# Optimal Placement of Speed Feedback Signs in the City of Boston

**Adriana D'Souza, Brian Roach, Jessica McAloon, Monica Chiu**

## Introduction and Motivation

Our team set out to determine an optimal placement of Speed Feedback Signs in the City of Boston. To do this, we analyzed where accident hotspots were around the city (further referred to as clusters), and where vulnerable areas are. Despite a plethora of types of potentially vulnerable areas, we decided to focus on schools, hospitals, and open spaces, such as parks. We used these locations collectively as 'triggers', or sites of equal weight in our scoring algorithm. We also clustered accidents to get locations where accidents frequently occur, and used these locations as another trigger of equal weight. We hope to solve this problem by placing signs so that the average distance between each sign and its nearby triggers is minimized, in order to protect vulnerable areas. We also have a constraint that each sign must be at least half a mile apart, so our signs don't clump in one area. This problem was previously hosted as an open data challenge, but we sought to give it an additional look.

## Datasets:

Original datasets:

- Motor Vehicle Accidents (Analyze Boston)
- Hospital Locations (Analyze Boston)
- Street Intersections (Boston Open Data)
- Open Spaces (Boston Open Data)
- Schools (boston.opendatasoft.com)

Datasets assembled manually from the above sets:

- Nodes (fetch_nodes.py) - Collects possible places to place zoning signs based on intersection locations.
- Clean Triggers (clean_triggers.py) - Collects and cleans accident clusters, schools, open spaces, hospitals, candidate intersections for placement for use as points in the k-means clustering done in get_signal_placements.
- Accident Clusters (get_accident_clusters.py) - Performs k-means on the input accidents to reduce accidents into accident clusters, which are later used as points of influence as to where feedback signs should be placed.
- Signal Placements (get_signal_placements.py) - Consumes the triggers produced by clean_triggers to determine the optimal placement of speed.
- Average Distances (get_avgs.py) - Gets average distances between signs and each trigger.

## Process

We categorized our approach into two parts.

**Part 1** - Placement of Speed Feedback Signs

Phase 1:

- Cluster accidents into accident hot spots via k-means, where the number of means is proportional to the number of input nodes.
- To look at signal placement results, open placements.html within the same directory.

Phase 2:

- Filter intersections by proximity to accident clusters. For an intersection to be a candidate placement site, the intersection must be in the lower 50th percentile with regards to distance to closest accident cluster. This ensures that final placements are not skewed by proximity to vulnerable sights alone, but must also be close to where accidents are known to occur.

Phase 3:

- We consider school, hospital, and open space locations, as well as accident cluster locations, as equally weighted 'triggers', or data points. We then run k-means on this data set, with k = 30 (hard-coded, imagining 30 of these signs are available). We then find the candidate intersection closest to each of these determined means (output of k-means), and output these 30 intersections as the sites of the speed feedback sign placements.

**Part 2** - Statistical Analysis

In order to ensure that the placement of speed feedback signs would not be affected by any other variables that can be changed, namely, the number of clusters that is chosen to create the accident hotspots (from Phase 1 of Part 1), we must determine a reasonable number of clusters, a, that gives the most optimal placement of s signs.

Phase 1:

- For values from B = {200 to 80 in multiples of 10}, run all of Part 1 for each value of the "cluster_divisor" variable, c, in get_accident_clusters being a value within the set B (i.e., c will be 200, then 190, etc.). As you run these parts, get_avgs will store the average distances of each sign placed to schools, parks, hospitals, and accident clusters nearby for each value of c.

Phase 2:

- For each value of c, find the average of the averages of all the triggers.

Phase 3:

- An optimal value for a is one that helps in minimizing the distance of a sign to other triggers, so the best value of a would be the number of accidents divided by c, which will be the number of clusters that minimizes the average overall distances of all trigger distances.

## Issues and Limitations

Current known issues:

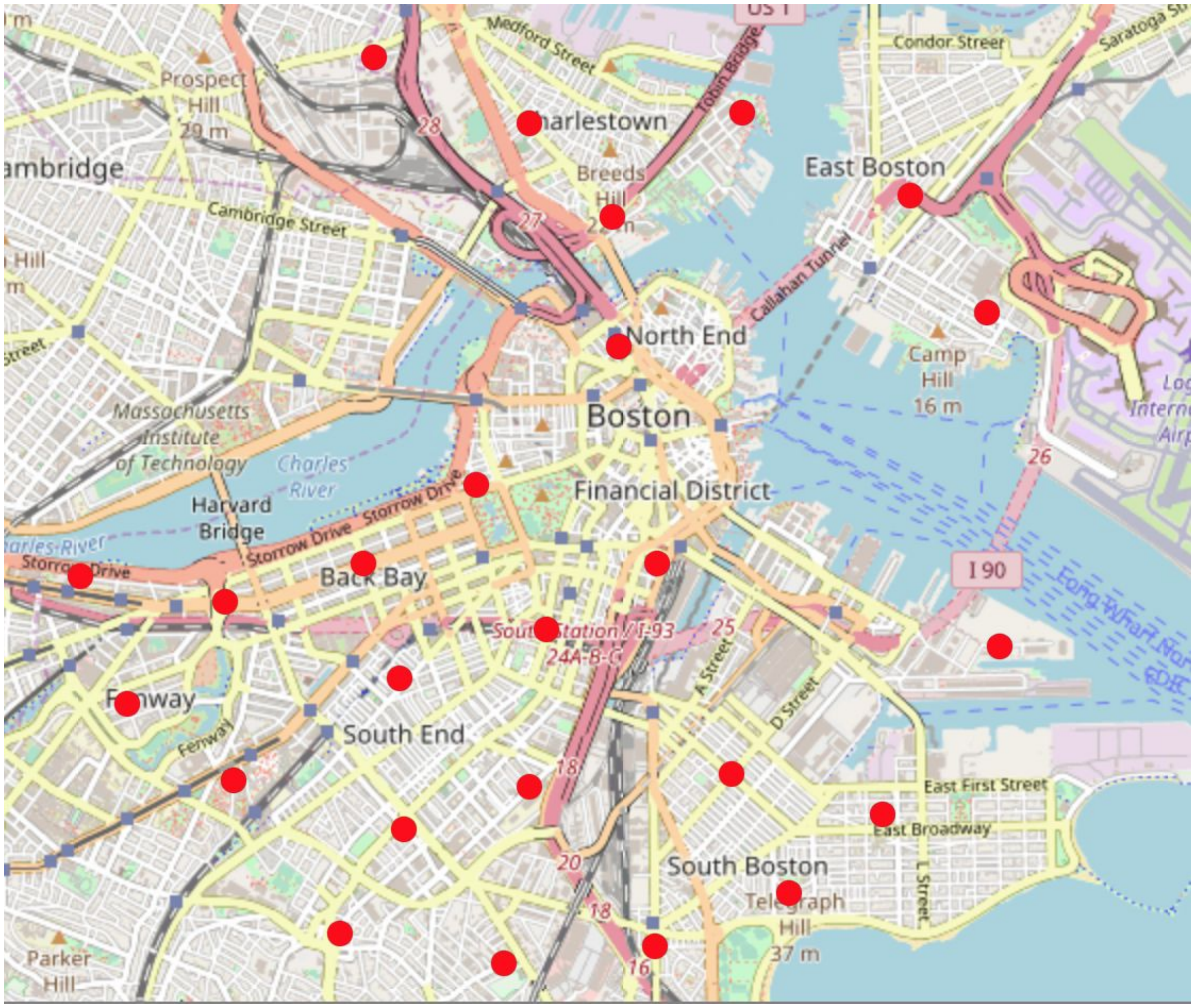
- One of our resources will give an http error when the project is run, so we cannot access the open spaces resource on the first try. When running the project, try to run it a couple of times, as the error will cease after 2 to 4 tries.

General issues & limitations:

- The nodes that come from the intersections dataset refer to all intersections within the Boston area, so they do not exclude intersections where there is very little car volume (such as back alleys). However, with a database that includes these locations, this issue can be resolved.

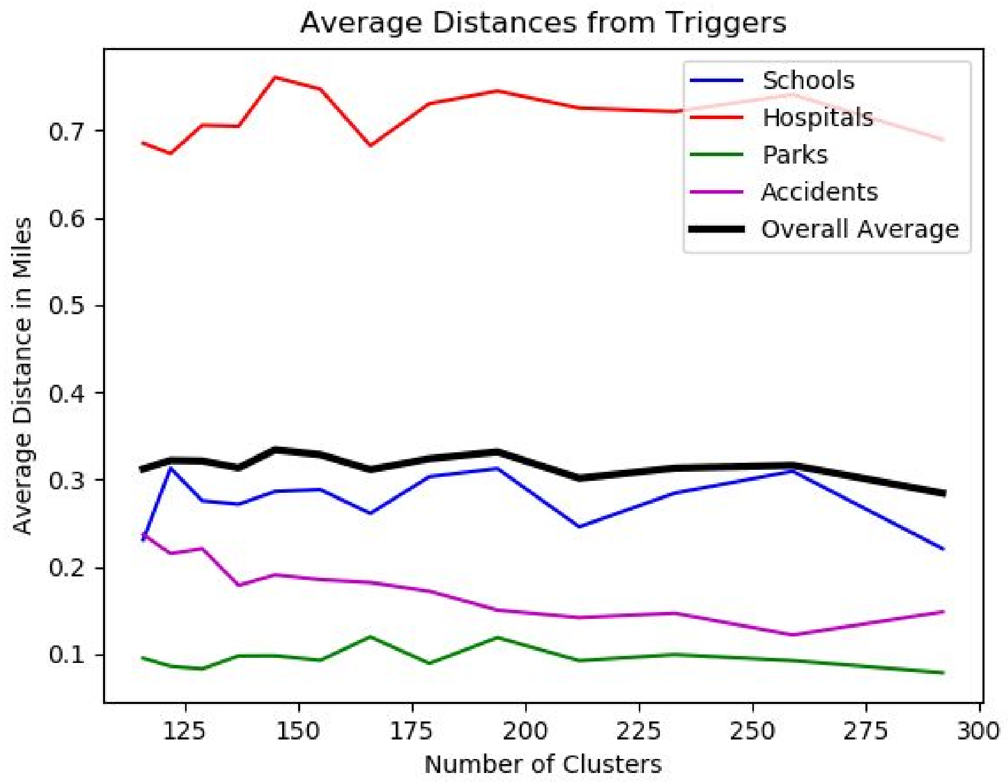
## Statistical Findings/Results

Using the stated processes above, we were able to map an optimal placement of speed zoning signs, while also finding the amount of “accident hotspots” that would minimize a a sign’s average distance from triggers. The placement of these signs given the default values stated in the web servers can be seen on the map below, where each red dot is a placement:



When placing 30 signs that were at least 2 miles apart throughout Boston, with a total of approximately 23,200 accidents from our data set, we found that after finding the average distances from each sign to a specific trigger (schools, hospitals, open spaces and accidents), the number of accident hotspots that needed to be used to minimize the overall sign-trigger distance was 212 hotspots.

It is important to know that hospitals are the only complicating trigger in these results, as their distances from signs are much higher than the other triggers. These higher averages are most likely due to their sparsity and scarcity. However, as this difference in distance applies to every different amount of clusters tested, it does not affect the overall minimum average. The graph for the average distances from signs to triggers, in addition to the overall average distances of these triggers, is provided below:



The solution we have found gives definitive results that can definitely be reproduced, as the parameters can be adjusted to account for different distances between signs and the number of signs placed. Additionally, determining a set amount of accident hotspots for a set of parameters provides an accurate approximation about which areas have a sufficiently high volume of cars, thus decreasing the chance of unoccupied intersections being zoned or dangerous intersections from being ignored.

## Future Works

In future iterations of this project, we would like to pursue extra features for our application, such as the option to focus optimization on sub-areas in Boston to allow a more concentrated placement of signs, and the ability to move signs around manually and dynamically relocate other signs based on a set of parameters. We would also like to narrow the optimization area to places with high volumes of traffic to maximize zoning efficiency, and use other critical areas like elderly homes as additional parameters for determining placement.