A scalable divide-and-conquer approach for network embeddings

DSG Team

December 2018

1 Patch Aligment via Synchronization

This section details the steps of the ASAP (As-Synchronized-As-Possible) algorithm, central to the divide-and-conquer pipeline we propose for the problem of recovering scalable network embeddings. Note that the difference between the original ASAP algorithm introduced in [1] and the approach described here lies in the graph decomposition method from Section 1.1, in the approach used to solve the local sub-problems (in this case, the GraphSAGE algorithm), and the scale synchronization step from Section 1.2.

The ASAP approach starts by decomposing the given graph G into overlapping subgraphs (referred to as *patches*), which are then embedded via the method of choice for (eg, the GraphSAGE algorithm). To every local patch embedding, there corresponds a scaling and an element of the Euclidean group Euc(d) of d-dimensional rigid transformations, and our goal is to estimate the scalings and group elements that will properly align all the patches in a globally consistent framework. The local optimal alignments between pairs of overlapping patches (whose intersection should be large enough - preferably tens of even hundreds of nodes) yield noisy measurements for the ratios of the above unknown group elements. Finding group elements from noisy measurements of their ratios is also known as the group synchronization problem. For this problem, Singer [3] introduced spectral and semidefinite programming (SDP) relaxations over the group SO(2) of planar rotations, which is a building block for the ASAP algorithm [1].

Table 1 gives an overview of the steps of the approach, Figure 1 shows a schematic view of our pipeline.

Algorithm 1 Modified ASAP algorithm that incorporates the scale synchronization step.

INPUT	$G = (V, E),\ \|V\| = n,\ \|E\| = m,\ MPS,\ d,\ OrdEmbed(\cdot)$
Choose Patches Embed Patches	1. Break G into N overlapping patches $P_1, \ldots, P_N$ following the steps in Sec. 1.1. 2. Embed each patch P_i separately via the method of choice (for eg, GraphSAGE).
Step 1 Scale	1. If a pair of patches (P_i, P_j) have enough nodes in common, let Λ_{ij} be the median of the ratios of distances realized in the embedding of P_i and their corresponding distances in P_j as in (1); otherwise set $\Lambda_{ij} = 0$. 2. Compute the eigenvector v_1^{Λ} corresponding to the largest eigenvalue of the sparse matrix Λ. 3. Scale each patch P_i by $v_1^{\Lambda}(i)$, for $i = 1, \ldots, n$
Step 2 Rotate & Reflect	1. Align all pairs of patches (P_i, P_j) that have enough nodes in common. 2. Estimate their relative rotation and possibly reflection $H_{ij} \in O(d) \subset \mathbb{R}^{d\times d}$. 3. Build a sparse $dN \times dN$ symmetric matrix $H = (H_{ij})$ where entry ij is itself a matrix in $O(d)$. 4. Define $\mathcal{H} = D^{-1}H$, where D is a diagonal matrix with $D_{1+d(i-1),1+d(i-1)} = \ldots = D_{di,di} = deg(i)$, $i = 1, \ldots, N$, where $deg(i)$ is the node degree of patch P_i. 5. Compute the top d eigenvectors $v_i^{\mathcal{H}}$ of $\mathcal{H}$ satisfying $\mathcal{H}v_i^{\mathcal{H}} = \lambda_i^{\mathcal{H}} v_i^{\mathcal{H}}, i = 1, \ldots, d$. 6. Estimate the global reflection and rotation of patch P_i by the orthogonal matrix $\hat{h}_i$ that is closest to $\widetilde{H}_i$ in Frobenius norm, where $\widetilde{H}_i$ is the submatrix corresponding to the i$^{\text{th}}$ patch in the $dN \times d$ matrix formed by the top d eigenvectors $[v_1^{\mathcal{H}} \ldots v_d^{\mathcal{H}}]$. 7. Update the embedding of patch P_i by applying the above orthogonal transformation $\hat{h}_i$
Step 3 Translate	Solve $m \times n$ overdetermined system of linear equations (3) for optimal translation in each dimension.
OUTPUT	Estimated coordinates $\hat{x}_1, \ldots, \hat{x}_n$

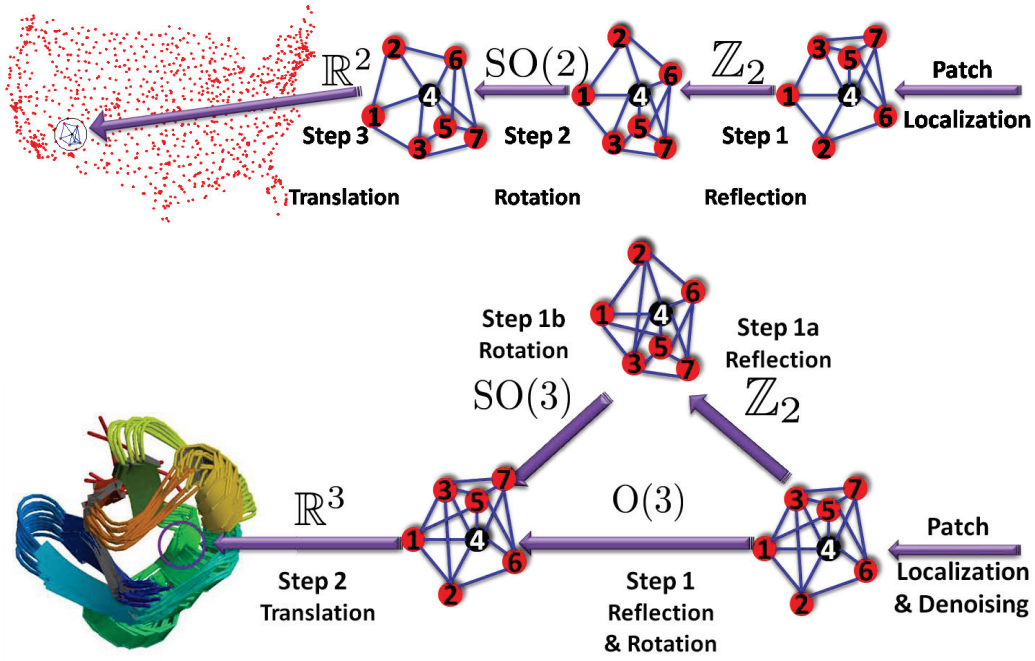

Figure 1: The ASAP recovery process in $d = 2$ and $d = 3$ dimensions, considered in [1], respectively [2].

1.1 Break up the kNN graph into patches and embed

The first step we use in breaking the graph into patches is to apply normalized spectral clustering [4] to a symmetrized version of the graph. Normalized spectral clustering partitions the nodes of a graph into $N \ll n$ clusters by performing k-means on the embedding given by the top N eigenvectors of the random-walk normalized graph Laplacian. It is shown [4] that normalized spectral clustering minimizes a relaxation of the normalized graph cut problem. Next, we enlarge the clusters by adding the graph-neighbors of each node, so that the resulting patches have significant overlap, a prerequisite for the ASAP synchronization algorithm. The higher the overlap between the patches, the more robust the pairwise group ratio estimates would be, thus leading overall to a more accurate final global solution.

1.2 Scale Synchronization

Before applying the original ASAP algorithm [1] to the embedded patches, we introduce an additional step that further improves our approach and is independent of the dimension d.

In the *graph realization problem* (GRP), one is given a graph $G = (V, E)$ together with a non-negative distance measurement d_{ij} associated with each edge, and is asked to compute a realization of G in $\mathbb{R}^d$. In other words, for any pair of adjacent nodes i and j, the distance $d_{ij} = d_{ji}$ is available, and the goal is to find a d-dimensional embedding $p_1, p_2, \ldots, p_n \in \mathbb{R}^d$ such that $\|p_i - p_j\| = d_{ij}$, for all $(i, j) \in E$.

In the GRP, the distances are readily available to the user and thus the local embedding of each patch is on the same scale as the ground truth. One only need to estimate the rigid transformation that aligns the local frame of each patch with the global solution. However, in the network embedding problem we are considering here, distances are unknown and the scale of one patch relative to another must be approximated. Any embedding approach has no way of relating the scaling of the local patch to the global scale. To this end, we augment the ASAP algorithm with a step where we synchronize local scaling information to recover an estimate for the global scaling of each patch, thus overall synchronizing over the group of similarity transformations.

We accomplish this as follows. Given a set of patches, $\{P_i\}_{i=1}^N$, we create a patch graph in which two patches are connected if and only if they have at least $d + 1$ (or more) nodes in common. We then construct a matrix

$\Lambda \in \mathbb{R}^{N\times N}$ as

$$\Lambda_{ij} = \begin{cases} \text{median}\left\{\frac{D^{P_i}_{a,b}}{D^{P_j}_{a,b}}\right\}_{a\neq b\in P_i\cap P_j} & \text{if } P_i \sim P_j,\ i \leq j, \\ 1/\Lambda_{ji} & \text{if } P_i \sim P_j,\ i > j, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $D^{P_i}_{a,b}$ is the distance between nodes a and b as realized in the embedded patch P_i.

The matrix Λ approximates the relative scales between patches. If all distances in all patches were recovered correctly up to scale, and all patches had sufficient overlap with each other, then each row of Λ would be a scalar multiple of the others and each column of Λ would be scalar multiple of the others, thus rendering Λ a rank-1 matrix. For the noisy case, in order to get a consistent estimate of global scaling, we compute the best rank-1 approximation of Λ, given by its leading eigenvector $v_1^{(\Lambda)}$. We use this approximation of global scaling to rescale the embedded patches before running ASAP. Note that the connectivity of the patch graph and the non-negativity of Λ guarantee, via the Perron-Frobenius Theorem, that all entries of $v_1^{(\Lambda)}$ are positive.

1.3 Optimal Rotation, Reflection and Translation

After applying the optimal scaling to each patch embedding, we use the original ASAP algorithm to integrate all patches in a global framework, as illustrated in the pipeline in Figure 2. We estimate, for each patch P_i, an element of the Euclidean group $\text{Euc}(d) = \text{O(d)} \times\mathbb{R}^d$ which, when applied to that patch embedding P_i, aligns all patches as best as possible in a single coordinate system. In doing so, we start by estimating, for each pair of overlapping patches P_i and P_j, their optimal relative rotation and reflection, i.e., an element H_{ij} of the orthogonal group O(d) that best aligns P_j with P_i. Whenever the patch embeddings perfectly match the ground truth, $H_{ij} = O_i O_j^{-1}$. We refer the reader to [1] for several methods on aligning pairs of patches and computing their relative reflections and rotations $H_{i,j}$. Finding group elements $\{O_i\}_{i=1}^N$ from noisy measurements H_{ij} of their ratios is also known as the group synchronization problem. Since this problem is NP-hard, we rely on the spectral relaxation [3] of

$$\min_{O_1,\ldots,O_N\in O(d)} \sum_{P_i\sim P_j} \|O_i O_j^{-1} - H_{ij}\|_F^2. \tag{2}$$

for synchronization over O(2), and estimate a consistent global rotation of each patch from the top d eigenvectors of the associated graph Connection Laplacian, as in Step 2.4 in Table 1. We estimate the optimal translation of each patch by solving, in a least squares sense, d overdetermined linear systems of the form

$$x_i - x_j = x_i^{(k)} - x_j^{(k)}, \quad (i,j)\in E_k, \quad k = 1,\ldots,N, \tag{3}$$

where x_i, respectively $x_i^{(k)}$, denote the unknown location of node i we are solving for, respectively, the known location of node i in the embedding of patch P_k. We refer the reader to [1] for a description of computing the optimal translations.

1.4 Extension to higher dimensions

Although we present experiments here on 2D and 3D data, the ASAP approach extends naturally to higher dimensions. In the 3D case, ASAP has been recently used as a scalable robust approach to the molecule problem in structural biology [2]. For the d-dimensional general case, one can extend ASAP by first using the same approach for scaling synchronization from Section 1.2, then synchronizing over O(d), and finally estimating the optimal translations over R^d by solving independently d overdetermined systems of linear equations via least-squares.

1.5 Summary of main steps for implementing the pipeline

Steps for generating a synthetic data set, for testing newly developed code:

1. Generate n=1000 random points in the $d = 2$ unit square (or any shape of your preference). Denote by B the embedding of the n points in $\mathbb{R}^2$.

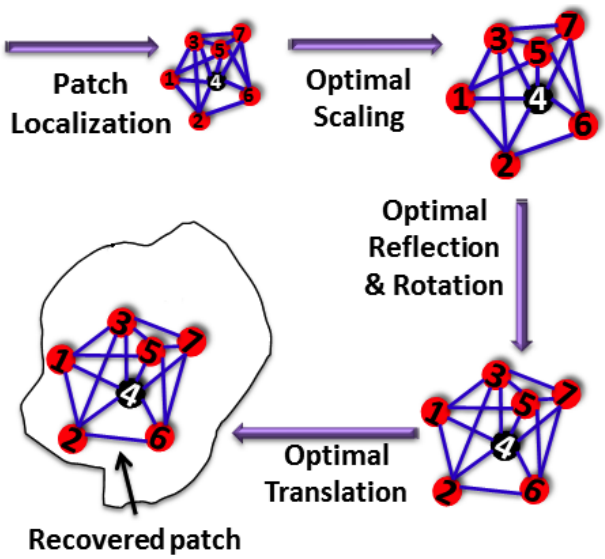

Figure 2: ASAP and scale synchronization pipeline.

2. Generate a disc graph G with a certain radius r, such that the resulting average degree $\bar{d}$ in the graph is relatively small (for eg., $\bar{d} = 50$). Recall that under the disc graph model, a pair of points is connected by an edge is they are less than distance r apart.

3. Run a clustering algorithm on G, and extract, say $k = 20$ (disjoint) clusters.

4. Enlarge each cluster to include its 1-hop neighbors. More specifically, for a given cluster C_i, consider the union of the 1-hop neighbots of each vertex $u \in C$

$$\mathcal{N}(C_i) := \cup_{u \in G} \mathcal{N}(u)$$

where $\mathcal{N}(u)$ denotes the set of neighbors of node u in G.

5. Update the vertex set of each cluster C_i, and denote its expanded version by $P_i, \forall i = 1, \ldots, k$.

6. For each cluster $\tilde{P}_i$ extract the embedding of its nodes from the initial ground truth embedding B, and denote this by E_i. For each embedding E_i, apply to it a random rotation matrix, followed by a shift along the x-axis and the y-axis. Note that a different randon rotation and shift will be applied to each initial embedding E_i. We shall refer to the clusters P_i and their corresponding updated embedding E_i as patches.

7. For each pair of patches that overlap in enough points[1], compute the Procrustes alignment of their overlapping nodes. For example, suppose we aim to align patches P_3 and P_7, say each of size 130 respectively 90. Suppose the common set of size has size 30; we then subset the 30 nodes from E_3 and E_7 and perform a Procrustes alignment of these two embeddings. In particular, we record the rotation $R_{3,7}$ (which is a $d \times d$ rotation matrix) and translation $T_{3,7}$ (which is a $d \times 1$ vector with the displacements for each dimension).

8. Denote by W the patch graph, which is a graph on k nodes, where each node corresponds to a patch. $W_{ij} = 1$ if patches i and j have been aligned, $W_{ij} = 0$ otherwise.

9. Consider the matrix R of pairwise rotations built above. Note that R is a "matrix of matrices", where entry (i, j) is itself a matrix. Thus altogether R is of dimension $kd \times kd$. To synchronize over O(d), consider the top d eigenvectors of R (or some normalized version of it), and collect them in a tall matrix of size $kd \times d$, whose columns are the top eigenvectors of R. Slice-up this tall matrix into k blocks of size

[1]Technically, in dimension d we only need $d + 1$ points, but for robustness we shall require more than 3 points in dimension $d = 2$, for example we shall align patches that only have at least 10 nodes in common

$d \times d$. For each block, compute the closest proper rotation matrix (can be achieved via a simple SVD). The end result of this procedure constitutes the solution to the synchronization problem over the orthogonal group.

10. Perform the synchronization over the translations, as detailed in the main text.

11. For each patch P_i, apply to its embedding E_i, its corresponding above-recovered rotation matrix and translation/shift. Collect the final recovered embedding into a matrix S of size $n \times d$.

12. At this point, you should have recovered the original global embedding E (of size $n \times d$), or a close approximation of it, if noise has been added to the problem. To this end, compute a final Procrustes alignment between E and S, and plot the two embeddings side by side, together with the alignment

References

[1] M. Cucuringu, Y. Lipman, and A. Singer, *Sensor network localization by eigenvector synchronization over the Euclidean group*, ACM Trans. Sen. Netw. **8** (August 2012), no. 3, 19:1–19:42.

[2] M. Cucuringu, A. Singer, and D. Cowburn, *Eigenvector synchronization, graph rigidity and the molecule problem*, Information and Inference **1** (2012), no. 1, 21–67.

[3] A. Singer, *Angular synchronization by eigenvectors and semidefinite programming*, Appl. Comput. Harmon. Anal. **30** (2011), no. 1, 20–36.

[4] U. von Luxburg, *A tutorial on spectral clustering*, Statistics and computing **17** (2007), no. 4, 395–416.