.042654
2	189	4715	132	4451	0.040085	0.042462

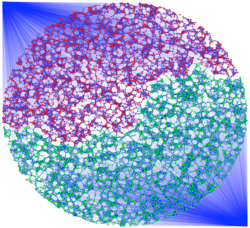
Original cut



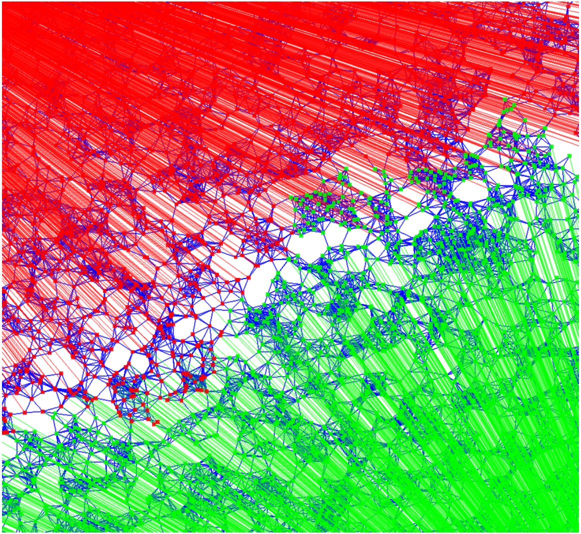
After 1 iteration



After 2 iterations
(done)



Detail of the set minimizing the relative quotient cost



Experiments

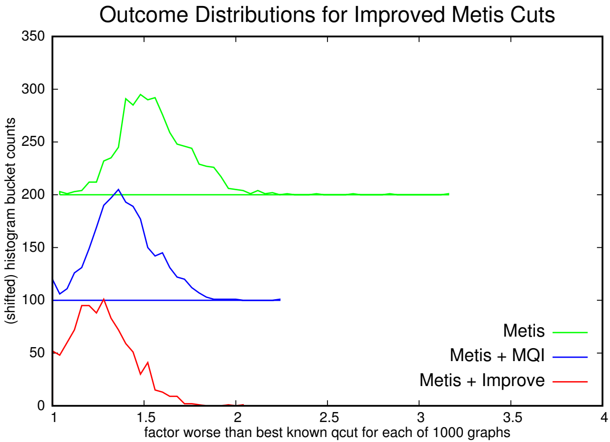
We ran off-the-shelf partitioning algorithms
(spectral partitioning and METIS)
on three different families of graphs.

We want to compare:
Partitioning algorithm alone
Partitioning algorithm + `PFI`
Partitioning algorithm + `Improve`

First family of test graphs:
random geometric graphs plus random edges.
We generated 1000 random graphs from this distribution.

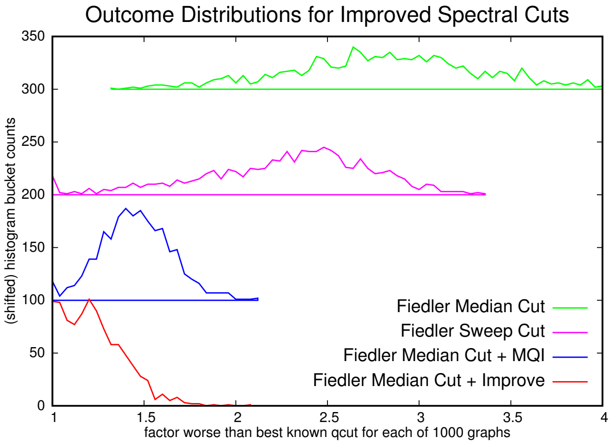
Improved METIS cuts

Histogram: for each of the 1000 graphs, how much worse is the resulting cut from the best cut we could find?



Improved spectral cuts

Solution quality histograms for 4 different methods for finding a cut from the Fiedler eigenvector.



Improve vs PFI on benchmark graphs

Test graphs Benchmark meshlike graphs from the graph partitioning archive (run by Chris Walshaw).

Experiment For each graph we ran METIS many times, improved the cuts using `Improve` and `PFI`, and took the best qcut cost over all runs.

Table 1 `Improve` always beats or ties `PFI`. The table shows how many times it beats it.

		Improve vs PFI		
name	n_nodes	wins	ties	losses
wing	62032	3914	2378	0
fe_tooth	78136	1970	625	0
fe_rotor	99617	1794	241	0
598a	110971	1263	181	0
144	144649	999	14	0
wave	156317	1132	0	0
m14b	214765	605	27	0

Improve vs PFI on benchmark graphs

Table 2

This table shows the ratio between the best balanced qcut obtained by METIS+`Improve` and the best balanced cut from the graph partitioning archive.

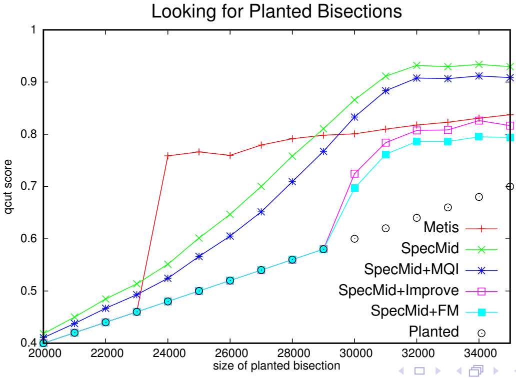
Both cuts are required to contain at least 47.5% of the graph

name	best qcut found with at least 475:525 balance	compared to archive cut
wing	0.025504 = 791 / 31015	1.000032
fe_tooth	0.097809 = 3821 / 39066	0.992518
fe_rotor	0.041964 = 2004 / 47755	1.036868
598a	0.043219 = 2398 / 55485	1.000000
144	0.089733 = 6488 / 72303	0.999982
wave	0.111400 = 8702 / 78115	1.001702
m14b	0.035723 = 3836 / 107382	1.000000

Random graphs with planted bisections

Test graphs Take two random 4-regular graphs with $50k$ nodes and $100k$ edges, then add c random edges between the two sides.

Experiment Generate 16 graphs with varying values of c. Compare the qcuts found by 5 algorithms.



Concluding remarks