Simultaneous clustering and layout

Convex clustering

Given observations $x_i \in \mathbb{R}^d$, the convex clustering problem is

$$\arg \min_{U \in \mathbb{R}^{d \times N}} \sum_{i=1}^{N} ||X_i - U_i||^2 + \lambda \sum_{i,j=1}^{N} W_{ij} \Omega(U_i, U_j) \tag{1}$$

where W_{ij} are coupling weights, and Ω encourages coupled observations to share the same centroid. Ω is often chosen to be the ℓ_q-norm, with $q \geq 1$. The solution path of $U^{(\lambda)}$ is analogous to the coalscence path recovered by hierarchical clustering, and the regularization parameter λ corresponds to the different levels in the hierarchical clustering dendrogram. For example, for $\lambda = 0$, the solution to Eq. 1 is $U^{(0)} = X$ - each point x_i belongs to its own cluster. As λ increases, the penalty term induces the centroids u_i to fuse, until in the limit, these centroids reach a consensus and form a single cluster.

The objective of the above problem is strictly convex, which guarantees the existence of a globally optimal solution, as well as its robustness to perturbations.

Convex clustering of graph-structured data

In the context of VLSI placement, the data corresponds to a weighted-hypergraph (i.e. a vector-representation of cells is not immediately obvious). There are a variety of methods to approximate this hypergraph with a graph (e.g. clique, star, path, etc.). Under the star framework, the graph is assumed to be characterized by a sparse, SPD similarity/adjacency matrix K between N elements (cells). Note that by application of the spectral lemma, K corresponds to a *gram matrix*, and we can re-write K: $K = \Phi^T \Phi$, $K_{ij} = \Phi(x_i)^T \Phi(x_j)$. Additionally, each cluster centroid lies in the convex hull of its corresponding vectors: $U = \Phi(X)\pi$, where π is *doubly stochastic* (rows and columns sum to one). More concretely, $\pi \mathbf{1} = \mathbf{1}$, $\mathbf{1}^T \pi = \mathbf{1}^T$, and $\pi \geq 0$. In other words, the columns of π correspond to the centroid representation in the space spanned by the data, and the rows of π may be interpreted as soft membership assignments of observations to clusters.

These constraints add additional complexity to the problem. The Birkhoff-polytope (set of all doubly stochastic matrices) projection operator is implemented according to the *Bregmanian Bi-Stochastication (BBS)* algorithm proposed by Wang et al., 10. One question is if these constraints are practically necessary, and if we could instead just replace them with computationally easier structured sparsity regularization terms.

Using the kernel trick, Eq 1 can be adapted to

$$\arg \min_{\pi \in \Delta_N} Tr[\pi^T K \pi - 2K\pi] + \lambda \sum_{i,j=1}^{N} K_{ij} \Omega(\pi_i, \pi_j) \tag{2}$$

where Δ_N is the set of doubly stochastic matrices. In Donnat and Holmes, 19, Ω is taken to be a mixed total-variation penalty: $\Omega(\pi_i, \pi_j) = \alpha||\pi_i - \pi_j||_{2,1} + (1-\alpha)||\pi_i - \pi_j||_1$. An analysis of the dual problem and a linearization of the objective facilitate the application of a number of efficient gradient-based methods to solve the problem.

Force-directed placement

Modern techniques for analytic placement typically rely on quadratic optimization with terms associated with attraction of connected cells, and repulsion of overlaping cells. A typical approach is to represent the netlist as a system of springs that apply force to each node according to a $\text{weight} \times \text{distance}$ formula, causing highly connected cells to be attracted to one another. Repulsion is often applied between overlapping nodes to reduce density. For example, a naive stress objective on a weighted graph W corresponds to the following:

$$\begin{aligned} \text{stress}(X) &= \sum_{i<j} Q_{ij}(X), \\ Q_{ij}(X) &= w_{ij}(||x_i - x_j|| - d_{ij})^2 \end{aligned} \tag{3}$$

Can we efficiently simultaneously place and cluster?

I want to do something like block coordinate descent & FISTA for cluster assignments π and cell positions X.

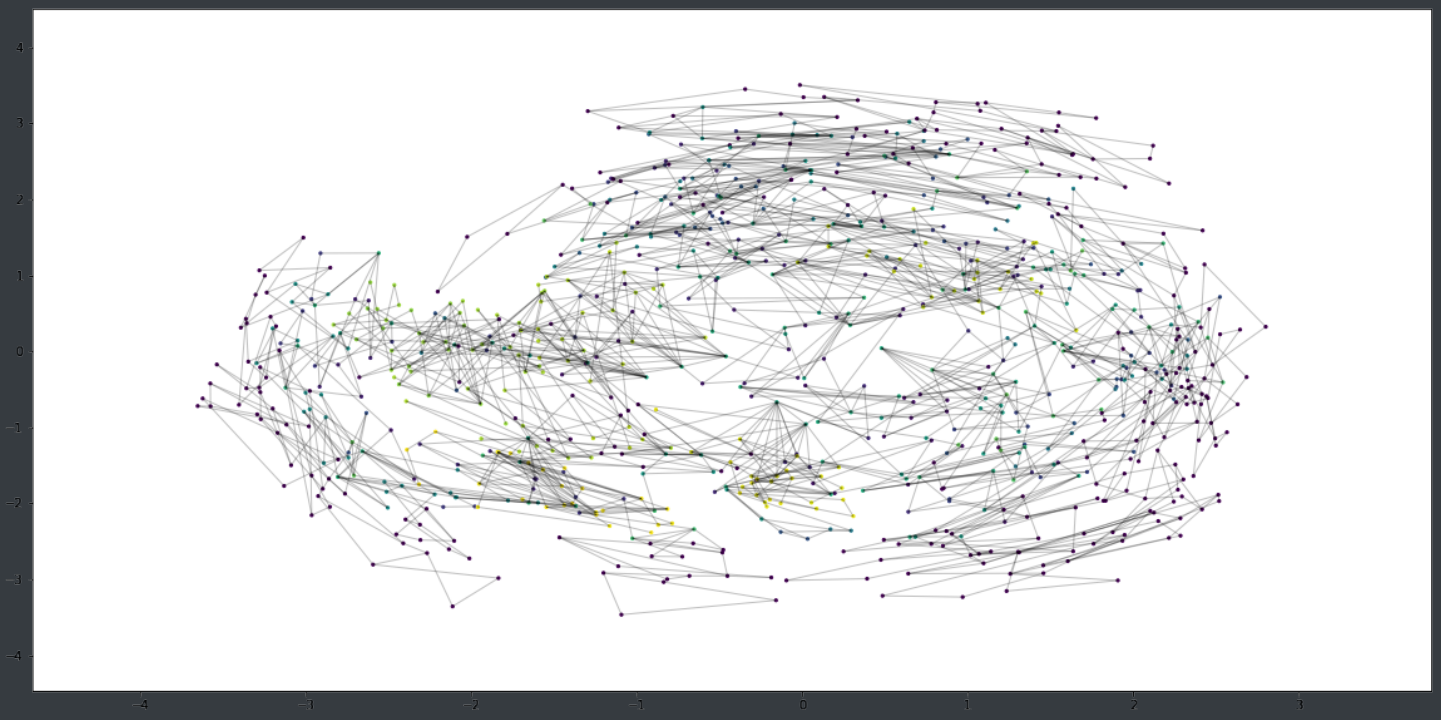
Want to avoid "un-vectorizable" quadratic ops. Need to scale to >100k - million nodes. The bottleneck is the clustering objective (K) and the BKP projection operator.

1. Alternating minimization for cluster assignments π and cell positions X:
    1. Initialize π such that all cells belong to one cluster, initialize X randomly, or seeded.
    2. Update π via FISTA
    3. Assign blocks to clusters according to assignment distributions induced by the rows of π.
    4. Update X via block cd for each cluster & iterate
2. The Birhoff projection operator takes a long time to converge, and may not produce an actual doubly stochastic matrix - I apply the sparsemax operator to the rows of π before sampling assignments. The sparesmax operator is defined as the following convex optimization problem wher Δ^{K-1} corresponds to the $K-1$-dimenssional simplex.

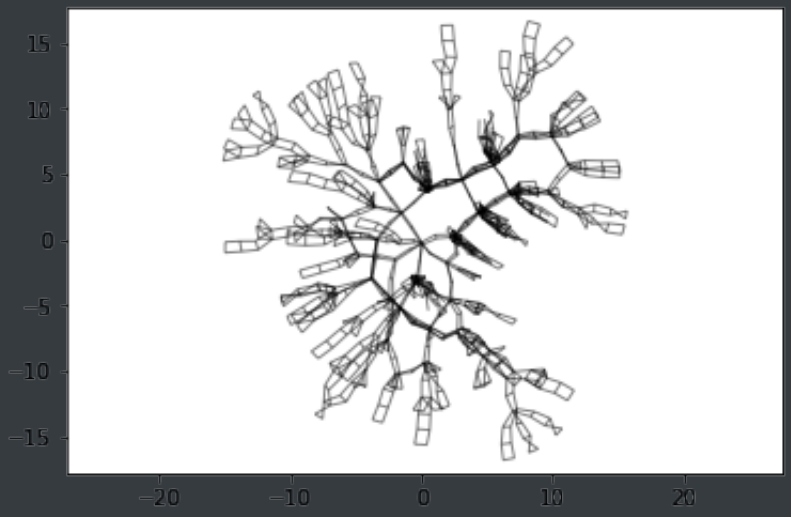
$$\text{sparsemax}(x) = \arg\min_{x \in \Delta^{K-1}} ||p - x||^2 \tag{4}$$

Initial results:

Stress + CCG CD minimization via projected block coordinate descent on the qh882 graph:



min-stress layout from Zheng et. al., 18 ()



Edge crossing minimization

Edge crossing minimization is well-studied in the context of VLSI routability optimization and graph layout. I adopt a smooth technique inspired by the soft SVM objective to regularize edge crossings as proposed by Shabbeer et al., 12.

Consider two edges $\mathcal{A}$ and $\mathcal{B}$ defined by endpoints $a = [a_x, a_y], c = [c_x, c_y] \in \mathbb{R}^2$ and $b = [b_x, b_y], d = [d_x, d_y] \in \mathbb{R}^2$ respectively. Note that *any* point on these edges may be written as a convex combination of the extreme points. For these edges to *not* intersect, there necessarily must be a $u \neq 0$ and γ such that $x'y - \gamma$ is nonpositive for all x lying on one edge, and nonnegative for all x lying on the other edge. We denote

the hyperplane defined by $\{x | x \in \mathbb{R}^2, x'u = \gamma\}$ the *separating hyperplane,* and we want to introduce a regularizer which encourages $\mathcal{A}$ and $\mathcal{B}$ to lie in different half-spaces.

We recover the following property: two edges do not intersect if and only if the following system has *no solution*.

Let $A = \begin{bmatrix} a_x & a_y \\ c_x & c_y \end{bmatrix}$ and B defined accordingly. Intuitively, two edges don't intersect if there is no point (a convex combination of the extreme points) shared by both edges.

$$\exists \delta_A, \delta_B \in \mathbb{R}^2 \text{ such that } A'\delta_A = B'\delta_B \quad e'\delta_A, e'\delta_B = 1 \quad \delta_A, \delta_B \geq 0 \tag{5}$$

Farkas' Lemma then provides the conditions that are satisfied if the edges do not cross, i.e. that (5) has no solution (Farkas Lemma: For matrix D, the linear system $Du \leq 0$, $d'u \geq 0$ has no solution iff $D'v = d$, $v \geq 0$ has a solution.):

$$\begin{gathered} Au \geq \alpha e \quad Bu \leq \beta e \quad \alpha - \beta > 0 \\ \implies Au - \gamma e \geq e, \quad Bu - \gamma e \leq -e \end{gathered}$$

More generally two edges do not intersect if and only if

$$0 = \min_{u,\gamma} f(A, B, u, \gamma) = \min_{u,\gamma} ||(-Au + (\gamma + 1)e)_+|| + ||(Bu - (\gamma - 1)e)_+||$$

Including these terms as regularizers, we get the following layout: