Multilevel Graph Coarsening with Applications

Benjamin Quiring, Northeastern University and Panayot Vassilevski, CASC, LLNL

Setting

Given a graph G with vertices V and edges E.

- Each edge $e = (i, j)$ has an edge weight a_{ij}
- Each vertex i has a (weighted) degree $d_i = \sum_{e=(i,j)} a_{ij}$

Note: $a_{ij} < 0$ is allowed, but $d_i > 0$ must hold.

Abstract

Goal: to generate a hierarchy of coarser graphs that maintain certain properties of the original graph by using pairwise aggregation of strongly connected vertices and monitoring the *modularity* functional $\mathcal{Q}$. After creating the hierarchy, embed the graph in $\mathbb{R}^d$ in a multilevel fashion.
Results: The aggregation algorithm achieves higher maximum $\mathcal{Q}$ values than the state of the art, Louvain's Algorithm. Addtionally, we applied the aggregation to classification and algebraic multigrid methods, forming aggregates that correspond to the dominant direction of the distcretized anisotropic diffusion operator.

Multilevel Graph Coarsening

Goal: Find a "balanced" partition $\{\mathcal{A}\}$ of G;
The coarse graph has vertices $\{\mathcal{A}\}$, with the following values:

$$d_{\mathcal{A}} = \sum_{i,j\in\mathcal{A}} a_{ij} + \sum_{i\in\mathcal{A}} d_i \qquad a_{\mathcal{A},\mathcal{A}'} = \sum_{i\in\mathcal{A}, j\in\mathcal{A}'} a_{ij} \qquad \alpha_{\mathcal{A}} = \frac{(d_{\mathcal{A}} + \sum_{\mathcal{A}'} a_{\mathcal{A},\mathcal{A}'})}{T} \in (0,1)$$

With

$$T = \sum_{i,j} a_{ij} + \sum_i d_i$$

We maximize the modularity functional $\mathcal{Q}$:

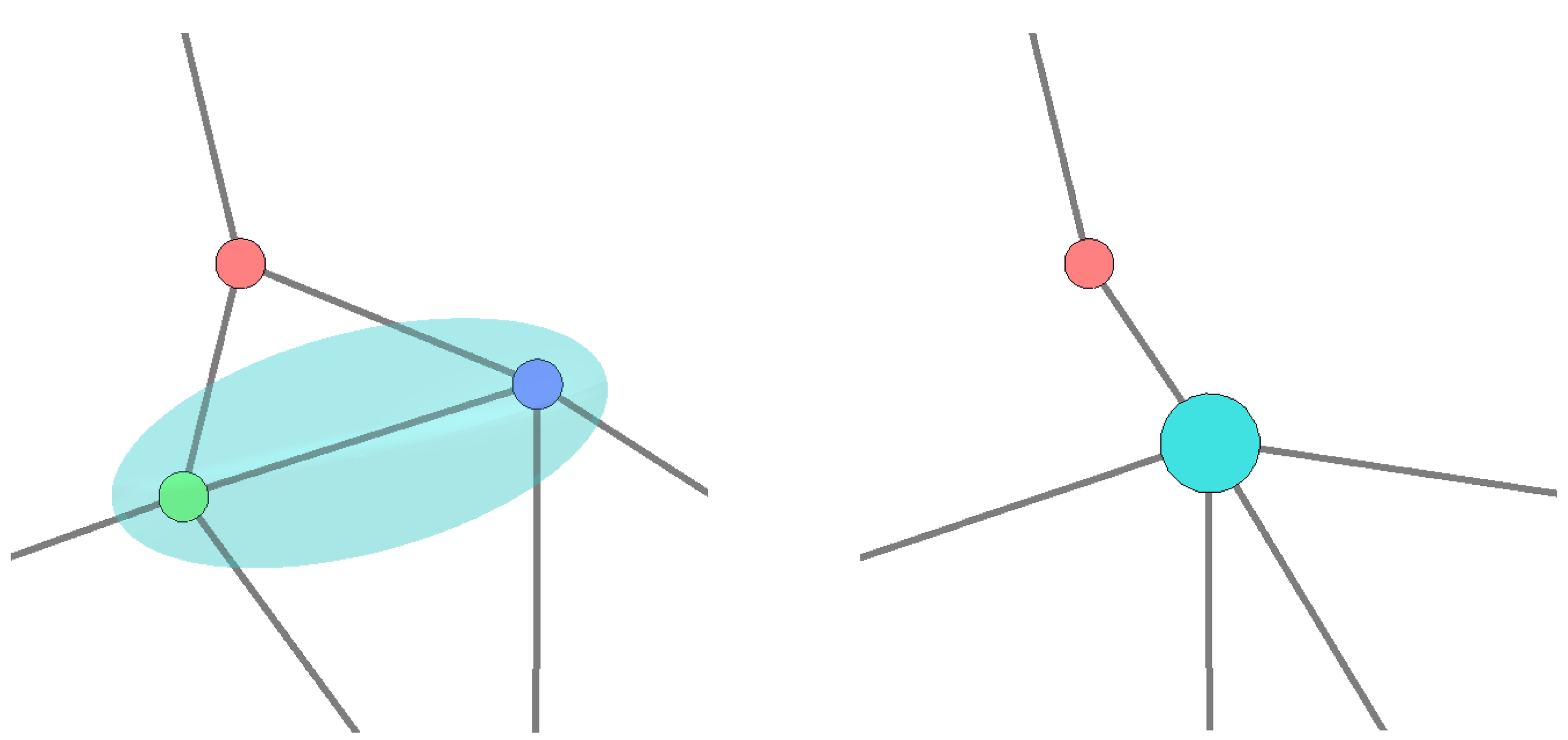
$$\mathcal{Q} = \frac{1}{T}\sum_{\mathcal{A}} \left[(1-\alpha_{\mathcal{A}})d_{\mathcal{A}} - \alpha_{\mathcal{A}}\sum_{\mathcal{A}'} a_{\mathcal{A},\mathcal{A}'}\right] = \frac{1}{T}\sum_{\mathcal{A}} d_{\mathcal{A}} - \sum_{\mathcal{A}} \alpha_{\mathcal{A}}^2$$

Assign to each edge a score η, a *surface-to-volume ratio* between $\mathcal{A}$ and $\mathcal{A}'$:

$$\eta_{\mathcal{A},\mathcal{A}'} = \frac{a_{\mathcal{A},\mathcal{A}'}}{\sqrt{d_{\mathcal{A}} d_{\mathcal{A}'}}} \quad (1) \qquad \text{or} \qquad \eta_{\mathcal{A},\mathcal{A}'} = a_{\mathcal{A},\mathcal{A}'}\,(log(d_{\mathcal{A}}^2 + d_{\mathcal{A}'}^2))^2\,\frac{(1-\alpha_{\mathcal{A}})(1-\alpha_{\mathcal{A}'})}{\alpha_{\mathcal{A}}\alpha_{\mathcal{A}'}} \quad (2)$$

Algorithm: Multilevel Graph Coarsening

- Initialize each $\mathcal{A} = \{i\}$, $d_{\mathcal{A}}$, $a_{\mathcal{A},\mathcal{A}'}$, and $\alpha_{\mathcal{A}}$ for each i. Initialize each $\eta_{\mathcal{A},\mathcal{A}'}$ and sort.
- Take the maximum $\eta_{\mathcal{A},\mathcal{A}'}$; merge aggregates $\mathcal{A}$ and $\mathcal{B}$ into a new aggregate $\mathcal{C}$. Compute $d_{\mathcal{C}}$, $a_{\mathcal{C},\mathcal{A}'}$, and $\alpha_{\mathcal{C}}$:
  $$d_{\mathcal{C}} = d_{\mathcal{A}} + d_{\mathcal{B}} + 2a_{\mathcal{A},\mathcal{B}} \qquad a_{\mathcal{C},\mathcal{A}'} = a_{\mathcal{A},\mathcal{A}'} + a_{\mathcal{B},\mathcal{A}'} \qquad \alpha_{\mathcal{C}} = \alpha_{\mathcal{A}} + \alpha_{\mathcal{B}}$$
- Recompute $\eta_{\mathcal{A},\mathcal{A}'}$ and update $\mathcal{Q}$. If $\mathcal{Q}$ had reached a maximum and has now started decreasing, create a partition.
- Iterate until there are no more edges, and return the heirarchy of partitions.



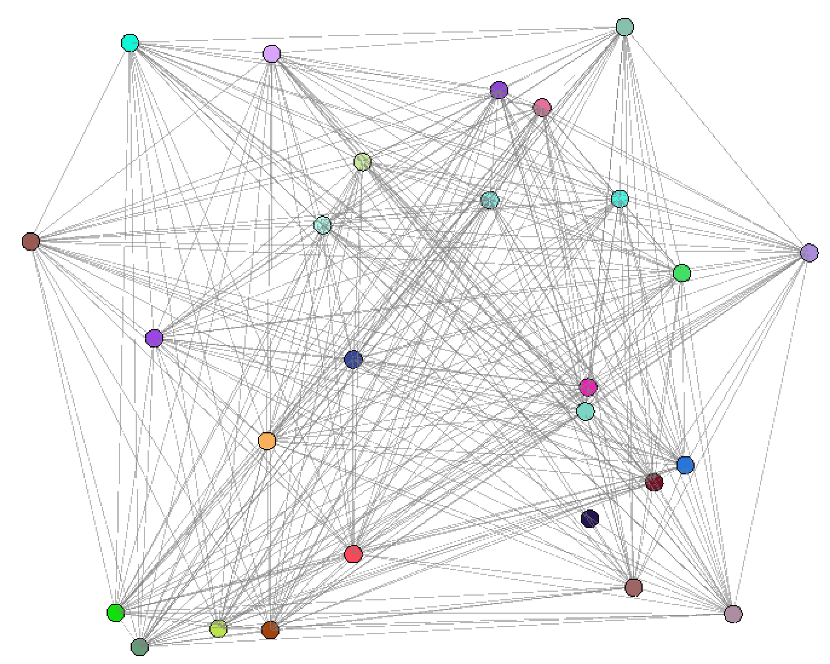
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Application: Embedding in $\mathbb{R}^d$

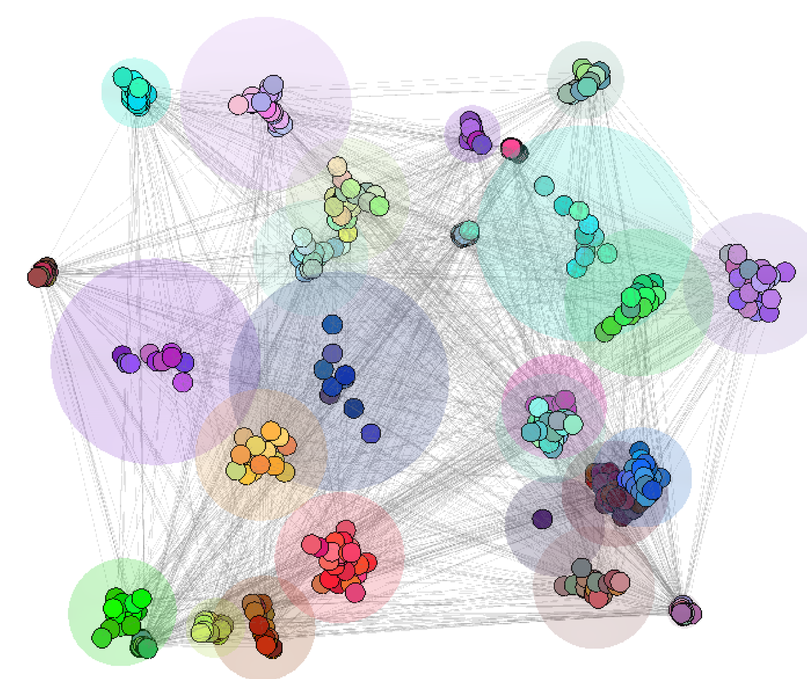
Using the hierarchy of coarser graphs, embed G in the unit d-cube $\subset \mathbb{R}^d$ for a given d.
Given an embedding for the coarsest level graph, move *coarse-to-fine* to assign coordinates to the fine-level vertices.
Assume: Each $\mathcal{A}$ has coordinates $\mathbf{x}^{\mathcal{A}} = (x_k^{\mathcal{A}})_{k=1}^d \in \mathbb{R}^d$.
Goal: Assign to each i a coordinate vector $\mathbf{x}_i = (\mathbf{x}_{i,k})_{k=1}^d \in \mathbb{R}^d$ maintaing well-separated aggregate structure.
For given initial values of $\mathbf{x}_i = (x_{i,k})_{k=1}^d$ and a radius $r_k^{\mathcal{A}}$ for each coarse vertex A and dimension k, we run over each vertex i and any fixed $k = 1, \ldots, d$ and apply the minimization process:

$$J_{loc}(t) = \sum_{j:\, e=(i,j)} \frac{1}{|t - x_{j,k}|} + \frac{1}{x_k^{\mathcal{A}} + r_k^{\mathcal{A}} - t} + \frac{1}{t - x_k^{\mathcal{A}} + r_k^{\mathcal{A}}} \mapsto \min,$$

for values of t such that $0 < x_k^{\mathcal{A}} - r_k^{\mathcal{A}} < t < x_k^{\mathcal{A}} + r_k^{\mathcal{A}} < 1$.
Then using the value $t = t_*$ we set $x_{i,k} = t_*$.



The coarse vertexices $\{\mathcal{A}\}$ at $\{x_{\mathcal{A}}\}$



The finer vertices $i \in \mathcal{A}$ at $\{x_i\}$, spread within the ball of radius $r_{\mathcal{A}}$

Algorithm: Embedding in $\mathbb{R}^d$

- Initialize $x_{i,k} = x_k^{\mathcal{A}} + \epsilon_{i,k} d_k^{\mathcal{A}}$, $\mathcal{A}: \; i \in \mathcal{A}$, $k = 1, \ldots, d$, with $\epsilon_{i,k} \in (-1, 1)$ a random number.
- Loop over each i and each $k = 1, \ldots, d$, minimizing $J_{loc}(t)$ to update $x_{i,k}$. Sort $\{x_{j,k}\}_{j:\, e=(i,j)}$, $x_k^{\mathcal{A}} - d_k^{\mathcal{A}}$, and $x_k^{\mathcal{A}} + d_k^{\mathcal{A}}$, and let these be $m+2$ in total. Then we have
  $$0 < x_{j_0,k} = x_k^{\mathcal{A}} - d_k^{\mathcal{A}} < x_{j_1,k} \le \cdots < x_{j_{s-1},k} < x_{j_s,k} \le \cdots < x_{j_{m+1},k} = x_k^{\mathcal{A}} + d_k^{\mathcal{A}} < 1.$$
  For each interval $t \in (x_{j_{s-1},k}, x_{j_s,k})$, the functional $J_{loc}(t)$ is convex and takes the form
  $$J_{loc}(t) = \sum_{r<s} \frac{1}{t - x_{j_r,k}} + \sum_{r\ge s} \frac{1}{x_{j_r,k} - t}.$$
  Its derivative and second derivative are
  $$\frac{dJ_{loc}}{dt} = -\sum_{r<s} \frac{1}{(t - x_{j_r,k})^2} + \sum_{r\ge s} \frac{1}{(x_{j_r,k} - t)^2}.$$
  $$\frac{d^2 J_{loc}}{dt^2} = 2\sum_{r<s} \frac{1}{(t - x_{j_r,k})^3} + 2\sum_{r\ge s} \frac{1}{(x_{j_r,k} - t)^3} > 0.$$
  Therefore the equation $\frac{dJ_{loc}}{dt} = 0$ has a unique solution $t = t_*$ in the interval $t \in (x_{j_{s-1},k}, x_{j_s,k})$.
- Iterate until desired, and then recursively apply to the next level in the heirarchy until the finest level has been embedded.

Acknowledgements

Stephan Gelever. `https://github.com/gelever/linalgcpp/`

Results and Other Applications

The partitioning algorithm achieved good results compared to the state of the art: Louvain's algorithm, which similarly monitors the modularity, $\mathcal{Q}$.

- Our algorithm maximizes $\mathcal{Q}$ better in almost all cases.
- The number of aggregates at the coarsest level of the algorithm remain similar (an important consideration for multilevel algorithms).

Maximum modularity $\mathcal{Q}$ of different $\eta_{\mathcal{A},\mathcal{A}'}$ and Louvain's algorithm on graphs of size $|V|$, $|E|$

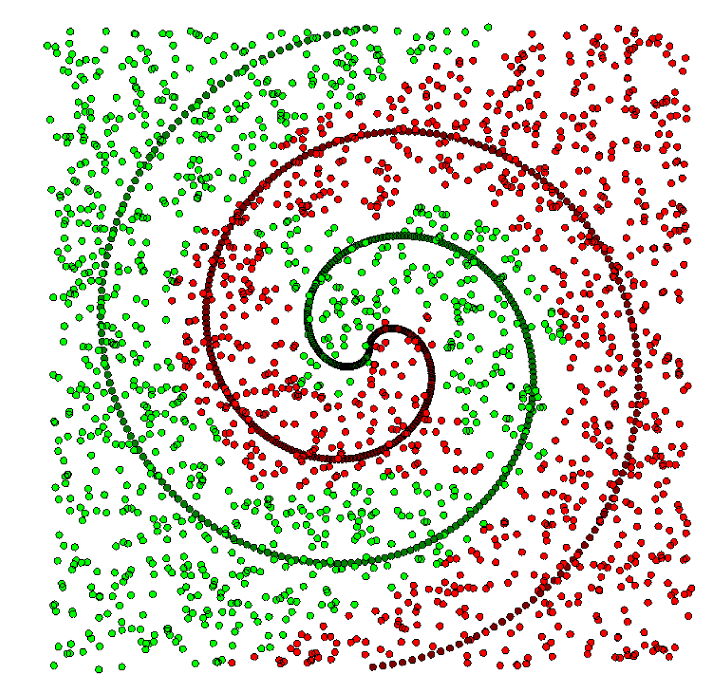
	35, 132	116, 1.2K	487, 4.4K	493, 3K	1.3K, 3K	1.8K, 18K	2.2K, 107K	14K, 763K	33K, 361K
$\eta_{\mathcal{A},\mathcal{A}'}$ (1)	0.7454	0.7523	0.6165	0.7014	0.9345	0.4863	0.5964	0.5923	0.7147
$\eta_{\mathcal{A},\mathcal{A}'}$ (2)	0.7373	0.7525	0.6253	0.7117	0.9368	0.5346	0.6246	0.6543	0.7489
Louvain	0.7337	0.6055	0.3367	0.5921	0.9056	0.1921	0.3517	0.4763	0.8055

Coarsest layer's $|\{\mathcal{A}\}|$ of different $\eta_{\mathcal{A},\mathcal{A}'}$ and Louvain's algorithm on graphs of size $|V|$, $|E|$

	35, 132	116, 1.2K	487, 4.4K	493, 3K	1.3K, 3K	1.8K, 18K	2.2K, 107K	14K, 763K	33K, 361K
$\eta_{\mathcal{A},\mathcal{A}'}$ (1)	8	13	61	104	125	856	553	2249	4265
$\eta_{\mathcal{A},\mathcal{A}'}$ (2)	6	13	30	65	124	37	40	51	67
Louvain	7	10	12	87	113	171	15	189	3907

We used aggregation for classification:

- Never merge two vertices with different labels
- Use the coarsest level to determine new labels



Unlabeled vertices (bright) with respect to a labeled double spiral (dark)

Aggregation following the dominated direction ($\theta = \frac{\pi}{6}$, $\epsilon = 0.001$) of the anisotropic diffusion operator.

The communities can also be used to help generate hierarchy of aggregates for use in *algebraic multigrid* (AMG). In particular, our algorithm generates aggregates that follow dominated direction for discretized anisotropic diffusion operator:

$$-\mathbf{div}\left((\epsilon I + \mathbf{b}\mathbf{b}^T)\nabla u\right)$$

for small ϵ and $\mathbf{b} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$ discretized using finite elements leading to non M-matrices.

Future Work

- Parallelization of aggregation via independent set
- Other multilevel algorithms (shortest path)

References

[N10] M.E.J. Newman, "*Networks. An Introduction*", Oxford University Press, New York, 2010.

[1] P. S. Vassilevski, "*Assigning Edge Weights in Graphs for Measuring Strength of Connectivity*" Presentation at Portland State University, 9 February 2015 (based on LLNL-PRES-663867).

[BAMGp] P. D'Ambra, S. Filippone, P. S. Vassilevski, "*BootCMatch: a software package for bootstrap AMG based on graph weighted matching,*" ACM Transactions on Mathematical Software (TOMS) **44**(4)(2018) Article No. 39, `https://dl.acm.org/citation.cfm?doid=3233179.3190647`.

[BAMGs] P. D'Ambra, S. Filippone, P. S. Vassilevski, "*Bootstrap AMG based on Compatible weighted Matching,*" `https://github.com/bootcmatch/BootCMatch`.