Diversifying Populated Districts via an Inverse Gerrymandering Optimisation Algorithm

Mels Habold

May 16, 2023

Summary

In this paper we worked on using an Inverse Gerrymandering Optimisation Algorithm, which focused on drawing Geographical Borders through cities to define Communities that are more equal and diverse. The goal is to redefine how we look at cities to connect people of different social standings and backgrounds in order to create a more resilient communities. The algorithm uses so-called buurten (which often contains part of a handful of streets) in Amsterdam, The Netherlands as building blocks of communities. It first initialises certain buurten as the center of communities to be with a K-Means clustering algorithm, to make sure the communities are located sparsely over the whole area. Then it lets them spread out iteratively like a virus by using the best socio-economic score as a metric of optimisation. Lastly, it refines these communities via Potts model, by focusing on the variances in population-sizes, socio-economic value, and distribution of education levels between communities. We have shown that our method is well capable to create new communities with better average social scores, as named before.

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 distribution of educational levels. We want people in communities to be close to each other, and therefore, we start out with defining them by one continuous geographical border. 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.

We start out by describing the problem together with the parameters and constraints in section 2. Next up, we will describe the method in section 3. In section 4 we will describe the results and benchmark them. Lastly, in section 5, we will discuss the results and draw conclusions.

2 Problem Description

This method is based on data from CBS on socio-economic scores, distribution of educational levels, population sizes, etc per city, wijk and buurt [1]. We choose to define the building-blocks of the data to be 'buurten', which is a Dutch term for an area that is part of a neighbourhood. These buurten contain 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 take into account 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 j. 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 the center of community 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 communities and buurten (size: $N_B \times N_C$)

2.3 Mapping the Parameters

Next, we would like to know exactly how the labeling θ impacts the parameters. First we take a look at how the populations of the new communities are distributed,

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

Which is the population of community j. $\mathbb{I}(\theta_i = j)$ is a function that equals 1 if the condition is satisfied, meaning buurt i is labeled under community j and 0 otherwise. Education levels are

distributed as percentages over the population, which means that we first would need to multiply the percentage of E_{ik} with the population of its buurt,

$$\hat{E}_{jk} = \frac{1}{\hat{P}_j} \sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot E_{ik} P_i \tag{2}$$

For the socio-economic value, this is not calculated per person but as a total of the buurt. This means that it is only cumulative,

$$\hat{S}_j = \sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot S_i \tag{3}$$

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 Variance of a parameter between Communities. For the socio-economic value this should look like,

$$L_S = Var[\hat{S}] = \frac{\sum_{j=1}^{N_C} \hat{S}_j^2}{N_C} = \sum_{j=1}^{N_C} \frac{1}{N_C} \left(\sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot S_i \right)^2 \tag{4}$$

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,

$$L_E = \sum_{k=1}^{3} Var[\hat{E}_k] = \frac{\sum_{j=1}^{N_C} \hat{E}_j^2}{N_C} = \sum_{k=1}^{3} \sum_{j=1}^{N_C} \frac{1}{N_C \hat{P}_j^2} \left(\sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot E_{ik} P_i \right)^2 \tag{5}$$

2.5 Constraint

Now that we have decided on how to optimise the socio-economic values and education levels, we can take a look at the constraints of the model. For this we will focus on the distances, to make sure the population per community lives close together, and the population sizes, to make sure communities are similar in size. For the optimisation of the distance, we take the mean distance for each buurt to the center of the community that they are part of,

$$L_D = \sum_{j=1}^{N_C} \frac{1}{\sum_i^{N_B} \mathbb{I}(\theta_i = j)} \sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot D_{ij} \tag{6}$$

In which we normalise the distance matrix D_{ij} by the number of buurten in the neighbourhood j in order to make it the mean distance. Next, 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 again focus on optimising the variance of the population-sizes per community,

$$L_S = Var[\hat{P}] = \frac{\sum_{j=1}^{N_C} \hat{P}_j^2}{N_C} = \sum_{j=1}^{N_C} \frac{1}{N_C} \left(\sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot P_i \right)^2 \tag{7}$$

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 in an objective function, we want to decide on the regularization term for each parameter.

As we will explain in section 3.2, the socio-economic score will be the deciding factor in the initialisation of the communities. Therefore, we let this term take a background role during the optimisation, which is why it gets L1 regularisation. This also means that the education, distance and population parameters should get a more primary role in optimisation. Therefore, both population bounds and the distances will get L2 regularisation. Experience teaches us that the education parameter 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:

$$\begin{aligned}
&\arg\min_{\theta} \left\{ w_S L_S + w_P L_P^2 + w_E L_E^3 + w_D L_D^2 \right\} = \qquad (8)\\
&\arg\min_{\theta} \Bigg\{ w_S \sum_{j=1}^{N_C} \left(\sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot S_i \right)^2 + w_P \left(\sum_{j=1}^{N_C} \frac{1}{N_C} \left(\sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot P_i \right)^2 \right)^2 \\
&+ w_E \left(\sum_{k=1}^{3} \sum_{j=1}^{N_C} \frac{1}{\hat{P}_j^2} \Big(\sum_{i=1}^{N_B} \mathbb{I}(\theta_i = j) \cdot E_{ik} P_i \Big)^2 \right)^3 + w_D \left(\sum_{j=1}^{N_C} \frac{1}{\sum_i^{N_B} \mathbb{I}(\theta_i = j)} \sum_{i=0}^{N_B} \mathbb{I}(\theta_i = j) \cdot D_{ij} \right)^2 \Bigg\}
\end{aligned}$$

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 cost function as derived in equation 2.6, which is calculated over all communities, which means the game of risk is rather cooperative.

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.

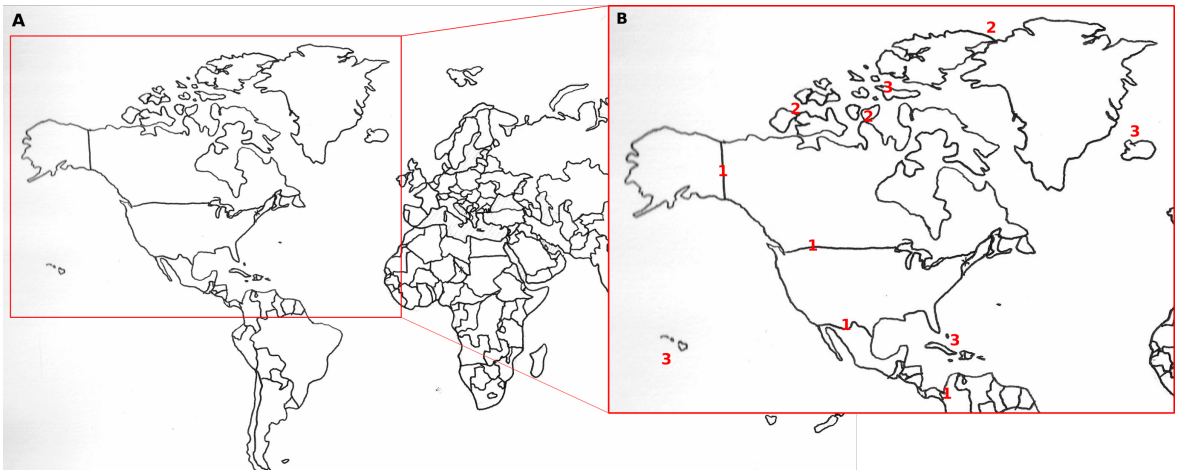

Figure 1: Generating Neighbours. In **A** the total map is shown over which we will run our algorithm as an example. This is purely to show the workings and therefore, except for the geographical borders, no information about the real world should translate to this figure. In **B** we zoom in on a part 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. 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.

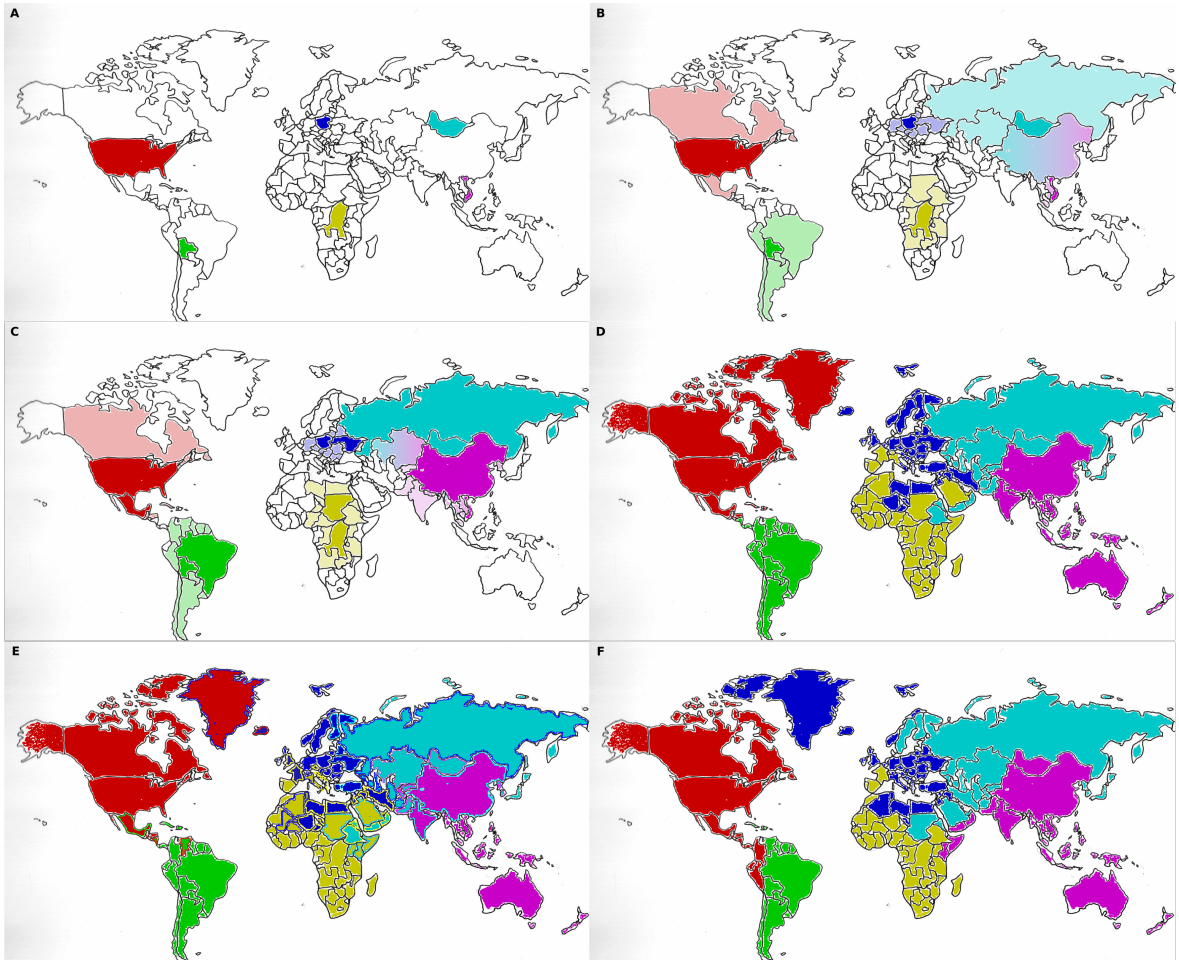

Figure 2: Showing how the algorithm works on the world map. Except for the geographical shapes of the borders, we must assume that this map has no similarities to the real world. In **A** we initialise 6 communities located sparsely over the map using K-means clustering. Each community is assigned its 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 optimal 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).

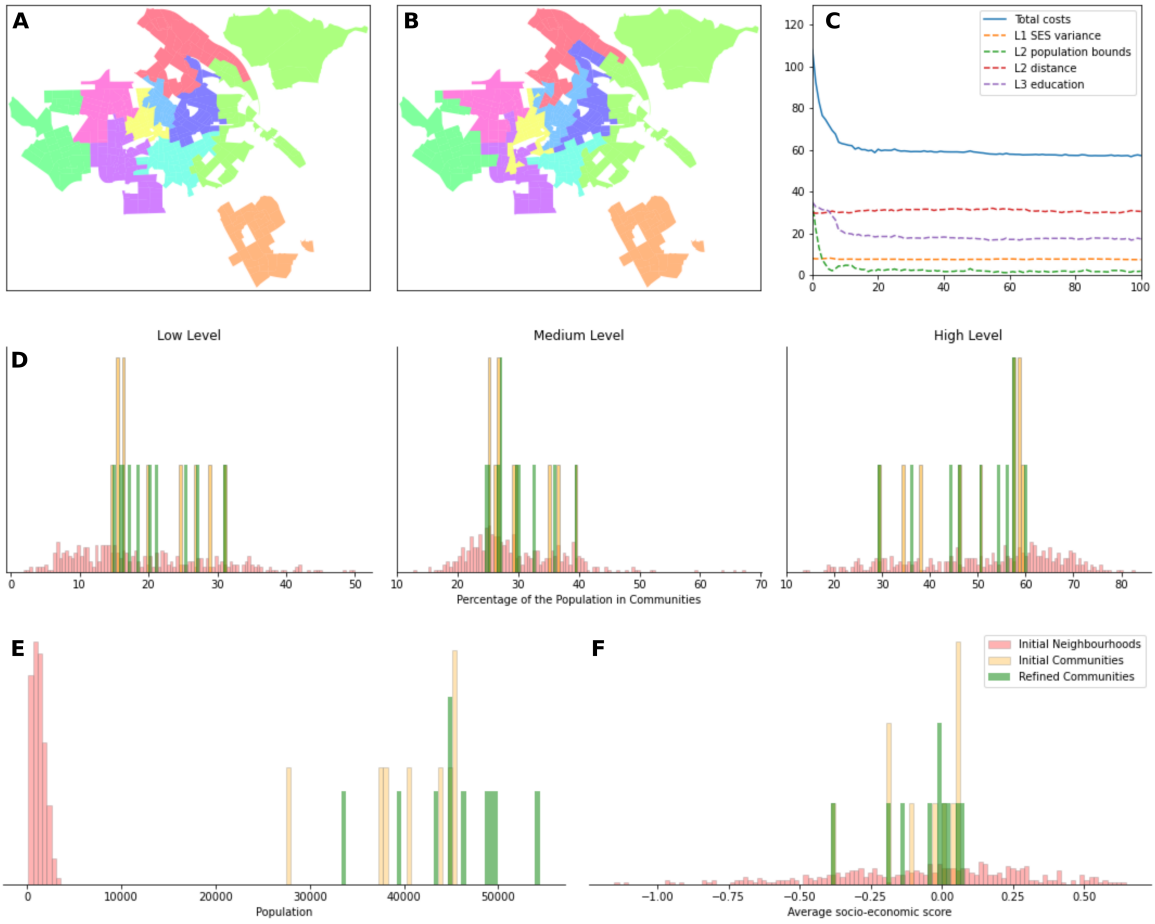

Figure 3: After initialising communities in Amsterdam via the method described in section 3.2, we arrive at **A**, the initialised communities based on socio-economic data. After running the refinement algorithm from section 3.3 (with a temperature of 0.05), according to the cost function of equation 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**.

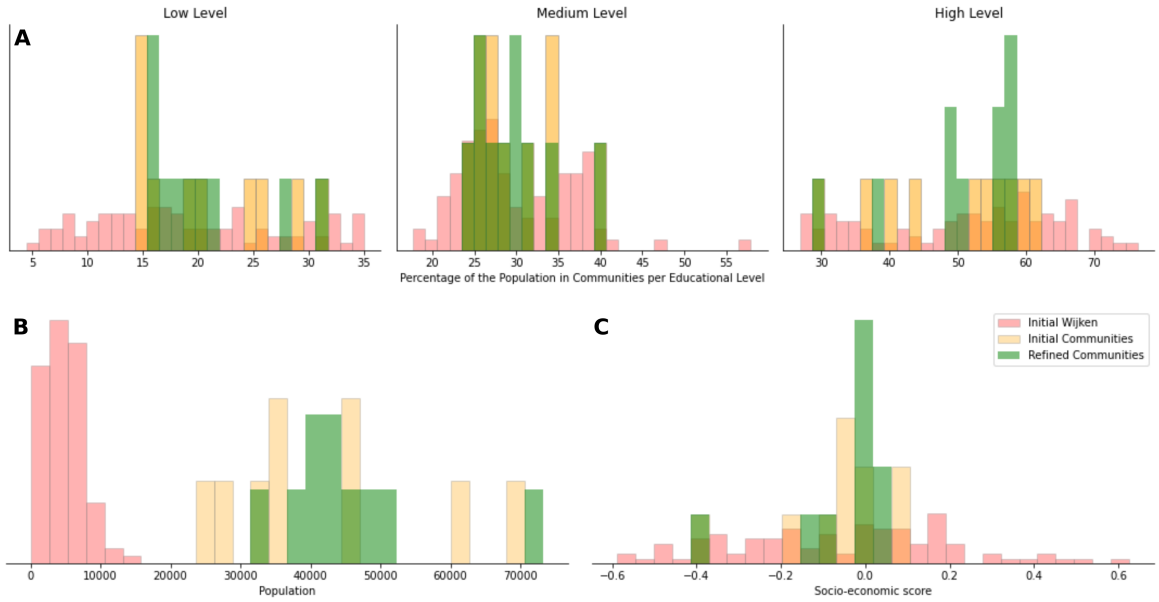

Figure 4: We compare the results gained by the previous method to the way things are ordered now. For this, we take the neighbourhoods (wijken) as they are composed of currently. In **A** we see a bar plot of the education levels of the neighbourhoods, in the initial communities, and after refinement. The same has been done for the population sizes in **B** and the socio-economic value in **C**.

4.5 Comparison to original Neighbourhoods

We would like to compare our results to the way neighbourhoods are composed as of yet. Amsterdam consists of 100 neighbourhoods (wijken) that all contain a handful of buurten. The comparison between the neighbourhoods as they are now, and the communities that we have generated, can be found in figure 4. As we can see, the communities are distributed much more equally.

4.6 Comparison to random Neighbourhoods

As the original neighbourhoods only contain a few buurten, this comparison is insufficient to validate our model. To benchmark our results, we would like to see how much our method improves the parameters compared to a randomly generated dataset. For this, we use a similar method as we have done in the initialisation of the communities (as described in section 3.2), but now we let the communities spread out randomly instead of towards the best socio-economic value. After doing this 50 times, we can average out the random results and compare them to our previous results. The comparison can be found in figure 5.

We can clearly see that the socio-economic value and the population sizes of our method are a good bit better than the randomly generated ones (ignoring the outlier in this case). If we look at the education levels, we see some improvements, but we do notice that the overall variance is still a bit high. It is important to note that the scale on the x-axis is much smaller the one seen in figure 3D. This clearly shows that adding together buurten in a random manner already results in a very strong improvement in the variance of a community. The same argument can be made for the socio-economic score.

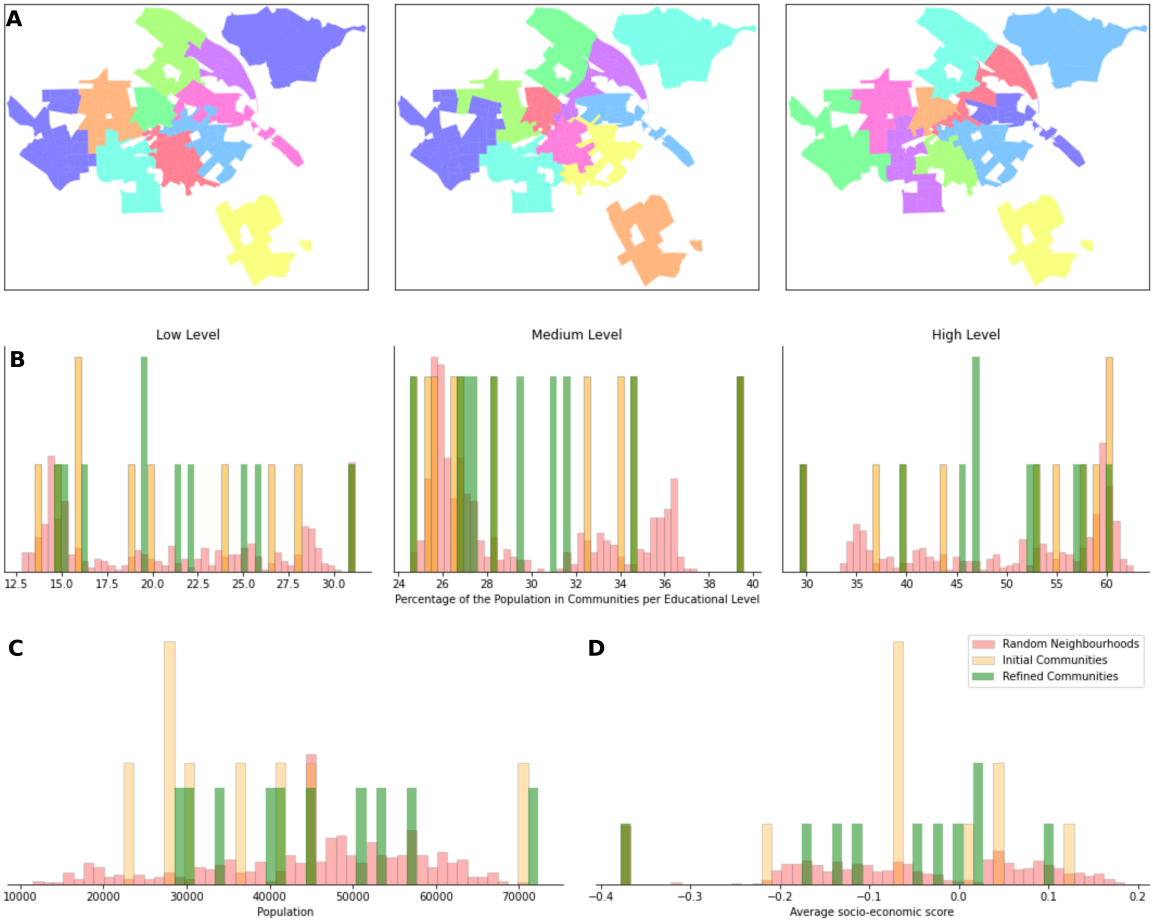

Figure 5: **A** 50 (of which 3 are shown) communities are created by letting them spread out randomly. We can compare the results from our method with these randomly generated communities by averaging out their results. In **B** we compare the distribution per education level for both the average randomly generated neighbourhoods. The same has been done for the population sizes in **C** and the socio-economic value in **D**.

Table 1: Standard Deviation (σ) Different Parameters between Communities as defined by our Method. Each row contains the standard deviation per parameter for different versions of labeling of buurten per column. The first column contains the Wijken, which is the neighbourhoods as they are defined now. This is mainly put here for reference, as the Wijken are much smaller than the Communities we want to create. Secondly, we have put the Random Communities from figure 5 in there. These Communities are created by letting them spread out without preference. The third column contains the Initial Communities as defined by section 3.2, which lets Communities spread out based on the Socio-economic score. The last column contains the Refined Communities from section 3.3, which takes the previously defined Initial Communities and iteratively refines them based on variances in all parameters.

$\sigma_{parameter}$	Wijken	Random Communities	Initial Communities	Refined Communities
Socio-economic Score	0.259	0.148	0.136	0.131
Population Size	—	11,680	11,348	5,489
(Lower) Education	8.358	6.050	5.974	5.179
(Middle) Education	6.789	4.866	4.994	4.615
(Higher) Education	13.356	10.787	10.840	9.640

5 Conclusion and Discussion

We have clearly shown to our method is well able to improve the variance of different parameters. We have also compared our results with the original wijken have even benchmarked our results by comparing it to randomly created communities. The results are best examplified by looking at the eventual standard deviation:

We clearly see that all scores are greatly increased by adding buurten together. And compared to the randomly adding together, we even see a big improvement in our initialisation method and refinement method.

Of course, our algorithm also has some limits. Most importantly, right now the method is not so good at finding neighbours for buurten that are geographically separated. This means some communities are not really able to be optimised as they are locked by the geographical boundaries. This might however not really be a big problem as we do want communities to be geographically close together.

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.