Spectral Community Detection

Satyaki Sikdar
Ph.D. Student
Weninger Lab
University of Notre Dame

Laplacian Matrix $\mathcal{L}$

For an undirected graph $G = (V, E)$, the Laplacian Matrix $\mathcal{L}$ is given by $\mathcal{L} = D - A$.

- *Hermitian*: eigenvalues are positive and eigenvectors are orthogonal
- smallest (trivial) eigenpair $(\lambda_0 : 0,\ v_0 : (1, \ldots, 1)^T)$
- second smallest eigenvector (Fiedler vector) approximates the *sparsest cut* (1)

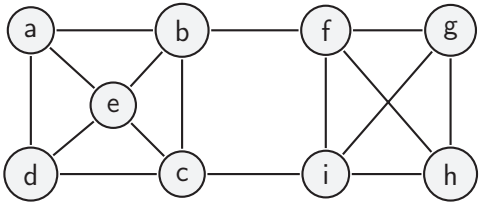


Figure 1: Toy undirected graph G

$\mathcal{L}$	a	b	c	d	e	f	g	h	i
a	3	-1		-1	-1				
b	-1	4	-1		-1	-1			
c		-1	4	-1	-1				-1
d	-1		-1	3	-1				
e	-1	-1	-1	-1	4				
f		-1				4	-1	-1	-1
g						-1	3	-1	-1
h						-1	-1	3	-1
i			-1			-1	-1	-1	4

Laplacian Matrix $\mathcal{L}$ of G

Informal Algorithm (3)

1. Compute the Fiedler vector $\mathcal{F}$ corresponding to the Laplacian $\mathcal{L}$
2. Partition $\mathcal{F}$ across threshold r into two disjoint sets p_1 and p_2

Sorted eigenvalues

λ_0	$9.54\text{e}-17$
λ_1	$6.498\text{e}-01$
λ_2	3.198
λ_3	3.326
λ_4	4
λ_5	4.554
λ_6	4.641
λ_7	5.382
λ_8	6.246

Fiedler vector $\mathcal{F}$

a	-0.378
b	-0.178
c	-0.378
d	-0.332
e	-0.178
f	$+0.291$
g	$+0.291$
h	$+0.431$
i	$+0.431$

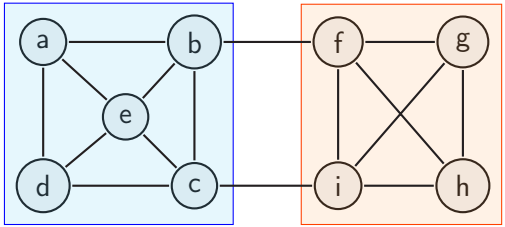


Figure 2: Graph G with 2 clusters {a,b,c,d,e} and {f,g,h,i}

Informal Algorithm (2)

1. Find k-smallest *non-trivial* eigenvectors of $\mathcal{L} : \mathcal{V} = (v_1, \ldots, v_k)$
2. Use k-means on $\mathcal{V}$ to find k clusters using Euclidean distance

k-smallest eigenvectors ($k = 3$)

	v_1	v_2	v_3
a	−0.378	0.521	−0.428
b	−0.178	0.418	0.463
c	−0.378	−0.521	−0.428
d	−0.332	0	0.104
e	−0.178	−0.418	0.463
f	0.291	0.232	0.172
g	0.291	−0.232	0.172
h	0.431	0	−0.26
i	0.431	0	−0.26

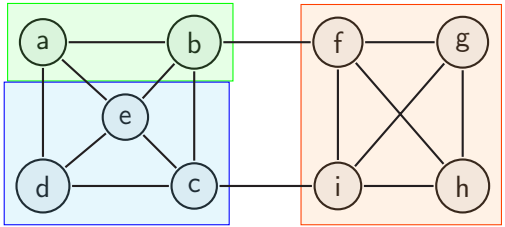


Figure 3: Graph G with 3 clusters {a,b}, {c,d,e}, and {f,g,h,i}

Algorithm 1 Approximate minimum cut of a connected graph G for a given threshold r

```
1: procedure approx_min_cut(G = (V, E), r)
2:     clusters ← ∅
3:     if G has fewer than 2 nodes then
4:         clusters ← V
5:     else
6:         fiedler ← Fiedler vector of G
7:         p_1 ← nodes ids having value less than r in
   fiedler
8:         p_2 ← rest of the nodes in G
9:         clusters ← clusters ∪ { p_1}
10:        clusters ← clusters ∪ { p_2}
11:    return clusters
```

Time Complexity

- Dependent on eigen-decomposition
- Fastest approximate decomposition runs in $\mathcal{O}(|V| + |E|)$
- Overall complexity $\mathcal{O}(|V| + |E|)$

Algorithm 2 k-way spectral partitioning of a connected graph G

procedure `k_way_spectral`$(G = (V, E),\ k)$
    `clusters` $\leftarrow \emptyset$
    if G has fewer than k nodes **then**
        `clusters` $\leftarrow V$
    else
        $\mathcal{L} \leftarrow$ Laplacian matrix of G
        $\mathcal{V} \leftarrow k$-smallest non-trivial eigenvectors of $\mathcal{L}$
        Run `K-means` clustering on $\mathcal{V}$ to find k clusters $\mathbb{C} = \{C_1, \cdots, C_k\}$ using Euclidean distance
        `clusters` $\leftarrow \mathbb{C}$
    return `clusters`

Time Complexity

- Dependent on eigen-decomposition & K-means
- Fastest approximate decomposition runs in $\tilde{\mathcal{O}}(k \cdot (|V| + |E|))$
- K-means runs in $\tilde{\mathcal{O}}(k \cdot |V|)$
- Overall complexity $\tilde{\mathcal{O}}(k \cdot (|V| + |E|))$

Software: PyPy (www.pypy.org)

- A *just-in-time* compiler - speeds up pure Python code
- "If you want your code to run faster, you should probably just use PyPy." - Guido van Rossum

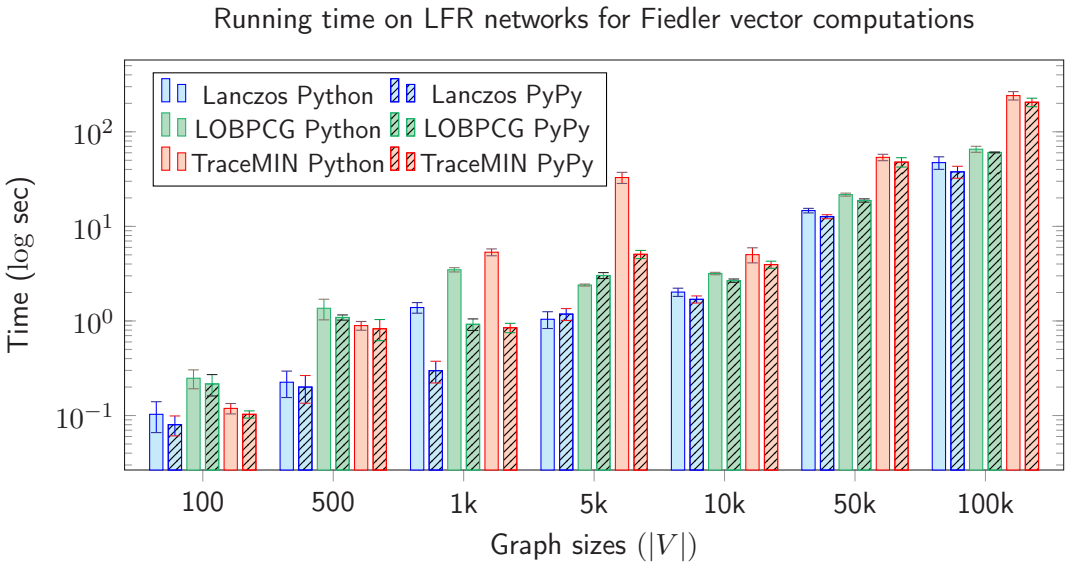
pypy

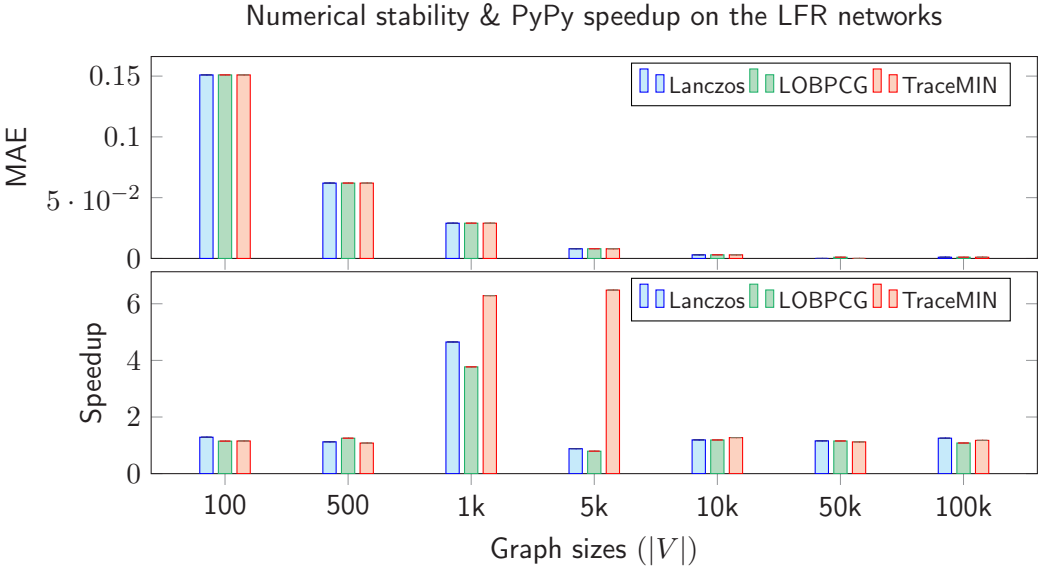
Table 1: Comparison of popular eigen-decomposition algorithms

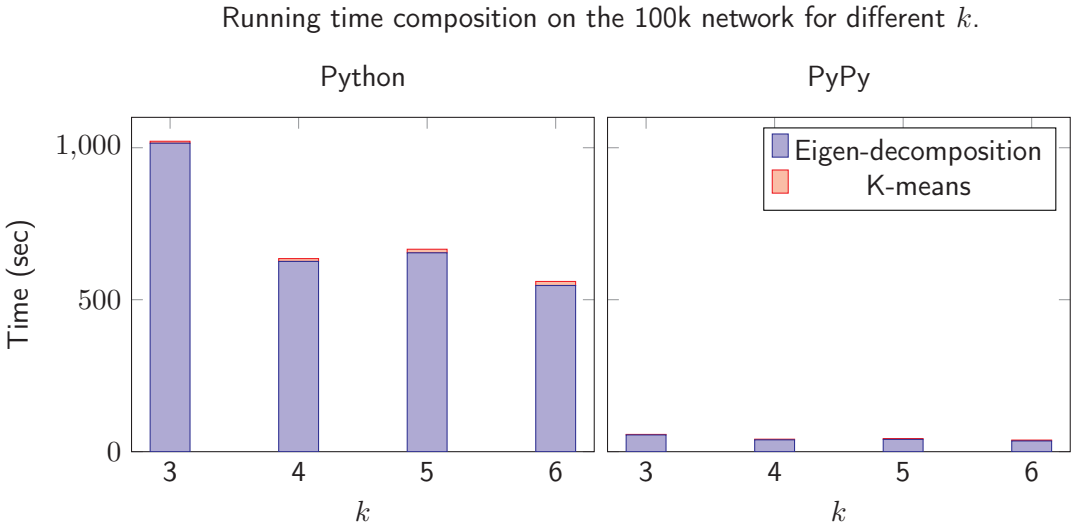
Feature	Lanczos (4)	LOBPCG (5)	TraceMIN (6)
Numerically stable?	✗	✓	✓✓
Speed	✓✓	✓	✗
Parallel?	✓✓	✓	✗

Table 2: Graph statistics for the LFR networks with $\langle k \rangle \approx 16,\ \gamma = -2,\ \beta = -1,\ \mu = 0.1$

Name	$\lvert V \rvert$	$\lvert E \rvert$	$\langle k \rangle$	ρ	$\#\Delta$s
100	100	795	15.9	16%	2 834
500	500	3 825	15.3	3%	12 296
1k	1 000	7 692	15.384	1.5%	21 491
5k	5 000	38 247	15.298	0.3%	31 063
10k	10 000	76 325	15.265	0.15%	47 136
50k	50 000	383 778	15.351	0.03%	183 152
100k	100 000	765 073	15.30	0.01%	329 364


Running time on LFR networks for Fiedler vector computations
Lanczos Python
Lanczos PyPy
LOBPCG Python
LOBPCG PyPy
TraceMIN Python
TraceMIN PyPy
Time (log sec)
10^2
10^1
10^0
10^{-1}
100
500
1k
5k
10k
50k
100k
Graph sizes ($|V|$)


Numerical stability & PyPy speedup on the LFR networks
MAE
0.15
0.1
$5 \cdot 10^{-2}$
0
Lanczos
LOBPCG
TraceMIN
Speedup
6
4
2
0
Lanczos
LOBPCG
TraceMIN
100
500
1k
5k
10k
50k
100k
Graph sizes ($|V|$)


Running time composition on the 100k network for different k.
Python
PyPy
Eigen-decomposition
K-means
Time (sec)
1,000
500
0
3
4
5
6
k

Conclusions

- PyPy is very effective in optimizing pure Python code
- Lanczos is usually good enough for clustering purposes
- Eigen-decomposition dominates the time consumption in spectral k-means
- BLAS and ARPACK ✓✓

References

(1) Fiedler, M. (1973). Algebraic connectivity of graphs. Czechoslovak mathematical journal, 23(2), 298-305.

(2) Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In Advances in neural information processing systems (pp. 849-856).

(3) Hagen, L., & Kahng, A. B. (1992). New spectral methods for ratio cut partitioning and clustering. IEEE transactions on computer-aided design of integrated circuits and systems, 11(9), 1074-1085.

(4) Lehoucq, R. B., Sorensen, D. C., & Yang, C. (1998). ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods (Vol. 6). Siam.

(5) Samokish, B. A. (1958). The steepest descent method for an eigenvalue problem with semi-bounded operators. Izv. Vyssh. Uchebn. Zaved. Mat, 5, 105-114.

(6) Manguoglu, M., Cox, E., Saied, F., & Sameh, A. (2010, June). TRACEMIN-fiedler: A parallel algorithm for computing the Fiedler vector. In International Conference on High Performance Computing for Computational Science (pp. 449-455). Springer, Berlin, Heidelberg.

Thanks!