Diversifying Populated Districts via an Inverse Gerrymandering Optimization Algorithm

Mels Habold

April 26, 2023

Summary

1 Introduction

In developing a more fair and equal world, one should strive to connect humans from diverse backgrounds with each other. We focus on creating diverse communities that bring together a variety of people for problem solving and mutual aid. For this, we would first need to make a just way of generating these fair communities.

This paper will describe an algorithm that will do just that. The main focus will be the socio-economic value of neighbourhoods, but we will also take in other parameters such as educational level. We do start out by limiting the population to generate communities via geographical boundaries. We do this in order to make sure communities lie close together and are connected to each other, as we believe this makes them more accessible for its members. This means that the resulting method should have a lot of overlap with Gerrymandering, but now the opposite is happening with the goals of diversifying the population of districts.

This method is built upon an older method by me described in appendix B. This method focused on mapping populations from one neighbourhood onto the communities, and was therefore a continuous problem. However, this method resulted in the splitting up of neighbourhoods, and the communities being geographically disjointed. This method tries to circumvent this problem by taking the geographical areas of neighbourhoods as input, and mapping them whole unto a community.

2 Problem Description

This method is based on data from [1]. We choose to define the data via 'buurten', which is a Dutch term for an area that is part of a neigbhourhood. These buurten make up about 1000 households and often consists of a few streets.

2.1 Decision Variables

We will view these buurten as the smallest possible piece of data which adds up to the total area. This means that our method will depend on the discrete allocation of these buurten. For this, we define the parameter θ which is a list of numbers of the same length as the data (N_B) that define which buurt should be allocated to which community. For this, we also define when initializing the model the total amount of communities that we want to end up with (N_C).

2.2 Parameters

For the starting problem, we choose to optimise 4 parameters:

- s_i: The socio-economic value for buurt i.
- e_{ij}: The percentage of people in buurt i that belong to a certain educational level. There are 3 possible levels for j: lower, middle and higher.
- p_i: The number of households in buurt i.
- d_{ij}: The distance between buurt i and j.

These parameters need to be fed into an objective function. For this, we first need to write them down in their matrix form:

- S: Socio-economic value (size: N_B)
- E: Education levels (size: $N_B \times 3$)
- P:Number of Households (size: N_B)
- D: Distances between Buurten (size: $N_B \times N_B$)

2.3 Mapping the Parameters

Next, we would like to know exactly how the labeling θ impacts the parameters. For this, we would like to write down the function $M^j_\theta(X)$, which maps the matrix X onto community j according to labeling θ. As both the education level and socio-economic value are parameters that are averaged out over the population, we need to make sure that the map also takes this into account as a normalisation. This results in,

$$M^j_\theta(X) = \frac{1}{\hat{P}_j} \sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot X_i P_i \tag{1}$$

In which $\mathbb{I}(\theta_i = j)$ equals 1 if the condition is satisfied, meaning buurt i is labeled under community j. The normalisation constant $\hat{P}_j$ is the population of community j and is defined by a similar map,

$$\hat{P}_j = \sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot P_i \tag{2}$$

2.4 Optimisation

We can now choose how to optimise the different parameters. As we want to make buurten more fair, we need to choose an optimisation function in such a way that it equalises the parameters S and E over all neighbourhoods. Therefore, we choose to optimise the variation of a parameter over all buurten. For the socio-economic value this should look like,

$$L_S = Var[M_\theta(S)] = \frac{\sum_{j=1}^{N_C} M^j_\theta(S)^2}{N_C} = \sum_{j=1}^{N_C} \frac{1}{N_C \hat{P}_j^2} \left(\sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot S_i P_i \right)^2 \tag{3}$$

We choose to optimise the education levels via the same method, but of course, now there are 3 different levels over which we need to calculate the variance, denoted by index k,

$$L_E = \sum_{k=1}^{3} Var[M_\theta(E_k)] = \frac{\sum_{j=1}^{N_C} M_\theta^j(S)^2}{N_C} = \sum_{k=1}^{3}\sum_{j=1}^{N_C} \frac{1}{N_C \hat{P}_j^2}\left(\sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j)\cdot E_{ik}P_i\right)^2 \tag{4}$$

For the distance we need to implement some extra steps. We want to make sure that all buurten that are part of the same community lie close together. As the matrix D contains all the distances between each and every buurt, we need to filter for only the distances that share the same label. For this, we use the operator $\mathbb{I}(\theta_i = \theta_j)$ to only use the distances for buurten that have the same labels,

$$L_D = \sum_{i=1}^{N_B}\sum_{j=1}^{N_B} \mathbb{I}(\theta_i = \theta_j)\cdot D_{ij} \tag{5}$$

2.5 Constraint

Now that we have decided on how to optimise the socio-economic values, education levels and distances, we can take a look at the population sizes. Our goal is to constrain the population sizes in such a way that we end up with communities that are somewhat equal in size. For this, we say that a community should not be smaller than 0.8 times the average community size, and now bigger than 1.2 times that. This results in,

$$L_P = \sum_{j=1}^{N_C} \mathbb{I}(\hat{P}_j < 0.8N_C \vee \hat{P}_j > 1.2N_C)\cdot \hat{P}_j \tag{6}$$

2.6 Objective

We have now derived the objectives for the socio-economic value, education level and the distances, as well as the constraint for population size.

Before adding them together, we want to decide on what regularization terms each term should get. As we will explain in section 3, the socio-economic term will be the deciding factor in our first estimate of the system. Therefore, we are okay with this term taking more of a background role during the refinement problem, which is why it gets L1 regularisation.

During refinement, the educational levels and population bounds should take a primary role, as it has been ignored in the first estimate. Experience also teaches us that the distance is a parameter that the program has a lot of trouble with in optimisation. Therefore, both population bounds and the distances will get L2 regularisation. Education seems to be harder to optimise, which is why we give it L3 regularisation.

We can now add everything together to create the next objective function:

$$\arg\min_{\theta} \left\{ w_S L_S + w_P L_P^2 + w_E L_E^3 + w_D L_D^2 \right\} = \tag{7}$$

$$\arg\min_{\theta} \left\{ w_S \sum_{j=1}^{N_C} \frac{1}{\hat{P}_j^2} \left(\sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot S_i P_i \right)^2 + w_P \left(\sum_{j=1}^{N_C} \mathbb{I}(\hat{P}_j < 0.8 N_C \vee \hat{P}_j > 1.2 N_C) \cdot \hat{P}_j \right)^2 \right.$$

$$\left. + w_E \left(\sum_{k=1}^{3} \sum_{j=1}^{N_C} \frac{1}{\hat{P}_j^2} \Big(\sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot E_{ik} P_i \Big)^2 \right)^3 + w_D \left(\sum_{i=1}^{N_B} \sum_{j=1}^{N_B} \mathbb{I}(\theta_i = \theta_j) \cdot D_{ij} \right)^2 \right\}$$

Costs are normalised and multiplied with their own weight so that we can decide their strength. We choose the weights 8, 35, 30, 35 for socio-economic score, population bounds, distance and education. This is done according to the same logic as named previously.

3 Method

To optimise the parameters, the next method has been implemented (code can be found at [2]). This method uses three distinct steps to generate the fairer communities. First, it initialises N_C buurten as communities, using K-Means clustering to make sure they are sparsely located. The $N_B - N_C$ leftover buurten are left unlabeled. We then make these communities spread out like a virus until all buurten have a label. Each community spreads out in such a manner that it equalises its own socio-economic score. After this is done, we let the communities fight over each-others territories in a game of risk. Now however, the leading factor for taking over buurten is the overall cost function as derived in equation 2.6.

3.1 K-Means Clustering Sparse Community Initialisation

Firstly, the starting positions for communities are calculated using the K-means clustering method to ensure they are located sparsely. The method involves the following steps:

1. Initialise the K-Means object and fit the data to it, with K equal to N_C.
2. Calculate the distances between each pair of centroids.
3. Find the pair of centroids with the maximum distance.
4. Merge the two centroids and re-fit the data to the K-Means object.
5. Repeat steps 2-4 until N_C centroids are obtained.
6. Return the final centroids.

Polygons downloaded from [3] are used to determine which buurten share a border, creating a list of neighbours. It is assumed that polygons that intersect at more than one location share a border. This method works 98% of the time. However, due to geographical dependencies, some buurten do not neighbour in such a manner. Therefore, buurten that have not found neighbours following this algorithm are given neighbours according to a calculation which only takes one intersection into account, and two extra via a nearest neighbour algorithm.

3.2 Initialising the Labels

The labels are initialised using an algorithm that is able to label the buurten accordingly. The algorithm runs until all buurten belong to a community, and each step involves shuffling the communities to introduce randomness. The communities then take turns iterating over their neighbours and choose to add a neighbouring buurt to themselves by selecting the buurt that improves the overall socio-economic score of the community the most, meaning it tries to converge on the average socio-economic score of the total area. Communities are not allowed to take over a buurt that already belongs to another community. The model updates all values according to the newly chosen label. In case the labels don't change for over 100 iterations, the model force quits itself, and buurten that have not been initialised yet are assigned to the label that their neighbour belongs to. This method is similar to the spreading of a virus, ensuring that the socioeconomic value of the total is optimised and all buurten are labeled.

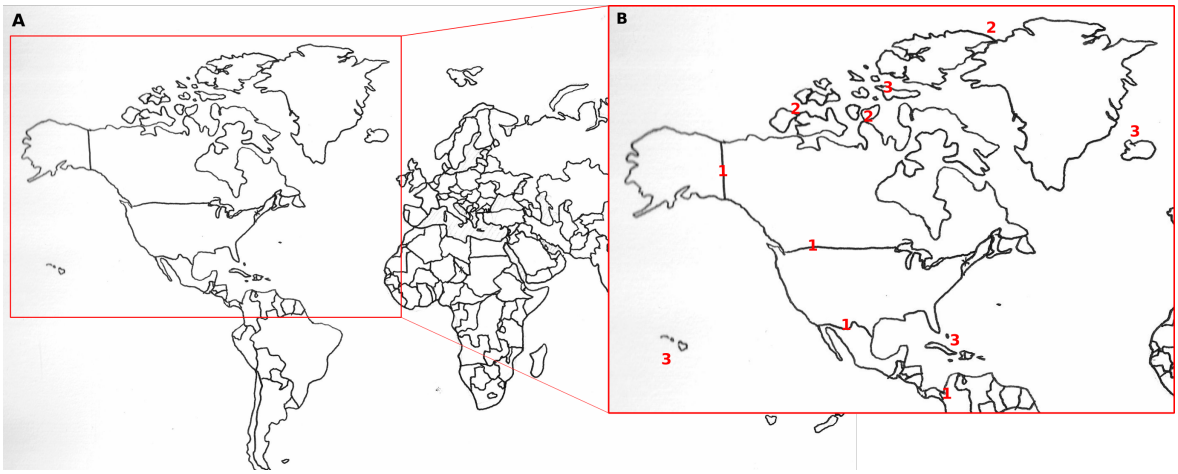

Figure 1: Generating Neighbours. In **A** the total map is shown over which we will run our algorithm as an example. This is purely to show the workings and therefore, except for the geographical borders, no information about the real world should translate to this figure. In **B** we zoom in on a part of the map to show the particular cases of neighbour generation. Firstly, neigbhours are calculated as polygons that intersect in more than one point, which often translates to having a shared border (indicated by the number 1). This should generate neighbours for most of the areas. For the few areas that haven't been assigned a neighbour yet, we define neighbours as intersecting at one point (indicated by the number 2). As these last areas often only touch one other area, we will generate 2 extra neighbours via a nearest neighbour algorithm (indicated by the number 2).

3.3 Refining the Labels using the Potts Model

Thereafter, the labeling of a set of N_B points is refined using the Potts model to minimise a cost function. The algorithm iteratively updates the label of each point, one at a time, while keeping the labels of all other points fixed. The goal is to find a labeling that minimises the cost function. The algorithm is run for a fixed number of iterations and a temperature is added to introduce a level of randomness to the whole optimisation process.

This algorithm looks a lot like a game of Risk, in which factions fight over each others territories. Now however, territories are stolen according to the overall cost function used by the algorithm as specified by the equation 2.6 as derived in the previous section. This cost function focuses on socio-economic value, education levels, population sizes, and distances, and as it is calculated over each and every faction, this is more a cooperative game than an competitive one, as the communities must work together to get the best overall score.

During refinement, the cost of each labeling is calculated and saved, and the temperature parameter is gradually reduced by a factor of 0.99 at each iteration, reducing the randomness and focusing the algorithm on finding more precise solutions. To ensure that the community is continuous, an error is built in that does not allow blobs to be cut in two, meaning a community must always be connected to all its territories.

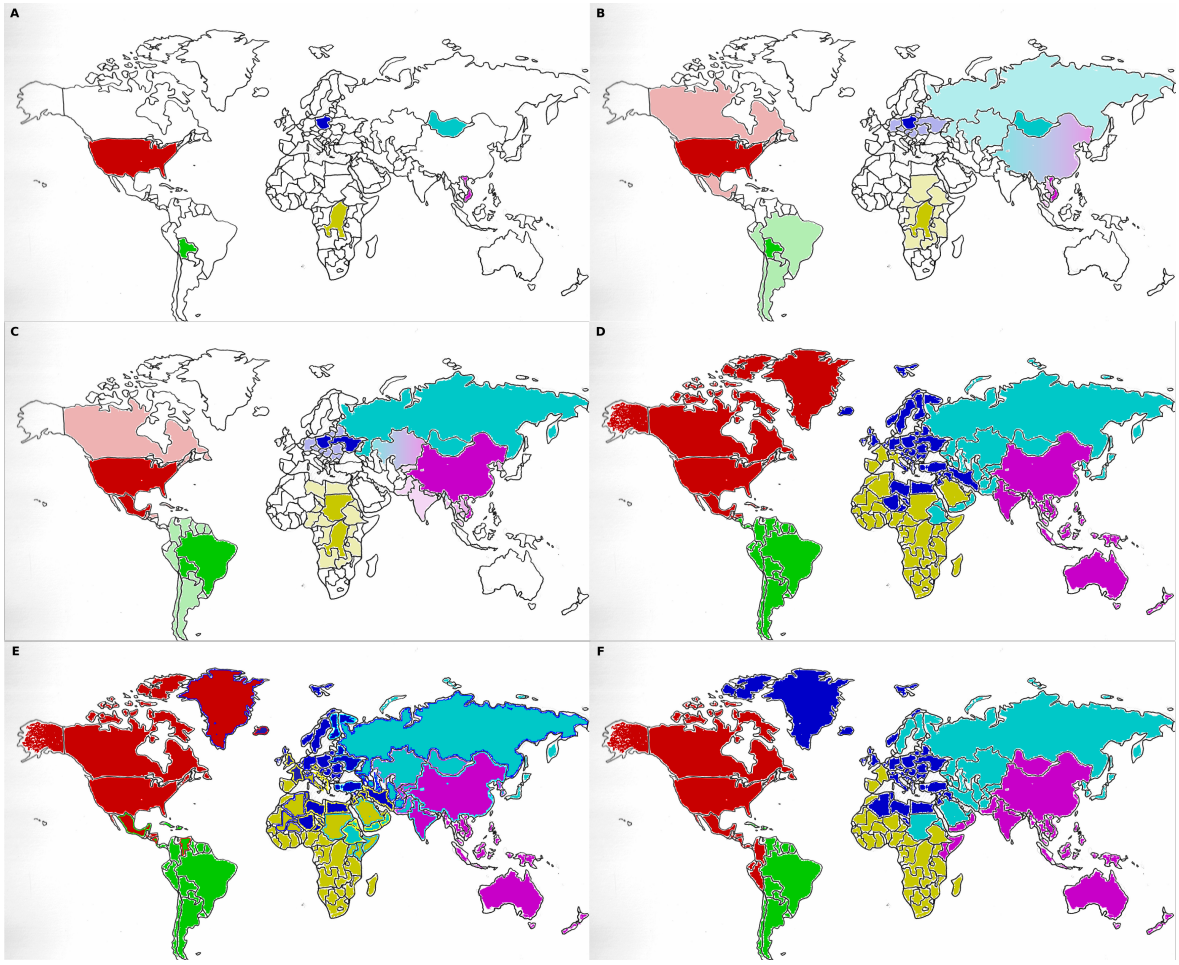

Figure 2: Showing how the algorithm works on the world map. Except for the geographical shapes of the borders, we must assume that this map has no similarities to the real world. In **A** we initialise 6 communities located sparsely over the map using K-means clustering. Each community is assigned its own color, all other areas are not initialised as part of a community just yet. We then calculate how the socio-economic scores of each neighbour would change the community in **B**. The neighbour that optimally averages out the socio-economic score of that community is then chosen in **C**, after which we again calculate the socio-economic scores of all the neighbours. This process is repeated, such that the communities will spread out in an organic way, till each area is assigned a community as in **D**. Communities are not allowed to steal areas from each other. This state is the first estimate of our model. Next, we will try to refine the communities via the Potts model. In **E** communities are allowed to steal each others territories, if this improves the total cost score calculated by equation 2.6, which in this case focuses not only on the socio-economic value, but also the education levels, population sizes of the communities and the distances between areas within the same community. One hard limit is that communities are not allowed to steal territories from each other if that means another community will be split in two. This process is repeated for a predetermined amount of iterations, after which we hopefully end up with **F**, the optimal communities according to the cost function.

4 Results

After running the algorithm on the data from Amsterdam, we arrive on the results in figure 3. In it, we see the geographical shape of the communities after initialisation (section 3.2) and after refinement (section 3.3). We also show how the costs (equation 2.6) develop over 100 iterations. Lastly, we show the education levels, the population sizes and the socio-economic value for the neighbourhoods and for the communities before and after refinement.

4.1 Distances

From figure 3A and B we can see that the refinement process does not greatly improve the distances too much. This is also confirmed when we look at how the costs progress in figure 3C, in which we see the distance staying somewhat constant. This indicates that either the costs of distances are extremely hard to optimise, or that the initialisation process already arrives at a pretty good optimisation costs. This last theory is likely, especially if we look at figure 3A, which we find to be satisfactory. This means that the distance costs as implemented in this model effectively work as a constraint that does not allow the communities to spread out too much. We could give the distance a higher weight in case we think initialisation does not work satisfactory, but this is not the case as of yet.

4.2 Education

According to figure 3C we see a big improvement in the variance within education levels. Although figure 3D does confirm a bit of improvement, it is clear that this improvement is not as strong compared to the other parameters, even though this was part of the main focus of the refinement process. Other regularisation terms or weights for this parameter did not result in a better variance, indicating that this is more or less a hard limit.

4.3 Population

In figure 3E we see how much the population sizes of the various stages of the communities are distributed. As the initialisation does not generate equal size communities, this is an important thing to optimise during refinement. In figure 3C we see that our method does vastly increase the costs of the population bounds. In figure 3E we see that, although overall the sizes are improved, there are still a few outliers. Two of these are probably the islands seen in figure 3A and B that are not connected to the main area, and are therefore unable to be labeled differently. To solve this, we should implement a neighbouring algorithm that is able to cross blank spaces.

4.4 Socio-Economic value

In figure 3F we see that the variance socio-economic value is already pretty good after initialisation. Shockingly however, the variance is even improved a bit better during refinement, which is also confirmed in figure 3C. We expected the variance to already be optimum after initialisation, as that is what the process focuses on, but now we see that this score can be improved upon further during refinement. This is probably due to the communities having more freedom (in stealing territories from each other).

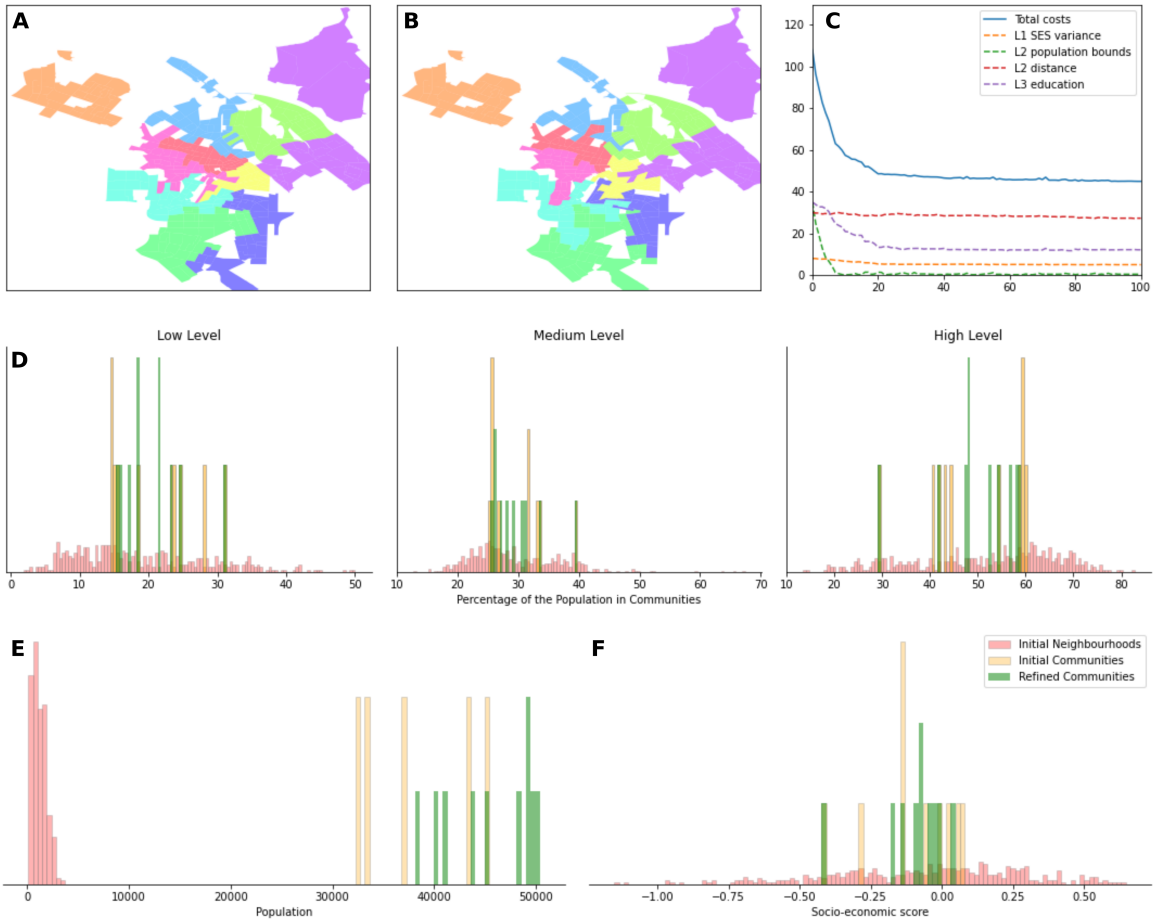

Figure 3: After initialising communities in Amsterdam via the method described in section 3.2, we arrive at **A**, the initialised communities based on socio-economic data. After running the refinement algorithm from section 3.3 (with a temperature of 0.05), according to the cost function of equation 2.6, we arrive at **B**. The progression of the costs over 100 iterations can be seen in **C**. In **D**, we see a bar plot of the education levels of the neighbourhoods, in the initial communities, and after refinement. The same has been done for the population sizes in **E** and the socio-economic value in **F**.

5 Conclusion and Discussion

Not average economic value but a diverse one. What is the difference.
Other parameters

A Comparing the Algorithm for Different Temperatures

B The Continuous Population Based Algorithm

References

[1] CBS. *Sociaal-economische status; scores per wijk en buurt, regio-indeling*. 2021. URL: https://opendata.cbs.nl/statline/{\#}/CBS/nl/dataset/85163NED/table?dl=7A1C1.

[2] HMels. *GitHub Inverse Gerrymandering*. 26-04-2023. URL: https://github.com/HMels/inverseGerrymandering.

[3] /www.atlasleefomgeving.nl. *Wijk- en buurt informatie*. 2022. URL: https://www.atlasleefomgeving.nl/kaarten.