Application of Genetic Algorithms for Urban Planning

Final Project

CPE 672: Applied Game Theory and Evolutionary Algorithms

Cameron Calder
Aleksandar Dimoski
Nawin Sakthivelan

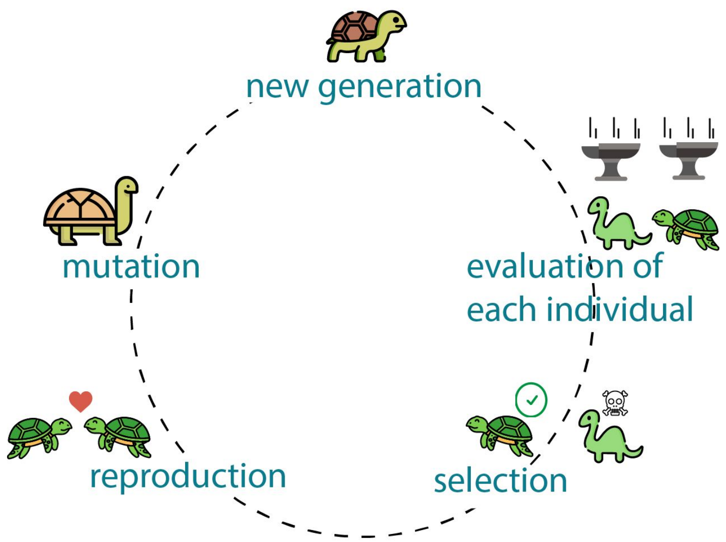
Introduction

In urban planning, the goal is to optimally utilize a given space given the needs of residents, businesses, and community spaces and possible arrangements given land constraints.

- Commercial needs
- Housing
- Environmental risk (floods)
- Infrastructure (fire, police, etc.)
- Environmental usability
- Traffic flow, road quality
- Public works (septic, water)

Why a Genetic Algorithm?

- No single aspect of urban planning can reflect the overall urban development needs
- Traditional optimization methods do not always produce a satisfactory solution, since the data is often non-spatial, dynamic, and/or nonlinear
- Genetic algorithms are an effective way of navigating through the large data space while maintaining flexibility when taking constraints into account

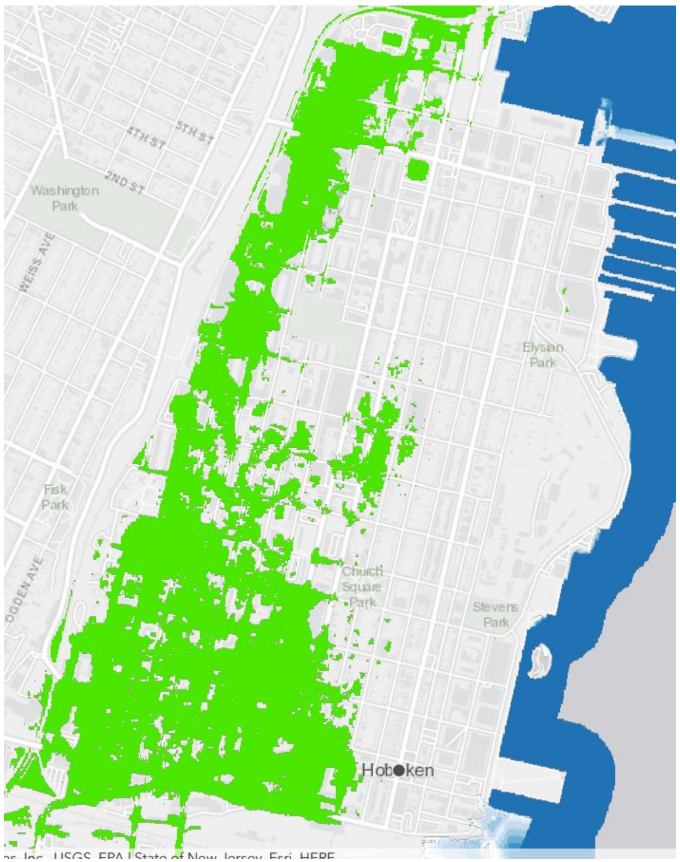


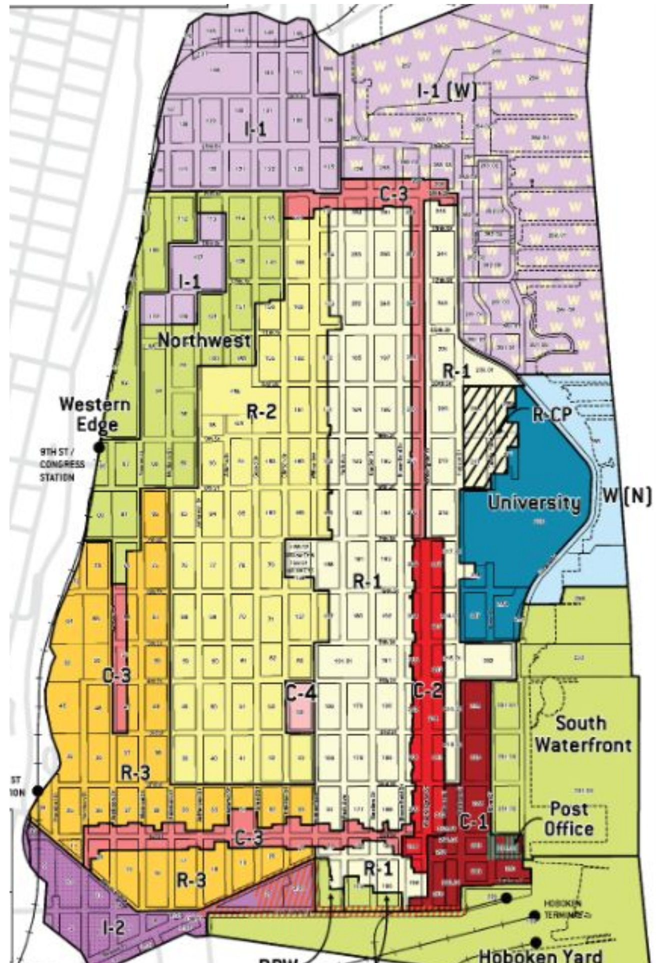
Why Hoboken?

- Hoboken is a great case for applying this type of algorithm because of its structure and diverse land usage, given its compactness (2 mi^2)
  - Grid system
  - Constrained by land that only has one use
  - Still flexible in what can occupy the majority of land space

Collecting Map Data

- Designate each land cell by its flood risk, its type, and chromosome inclusion (movable y/n)
- Areas where the 2-year flood average is 2.4 ft. above the MHHW (mean higher high water)





R: Residential
C: Commercial
I: Industrial

Quantifying Map Data

- Used current city wards/zoning and Google Maps to choose land cell tuple values

(1,4,1)	(1,4,1)	(1,5,1)	(1,5,1)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)
(1,4,1)	(1,9,1)	(1,9,1)	(1,5,1)	(-1,0,0)	(1,5,1)	(1,5,1)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)
(1,4,1)	(1,5,1)	(1,9,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)
(1,4,1)	(1,5,1)	(1,9,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,6,1)	(1,5,1)	(1,2,1)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)
(1,4,1)	(1,5,1)	(1,9,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,6,1)	(1,5,1)	(1,2,1)	(1,2,1)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)
(1,4,1)	(1,5,1)	(1,9,1)	(1,5,1)	(1,5,1)	(1,9,1)	(1,9,1)	(1,9,1)	(1,2,1)	(1,2,1)	(1,2,1)	(1,2,1)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)
(1,4,1)	(1,5,1)	(1,9,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,2,1)	(1,4,1)	(1,2,1)	(1,2,1)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)
(1,4,1)	(1,5,1)	(1,9,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,2,1)	(1,4,1)	(1,4,1)	(1,2,1)	(1,2,1)	(1,4,1)	(-1,0,0)	(-1,0,0)	(-1,0,0)	(-1,0,0)
(1,4,1)	(1,5,1)	(1,9,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,2,1)	(1,4,1)	(1,4,1)	(1,2,1)	(1,2,1)	(1,4,1)	(1,4,1)	(-1,0,0)	(-1,0,0)	(-1,0,0)
(1,4,1)	(1,5,1)	(1,9,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,2,1)	(1,2,1)	(1,4,1)	(1,4,1)	(1,2,1)	(1,4,1)	(1,4,1)	(1,4,1)	(1,4,1)	(-1,0,0)
(1,4,1)	(1,5,1)	(1,9,1)	(1,5,1)	(1,8,0)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,2,1)	(0,2,1)	(0,6,1)	(1,4,1)	(1,2,1)	(1,4,1)	(1,4,1)	(1,4,1)	(1,4,1)	(-1,0,0)
(1,2,1)	(1,8,0)	(1,9,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(0,5,1)	(0,6,1)	(0,4,1)	(0,2,1)	(0,4,1)	(1,4,1)	(1,4,1)	(1,4,1)	(1,4,1)
(1,2,1)	(1,8,0)	(1,9,1)	(1,6,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(0,5,1)	(0,6,1)	(0,2,1)	(0,2,1)	(0,4,1)	(1,4,1)	(1,4,1)	(1,4,1)	(1,4,1)
(1,2,1)	(1,8,0)	(1,9,1)	(1,6,1)	(1,5,1)	(0,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,6,1)	(0,2,1)	(0,2,1)	(0,4,1)	(1,4,1)	(0,4,1)	(0,4,1)	(0,4,1)
(1,2,1)	(1,4,1)	(1,9,1)	(1,5,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,6,1)	(0,2,1)	(0,2,1)	(0,4,1)	(1,4,1)	(0,4,1)	(0,4,1)	(0,4,1)
(1,2,1)	(1,4,1)	(1,9,1)	(1,5,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,6,1)	(0,2,1)	(0,2,1)	(0,4,1)	(1,4,1)	(0,4,1)	(0,4,1)	(0,4,1)
(1,2,1)	(1,4,1)	(1,9,1)	(1,5,1)	(1,5,1)	(0,7,0)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,6,1)	(0,5,1)	(0,9,1)	(1,4,1)	(0,4,1)	(0,4,1)	(0,4,1)	(0,4,1)
(1,2,1)	(1,4,1)	(1,9,1)	(1,5,1)	(1,5,1)	(0,7,0)	(0,5,1)	(0,5,1)	(0,5,1)	(0,8,0)	(0,5,1)	(1,5,1)	(0,5,1)	(0,9,1)	(1,4,1)	(0,4,1)	(0,4,1)	(0,4,1)	(-1,0,0)
(1,2,1)	(1,4,1)	(1,9,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,6,1)	(0,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,9,1)	(0,4,1)	(0,4,1)	(0,4,1)	(0,4,1)	(-1,0,0)
(1,2,1)	(1,4,1)	(1,9,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,6,1)	(0,5,1)	(1,5,1)	(1,5,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,9,1)	(0,2,1)	(0,6,1)	(0,6,1)	(-1,10,0)	(-1,0,0)
(1,2,1)	(1,4,1)	(1,9,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,6,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,9,1)	(0,2,1)	(0,6,1)	(0,6,1)	(-1,10,0)	(-1,0,0)
(1,2,1)	(1,4,1)	(1,9,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,6,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,9,1)	(0,2,1)	(0,6,1)	(-1,10,0)	(-1,10,0)	(-1,0,0)
(1,2,1)	(1,4,1)	(1,9,1)	(1,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,9,1)	(0,2,1)	(-1,10,0)	(-1,10,0)	(-1,10,0)	(-1,0,0)
(0,2,1)	(0,4,1)	(0,9,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,9,1)	(0,2,1)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,2,1)	(0,5,1)	(0,9,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,5,1)	(0,9,1)	(0,2,1)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,2,1)	(0,5,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,8,0)	(0,2,1)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,2,1)	(0,9,1)	(0,9,1)	(0,2,1)	(0,2,1)	(0,2,1)	(0,2,1)	(0,2,1)	(0,2,1)	(0,2,1)	(0,2,1)	(0,2,1)	(0,2,1)	(0,2,1)	(0,2,1)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,2,1)	(0,9,1)	(0,9,1)	(0,2,1)	(0,2,1)	(0,2,1)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,2,1)	(0,2,1)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,4,1)	(0,4,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,6,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,6,0)	(0,2,1)	(0,2,1)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,4,1)	(0,4,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,9,1)	(0,6,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,6,0)	(0,2,1)	(0,2,1)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,4,1)	(0,4,1)	(0,9,1)	(-1,10,0)	(-1,10,0)	(-1,10,0)	(0,6,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,6,0)	(0,2,1)	(0,2,1)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,4,1)	(0,4,1)	(0,4,1)	(-1,10,0)	(-1,10,0)	(-1,10,0)	(0,6,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,6,0)	(0,6,0)	(0,6,0)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,4,1)	(0,4,1)	(0,4,1)	(-1,10,0)	(-1,1,0)	(-1,1,0)	(0,6,0)	(0,6,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,6,0)	(0,6,0)	(0,6,0)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,4,1)	(0,4,1)	(0,4,1)	(-1,10,0)	(-1,1,0)	(-1,1,0)	(0,6,0)	(0,6,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,6,0)	(-1,1,0)	(0,6,0)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,4,1)	(0,4,1)	(0,4,1)	(-1,10,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,3,0)	(0,6,0)	(-1,1,0)	(0,6,0)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,4,1)	(0,4,1)	(0,4,1)	(-1,10,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(0,3,0)	(0,3,0)	(0,3,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(0,6,0)	(-1,10,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)
(0,4,1)	(-1,1,0)	(-1,1,0)	(-1,10,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)
(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)	(-1,1,0)

LEGEND	
(-1,0,0)	non-Hoboken area
(-1,1,0)	water
(n,2,p)	undecided
(0,3,0)	university
(n,4,p)	industrial
(n,5,p)	residential
(n,6,p)	recreation
(n,7,0)	hospital
(n,8,0)	fire station
(n,9,p)	commercial
(-1,10,0)	waterfront

p=0	permanent
p=1	movable
n=1	high flood risk
n=0	low flood risk
n=-1	not applicable

Quantifying Map Data (Flood Only)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
0	0	(1,2)	(1,2)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	(1,2)	(1,2)	(1,2)	0	(1,2)	(1,2)	0	0	0	0	0	0	0	0	0	0	0	0
0	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	0	0	0	0	0	0	0	0	0	0	0
(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	0	0	0	0	0	0	0	0	0	0
(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	0	0	0	0	0	0	0	0	0
(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	0	0	0	0	0	0	0
(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	0	0	0	0	0	0
(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	0	0	0	0
0	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	0	0	0
0	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	0
0	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	0
0	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,2)	(1,2)	(1,2)	(1,2)
0	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,2)	(1,2)	(1,2)	(1,2)
0	(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,2)	(0,2)	(0,2)	(0,2)
0	(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,2)	(0,2)	(0,2)	(0,2)
0	(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,2)	(0,2)	(0,2)	(0,2)
0	(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)
0	(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,2)	(0,2)	(0,2)	(1,2)	(0,2)	(0,2)	(0,2)	0
0	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	0
0	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	0
(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	0
(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	0
(1,2)	(1,2)	(1,2)	(1,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	0
(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	1	1	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	1	1	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	1	1	1	1	(0,2)	(0,2)	(0,2)	(0,2)	(0,2)	1	(0,2)	(0,2)	(0,2)	1	1
(0,2)	(0,2)	(0,2)	(0,2)	1	1	1	1	(0,2)	(0,2)	(0,2)	1	1	1	(0,2)	(0,2)	(0,2)	1	1
(0,2)	1	1	(0,2)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Description of Variables

Social Factors

X_{11}: Distance to commerce

X_{12}: Distance to recreation

X_{13}: Distance to transportation

X_{14}: Distance to fire station

X_{15}: Distance to hospital

Environmental Factors

X_{31}: Density of greenspace/recreation areas

X_{32}: Flood mitigation potential

X_{33}: Waterfront usage (commercial bonus)

X_{34}: Distance to industrial area

X_{35}: Distance between residential and industrial

Commercial Factors

X_{21}: Density of commercial businesses

X_{22}: Distance to transportation

X_{23}: Level of pedestrian traffic

Constructing the Chromosomes

- We choose K genes, each representing a parcel land with a designated land code
- The number of coordinates per land code determines the code length/size
- Size of the population, m, determined by combining the codes of all movable cells
  - Cells valid for crossover

Key	Type	Size	Value
0	list	86	[(0, 4), (0, 5), (0, 6), (0, 7), (0, 8), (0, 9), (0,
1	list	79	[(23, 17), (23, 18), (24, 17), (24, 18), (25, 17), (2
2	list	79	[(3, 8), (4, 8), (4, 9), (5, 8), (5, 9), (5, 10), (5,
3	list	42	[(27, 6), (27, 7), (27, 8), (27, 9), (27, 10), (27, 1
4	list	102	[(0, 0), (0, 1), (1, 0), (2, 0), (3, 0), (4, 0), (5,
5	list	179	[(0, 2), (0, 3), (1, 3), (1, 5), (1, 6), (2, 1), (2,
6	list	42	[(3, 6), (4, 6), (10, 11), (11, 11), (12, 3), (12, 11
7	list	2	[(16, 5), (17, 5)]
8	list	6	[(10, 4), (11, 1), (12, 1), (13, 1), (17, 9), (25, 13
9	list	61	[(1, 1), (1, 2), (2, 2), (3, 2), (4, 2), (5, 2), (5,
10	list	44	[(19, 17), (20, 17), (21, 16), (21, 17), (22, 15), (2

Keys

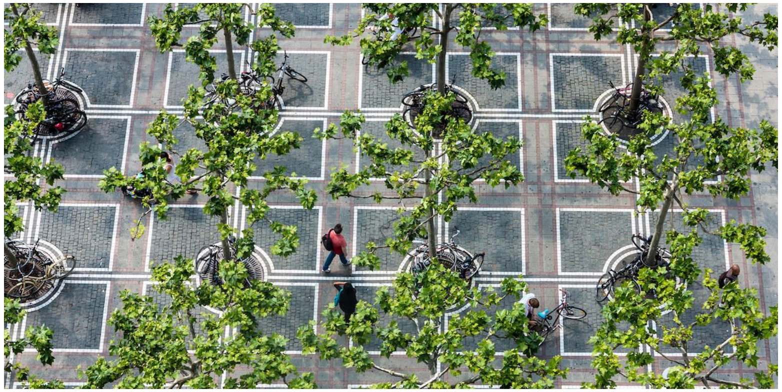
0: Ignore (Non-Hoboken)

1: Water
2: Undecided
3: University
4: Industrial
5: Residential

6: Recreation
7: Hospital
8: Fire Station
9: Commercial
10: Waterfront

Determining Fitness

- To determine fitness, distance to different areas was the main evaluation criteria
  - Distance to commercial, industrial, health and fire services, etc
- Used walking times as a threshold based on the “block” distance to calculate overall score
  - If you are 15 blocks away, 10 - 5 = 5 block error, 70% fitness
- Density of commercial areas and recreational areas was also used to factor in the distribution of community-focused areas
- Environmental fitness focused on the availability of greenspace to residents and the optimality of its location
  - Rainwater mitigation and distance to industrial areas



Code Sample: Fitness function

```
############################ distance to public transportation ####################################
dist3 = []
constant_exp_mny = 1.1 # increased exposure to metropolitan ny
for i in range(0, self.chromosome.shape[0]):
  for j in range(0, self.chromosome.shape[1]):
    land_zone_type = self.chromosome[i,j,0]
    x2 = i
    y2 = j
    x1 = 28
    y1 = 8 # Hoboken terminal is the south east corner of the city
    if (land_zone_type == 1 or land_zone_type == 2 or land_zone_type == 3):
      dist3.append(residitial_zones_have_people_per_floor * math.hypot(x2 - x1, y2 - y1))
    if (land_zone_type == 4 or land_zone_type == 5 or land_zone_type == 6):
      dist3.append(commercial_zones_have_people_per_floor * math.hypot(x2 - x1, y2 - y1))
dist_public_transtp = constant_exp_mny * sum(dist3)/len(dist3)

##################################### proximity to greenspace ###################################
dist4 = []
for i in range(0, self.chromosome.shape[0]):
  for j in range(0, self.chromosome.shape[1]):
    land_zone_type = self.chromosome[i,j,0]
    if (land_zone_type == 0):
      for i2 in range(0, self.chromosome.shape[0]):
        for j2 in range(0, self.chromosome.shape[1]):
          land_zone_type = self.chromosome[i2,j2,0]
          x2 = i2
          y2 = j2
          x1 = i
          y1 = j
          if (land_zone_type == 1 or land_zone_type == 2 or land_zone_type == 3):
            dist4.append(residitial_zones_have_people_per_floor * math.hypot(x2 - x1, y2 - y1))
          if (land_zone_type == 4 or land_zone_type == 5 or land_zone_type == 6):
            dist4.append(commercial_zones_have_people_per_floor * math.hypot(x2 - x1, y2 - y1))
          #print(dist4)
proximity_to_green_space = 1.1 * sum(dist4)/(len(dist4)+1)
```

Example of Fitness Calculation:

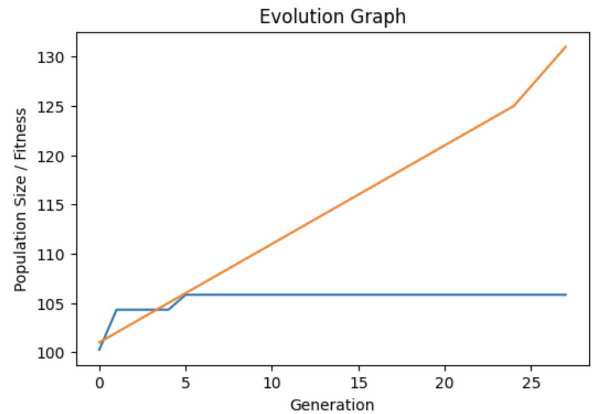
- Proximity to greenspace
- Distance to public transport
- Distance to riverfront

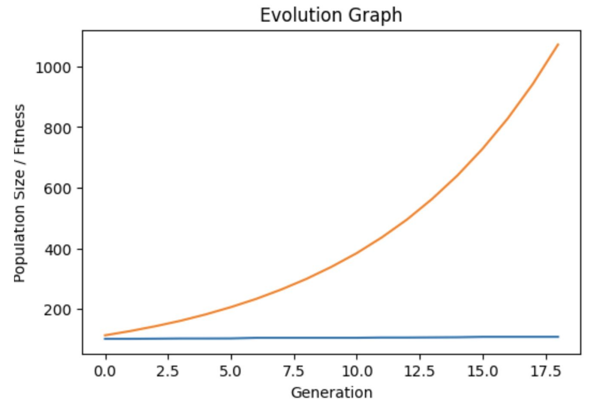
Code Sample: Main Evolution Loop

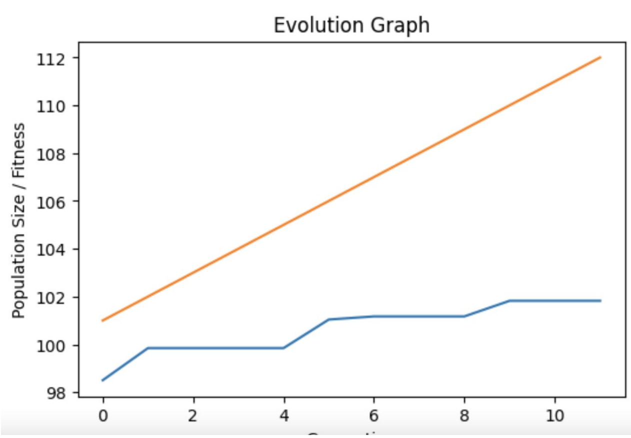
```
for i in range(0,gen):

  #random mutations
  for individual in self.population[5:]:
    if random.random() < PROP_MUTATION_OCCURING:
      individual.mutate_genes()
  for individual in self.population[5:]:
    if random.random() < PROP_MUTATION_OCCURING:
      individual.mutate_chromosomes()

  #sort, select parents, after mutation
  for individual in self.population:
      individual.fitness_eval()

  top_20 = self.crossover_parent_selection(self.population)

  #crossover, create children increase population size by 10 per
  for i in range(1,len(top_20)):
    child = self.crossover(top_20[i-1], top_20[i])
    child.fitness_eval()
    self.population.append(child)

  self.population.sort(key=lambda x: x.fitness_score, reverse=True)

  #evaluate fitness, sort, reduce by 5 per

  self.population = self.population[:int(len(self.population) - ((PERCENT_NEW_CHILDREN/2)  * len(self.population)))]

  print(self.population[0].fitness_score)
  print(len(self.population))
```

Optimizing GA

- Initial population 100, and then 1000
- Number of generations –5,10, 30, 1000
- Usually converges within 5-10 generations
- Tuning population increase ratio–How the much the population increases in every generation.
  - Trying 1 to 20 percent
- Tuning Mutation ratio – Probability of mutation happening in a selected individual within the population
  - 5 to 30 percent



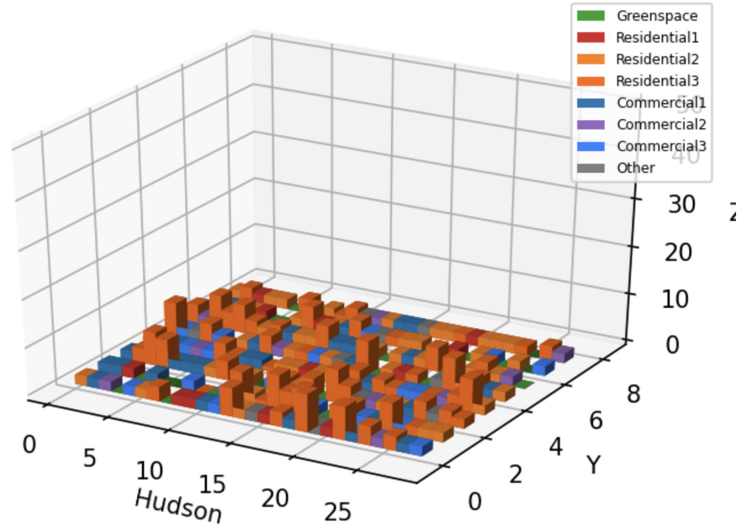


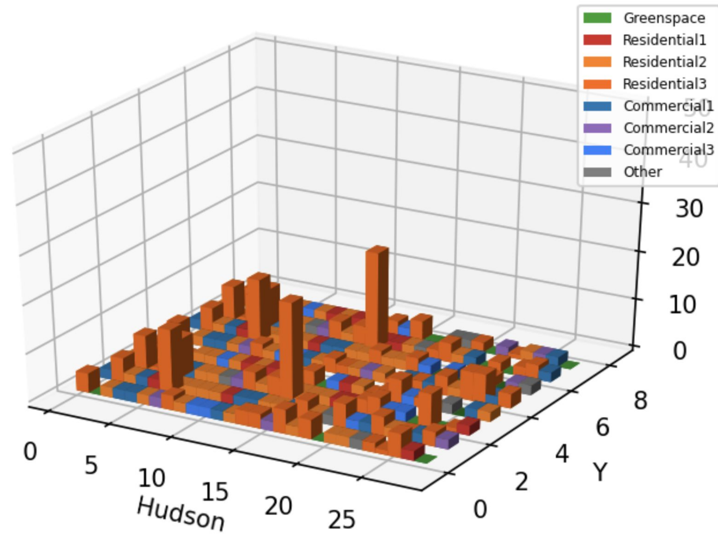



```
[101.42072605854456,
 101.42072605854456,
 101.82977615752704,
 102.51247550279496,
 102.51247550279496,
 102.74049122890176,
 104.807030838446 7,
 104.807030838446 7,
 104.807030838446 7,
 104.807030838446 7,
 104.807030838446 7,
 105.66904610397947,
 105.66904610397947,
 106.14150708487958,
 106.61045701946207,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775,
 107.97631777699775]
```

Results and Analysis

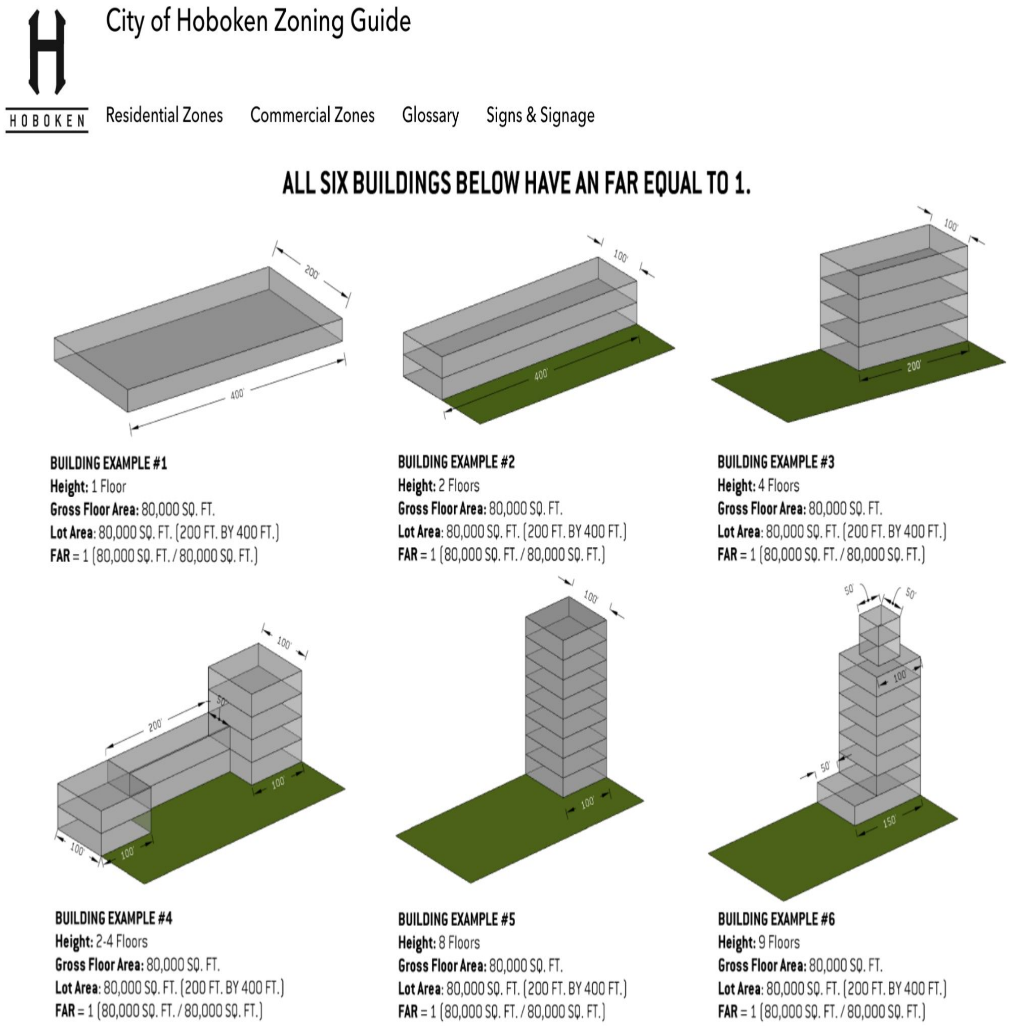
- We set the initial Grid to be a 28 x 8
- The best results were obtained for population size of 100 with relatively higher mutation ratio of 0.3
- Best results were also generated within 20 gens.
- Genetic algorithm seemed to optimize city map to an extent
- Average fitness did not increase by more than 10% from the average fitness of random starting point no matter where you started.
- It kept aggregating to symmetric pattern where all the tall building where on one side
- Maps with tall building seemed to have higher fitness than





Further improvements to the Algorithm

- Higher reward for clustering
- Add more properties to the genes — Especially building dimensions
- Block usage ratio
- Improve Initial Map
- Implement Logarithmic Population curve/Flattening
- Include more types of crossover and mutation
- Integrate various ratios within the fitness function itself, so the algorithm can optimize with respect to the ratios as well.
- Weighted average of fitness score
- Extending fitness function

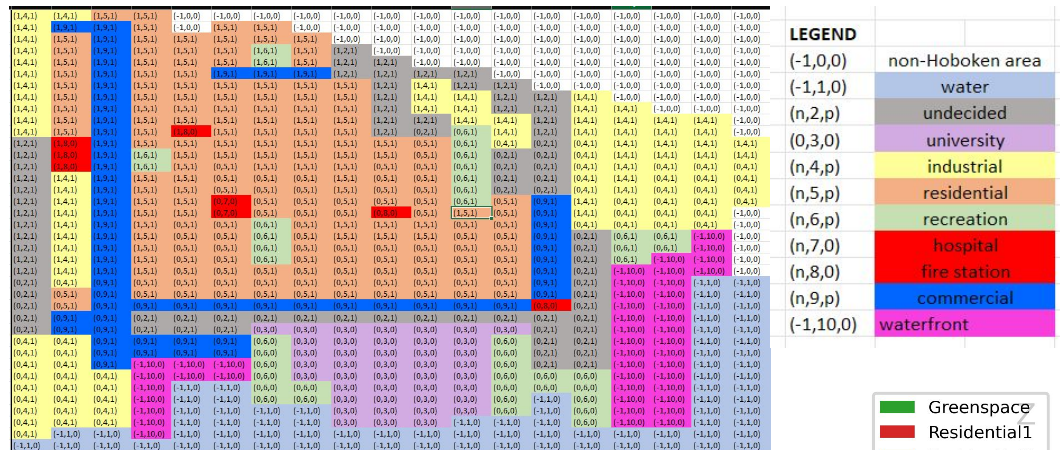


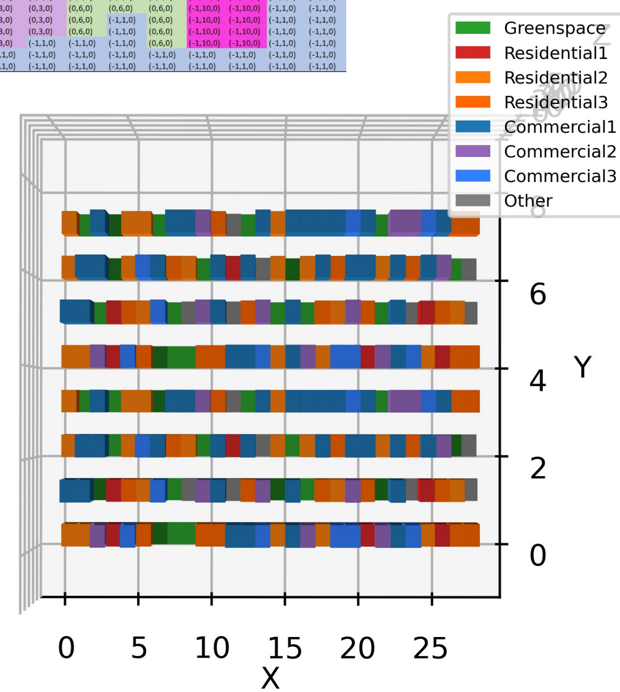
Land Type Distribution

- Hoboken has similar districts clustered together
- Results of our algorithm are distributed fairly evenly across the test space

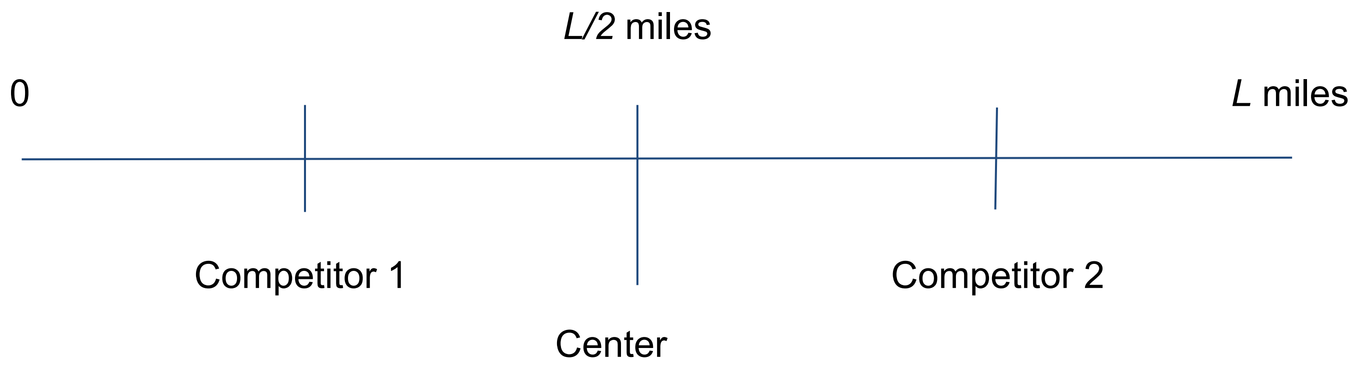
Why?

- Loss functions primarily take into account *distance for people to get to a given service*
- Having services most evenly distributed means that people are closest to most services
- Indicates an even spread of all kinds of services is optimal



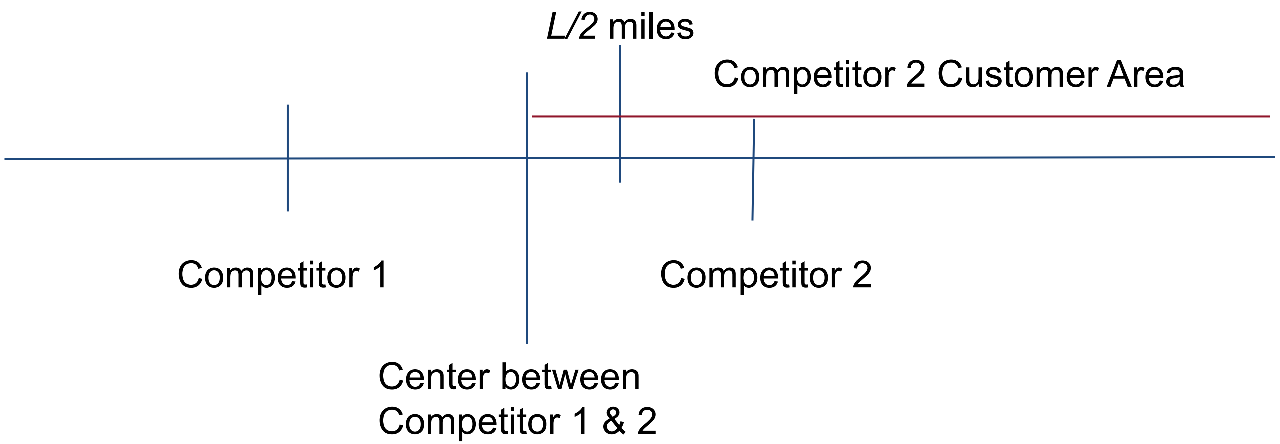


Clustering of Businesses - Example



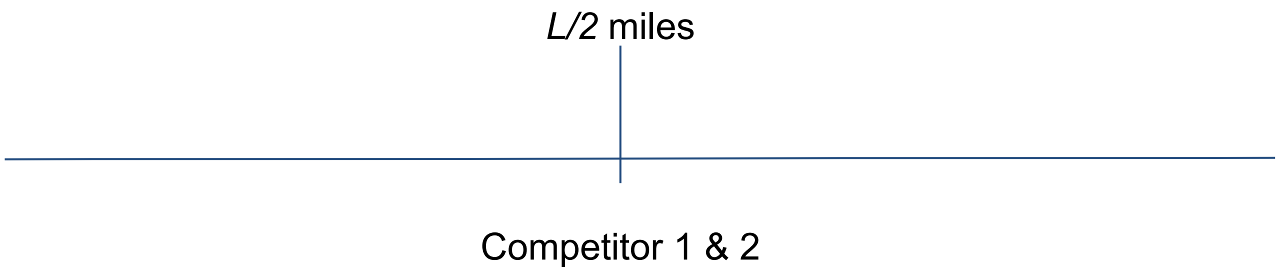
- In this case, as indicated by our model, consumers (assumed to be uniformly distributed along the axis) have the highest (most equitable) utility - the distance they have to travel to get to either competitor is minimized for all consumers - no consumer needs to travel more than *L/4* miles
- Both competitors then have an equal portion of consumers that will travel to them (assuming everything else is held constant, ie product quality etc)

Clustering of Businesses - Example



- Either competitor can increase their own utility by travelling farther to the center
- In this case, the distance that customers on the right end need to travel is increased, while those closer to the center don't need to travel as far
- This action also increases the utility of Competitor 2 and decreases it for Competitor 1.

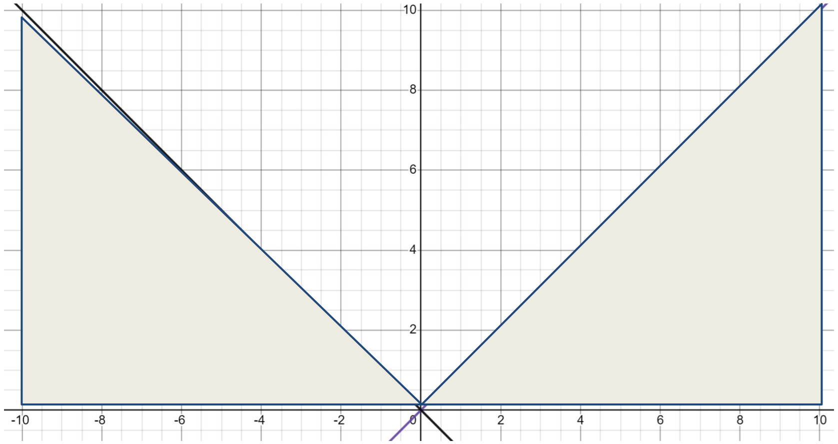
Clustering of Businesses - Example



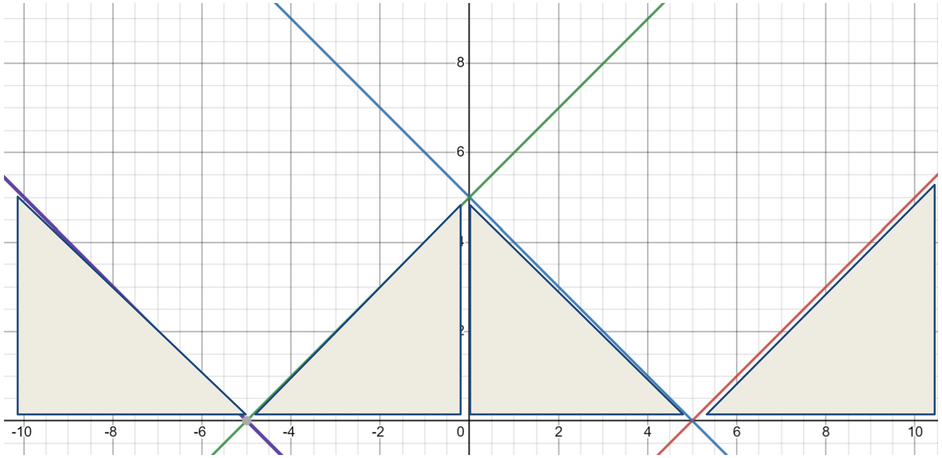
Eventually, both competitors will converge to the center point, as it minimizes both of their utilities.

Clustering of Businesses - Utility for Customers along Axis

‘Detriment’ / Reverse of Utility: Area under distance line
Detriment for Consumers increases as competitors get closer together!



Competitors in center
(A=100)



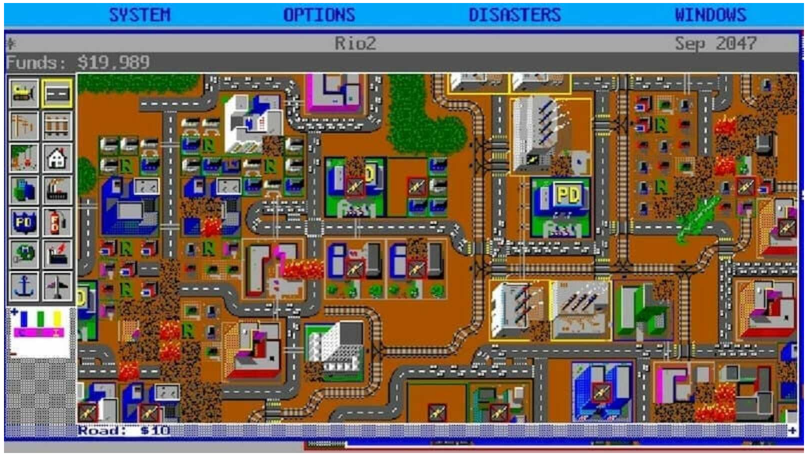
Competitors evenly distributed
(A=50)

Conclusion

- Urban planning is a complex problem with many interdependent factors
- Using a genetic algorithm makes it easy to create a model that can quickly iterate through hundreds of possible maps
- Further tuning of both the initial map and fitness function criteria can greatly improve the model
- Clear difference in model results vs. current Hoboken allocation as a result of Game Theoretic principles

Future Work

- Create a more accurate initial map; image processing - improves scalability
- Try different methods to evaluate the fitness
  - Add different weights/final scoring methods, spheres of influence, clustering
  - Factor in more historical data, or land use over time [3]
- Develop a model that takes into account individual incentives/utility as opposed to the total utility of the arrangement; more realistic, less optimal distributions
- Weight utility (or detriment) differently for different runs, and analyze results of these simulations to obtain better understanding of effects on city arrangement
- Develop alternative model with a variety of competitive populations, which either compete or cooperate with members of their own or other groups; can simulate businesses and more interactions better



1870

STEVENS

INSTITUTE *of* TECHNOLOGY

THE INNOVATION UNIVERSITY®

stevens.edu

Sources

Papers

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

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

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

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

Map Websites

https://assets-global.website-files.com/58407e2ebca0e34c30a2d39c/618079f6cd979db2e3d607d1_HobokenWardMap.pdf

https://www.njfloodmapper.org/