Research Project Brief

Anthony Pizzimenti

Computer Science, Mathematics - The University of Iowa

anthony-pizzimenti@uiowa.edu

August 30, 2018

Contents

1 k-means and Spanning Trees

1.1 Overview

Given a graph $G = (V, E)$ where all $v \in V$ are weighted with some set of data, how can we find districting plans (based on spanning trees) that match our desired criteria (e.g. population balance, demographic balance, etc.)?

Proposition 1.1 (Deterministic Tree Generation)**.** Given V, use a k-means algorithm to create n clusters of points in V based on some weighting (population, average population, voting population, demographic data, etc.). Then, for each cluster, find a spanning tree T_k, where $k \in \{1, \ldots, n\}$. Finally, draw $n - 1$ edges between the clusters' spanning trees and call this new graph T. As we

have added $n-1$ edges, this is maximally acyclic on V; thus, T is a spanning tree. Using this divide-and-conquer method may be advantageous, as sampling randomly from spanning trees is costly, especially in terms of computing time and power.

However, we can possibly change this approach in two ways.

Proposition 1.2 (Deterministic Clustering, Random Tree Sampling)**.** Given V, use a k-means algorithm to create n clusters of points in V based on some weighting, as before. Then, for each cluster, sample a random spanning tree T_k, where $k \in \{1, \ldots, n\}$. Finally, draw $n-1$ edges between the clusters' spanning trees and call this new graph T. As we have added $n-1$ edges, this is maximally acyclic on V; thus, T is a spanning tree.

Proposition 1.3 (Tree Clustering)**.** Using either (1.1) or (1.2), generate a set of spanning trees $\mathcal{T}$. Then, given a metric $d(T_1, T_2)$ on $\mathcal{T}$ (maybe Hausdorff distance, based on embedding?), cluster the spanning trees themselves.

1.2 Challenges

A number of challenges crop up with this problem. Firstly, by using this method, we are no longer able to sample uniformly randomly from the set of spanning trees, as k-means algorithms are deterministic and tree-sampling algorithms (such as Broder's or Wilson's) are not. Secondly, as this is a divide-and-conquer approach, how can we implement our computations in parallel? Thirdly, how do we not violate the contiguity rule (that each cluster's points must be adjacent to each other in the original graph)? What is an appropriate measure of closeness? And finally, once we have found these trees, how can we serialize them, so as to minimize computational power and storage use?

1.3 Benefits

As readily as there are challenges, there are benefits. Firstly, this approach will be far faster than sampling from spanning trees and throwing out those that don't meet our criteria. By finding clusters that match criteria first and constructing a spanning tree from those clusters, we can more effectively seed Monte-Carlo Markov chain methods. Secondly, we can potentially run algorithm (1.1) in parallel:

```
1   // algorithm 1.1
2
3   // input(s): a set of weighted (and possibly embedded)
4   // vertices V; a set of edges E; number of components k;
5   // set of boundary edges B ⊂ E.
6
7   // output: a spanning tree with main components generated
8   // by k-means clusters.
9
10  function find_tree(V, E, B, k):
11      C = clusters(k)     // C is a set of clusters; each cluster is
12                          // made up of (possibly embedded) vertices.
13
14      T := parallel(
15          Dijkstra(c ∈ C) // run Dijkstra's algorithm on each c in C
16      )                   // in parallel; we will need to split k tasks
17                          // across n compute nodes; returns a set of trees
18                          // T that can be joined into one tree with the
19                          // addition of edges
20
21      added_edges := 0
22      while added_edges < k - 1:
23          edge := sample(B)   // randomly sample an edge from B
24          add_edge(T, edge)   // add this edge to the set of trees
25          added_edges += 1
26
27      return T
```

We can do the same with algorithm (1.2) by changing `Dijkstra` to `Wilson` on line 15. Algorithm (1.3), however, cannot be parallelized.

2 Tabu Search

2.1 Overview

Our current Markov chain model has a number of undesirable characteristics:

1. It has a poor step proposal procedure, resulting in approximately 7 steps being discarded for every 10 proposed;
2. We may be moving away from districting plans that are desirable;
3. It may not be possible to explore the entire space by starting from a random plan;
4. Computations take an incredibly long time, resulting in slow performance.

Remark. We will explore (3) in Section 3, and an exploration of improving computational performance (4) will accompany any revisit of the Markov chain model.

To address (1) and (2) in one fell swoop, we can introduce a new way to propose steps - the Tabu search. By expressing the exploration of plans as a combinatorial optimization problem (or modifying the Tabu search algorithm to meet our existing model's needs), we can mitigate both (1) and (2). The underlying principle of Tabu search is that it does not get stuck, as some optimization methods (e.g. gradient descent) do - consequently, the Tabu search will always iteratively progress until we've determined the current solution is "good enough," effectively alleviating (1). Furthermore, this approach always moves toward a solution, which meaningfully addresses (2).

2.2 Challenges

Most of these challenges are theoretical and require developing rigorous mathematical structure to effectively entertain them. A number of questions include:

1. How are we to reduce this search to a combinatorial optimization problem? (Or do we need to?)
2. What is an optimal solution with respect to the set of possible districting plans?
3. Will we re-think our Markov chain approach to incorporate principles of Tabu search?
4. How can we define the notion of a neighborhood in the space of plans (i.e. how do we express the space of plans as, at the very least, a topological space)?

2.3 Benefits

There is clear theoretical value to exploring the questions outlined in Section 2.2. Building on this theory may be fundamental to making breakthroughs in this area, as using this method of step proposal will dramatically increase the number of valid, "important" plans we're able to find. Furthermore, we may be able to use this theory to help address questions in Section 3. Finally, at the very least, we will drastically improve the speed of our current approach.

3 (Ir)reducibility of Markov Chain Approach

3.1 Overview

Is it possible that some states in the state space are unreachable from certain starting states? This may be true, especially considering some edge cases (e.g. a "cornered" district or a cyclic ordering).

3.2 Challenges

This is undoubtedly the most mathematically rigorous of the three problems presented here. How do we go about showing that some states are unreachable from others? Is there an algebraic structure (groups, groupoids, etc.) that can describe the state space effectively? Are there equivalence classes we can identify?

3.3 Benefits

The theoretical implications of exploring this problem are quite promising. I think that, regardless of which problem we choose to investigate first, this problem will be touched upon in one way or another.