BANA 8083: MS-BANA Capstone

District Configuration Analysis through Evolutionary Simulation

Matt Policastro

Professor Peng Wang, First Reader

Professor Michael Magazine, Second Reader

2017-07-20

Abstract

This capstone replicates a methodology for identifying biased electoral district schemes on a smaller scale in a new context. Rather than electoral districts this project examines three of the five Cincinnati Police Department (CPD) districts containing twenty-four of the City of Cincinnati's fifty neighbourhoods. It should be noted that this project does not constitute a rigourous analysis of potentially-biased districting practices on the part of the city; instead, this project identifies advantages, trade-offs, and other challenges related to implementation and analysis.

Contents

Introduction

Background

The practice of manipulating district configurations in order to effect specific outcomes—gerrymandering, as it is commonly known in politics and law—is well-established and pernicious. Courts have struggled a great deal with establishing robust tests for bias or malign intent, as legal precedent is too broad to cover the highly-specific techniques used to manipulate voting effects. Furthermore, quantitative measures (e.g. compactness, contiguity) have largely failed to make an impact as gerrymandered districts can be manifested in myriad ways (Cho and Liu [2016], p.351–353). In short, specific quantities rarely demonstrate biased districting.

As such, an alternative approach of developing context has been proposed. Of course, full enumeration has been suggested as it provides perfect context for an existing plan. All solutions to the problem of assigning districts become visible, and remote or unlikely configurations become consequently apparent. However, as is typically the case of problems with meaningful size, the computational effort is simply unfeasible. In the case of geographic districts, the total number of maps is Stirling number of the second kind ($S(n, k)$). As Cho and Liu [2016] state:

> Even with a modest number of units, the scale of the unconstrained map-making problem is awesome. If one wanted to divide n = 55 units into k = 6 districts, the number of possibilities is 8.7×10^{39} , a formidable number. There have been fewer than 10^{18} seconds since the beginning of the universe. Of course, the number becomes significantly smaller with relevant legal constraints, such as contiguity, in place. However, it does not become manageably smaller. The problem remains massive.

In response, there have been ongoing efforts to leverage simulation in exploring the solution space. As computing power has become radically inexpensive, the real costs of complex simulations have dropped significantly. A simulation with fairly simple constraints can be used to examine the solution space for a districting problem. At minimum, there are three categories of constraints for electorally-pertinent districting problems (Cho and Liu [2016], p. 358):

1. Each unit must be assigned to exactly one district.
2. The maximum population deviation across all K districts is no greater than a specified value.
3. The units in each district must form a geographically-continuous and -contiguous set.

These constraints can be augmented with additional constraints or by objectives, such as compactness factor. With this structure in place, it then becomes easy to simulate possible solutions through an evolutionary algorithm. The work undertaken in Cho and Liu [2016] and Liu et al.

[2016] resulted in the Parallel Evolutionary Algorithm for Redistricting (PEAR), which leverages asynchronous communication to minimise overlap between evolutionary processes in a supercomputing environment.

Overview

Examining an electoral problem at a state scale would have been simply infeasible for the scope of this project. The talismanic redistricting project required tens of thousands of processors (131,072 in the case of the Blue Waters supercomputer per Liu et al. [2016], p. 87) to conduct a simulation in a reasonable time frame. Furthermore, local or regional elections do not have the same data availability or reliability, particularly with respect to geospatial data. However, the methods developed to handle geospatial relationships and contiguity were of particular interest.

As such, local police districts were chosen for simulation and analysis. The City of Cincinnati provides a significant amount of their data to the public through their open data portal (Cincinnati [a]), including anonymised crime reports by street address and neighbourhood. And as was stressed in a review of other methods (Cho and Liu [2016], p. 354), populations and districts must remain constant to keep analysis valid. To satisfy that requirement, all population and geography data was sourced from the 2010 Census and the City of Cincinnati's statistical neighbourhood approximations (Cincinnati [b]).

Finally, as this project required a naïve implementation of the evolutionary algorithm developed by Liu et al. [2016], all code and data was housed in a version control system repository (specifically, Git). All code and a full project history including comments can be found at the author's personal repository on Github (Policastro), available at the following URL: github.com/mattpolicastro/BANA-8083-Capstone.

Methods & Process

Sources

Data on crime reports between the beginning of 2008 and the end of 2012 was requested from Cincinnati's open data portal using `data/import_crime.py` and ignored any reports without explicit neighbourhood data. Exploratory data analysis found the report data to be virtually non-existent before July of 2010 (Figure 1). To infer the missing crime report counts, a naïve linear regression was chosen over a time series model for the sake of simplicity. The regression was developed on the following three years of crime data (the ratio of crimes committed in the first and second halves of the year, 2011–2013), yielding an estimated ratio (r) of 0.915. As

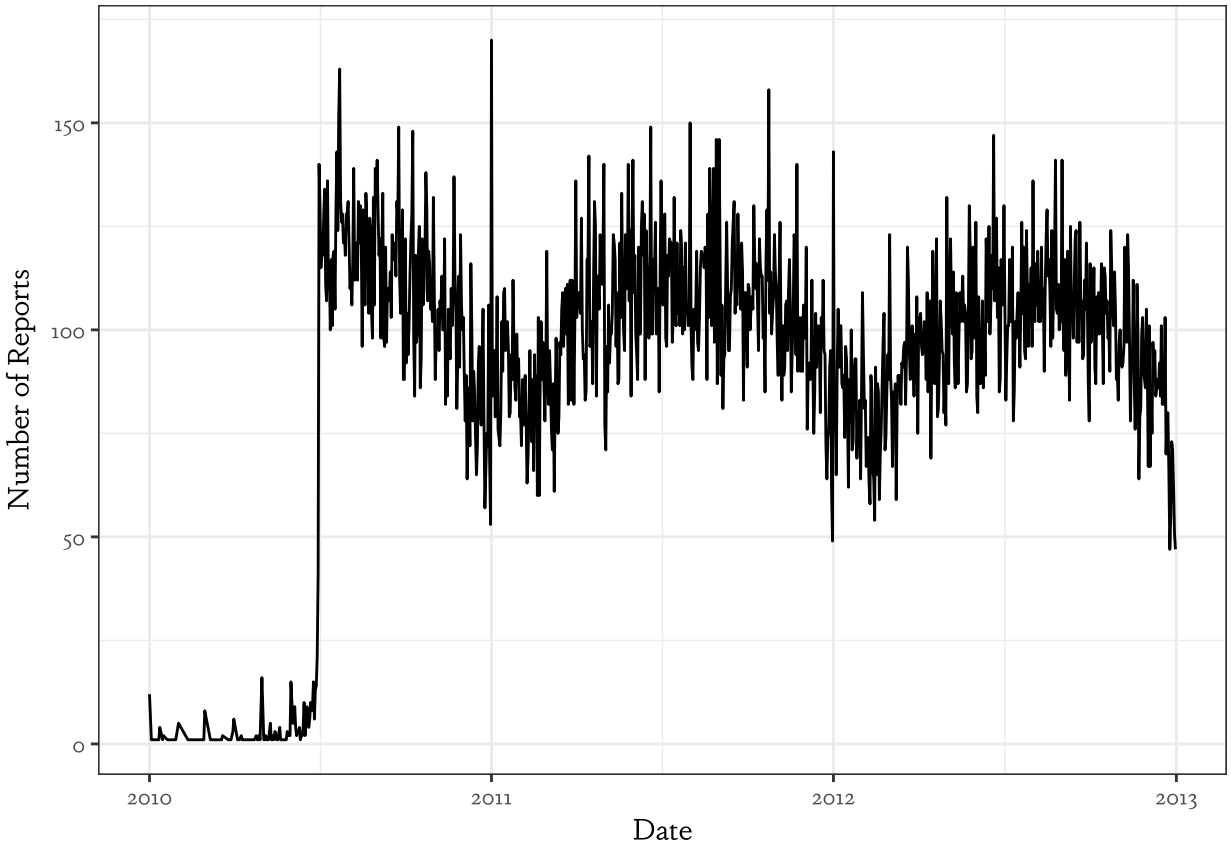


Figure 1: Crime Report Daily Totals

the simulation was only concerned with annual totals, per-neighbourhood crime rates were considered constant and equal throughout 2010.

The final crime report counts (C) were calculated as $C = C_{\text{2nd half}} \cdot [1 + r]$. Neighbourhoods, police districts, and population counts were extracted and checked by hand from the city's website (neighbourhood-specific approximations from the 2010 census were only available as PDFs). Two consolidations were made to match the city's official neighbourhood designations: Clifton Heights, University Heights, and Fairview were consolidated into one neighbourhood, "CUF"; the Fay Apartments were re-designated as "The Villages of Roll Hill" as part of a public redevelopment project (Simes). These changes were incorporated into the mappings between the irregular names and the city's standard nomenclature so all counts were summed appropriately. Finally, the population data was mapped onto the summarised crime report data and saved into the final data set.

The other major data component was geospatial data (commonly known as GIS or shapefile data) for the neighbourhoods, which proved to be challenging. Geospatial data consists of encoded shapes, lines, attributes, etc. in a coordinate system, allowing for complex geometric calculations and operations. Hamilton County and the city's Enterprise Technology Solutions department provide and update these files at regular intervals (County), but they were found to be extremely unreliable. Thousands of intersecting points prevented accurate calculations of area and perimeter, and the data was ultimately discarded.

An alternative geospatial data set was found and extracted as GeoJSON from the Cincinnati Police Department (CPD) using an open-source Python package (OpenAddresses). (The same name were used to clean the neighbourhood names.) This data was then edited slightly to prevent a false negative space in the map, but cursory analysis indicated the maps were a good match for the city's official records in terms of perimeters and areas. Using another open-source Python package (GeoPandas), the data was then transformed to remove three neighbourhoods not in the city (Edgemont, Elmwood Place, St. Bernard). Finally, a shape representing the exterior boundaries of the city was formed by subtracting each neighbourhood's geometry from a regional overlay.

The final data sets were then filtered to only include the three districts under consideration: districts one, four, and five. Districts two and three (the districts furthest east and west in Fig. 2) introduced several problems and were excluded for the following reasons:

- Each contains a number of neighbourhoods which only have one or two neighbours and are on the city's edges, meaning they would have little opportunity to change district memberships during the simulation.
- Many of the neighbourhoods on the east side of the city are eccentrically shaped and introduce "holes" in the area, adding computational complexity and de facto compactness penalties.

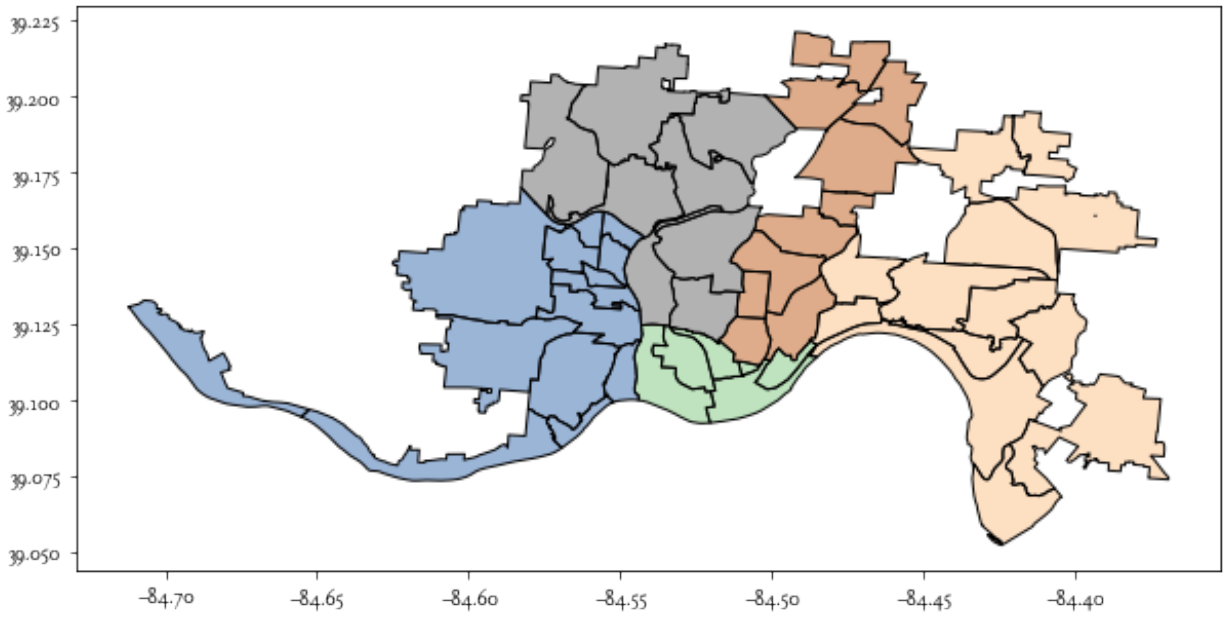


Figure 2: City of Cincinnati neighbourhoods by police district

- The number of possible solutions for set assignments is $S(50, 5)$, or 7.4×10^{32}. Removing these two districts brings that size down to $S(25, 3)$, or 1.4×10^{11}. As this project was carried out on consumer-grade computing hardware, the reduction in problem size is meaningful when attempting to generate a representative set.

Preparation

The terminology used with PEAR is also applied here: *units* are the indivisible sub-regions, which in this case are the city's neighbourhoods; police districts are *zones*, and are comprised of units; the entire city is the *region*, which is the entirety of the study area. Geospatial analysis is undoubtedly powerful, but it is relatively expensive in terms of computational operations. As such, the data sets required transformation into new data structures, per the methods outlined by Liu et al. [2016] (p.82).

The most important primary data structure used in the evolutionary algorithm was the chromosome: an array of values indicating the zone membership of each unit, where index position indicates the given unit and the value its assigned zone. Then, a network graph was extracted from the geospatial data, detailing the neighbours of each unit in the region using another open-source Python package for geospatial analysis (PySAL). However, rather than "queen" adjacency (units are considered adjacent if they share a corner or a border), this project only considered "rook" adjacency (units are considered adjacent only if they share a border).

Where electoral law has established precedence for queen adjacency, this project was con-

cerned with the physical relation of units and opted for shared borders. Another network graph was developed for zones and their rook-adjacent neighbours. This graph was generated and revised in each mutation executed by the algorithm to determine the order of operations and to check for violations of contiguity. (If a zone has only one neighbour in the graph, it has been surrounded by that neighbour and is considered invalid.) Finally, a data frame (a table-like data structure) was used as a canonical reference containing the geometries, crime counts, and populations for each unit in the simulation.

An initial population for the simulation was generated with the PySAL library, which includes a `Random_Regions()` method for generating random zone assignments from a provided adjacency graph. One thousand of these random assignments were generated, converted into chromosomes, and checked for violations of contiguity. Of the generated chromosomes, 94.1% were considered valid, and one hundred of those chromosomes were randomly sampled to create the initial population.

Simulation

The open-source framework Distributed Evolutionary Algorithms in Python (DEAP) was used directly and as a design reference in implementing the evolutionary algorithm (Fortin et al. [2012]). Individuals (the chromosomes representing zone membership) constitute a population, which is grown over a succession of generations in the following process (beginning with the randomly generated initial population):

1. From a starting population, a generation of clones is created (the "offspring").
2. Each individual in the offspring is mutated and evaluated for fitness, yielding a new population (the "mutants").
3. The mutants and their fitnesses are stored in an external data structure and passed to the next generation as the starting population.

This process was repeated for one hundred generations in a semi-parallel fashion. Each population as subdivided, dispatched to multiple child processes using multiprocessing pool, and then re-combined before proceeding to the next step. Fitness checking consisted of three values on a $[0, 1]$ range: compactness, maximum inter-zone population deviation, and maximum inter-zone crime count deviation. Compactness quantifies the spatial efficiency of a zone, where one is the maximum value (e.g. a circle, which is the most efficient ratio of area to perimeter). The maximum observed deviation, however, makes pairwise comparisons between zones to identify the most imbalanced assignments, where any sufficiently large difference is capped to a value of 1. The same calculations used by Cho and Liu [2016] (p. 358) for compactness and deviation were used here.

Similarly, the implementation of the mutation operation closely followed the algorithm refer-

ence provided in Liu et al. [2016] (p.84). While DEAP provides a wealth of efficient evolutionary operators, none were able to perform contiguity and validity checks on geospatial data. The custom operator followed this general procedure:

1. Generate a random sequence of source zones, finding a neighbouring destination zone for each.
2. If the source zone contains more than one unit, continue the mutation; else, move to the next source zone.
3. Beginning with a unit on the border between the source and destination zones, select a random number of neighbouring units without selecting the entire source zone.
4. If the "donation" of the selected units does not violate contiguity or validity, shift those units into the destination zone; else, if the mutation has been attempted x times, move to the next zone; else, shift the selected units to the destination zone and begin 2. with the next source zone until that sequence is complete.

This operator only makes one significant departure from the referenced algorithm, which is the arbitrary cut-off of x for indefinitely-cycling solutions. The included checks avoid purely stochastic changes to the chromosome which require additional checking and validation later on in the simulation.

It is also worth noting that this simulation did not leverage any selection criteria to create a "hall of fame" of best performers. As this project leveraged approximations for quality rather than more tangible criteria (e.g. time to respond to emergency call) and could only produce ten thousand variations at most, all valid results were recorded for the analysis.

Analysis

District Comparison

Of the ten thousand variations generated by the simulation, 84.51% of them were found to be unique variations. As this simulation did not implement inter-process messaging to prevent duplication of efforts, this was expected. The set used for primary analysis was filtered to ignore duplicates to focus on the realm of potential solutions rather than their frequency. Table 1 contains the summary statistics for each of the three fitness criteria. For each fitness criteria, beta distributions with fixed location and scale parameters were fitted using maximum likelihood estimates (MLE). To approximate goodness of fit, Kolmogorov-Smirnov (K-S) tests were conducted using those estimated parameters at 95% significance; the null hypothesis being that the observed and theoretical distributions are identical, and the alternative that the observed distribution is either greater than or less than the theoretical.

As can be seen in Figure 3, the real-world CPD districts are not particularly compact, scoring

in the 99th percentile of the simulated data. Presumably due to the large spikes seen there, the null hypothesis was soundly rejected with a p-value of 6.0350×10^{-16}. The crime count deviation criteria also rejected goodness of fit, albeit in a relatively less severe fashion with a p-value of 4.9917×10^{-7}. Fig. 4 shows departures from the fitted distribution, but much less significantly than the previous criteria. That said, the actual CPD districting outperformed many of the simulations, scoring in the 6th percentile. Contrary to the other two fitness criteria, the population scores fell fairly neatly around the real-world maximum deviation (the 44th percentile) and the K-S test rejected the null but at an even less-severe p-value of ~0.0017. Fig. 5, showing the the simulation against the baseline and fitted distribution, supports that conclusion.

However, another criteria was considered in the post-simulation analysis: the Adjusted Rand Index (ARI). Typically used in clustering, the ARI quantifies agreement between two sets of group assignments through pairwise group membership comparison. The traditional Rand Index ranges from zero to one—zero meaning no agreement, one meaning full agreement—and the adjusted version accounts for random chance and allows for negative values, indicating the score for the given assignment falls beneath the expected value. Crucially, the ARI is agnostic toward the actual group labels; it is only concerned with pairwise membership across the entire set (i.e. if units 4 and 17 are in zone A, are they in agreement in the alternative zone plan?). Similar to the previous criteria, a beta distribution was fitted to the ARI scores and was tested for goodness of fit. As seen in Fig 6, the ARI seemed to stop abruptly around -0.10 in agreement/disagreement, yielding a p-value of 2.6896×1^{-58} in the K-S test. However, as a score of 1 indicates total agreement, the relative lack of overall agreement between the simulation results an the actual memberships is striking.

Table 1: Descriptive Statistics for Fitness Criteria

Criteria	Actual	Max.	Mean	Median	Min.	Var.
Compactness	0.0823	0.0921	0.0564	0.0567	0.0243	0.0001
Crime Dev.	0.3214	0.9154	0.3417	0.3365	0.0036	0.0243
Pop. Dev.	0.3214	0.9359	0.3521	0.3438	0.0005	0.0264
Adj. Rand Ind.	N/A	0.7331	0.1128	0.0978	-0.0829	0.0124

Cycling

Of the duplicate solutions, four configurations occurred at least one hundred times each over the course of the simulation, suggesting cycling was an issue. (The configurations in Figure 7(a) are identical, only using different district labels for the same geometries). Although some logs were kept during the simulation runtime, detailed data about the relative frequency of

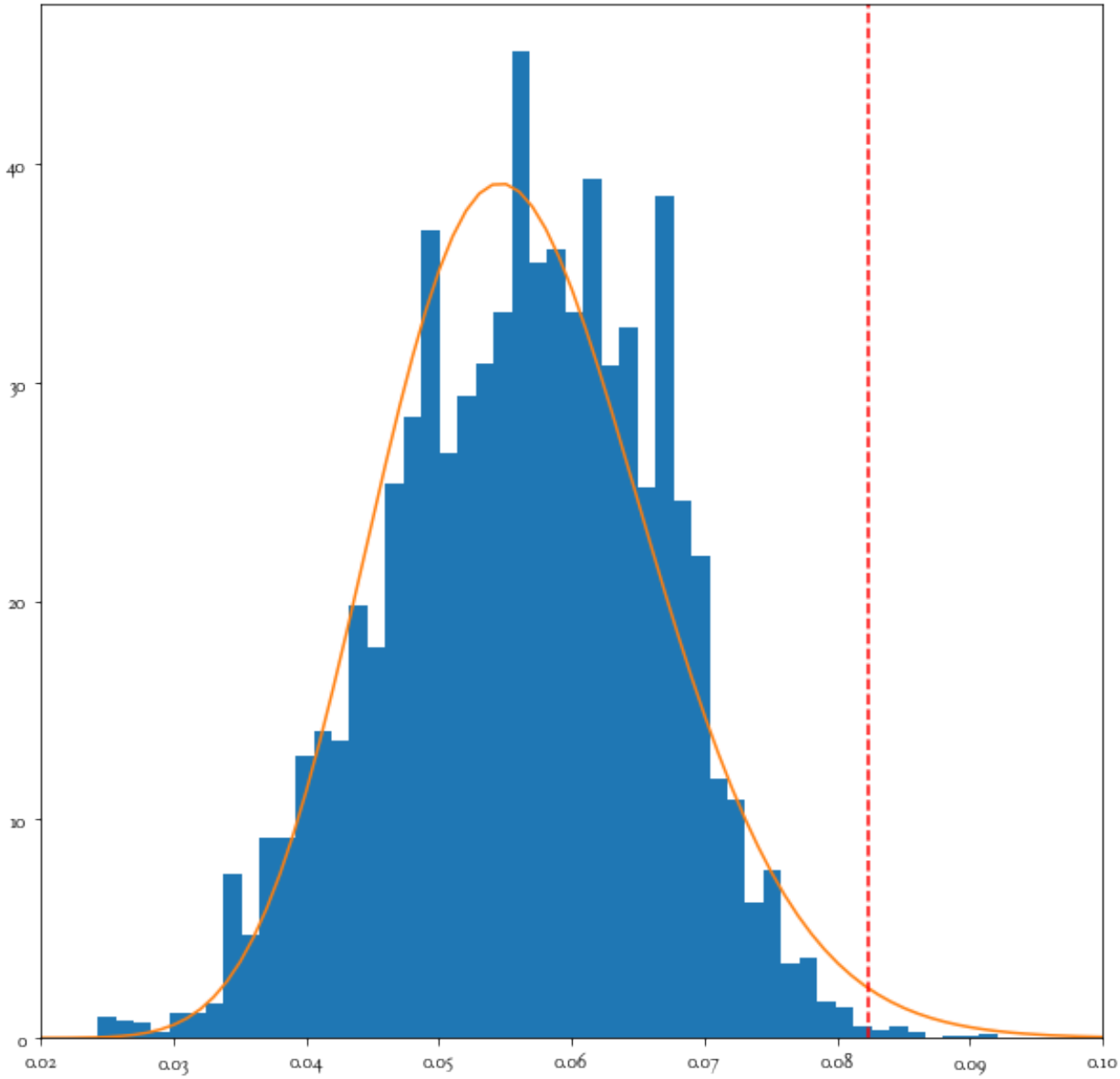


Figure 3: Simulated Compactness scores vs. actual districting score

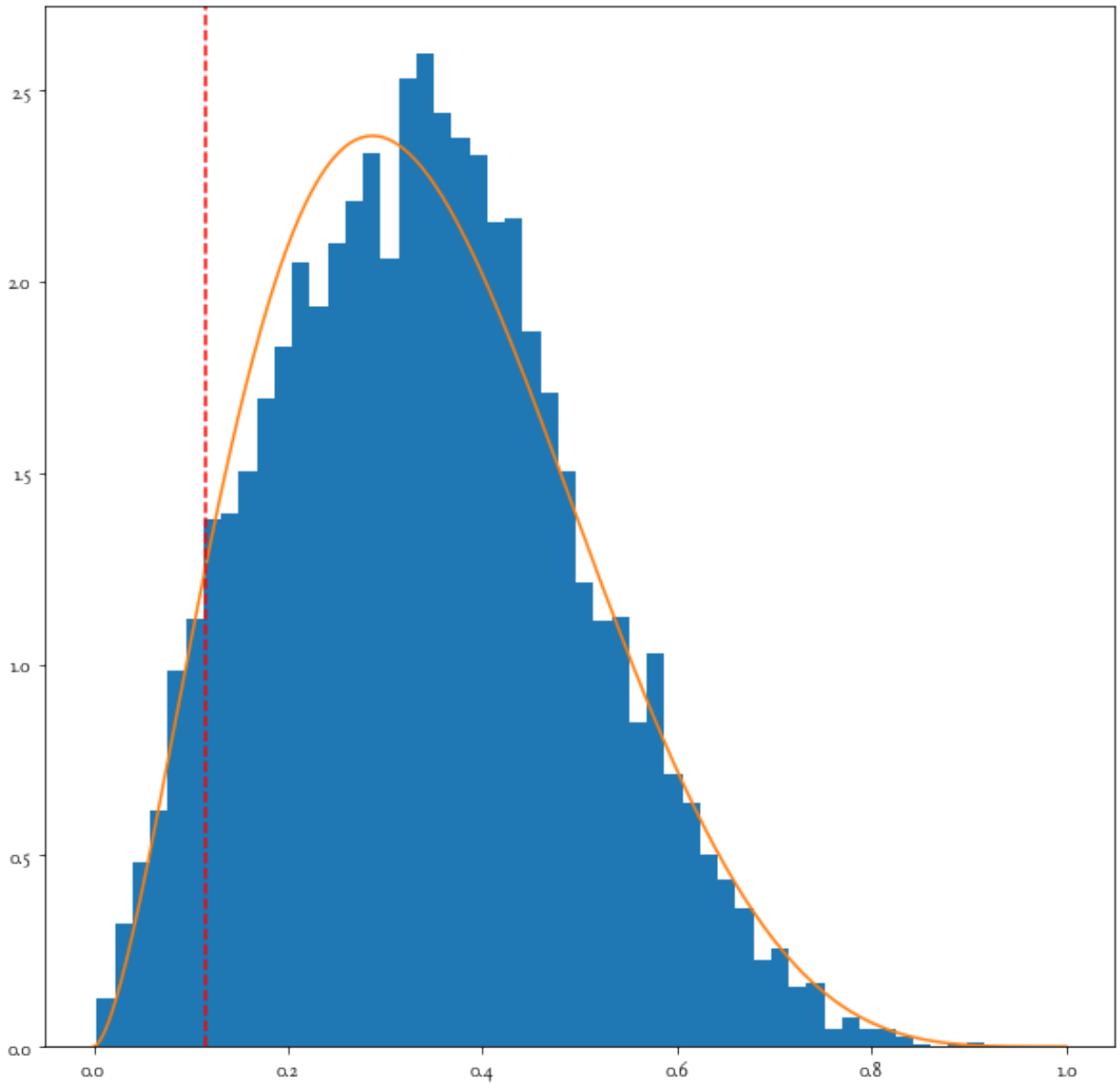


Figure 4: Simulated crime scores vs. actual districting score

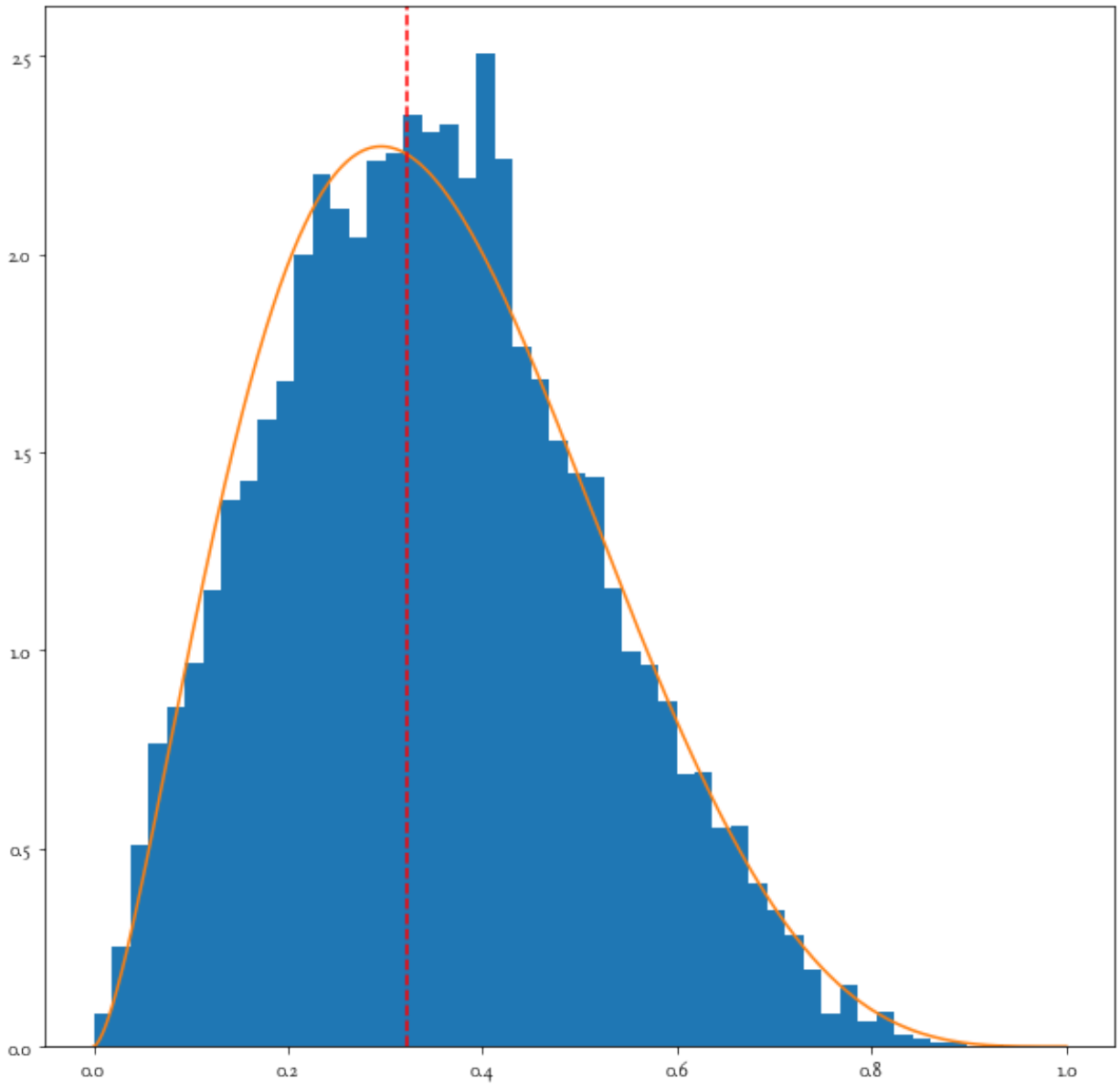


Figure 5: Simulated population scores vs. actual districting score

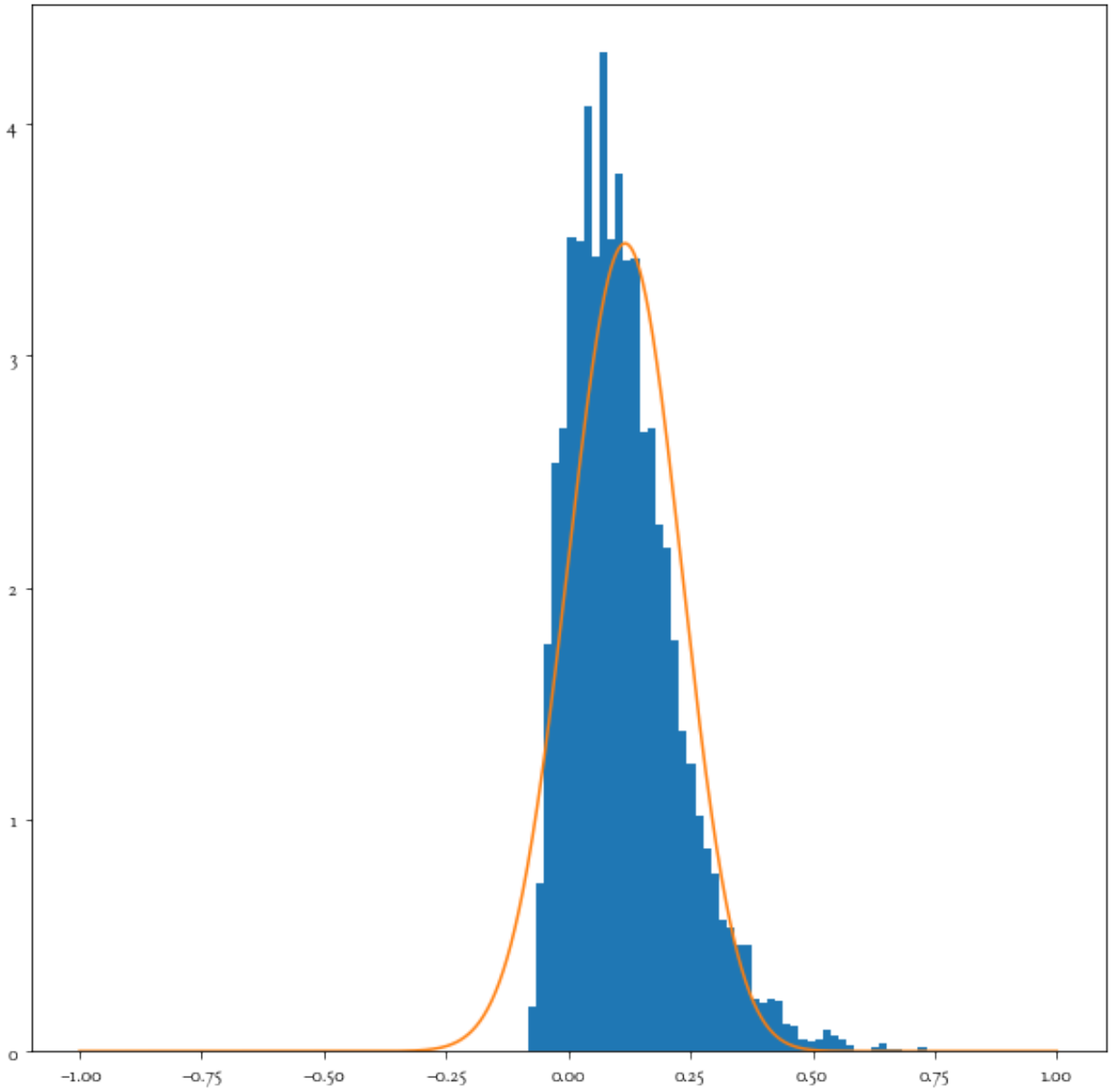


Figure 6: Adjusted Rand Indices for simulations against the actual

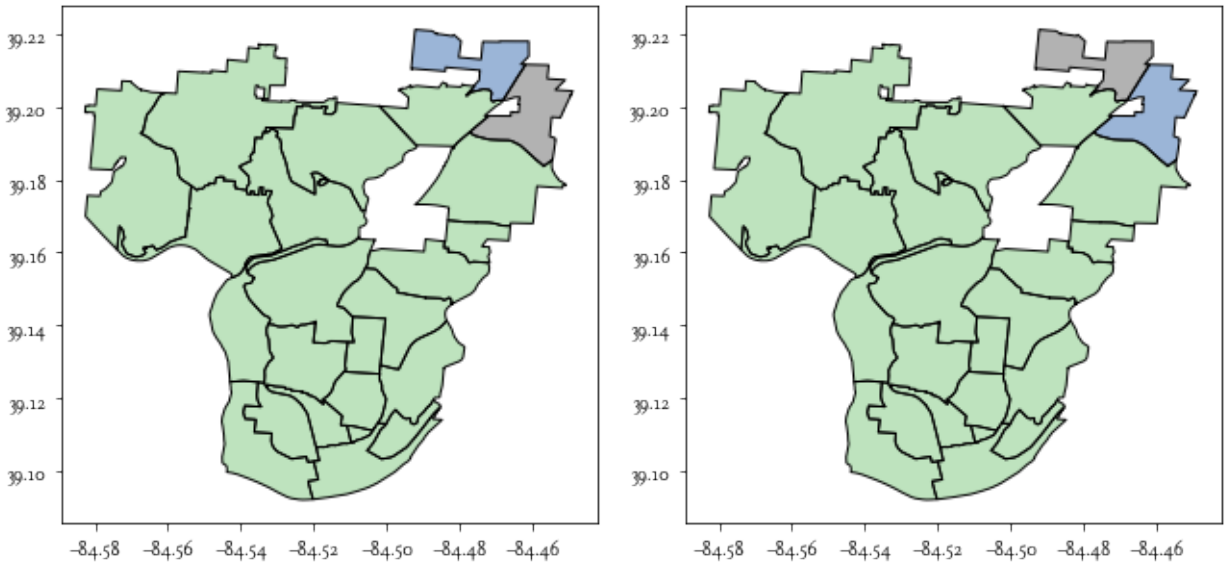
these solutions was not available. However, the small number of neighbours for each of the constituent units does seem to suggest the mutation operator was unable to "leap" out of the solution over successive generations. Furthermore, the spikes seen in Figures 3–6 might suggest these duplications were part of a larger trend of "plateaus" in the simulation results.

Conclusions

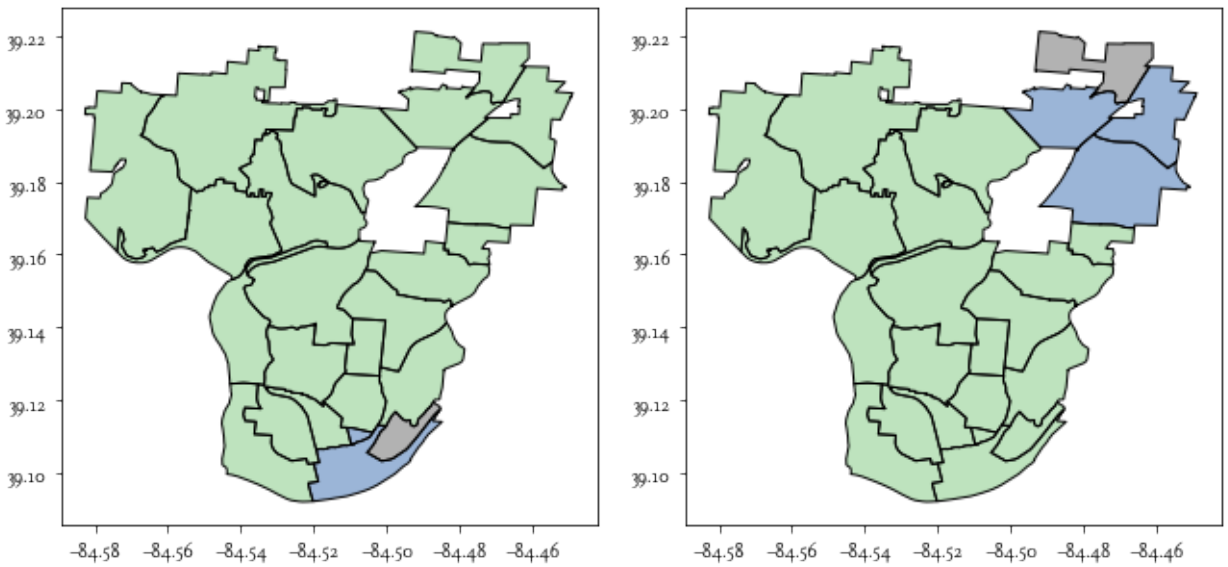
Using a few simple criteria for evaluating fitness, the simulation approach seems to theoretically satisfactory. However, the current implementation could be improved. Each generation ran for approximately forty-three seconds, meaning the total run time was just short of an hour and twelve minutes. Admittedly, the hardware used was not appropriate for larger-scale projects (a 1.7 GHz Intel Core i7 processor with 8GB of DDR3 RAM), but 16.49% of the generated plans were duplicates, and each generation required approximately forty-three seconds of running time. Improvements to the evolutionary operators are certainly possible, and fall into three categories:

- *Cycle avoidance.* Additional checks could be designed or thresholds could be adjusted to prevent the algorithm from conducting expensive spatial operations when the selected mutations are cycling.
- *Parallelization.* The current implementation is straightforward in dividing and recombining results, but each stage of the algorithm requires all individuals to finish, meaning those needing additional mutations or quality checks hold back the entire generation.
- *Messaging.* Similarly, the message-passing framework to update each process' knowledge of best solutions and potentially unexplored areas is possible, but also a significant undertaking.

That said, the general methodology appears to be sound. Each of the fitness criteria demonstrated reasonable spread across the simulation, and made significant departures from the current districting plan being used by CPD. Furthermore, the ARI data suggests the simulation's results contained a significant portion of plans which were extremely dissimilar in composition to the actual plan. Although there is casual evidence of plateaus and duplicated efforts, experimentation with mutation thresholds may be sufficient to address that issue. And, perhaps most importantly, designing more complex fitness criteria or selection operators to develop nuanced analysis is straightforward and the implementation provides a solid foundation for future streamlining and expansion.



(a)



(b)

Figure 7: Recurring district configurations

References

Wendy K. Tam Cho and Yan Y. Liu. Toward a talismanic redistricting tool: A computational method for identifying extreme redistricting plans. *Election Law Journal*, 15(4):351–366, 2016. doi: http://doi.org/10.1089/elj.2016.0384.

Cincinnati. Cincinnati open data portal | open data, a. URL https://data.cincinnati-oh.gov/.

Cincinnati. Census & demographics, b. URL http://www.cincinnati-oh.gov/planning/reports-data/census-demographics/.

Hamilton County. Cincinnati area geographic information system (cagis). URL http://cagismaps.hamilton-co.org/cagisportal/mapdata/.

CPD. Cpd neighborhoods. URL https://services.arcgis.com/JyZag70O4NteHGiq/arcgis/rest/services/CPD_Neighborhoods/FeatureServer.

Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

GeoPandas. Geopandas. URL https://github.com/geopandas/geopandas.

Git. URL https://git-scm.com/.

Yan Y. Liu, Wendy K. Tam Cho, and Shaowen Wang. Pear: a massively parallel evolutionary computation approach for political redistricting optimization and analysis. *Swarm and Evolutionary Computation*, 30:78–92, 2016. doi: http://doi.org/10.1016/j.swevo.2016.04.004.

OpenAddresses. esri-dump. URL https://github.com/openaddresses/pyesridump.

Matt Policastro. Ms-bana capstone project repository. URL https://github.com/mattpolicastro/BANA-8083-Capstone.

PySAL. Pysal. URL https://github.com/pysal/pysal.

Randy A. Simes. Fay apartments to be renamed the villages of roll hill, undergo $36m renovation. URL http://www.urbancincy.com/2010/10/fay-apartments-to-be-renamed-the-villages-of-roll-hill-undergo-36m-renovation.