Application of Genetic Algorithms for Urban Planning

By Cameron Calder

AAI 646 Spring 2023

ABSTRACT

This project aims to explain the methods and accuracy of converting a city map to a "chromosome" that can be used in a genetic algorithm to understand the relationships between different industries during urban planning. Using this algorithm, the effects of different parameters such as distance to public transport or commercial density can be studied to choose the best land type for regions up for redevelopment as well as the best parameters for quantifying the "success" of a city map.

Public resources from the U.S. Census Bureau and free ecological modeling tools are used to pull data about land use and allocation. A genetic algorithm performs two-point crossover and random index mutations with a random probability to introduce more complexity into the data and create new map combinations. The fitness of residential, industrial, and commercial cells is then calculated to find the highest scoring configurations in each generation and keep the top 50 "fittest" individuals out a population of 100. Multiple "evolutions" were then run using different weight initializations to expand the search space and find the best maps overall in comparison to the original configuration. These generational maps showed the effectiveness of the chosen parameters and how cells are "naturally" arranged in a comparatively simple model for the problem of urban planning.

INTRODUCTION

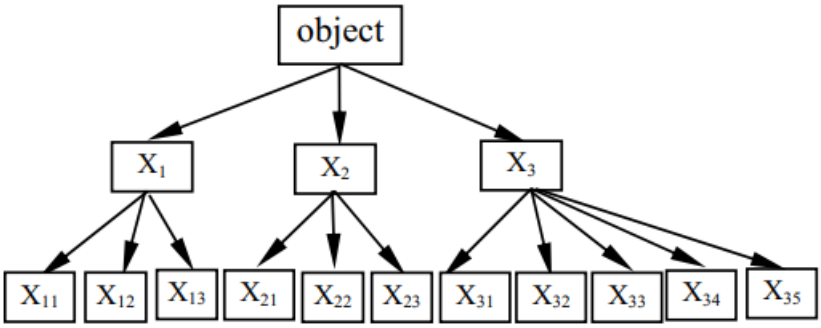
Urban planning is a highly complex problem where the needs of multiple interests must be balanced while staying robust to changing dynamics over time. The city of Hoboken provides an excellent case study for this problem where the combination of industry, commercial business, and high-density residential living is compressed into two square miles. Many redevelopment plans have been proposed to keep up with changing needs, most recently with unused areas that used to be heavily industrial during a time of high residential demand. With the increased pressure for housing and businesses following the suburban sprawl during the COVID-19 pandemic, urban planners must stay innovative to draw in residents and the resulting revenue.

The grid structure of Hoboken also makes it easy to section and encode into a "chromosome" that can be recombined and shuffled to explore the effects of different land distributions. In this project, a grid map was made with specific parameters for the fitness of residential, commercial, and industrial needs. Other factors such as access to public transport and areas of

high foot traffic such as the city center and near Stevens are used to incorporate density information. This paper provides an in-depth analysis of the methods used to create this dataset and how a genetic algorithm can be applied to visualize the effects of spatial distribution and parameter weights to maximize performance in urban planning.

RELATED WORK

Multiple sources were considered to decide on the cell types and fitness parameters when applying the algorithm. The paper by H. Xin and Z. Zhi-xia [4] shows how genetic algorithms can be applied to spatial distribution in urban planning using an indexing system to create relationships between sectors and their parameters, shown in Fig. 1. A similar structure was employed in this project.



$$f(x) = c_1 g(x_1) + c_2 g(x_2) + c_3 (x_3)$$

$$y = \min f(X_1, X_2, X_3)$$

Fig. 1: Target evaluation system, fitness function, cost function used in [4].

The overall fitness for an arrangement was determined by the proportion of 'profit' from the cost function relative to each sector. Crossovers and mutations are applied at each generation to create new distributions within a population of 100 individuals.

A paper comparing different parameters and types of urban growth models to study land allocation and configuration [1] was also used to choose fitness parameters. Based on their results, parameters like flood chance and public transport distance were added to this project's model. The calibration methods used in the reference study are shown below.

Table 2. Input data use to calibrate each urban growth model. All data are 30 m resolution.

Parameter	Description	Base Data	Year(s)	Data Source
Landcover	Lands designated as developed and undeveloped	Landsat MSS and TM imagery	1976, 1985, 1996, 2006, 2015	United States Geological Survey
Roads	Road networks, proximity to roads, distance to interchanges	Primary and secondary road networks	1976, 1985, 1996, 2006	North Carolina Department of Transportation
Topography	Slope, elevation, and aspect	National Elevation Dataset	2006	United States Geological Survey
Population	Observed population	State demographic data	1976, 1985, 1996–2015	State Demographics Branch of the North Carolina Office of State and Budget Management
Hydrograpy	Proximity to water bodies	Rivers, lakes, and reservoirs	2006	National Hydrography Dataset
Municipal Centers	Proximity to municipalities	Locations of cities, towns, and counties	2006	United States Census

Fig. 2: Calibration parameters used in [1].

METHODS

a) Data Collection

Based on data collected by the U.S. Census Bureau, the "city [of Hoboken] has had a total area of 2.00 square miles (5.18 km^2), including 1.25 square miles (3.24 km^2) of land and 0.75 square miles (1.94 km^2) of water" [6]. A map of Hoboken from the Census Bureau, shown in Fig. 3, was copied into Excel and uniformly resized to fit each city block to approximately one cell. The contained cells were classified as "land" by changing their fill color.

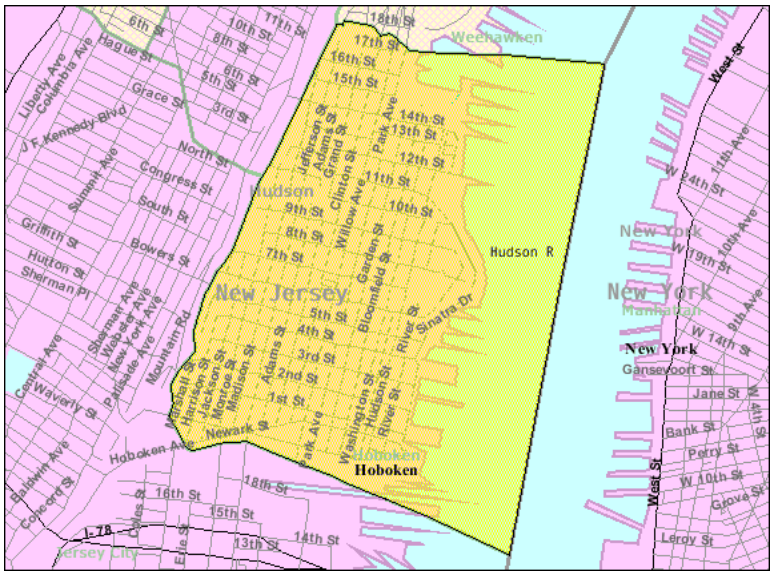


Fig. 3: Census Bureau Map of Hoboken.

A grid boundary was then created around the entire map, and remaining cells were classified as “land”, “water”, or “non-Hoboken”. Classifications were adjusted to fit the map as closely as possible while maintaining the same ratio of land to water specified by the Census. Cells were then re-classified based on a Hoboken zoning map from a recent redevelopment plan and property type classification codes from the NY State Dept. of Taxation and Finance, as well as Google Maps for the locations of features like fire stations (see Appendix A,B for details). The redevelopment plan was used to separate the commercial, residential, and industrial zones as well as “undecided” areas marked for redevelopment.

After deciding the possible cell labels, each color was assigned a number to populate the entire map for further analysis. Some rows of water were also removed, as only the location of the land/river boundary are needed for fitness evaluation after applying the evolutionary algorithm. The resulting initial map is plotted in Fig. 4, rotated 90 deg. clockwise to orient the Hudson River along the x-axis for visual clarity.

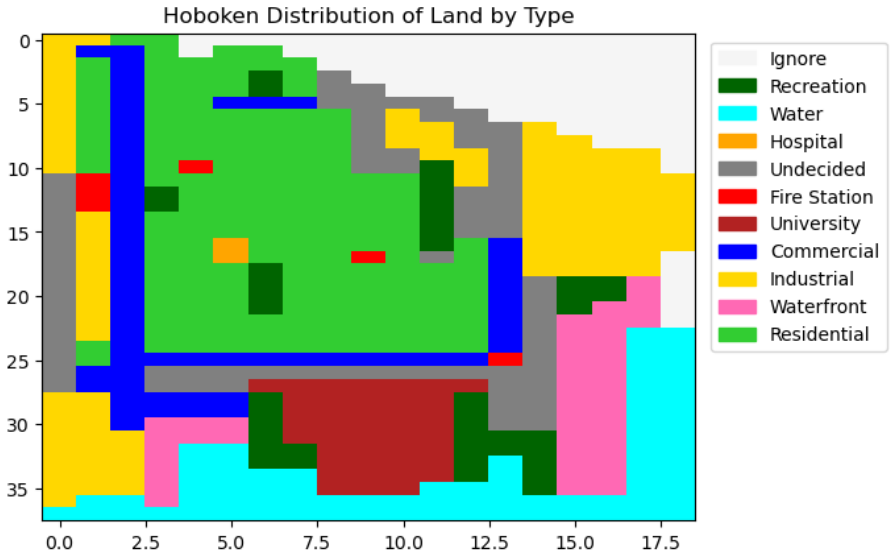


Fig. 4: Labeled map of property code distribution.

Another factor that is especially relevant in Hoboken is flood risk. This was quantified by extracting areas of high risk where the 2-year flood inundation average is >= 2.4 ft. above the mean higher high water based on data collected with the NJFloodMapper tool (Appendix C). This metric means these areas during flood events have at least 2.4 ft. more water than the average height of the highest tides during an average tidal cycle. This can also be thought of as areas with high likelihood of severe storm surge. This map was converted into a labeled grid using the same methods for the full property code map, resulting in the map shown in Fig. 5. The waterfront, river, and non-Hoboken areas are considered non-applicable for flood risk and marked in white.

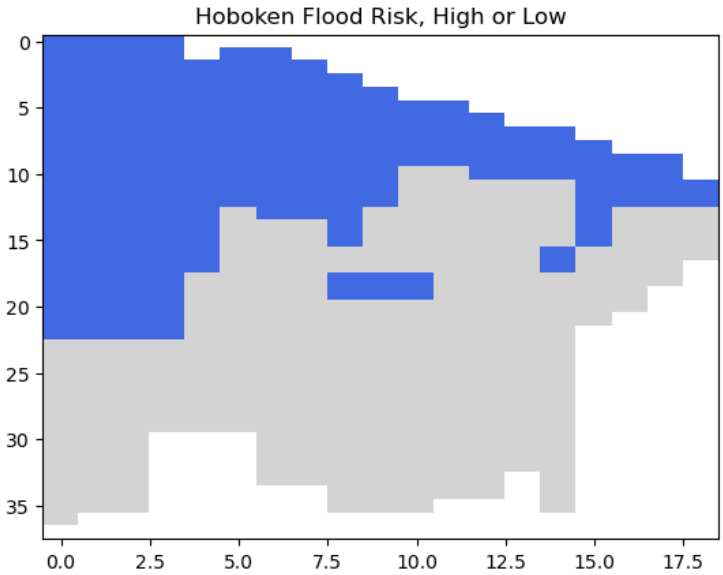


Fig. 5: Grid of map flood risk (blue = HIGH, grey= LOW, white=not applicable).

After finalizing the maps, each cell was tagged based on crossover applicability to decide if it should be included in the "chromosome" during "evolution" as 0 (permanent) or 1 (movable). Water, non-Hoboken areas, university, hospital, fire station, and waterfront cells were classified as 0 and all other cells as 1. The color-coded maps for land codes, flooding, and crossover inclusion are separated into three sheets within the 'map_tuples_color' workbook, with the simplified sheets used in the actual programming implementation in the 'map_layers' workbook.

b) Population Initialization

To initialize the population and create the mating pool, the land code grid and crossover inclusion grid are first converted to numpy arrays. The cells at the locations where crossover = True (1) are then extracted. The cells marked "undecided" (2) are replaced with random land codes corresponding to crossover-eligible types. The array is then flattened to a 441x1 vector to represent the chromosome for a single individual in the population. The entire initial population is created by combining this vector with 99 vectors of the same length containing random combinations of eligible land types to create a population array of size (100, (441,)). This first population is used to initialize the mating pool.

c) Evolutionary Algorithm

The initialized mating pool is then passed to the evolutionary algorithm function evo_algorithm() with the following parameters:

mating_pool: Initial mating pool (includes original map)
i_cells: grid cell crossover eligibility tags
l_cells: grid cell land type tags (full map)
generations: number of times to “mate” population couples
crossover_rate: threshold for crossover chance (0.7)
mutation_rate: threshold for mutation chance (0.02)
size: number of individuals in the population (100)

In a single generation, the mating pool is split into consecutive mating pairs. For each pair of parents, children are created using the crossover_event(parent1, parent2, crossover_rate) function. The “random” module is used to generate a pseudo-random float from 0 to 1.0. If the random float is less than the crossover rate, two-point crossover is performed by picking a random slice and switching the elements within the slice between the two parents. If the generated number is larger than the crossover rate, the same parents are used as the children.

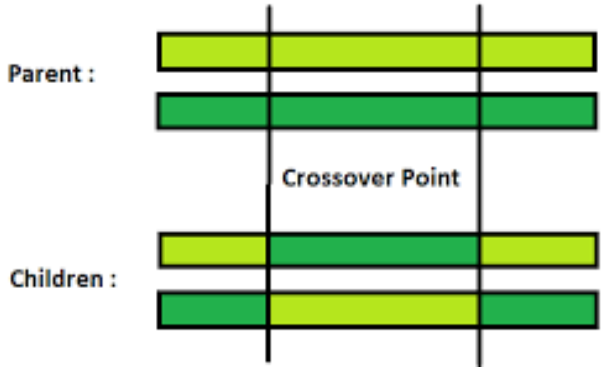


Fig. 6: Two-point crossover example.

The children then move onto the mutation event. Two random floats are generated to represent the mutation chance for each child. If the mutation chance is less than the mutation rate, two “alleles” are randomly switched within the child’s chromosome. The resulting children from all the mating pairs are used to update the mating pool and re-mapped onto the original map cells. The fitness of each child’s map is evaluated and the top 50 children are kept for the next generation. 50 more randomly generated children are used to make up the rest of the generation. These steps are repeated until all generations are completed to get the final population.

d) Fitness Evaluation

To quantify fitness for residential, commercial, and industrial cells, the average distances to different cell types as well as densities of the same cells are used. The final fitness scores for each type – ‘r’, ‘c’, ‘i' – are multiplied by their respective flood score. Flood score is

calculated using the get_density(padded, pad, cond) function, where the grid is padded using a specific ‘pad’ integer. The ratio of high risk to low risk cells that within a surrounding area is calculated to get flood risk, shown in Fig. 7 below.

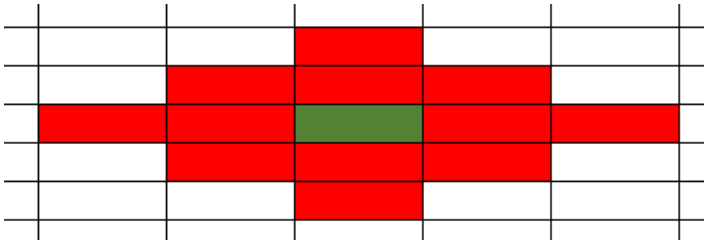


Fig. 7: High risk cell density (red) for a given cell (green).

High risk tags are specified by ‘cond’. The average flood score for each cell type is multiplied by a given number based on the perceived importance of flood mitigation, shown below.

```
flood_weight = {
    'r':flood_score(5)*1,
    'c':flood_score(9)*1.3,
    'i':flood_score(4)*1.8
}
```

Fig. 8: Flood weight specifications and perceived level of importance.

The fitness of a child is determined based on the average fitness score for residential, commercial, and industrial cells. The indexes of cells within each group, as well as the location of transportation, fire/health services, and the center city cell are stored in a dictionary for distance and density calculations in each sector. Weights for each sector are initialized as ‘1.0’ for each parameter.

Residential

For residential fitness, the parameters are calculated for all ‘r’ cells using Euclidian distance:

comm_dist_fit: Average distance to commercial cells
center_fit: Distance to the city center
ind_dist_fit: Average distance to industrial cells
park_dist_fit: Average distance to recreational areas/parks
trans_dist_fit: Average distance to public transport
services_fit: Average distance to fire/health services

The fitness score for each parameter is adjusted by its respective mean and averaged. If the parameter needs to be maximized, such as distance to industrial areas, is adjusted by (1-avg. score). The mean of all the average scores is used as the final recreational fitness score.

Commercial

For commercial cells, the following parameters were used:

comm_dens_fit = Ratio of commercial to non-commercial cells in surrounding cells
water_dist_fit = Average distance to the waterfront
trans_comm_fit = Average distance to public transport
uni_dist_fit = Average distance to Stevens
comm_ind_fit = Average distance to industrial cells

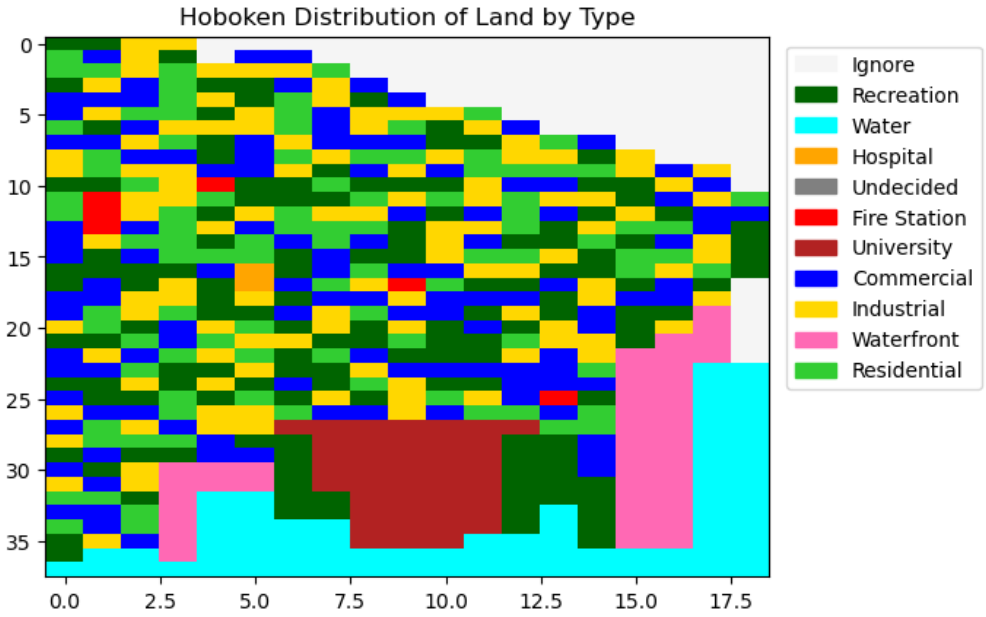
Distances were calculated using the same Euclidian formula as residential. Distance to the waterfront and Stevens are prioritized for commercial cells due to increased foot traffic, and distance to industrial areas are maximized. Commercial density is also considered and maximized due to increased foot traffic from customers of surrounding businesses. Density was calculated using the same procedures as the flood score using '-1' for padding and '9' as 'cond' for commercial cell tags.

Industrial

For industrial cells, industrial density was maximized using the same procedures described for 'c' to encourage creation of industrial districts. Distances to residential cells were also maximized due to air and noise pollution. Distances to transit as well as health/fire services were also considered using the same procedures are 'r' and 'c'.

RESULTS

After the initial "evolved" map was made, the parameter weights were randomized between 0 and 3 and used to create a new map. The highest scoring map for each evolution over 5 are shown. An example of a map with a score of 74% is shown below.




```
Weights: {'r': [2.8361, 1.11353, 2.6255, 2.6653, 2.90666, 0.3254],
'c': [2.9317, 1.4040, 2.4878, 2.2280, 2.5409],
'i': [1.9232, 1.5483, 2.9698, 1.7508]}
```

Fig. 9: Randomly weighted map and weights with overall score of 74%.

In this result, while the score was very high compared to the initial map's score of 34%, the weights were all close to the maximum of 3. This map also had a higher number of recreational and commercial cells which helped increase the score. The weight to industrial distance for 'c' was also unexpectedly high, as it was overshadowed by the high scores for 'r' distance and 'c' density.

The lack of clustering of industrial areas also shows the scoring parameters and formulas need to be further explored to better represent the needs of all the interest groups in Hoboken. Commercial areas tended to create borders around areas with high 'i' concentration to balance out lower scores due to 'r' proximity.

DISCUSSION

Compared to the original map, the areas classified as 'unknown' were most consistently parks and residential around Stevens, businesses around public transportation, and a mixture of businesses, residential, industrial, and recreation towards the northeast corner. The lack of industrial clusters and tendency towards the weight boundaries of 0 and 3 for 'better' maps shows that the parameters still need improvement. Exploring more complex urban planning models would help performance, such as an LULC (land-use land-cover) classifier as proposed in

the Future Work section of S. Nagappan et. al. [3]. Incorporating more data such as Google Earth images or historical land use maps would also improve performance.

While an evolutionary algorithm alone was not complex enough to accurately model the data, it was a powerful tool for exploring how different parameters and weights affect the simple representation of the extremely complex problems in urban planning. Combining a version of the algorithm used in this project for parameter search and optimization with an image feature classification model shows promise in furthering this research.

COMPILING INSTRUCTIONS

To run the program, open the file 'urban_planning_final.ipynb' in Jupyter Notebook or Google Colab with the 'map_layers.xslx' workbook in the working directory and run the .ipynb file. Configure your environment so you can make the following imports:

```
import pandas as pd
from itertools import product
import openpyxl
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import matplotlib.patches as mpatches
import math
import random
```

REFERENCES

[1] B. Pickard, J. Gray, and R. Meentemeyer, “Comparing quantity, allocation and configuration accuracy of multiple land change models,” Land, vol. 6, no. 3, pp. 52, 2017.

[2] R. Lopez-Farias, S. I. Valdez and A. Garcia-Robledo, "Parameter Calibration of the Patch Growing Algorithm for Urban Land Change Simulations," in Proceedings of the 2021 Mexican International Conference on Computer Science (ENC), 2021, pp. 1-8, doi: 10.1109/ENC53357.2021.9534789.

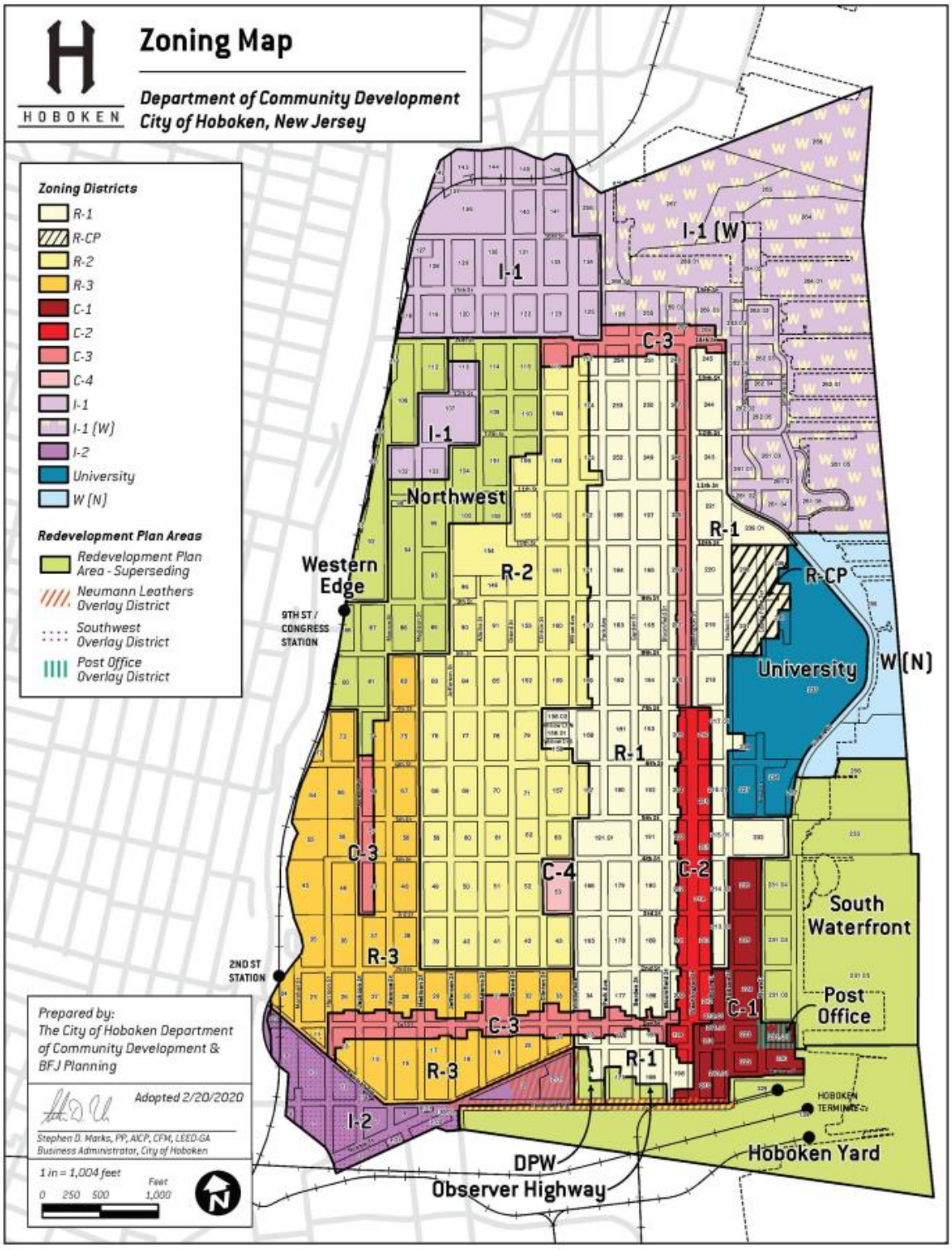
[3] S. Nagappan and S. Daud, “Machine Learning Predictors for Sustainable Urban Planning,” International Journal of Advanced Computer Science and Applications, vol. 12, no. 7, 2021, doi: 10.14569/IJACSA.2021.0120787.

[4] H. Xin and Z. Zhi-xia, "Application of Genetic Algorithm to Spatial Distribution in Urban Planning," in Proceedings of the 2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop, 2008, pp. 1026-1029, doi: 10.1109/KAMW.2008.4810667.

[5] Z. Wu and Z. Li, "Modular Information Fusion Model of Urban Landscape Design Based on Genetic Algorithm," in Proceedings of the 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), 2022, pp. 1275-1278, doi: 10.1109/IPEC54454.2022.9777337.

[6] Wikipedia contributors, "Hoboken, New Jersey," Wikipedia, The Free Encyclopedia, April 12, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Hoboken,_New_Jersey. [Accessed: April 12, 2023].

APPENDIX

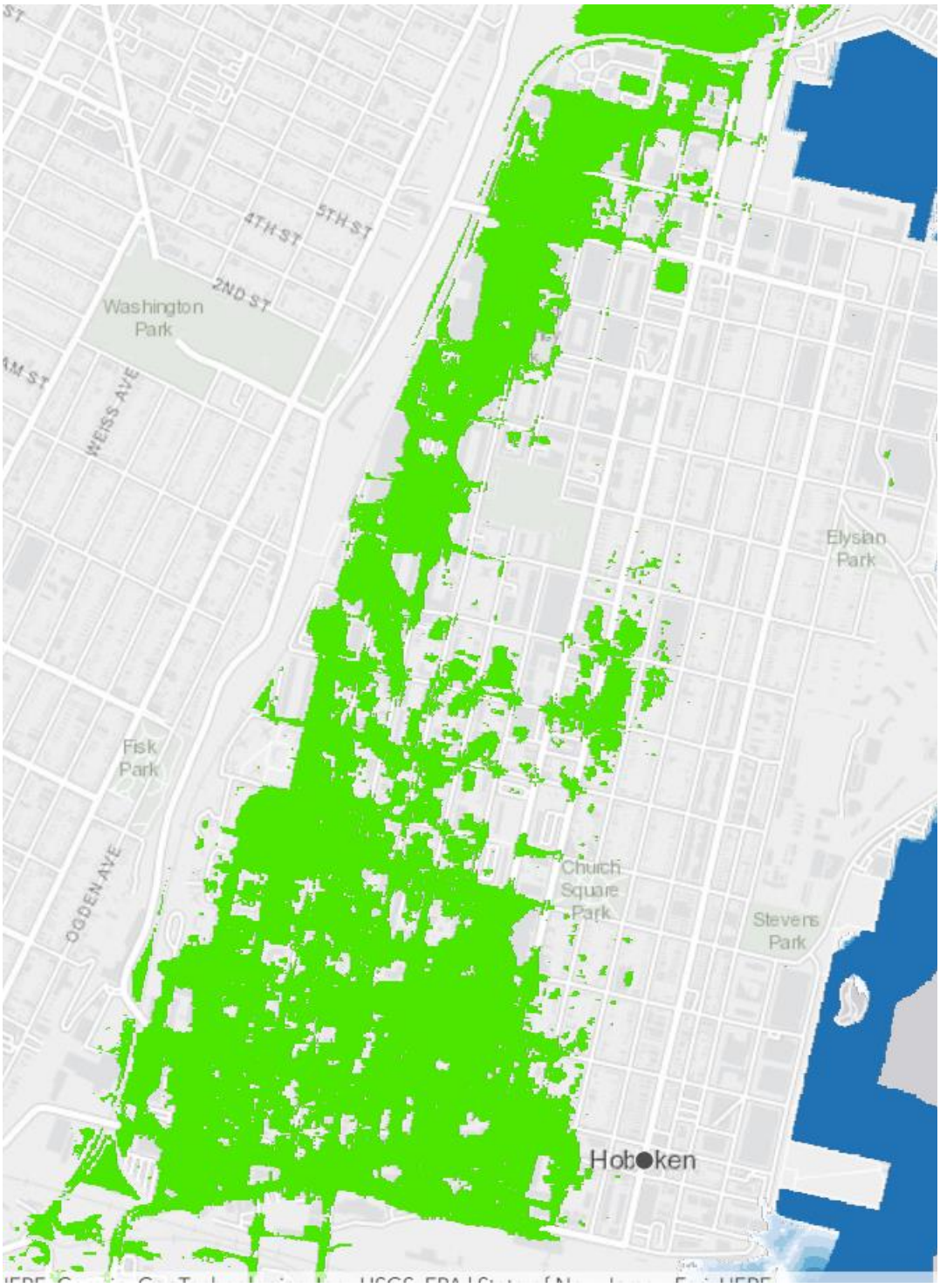


Appendix A: Zoning map; https://ecode360.com/15237147

Property type classification codes

Code	Type	Description
100	Agricultural	Property used for the production of crops or livestock.
200	Residential	Property used for human habitation. Living accommodations such as hotels, motels, and apartments are in the Commercial category - 400.
300	Vacant land	Property that is not in use, is in temporary use, or lacks permanent improvement.
400	Commercial	Property used for the sale of goods and/or services.
500	Recreation & entertainment	Property used by groups for recreation, amusement, or entertainment.
600	Community services	Property used for the well being of the community.
700	Industrial	Property used for the production and fabrication of durable and nondurable man-made goods.
800	Public services	Property used to provide services to the general public.
900	Wild, forested, conservation lands and public parks	Reforested lands, preserves, and private hunting and fishing clubs.

Appendix B: Property type classification codes;
https://www.tax.ny.gov/research/property/assess/manuals/prclas.htm

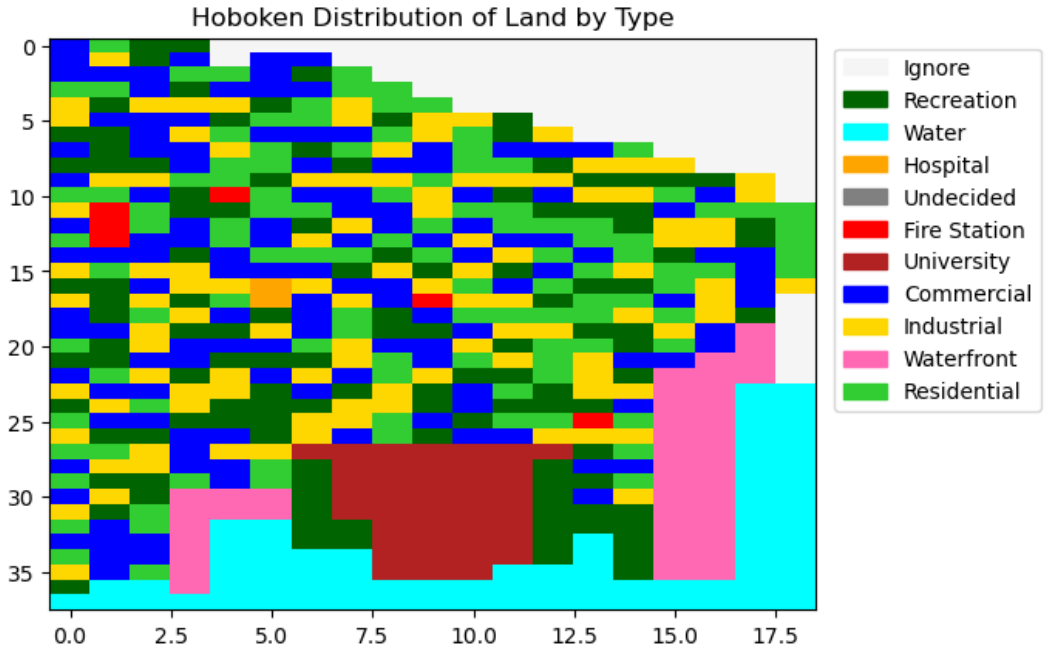


Appendix C: MHHW map; https://www.njfloodmapper.org/

Appendix D: Example of all maps and weights from randomizations.

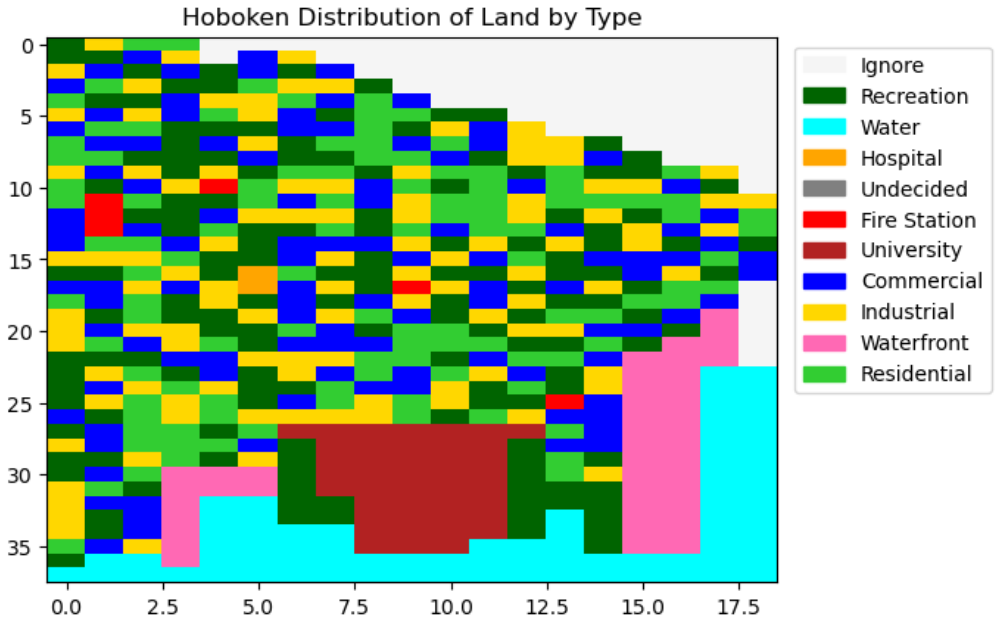
```
Weights: {'r': [1.0, 1.0, 1.0, 1.0, 1.0, 1.0], 'c': [1.0, 1.0, 1.0, 1.0,
1.0], 'i': [1.0, 1.0, 1.0, 1.0]}

Highest fitness score: 0.3561458786152669
```



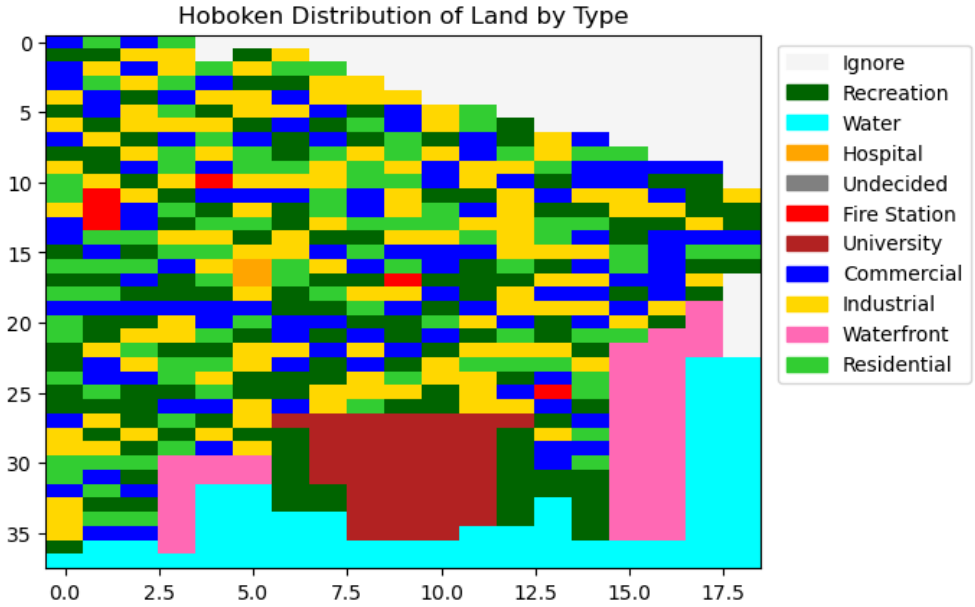

```
Weights: {'r': [1.969759206169063, 0.5869748483788861, 0.5577472993352424,
1.0062059161715986, 1.2056240699483198, 2.827875326643669], 'c':
[2.3201873372870465, 2.118908697436234, 0.427242076371376, 0.620324692707132,
0.5149420729009574], 'i': [1.0965297273433623, 0.40632031606092645,
2.426238075681436, 1.667096061197714]}

Highest fitness score: 0.41625816739383675
```



Weights: {'r': [2.9210237504480334, 0.015740753717331413, 1.5660762940019843, 0.8939824828528893, 0.18445018241469469, 1.6429652233565957], 'c': [2.855531914834777, 0.11301830917460698, 2.8074835910430394, 0.4480301684941588, 0.4137551417358495], 'i': [2.423522272974903, 2.5911069170924246, 0.7539958232226309, 1.3945732498219559]}

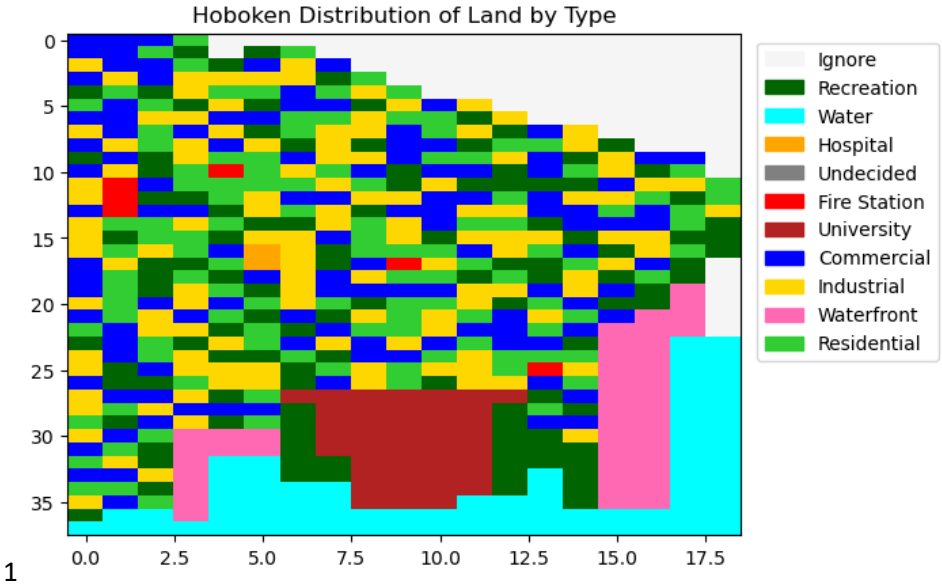
Highest fitness score: 0.5180063617697696



Weights: {'r': [2.778465885773809, 2.6241531329125873, 2.5993655233500164, 2.8986187179716545, 0.517681713083735, 0.06511533653836643], 'c': [0.6881703261500193, 2.829503761370818, 1.5120248417178712,

1.5818712330856108, 0.23946654046860005], 'i': [1.6704178976307777, 2.924428686148546, 0.599952953775573, 1.2348617506929935]}

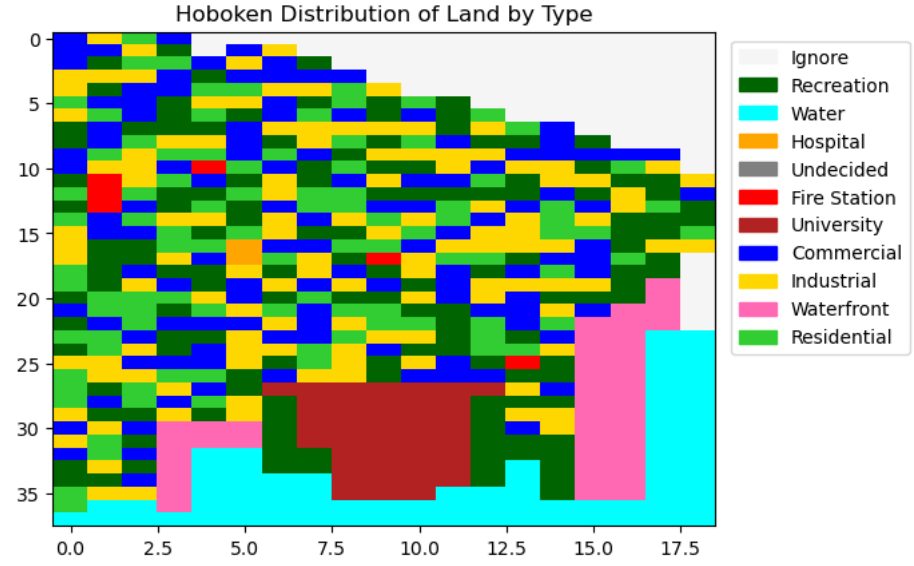
Highest fitness score: 0.5975412271121446



1

Weights: {'r': [2.1406449179557887, 2.2500815812381694, 0.05602061853850704, 1.033198044689339, 1.930271972283736, 1.9698843130817218], 'c': [1.2240301910035294, 2.885287746126938, 0.2001806366793144, 2.6923577527606493, 0.03367163998245559], 'i': [2.5217732200038223, 0.5869961082895198, 0.8115824763754564, 0.6071135470613441]}

Highest fitness score: 0.46030821412521994



Weights: {'r': [2.836191230338923, 1.1135370461359388, 2.625560790523421, 2.6653699709937264, 2.906652792447046, 0.32541657546974057], 'c': [2.931755043724575, 1.4040408323478095, 2.487846790522082, 2.228054223744046,

2.5409230446308957], 'i': [1.9232102464556502, 1.548305108068262,
2.9698692866382626, 1.7508572053482832]}

Highest fitness score: 0.7378270954222561