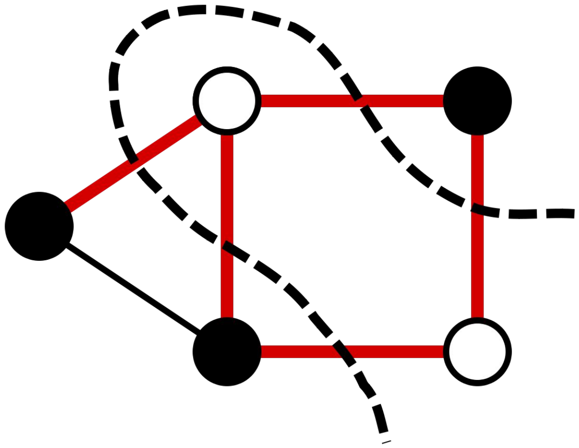
Parallelized MAXCUT

Jonathan Edelman

What is MAXCUT?

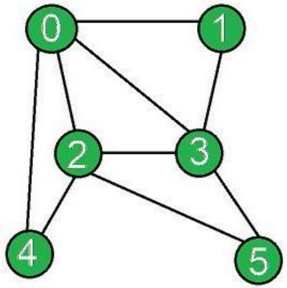
Given a graph G(V,E), partition the vertices into V_1 and V_2 s.t. the number of edges between V_1 and V_2 is maximized i.e.



MAXCUT as an optimization problem

$$\max_{y_i \in \{-1,1\}} \frac{1}{4} \sum_{i,j} w_{ij}(1 - y_i y_j),$$

$$\min_{y_i^2=1} \sum_{i,j} w_{ij} y_i y_j.$$



	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0

w_{ij} = 0 if (i,j) is not in E.

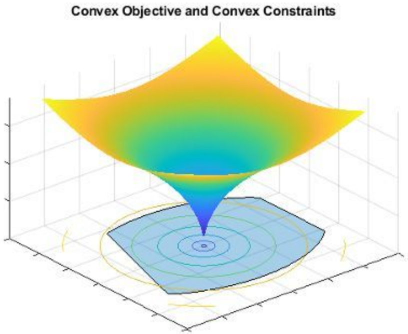
Otherwise it is the weight of edge (i,j)

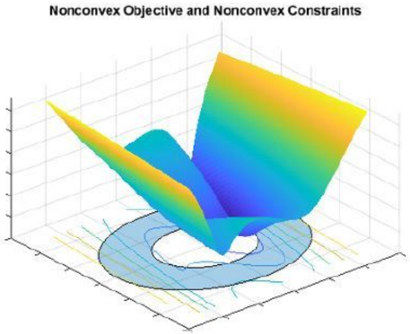
This is NP-hard.

Relaxation of MAXCUT

Convex optimization (an SDP):

$$\min_{Y \succeq 0 \in \mathbb{R}^{n \times n}} Tr(WY) \text{ s.t.} X_{ii} = 1, i = 1, ...n$$





Rank constraint (no longer convex).

$$\min_{V \in \mathbb{R}^{k \times n}} Tr(WV^TV) \text{ s.t. } ||V_i||_2 = 1, i = 1, ..., n$$

Mixing Method

If all other v_j are fixed, the last v_i is optimized with:

$$v_i = \text{normalize}(-\sum_j w_{ij} v_j)$$

Just keep looping through v_i until convergence.

Parallelization

Initial hypothesis: We can loop over all v_i to find the one that minimizes wrt to the other v_j in parallel.

Where I'm currently at

With n=30, k=4

SDP solver (way overkill, way too long): 64.722 ms (31442 allocations: 2.31 MiB)

Serial Mixing Method (a lot faster!): 1.756 ms (4762 allocations: 762.50 KiB)

Parallel Mixing Method is not working at the moment: doing the for loop in parallel does updates each vector independently, whereas when it is done serially the vectors actually converge. Instead it finds a v s.t. $v^T v = J$, the matrix of all 1s.

If this can be fixed, I think a n-times speedup could occur (where n is the number of processors)

Still working on this! Hoping to try to make even faster.

With n=100, k=16

SDP Solver: 1.125 s (311671 allocations: 25.15 MiB)

serial: 62.123 ms (47440 allocations: 19.51 MiB)

It seems the mixing method is still faster, but it may not scale as well with n as the generic solver? It is still better for smaller graphs.

Could also be an issue with implementation. Mixing method doesn’t seem to scale well with k.