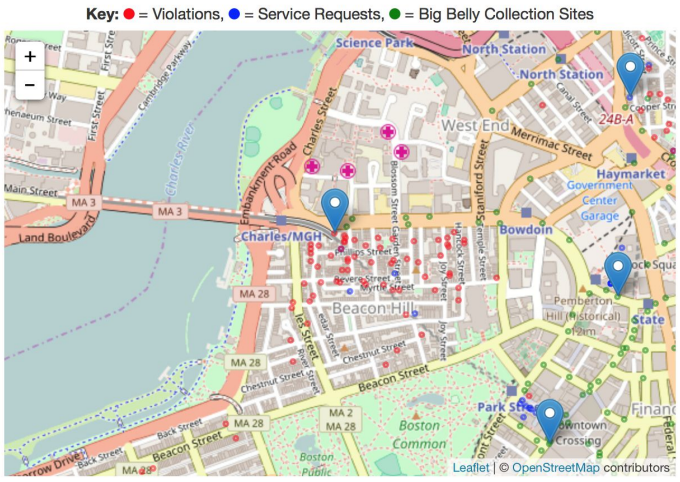
# Optimal Trash Collection Installation Sites

**Jacquelyn Andrade, Joseph Cho**
CS591 Data Mechanics
Boston University Computer Science Department

## 1. Introduction

*Instance of our web service finding optimal collection unit installations*




Our goal is to gather information on the sanitation-related trends of Boston neighborhoods. Essentially, we want to be able to analyze this data to propose the optimal location to install trash collection units in order to help reduce sanitary code violations, improve overall cleanliness, and prevent trash overfill. The goal of this application is to create an interactive application for the City of Boston. The user will be able to specify the number of trash sites the city wishes to install as well as the radius a single site is expected to cover. The application will display the optimal locations as markers on a map.

## 2. Data Sets

All our datasets are provided from the City of Boston Data Portal.

### 2.1 Big Belly Alerts 2014:

This dataset contains information from a selected group of Bigbelly waste receptacles within the City of Boston. Each entry in the file represents a single “call” from a Bigbelly which represents a snapshot of the current status of that Bigbelly receptacle at the time of the call. Utilized columns: Latitude, Longitude, Fullness.

Source:
https://data.cityofboston.gov/resource/nybq-xu5r.json

### 2.2 Master Address List:

The list of all property addresses in the city. Utilized columns: Geocoded_Location and trash_day.

Source:
https://data.cityofboston.gov/resource/je5q-tbjf.json

### 2.3 Code Enforcement - Building and Property Violations:

Inspectional Services violations found by Code Enforcement with citations issued. Utilized columns: Latitude, Longitude, Status, and Description.

Source: https://data.cityofboston.gov/resource/w39n-pvs8.json

### 2.4 Food Establishments Inspections:

Health inspections of licensed food establishments. Utilized columns: Location, ViolStatus, ViolDesc.

Source: https://data.cityofboston.gov/resource/427a-3cn5.json

### 2.5 Mayor's 24 Hour Hotline (Cases created **last 90 days):**

Constituent requests for city services. Utilized columns: Geocoded_Location, CASE_STATUS, CASE_TITLE.

Source: https://data.cityofboston.gov/resource/jbcd-dknd.json

## 3. Algorithm

### 3.1 Overview

In our optimization function, we find the optimal placements of trash collection units. When running the algorithm, the user has the option to specify the number of units the city is willing to install as well as the radius a single unit is expected to cover. The algorithm returns a list of relative location coordinates that fit these criteria.

### 3.2 Analysis

We look at the geolocations of sanitary violations and requests, the weights of those locations, and we also look at the locations of Big Belly's and their average fullness. In addition, the algorithm takes into consideration the geolocations of areas that already have heavy trash collection schedules and attempts to focus more on problematic areas where trash is collected less regularly. Based upon these datasets we find the optimal placement of trash collection units to reduce the overall amount of violations associated with excess waste. We find these placements using weight metrics and KD trees. Our algorithm is contained in optimization.py and has been adjusted to accept number of units and radius parameters and to return a set of optimization coordinates.

#### 3.2.1 Weighting

In order to find the optimal location for new collection units, we calculate the weights of areas that a new collection unit would cover within its expected radius. The higher the weight of the area that is within the radius of a collection unit, the more likely that area will benefit from a new unit. In general, when given any single location, weights are scaled based upon the association and severity of the violation, request, average fullness, or pickup frequency.

For code violations, overfilling or heavy amounts of trash are weighted higher. Incidents not necessarily (but are often) associated with excessive trash are weighted lower, such as 'insects/animals'.

For each given location we calculate the weight as follows: *weight = count * scale*. Where count is the number of violations found in this area and scale is given by the table below.

| Type | Scale |
|---|---|
| Improper Storage Trash | 0.75 |
| Illegal dumping | 0.75 |
| Overfilling of barrel/dumpster | 1 |
| Storage of Garbage & Rubbish | 0.75 |
| Insects Rodents Animals | 0.3 |
| Trash Illegally Dump Container | 0.75 |

For service requests, trash specific requests are weighted higher. Incidents not necessarily (but are often) associated with excessive trash are weighted lower, such as 'pest infestation'. For each given location we calculate the weight as follows: *weight = count * scale*. Where count is the number of requests found in this area and scale is given by the table below.

| Type | Scale |
|---|---|
| Illegal Dumping | 0.75 |
| Improper Storage of Trash | 0.75 |
| Mice Infestation | 0.3 |
| Missed Trash/Recycling/Yard Waste/Bulk Item | 0.5 |
| Overflowing or Un-Kept Dumpster | 1 |
| Pest Infestation | 0.3 |
| Rodent Activity | 0.5 |
| Unsanitary Conditions - Establishment | 0.3 |

For Big Belly's, units with high average fullness are weighted higher, but we also weighted Big Belly's lower overall as they appear to be concentrated only in specific areas. We adjusted accordingly as we did not want this dataset alone to skew our data one way or another. For each given location we calculate the weight as follows: *weight = count * scale*. Where count is the number of snapshots of a specific Bigbelly and scale is given by the table below.

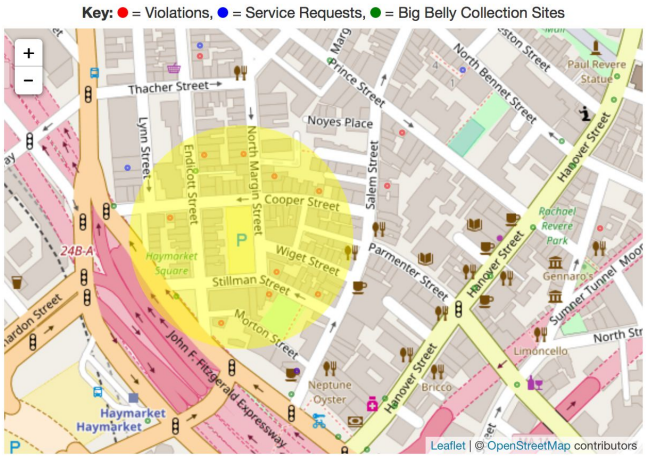
| Average Fullness | Scale |
|---|---|
| 0.9 - 1.0 | 0.3 |
| 0.7 - 0.89 | 0.25 |
| 0.5 - 0.69 | 0.2 |
| 0.3 - 4.9 | 0.15 |
| 0.0 - 0.29 | 0.1 |

For Trash Schedules, we add a negative weight to areas that have multiple collection days. Our reasoning is that these areas already have focused effort in terms of time and resources devoted to collecting trash. We want the areas that are not receiving as much attention to be weighted on an equal basis. For each given location we calculate the weight as follows: *weight = count * -1*. Where count is the number of times per week trash is collected.

### 3.2.2 KD-trees

“KD-trees” is a technique used to find coordinates within a specific radius given a central coordinate point.

For each trash collection unit the user wants to install, we choose x random coordinates (*x is specified from the iterations parameter when the optimized function is called*) from a master dataset that contains all violations, requests, Bigbelly, and pickup coordinates. We then use python's K-D tree library to find all points within a certain radius for each of these random coordinate points. Using the points within these radii we calculate the total weight of the region associated with that coordinate. The algorithm will find the location coordinates of the regions that have maximal weight out of all randomly selected coordinates.



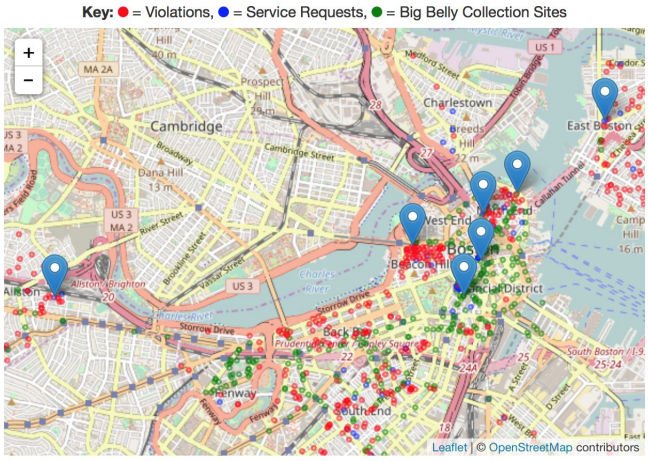

To ensure the algorithm does not return the same regions for every iteration, if the algorithm is searching for multiple trash collection units to install it will “ignore” regions that have already been selected by previous iterations.

### 3.2.3 Limitations

Note that because we are choosing coordinates from our pre existing dataset of incidents, our optimization returns coordinates corresponding to an incident. This decision was made to optimize our running time and to accurately choose locations that need a collection unit within the area. We wanted to choose coordinates that would closely align with the same areas as our incidents instead of randomly generating coordinates that may or may not lie within the city of Boston.

Our algorithm when under heavy loads, such as 1000 trash collection units, may slow down in performance. Additionally, if the user inputs a large radius with a large number of collection units, the optimization algorithm may throw errors as previous iterations may have exhausted the master list. This means that for sequential iterations trying to find additional locations, if the previous unit had a radius of 10000 meters, all coordinates may have been removed from the master dataset, so there are no other incident coordinates to choose from.

## 4. Summary




In our analysis, we found that the algorithm initially tends to favor the areas of Boston that are more metropolitan or have high levels of activity, but still tends to favor areas outside of densely trafficked areas when additional trash units are specified.

Also, in multiple iterations of running the application, the same exact coordinates are not always returned, however they are within the same relative areas.

When deriving weights for each violation or request, we did our best to be as logical as possible, but these weights may not be completely accurate when used to weight certain violations/requests over others as we do not know all details or severity of certain incidents over others.

Overall our project is a good estimator of where to install trash collection units. The only major limitation is from the City of Boston Data itself. Our data is derived from whether an incident has occurred or not, but it does not indicate the severity of a specific incident. If we were given the severity, our application would be able to predict optimal locations more definitively instead of treating each incident as a binary attribute.

## 5. Future Work

In future iterations of this project we hope to be able to also optimize trash schedules for the City of Boston. We would like to apply the same weighing techniques to pinpoint locations where trash could be collected more often and locations where trash is not a severe problem and funding to these locations can be allocated somewhere else.