Approximating p-centres in large δ-hyperbolic graphs
A quasi-linear time algorithm for distance-based clustering
Katherine Edwards¹, W. Sean Kennedy², Iraj Saniee²

Clustering via p-centres

The p-centre algorithm is a discrete variant of one of the most frequently used clustering algorithms, the k-means clustering. The goal of the p-centre algorithm is to identify on a given graph a p−specified number of vertices or centers, such that the maximum distance of any graph vertex to its nearest center is minimized. For any given p, the algorithm naturally partitions a graph into p clusters induced by the position of its p centers. Unfortunately, as a clustering algorithm the complexity of the p-centre algorithm is generally prohibitive, O(n^p) for an n−vertex graph, making it inapplicable to even moderate size graphs.

In this work we show how to approximate p-centres in a class of graphs which are common in real data, the δ−hyperbolic graphs. We show that by giving up in the (additive) approximation, one can achieve a quasi-linear time p-centre approximation. As such, this scheme is the first p-centre approximation applicable to large graphs, particularly when p is relatively small, for example in the range 10−10⁴ and n is large, for example, 10−10⁵ vertices.

The p-centre Problem

Given input graph G and integer p:
- Find p centres such that the p−radius is minimized
- Optimal value of r is called the p−radius r_p
- Centres may be vertices or lie on edges

Known Algorithms

In general graphs, p-centres is NP−hard. In fact it is NP−hard to approximate the p−radius to a factor smaller than 2. A (multiplicative) 2−approximation exists in time O(m log n).

In trees, p-centers can be solved exactly in linear time O(m).

In δ-hyperbolic graphs, previous result: p-centres can be solved in time O(n^3) with an additive error at most δ on the p−radius [1].

δ-hyperbolicity

Hyperbolicity is an invariant of a graph which roughly measures how close its distance metric is to the distance metric of a tree.

Definition: Let x,y,z be any three vertices in G and let \( d(x,y), d(x,z), d(y,z) \) be three shortest paths. The union of the paths is called a geodesic triangle.

\[ \text{Let the perimeter } \pi = d(x,y) + d(y,z) + d(x,z) \text{ and define } \]
\[ a_1 = \frac{1}{3} \pi - d(x,y) \]
\[ a_2 = \frac{1}{3} \pi - d(x,z) \]
\[ a_3 = \frac{1}{3} \pi - d(y,z) \]

The points \( m_x, m_y, m_z \) are located where the inscribed circle would meet the edges of a triangle with side lengths \( d(y,z), d(x,z) \) and \( d(x,y) \).

The insize of a geodesic triangle is \( \max_{i=1,2,3} \min_{j \neq i} d(p,q) \) where \( p,q \) are points on the geodesic triangle that are both at distance \( a_i \) from \( v \).

The hyperbolicity \( \delta \) of G is the maximum insize of a geodesic triangle.

Fact: Trees are 0-hyperbolic.

Fact: Graphs arising from real networks have small hyperbolicity. Kennedy et al. [3] studied a large number of publicly available graphs arising from social media, collaboration and citation networks, IP−layer networks and web graphs. They found that as the size of these graphs grows very large, their hyperbolicities \( \delta \) remain small (less than 10). This is in contrast with the random graph, which has logarithmic hyperbolicity.

So in this sense, real−world graphs are tree−like.

The objective

Real graphs are big: a quadratic−time algorithm is too slow in practice on a billion−vertex graph.

Real graphs are δ−hyperbolic: can get an additive constant error on the p−radius for graphs with fixed \( \delta \). We want an approximation algorithm with a small additive approximation error that runs in nearly linear time.

Our results

In δ−hyperbolic graphs:

Theorem: If \( p \in \{1,2\} \) and G is δ−hyperbolic: there is an approximation algorithm with
- additive error at most \( \delta \) on the p−radius
- running time \( O((2 + 1/(m + n)) \)

Theorem: If \( p \geq 3 \) and G is δ−hyperbolic: there is an approximation algorithm with
- additive error at most \( \delta \) on the p−radius
- running time \( O(p(\delta + 1)/(m + n) \log n) \)

Proof Ideas

Hyperbolic graphs are similar to trees, a class of graphs where p−centres is easy. So we want to exploit this similarity.

The p-centre problem is a covering problem: we want to cover the vertices of G with p balls of a smallest possible radius. The natural dual is a packing problem, known as (p+1)−dispersion, which asks for p+1 disjoint balls each with a largest possible diameter.

The (p+1)−dispersion Problem

Given input graph G and integer p+1:
- Find p vertices \( x_1, ..., x_p \) maximizing \( d(x_1, ..., x_p) \)
- Adding value of \( r(c_1, ..., c_p) \) is called the (p+1)−diameter \( d_{p+1} \)

In general graphs: \( r_p \geq d_{p+1} \) This follows from the pigeonhole principle (but \( r_p \) and \( d_{p+1} \) may be arbitrarily far apart).

In trees: \( r_p = d_{p+1} \)

In δ−hyperbolic graphs: \( r_p \leq d_{p+1} + \delta \)

Previous work on δ−hyperbolic graphs solved the primal p−centre problem and its dual (p+1)−dispersion problem simultaneously, resulting in a solution to p−centres with an additive error of \( \delta \) on the p−radius \( r_p \). But this cubic time algorithm is prohibitively slow.

Locally Dispersed Sets

Our key observation is that starting from a solution to a 'local' version of dispersion, we can easily obtain p-centres with an additive error of at most \( 3\delta \) on the p−radius. For small values of \( p \in \{1,2\} \) our additive error is just \( \delta \).

In the local (p+1)−dispersion, rather than look for \( p+1 \) vertices which are pairwise as far apart as possible in G, we look for a set of \( p+1 \) vertices which we can't improve by swapping a vertex in our set with a new vertex (i.e. by making a local improvement).

We give an algorithm to find an optimal locally dispersed set by performing a quasilinear number of vertex swaps. We then show how to optimize the approximate p-centres from it in constant time.

References & Contact

¹ University of Rome La Sapienza - katherine.edwards2@gmail.com
² Nokia Bell Labs - kennedy, is@research.bell-labs.com

