Master Thesis
Computer Science
Thesis no: MCS-2016-NN
01 2016

Time based procedural population generation with relationships graphs

Rickard Fridvall

Dept. Computer Science & Engineering
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Contact Information:
Author(s):
Rickard Fridvall
E-mail: rifa10@student.bth.se

University advisor:
Prof. Prashant Goswami
Dept. Department of Creative Technologies, Blekinge Institute of Technology

Abstract

Context. Games are very limited when it comes to creating large virtual populations. Most simply use very simple templates that are instantiated, with possible minor customisation. This quickly becomes repetetive. One way to solve this is to make more extensive use of procedural generation.

Objectives. This thesis will attempt to create a method by which its possible to quickly generate large city populations. That follows patterns exhibited by real cities complete with relationships graphs.

Methods. Develop techniques which makes it possible to expediently generate large populations from a small set. Compairing the result to data gathered from real cities.

Results. The simulation is able to quickly generate the required data, and it corresponds to real cities in most aspects.

Conclusions. Time for generation makes it a suitable solution for high performance applications. This in addition to the flexibility in of the simulation to generate suitable patterns makes it a good kandidate for generation of populations in RPGs.

Keywords: procedural generation, relationships graph, statistical agents.

List of Figures

Contents

Chapter 1

Glossary

Some commonly used words through this work.

Word	Definition
NPC	Non-Player Character, is any person in a game not directly controlled by a player.
Individual	Through this paper it's used interchangeably with NPC
RPG	Role Playing Games, games that focus on the interaction between player and NPCs as well as other players in some cases.
O(n)	Refers to the time complexity of the program. Expressed here in big o notation, where n is the growth rate. If written as here with only n, the growth is linear.
Group	When used refers to any amount of individuals that identify as belonging to the same group, examples includes professions and different cultures.

Chapter 2

Introduction

2.1 Problem

How do you create large virtual cities? This is a major issue in game development with only limited solutions. Some use procedural generation in some limited form [1]. While others just make the cities smaller. But even then to be able to make everything, a large amount of artist time is necessary. Even the biggest projects often have cities with only a few houses and a couple of inhabitants.

The most used form of procedural generation is to create people. Usually this means that people spawn and despawn as the player moves. This creates the impression of a large population. But this solution provides quite a weak illusion. As interaction beyond very simple ones (most often shooting them) will be difficult. Many larger games uses something like this, with examples such as GTA V and Watch Dogs.

When it comes to RPGs this approach proves problematic. Since they rely to a much larger degree on interactions with NPCs. Here the limits created by generating them one by one becomes clear. These generated people are just a collection of a few variables (age, gender, name etc.) with no connection between them. Reducing all possible interactions to a much smaller subset of what they could be.

2.2 Purpose

To create a base with more interconnected underlying NPC data. Where each NPC influences and is part of the larger population.

By creating this base data that other systems can use. You create a more cohesive world where each part fits together. For example a city where each house appearance depends on the people living in it.

So that instead of some disconnected and isolated NPCs. You instead get NPCs that are a part of their environment. With relations to family and friends.

Chapter 3

Background

3.1 Basics

This section describes some underlying concepts relevant to this thesis. Only relevant parts are described so large part of them are skimmed over.

3.1.1 Time Complexity

Refers to how the time a function takes to execute changes as the number of elements in its input increases. This is commonly expressed using big O notation which ignores constants and coefficients. Shown as O(f(n)) where O signifies the big O notation, n is the number of elements in the input and f is the function corresponding to its time increase. Some common ones are:

O(1) or constant time. This means that the time to completion is not influenced by its input size.

O(log(n)) or logarithmic time. A classic example is binary search, where the search space is halved after each iteration.

O(n) or linear time. Looping over all input elements a fixed number of times is O(n) (presuming constant time work is done on each element).

O(n log(n)) or quasilinear time. Many sorting algorithms has a worst case time complexity of O(n log(n)). Cause the loop over each element, O(n), and perform O(log n) work (binary search for example) in the worst case.

O(n*2) or sub-quadratic time. Looping over all elements and performing a linear time search for each is O(n*2). Generally very bad since the amount of time increases extremely rapidly compared to the other time complexities mentioned.

3.1.2 Sorting Algorithms

These algorithms as their name implies sorts a sequence of elements according to some comparision. Most often the comparison used results in: larger than, equal to or smaller than. This comparison is then used in the algorithm to place each element. Such that the sequence is either rising (each subsequent element is larger than the one before) or falling (same but smaller instead) in size.

If this is done in such a way so that all equal elements are below and above the same elements as before. It's said to be stable.

If the comparison used is the one described here then the best worst case time complexity is guaranteed to be at least O(n log(n)) [2].

Pigeonhole Sort

One of the sorting algorithms that doesn't use this comparison is Pigeonhole sort. Where an array is created for each possible value and then each element is placed in its corresponding bucket. Then iterating over the arrays the non empty ones are placed backed into the list in order. This algorithm is stable and have a worst case time complexity of O(n+N) where N is the number of possible values. Since it doesn't compare values directly it's possible to achieve in practice a worst case time complexity of O(n) if N is small and constant.

3.1.3 Artificial Intelligence

Whenever you are doing anything involving complex systems. You often find yourself stumbling into the field of artificial intelligence. For one succinct description of what it is: "the study and design of intelligent agents" [3]. Which is wide to say the least. Since this work simulates agents (in some manner) it quite easily fits this description.

This work does not have explicit agents in a form that is most often used. They reproduce and die of as well as taking no explicit actions. But the data interacts in a manner that resembles individual actions. These are probabilistically in nature and so forms statistical agents. Where agents actions exists as statistical patterns.

3.1.4 Emergent Behavior

When a large amount of agents interact using predefined actions larger patterns can emerge. This is called emergent behavior. Because of the need for a large amount of agents interacting with each other this is often also quite chaotic and can be difficult to control or create.

"In philosophy, systems theory, science, and art, emergence is a process whereby larger entities, patterns, and regularities arise through interactions among smaller or simpler entities that themselves do not exhibit such properties." / M. A. Bedau [4]

3.2 Previous Work

Procedural generation is a large field, and there is quite a lot of work on how to generate terrain and cities [5] [6]. Some of these also generates cities based on patterns found in the real world [7]. Other often used techniques are L-systems, or other grammar-based systems. There exists some very impressive papers describing different forms of formal grammars to generate buildings [8] and cities [9].

City generation is a large focus for many of these papers. Some focus more on realistic roads [10]. While others focus on larger sizes (including larger features such as parks and bridges) [9]. But work related to procedural generation of relations seems to be missing. This creates a gap that is critical. If we want to be able to generate an interconnected city population that exists as part of the city.

In addition to this the work touches on agent based AI system. Since each person simulated can be viewed as an agent taking actions. But since this work is very statistical based (mostly for speed) not much in form of techniques or methods have been utilized in this work. But there exists papers that focus much more on this approach. [11]

Some work has been done on landscape agents. Where agents interact to spread over some land based on interactions and rules [12]. But it has the same problem that the techniques utilized are to different to be able to do a proper comparison.

Chapter 4

Aim and Objectives

- To create a method by which it's possible to simulate a large interconnected city population. Where there exists large relationship networks between individuals. By also simulating the population over time, you capture the history of the population and individuals.
- To create groups of individuals with relations to other groups and individuals. With this an individual is part of larger groups. That themselves have relationships to other groups creating larger behavioral patterns.
- To utilize algorithms and data types that allow for fast simulation times. Making it possible for it to be incorporated into real time systems.
- To attribute physical locations to each person. So that their choice of where to live based on factors such as: population size, cultural and professional make up of the people living their and the value of the land. Will influence who they know and form connections too. Forming large scale emergent patterns through self reinforcing patterns (more people from one culture leads to more people from that culture wanting to live there).

All this is then used to create a city that has grown over time to become what it is. By shaping the city based on the changes in population, you create growth patterns. By letting groups have different cultural backgrounds, you can create areas that have distinct architectural styles. Similar to those found in real cities (for example *Chinatowns*). By modeling over time, changes and the age of a part of a city can be shown. Creating a visual history of the city.

Together you get a city and population that matches each others history and development. This creates individuals that do not only have their own fathers, sisters, friends and lovers but also a cultural heritage. Where they belonging to groups that have a long history, and living in a city with its own history shown through its current state.

4.1 Research questions

RQ1: Can the simulation create patterns that exists in the population of real cities?

RQ2: Will the resulting relationship graph have the same properties as real relationships graphs (in terms of path lengths, clustering and degree distributions [13])?

RQ3: Is it possible to do each iteration in O(n) time and O(n) space (where n is the population size)?

Chapter 5
Methods

In general how different objectives have been quantified and how to reach them.

5.1 Complexities

On goal for this work is that each iteration should have a worst case linear time complexity (O(n)). Since some of the work will require a sorted list this is quite difficult. Since sorting algorithms are usually worst case O(n log(n)).

To show this the source code is manually analyzed. With explainations of possible deviation from O(n) and how they are handled.

5.2 Pattern of social clustering

Based on the assumption that people identifying as belonging to one group are more likely to form relationships with and want to be around people from the same group. This will be referred to generally as culture but is broad enough in implementation that it can refer to almost anything.

Because of this a clustering effect should be seen, where people from the same group are more likely to live closer together. One real world example of this is chinatowns. Where the majority of the occupations is of one ethnicity.

5.3 Pattern of socioeconomic clustering

In addition to this a similar behavior should be expected when it comes to income. Here the professions used in the simulation are generalized to an average income for each profession. A clustering effect should be expected here as well but more centered on higher value areas. Such as town centers or with favorable view since higher income is required to live in more desirable areas (in general).

This can be validated against the limited real world data gathered.

5.4 Properties of relationship graph

The amount of properties related to graphs are quite large. But in general three ones are of the most interest: average path lengths between people, the amount of clustering and / or connectivity and the amount of connections both averages and probability distributions. [13] The properties that the relationships graph exhibits can then be compared against networks of different types. [14]

5.4.1 Path lengths

Social networks conform to the small-world property. [13] Which simply state that the distance between two individuals, counted as the number of friends friends etc. before any person is reached, is small. The average path length between all individuals can be measured and then compared to see how well it conforms to this and to what extent it can be varied.

5.4.2 Clustering coefficient

Basicly transitivity or if A has a relation to B and B has a relation to C, to what degree does that imply a relation between A and C. Shows how connected the graph is or how much it tends to form subgroups. Can be measured and compared to networks of different types as well as to what extent it can be varied.

5.4.3 Degree distributions

How many of the shortest path between different people pass through each individual, this indicates the amount of betweenness of different people. The distribution of the number of times this happens for the population will for social graphs follow a power law. Where a very small amount of individuals have an inordinate amount of the total betweenness. Comparison is made by comparing the scaling exponent to other networks.

Chapter 6

Implementation

6.1 Groups

This simulation implements two groups. Where in culture is a general implementation and profession is a specific one.

They both share a number of different aspects. Both influences the position of each individual. As well as their relations between each other. Both increases the chance of any children to belong to the same group as their parents. In addition to this both implement a dynamic system of relations between groups. Where a sample of people from each group determines the relations. Done by calculating the number of relations from this sample to different groups. This then forms the basis for the group relations values used in the next iteration. In turn influencing future relations, creating a feedback loop. However because of the random nature of the sample, this feedback loop is not guarantied to continue. So even strong group relations can change over time.

6.1.1 Culture

Meant to be a generic collection of groups that are non intersecting with no null space (no one can be in two at the same time, and its assumed that everyone is in one). The population as a whole have an average position, referred to as center. In addition to this all culture groups also have their own center. This center then exhibits a pull when choosing position.

6.1.2 Profession

For profession relations static values that don't change are also used. This is to be able to take into consideration socioeconomic factors. Depending on the average income specified for each profession.

6.2 Population spread

In the simulation a person that have an invalid position and is above a certain age will choose a position. By taking a number of possible locations randomly within a subset of the map. This subset is centered on the individuals parents position in the form of a square of a certain size. The best one from the resulting list becomes the chosen position. The evaluation of each position to determine how good it is depends on a number of factors:

terrain - A map with different values to indicate favorable or unfavorable terrain. Also used to indicate impossible terrain where no one can live.

concentration of population - How many people that already lives in a specific position. The larger the number the lower the value. Determined by a logarithmic function so that the influence is small for low values. But quickly grows larger as the number of people increases.

distance from center - A value that depends on the distance to the average position of the whole population. The closer the higher value. Maintains cohesion and simulates the effect of a city core.

professions of population - The value depends of relations between different profession as well as the profession of the individual. Used to simulate group cohesion, and group relations.

culture of population - The value depends of relations between different cultures as well as the culture of the individual. Used to simulate group cohesion, and group relations.

distance from cultural center - Specific to culture and behaves like distance from center. Increases group cohesion, used to form specific patterns.

6.3 Relations

When creating relations a number of people are taken from different "pools". These "pools" are created when sorting people base on different attributes. The number of people to take from each pool shapes the relations. By a set number of relations at random from these pools. The one non attribute based pool is previous friends friends. That pick at random from previously known peoples friends. This directly influences the clustering coefficient and can be used to increase and decrease it as well as provides some stability at larger sizes. Where random chans will lead to a very low clustering coefficient without it.

Then for choosing lover its simply a matter of picking a random person from the friends list. Because of the nature of how its created it will conform to all the same probabilities as when picking friends.

The size of the number of relations for each person varies between 100 to 250. These numbers are based on Dunbar's number. That comes from a paper published in 1992 that extrapolates data from primates group sizes to humans. [15]

6.4 Data retrieval and processing

For this work data gathered from the website: "hitta.se" is used. Hitta.se contains a web feature where in a map of Sweden exists. There you can move around and zoom to different levels. The maps used in this work were created by taking a screenshot of the city Trelleborg on the map 6.1. Before processing it in different ways.

6.4.1 Processing

Each processing of the map simplifies the map in some aspect. This to be able to extract data more easily (both for visual effect and for the simulation). The code for each process exists in the file "formatsMap.hs". Run through an interpretator for Haskell (named ghci) instead of compilation. List of all sub functions and their function:

clearMap - Clears the map of almost all data. Changes all colours that are not the colour used for the ocean to the default colour for land.

compareMap - Identifies the dominant colour for ether industrial area, water or living area (if none found defaults to default land colour). In a square matrix where each square is one fiftieth the total width/height of the map.

valueMapImage - The same as compareMap but without living area.

valueMapText - The same as valueMapImage but as a text file. Industrial and water areas marked as impossible terrain and zero for everything else.

valueMapText2 - The same as valueMapText, but with living areas marked with ones instead of zeros.

6.4.2 Economic data

In addition to the map itself. There is a sub feature called "livsstil" on "hitta.se". This feature displays a web element showing more information about a small area on the map. One piece of information displayed is the average income for the area. By requesting information from the webpage at different coordinates. Its possible to gain data in similar format to that generated by the simulation.

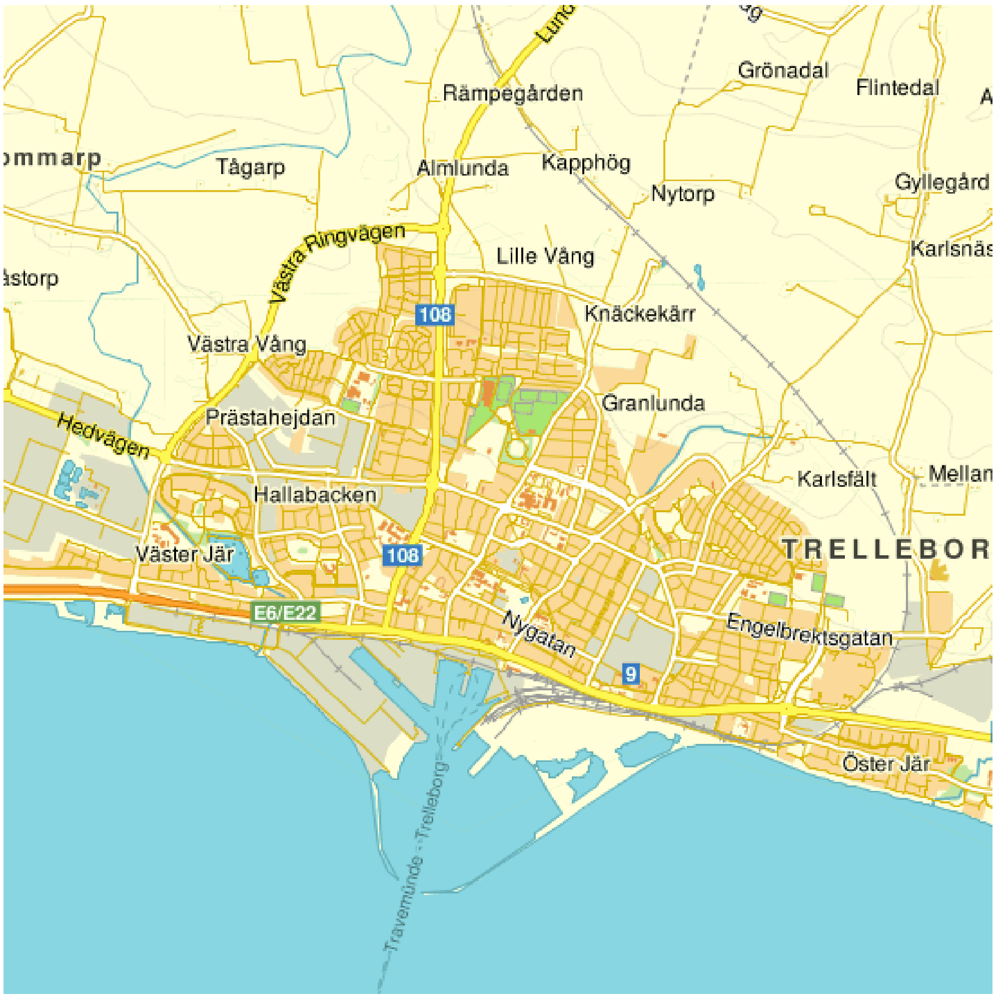

Figure 6.1: A cut screen-shot of Trelleborg taken from the website: "hitta.se" (uncut source: "http://www.hitta.se/kartan! 55.37592,13.15759,14z/tr!i=AJvqJoEC")

To gather this data a scrip named "getData.js" exists. This script depends on a program named "phantomjs-2.0.0" found at the website: "phantomjs.org". In the script are a number of constants, set to capture data about a Swedish city named Trelleborg. The precis value of each constant depends on the image used as background. Where the coordinates of landmarks in each corner of the background serves as the base. This plus experimentation assures a suitable fit.

Additionally the feature "livsstil" displays a circle of the presumed area covered. But after some testing it turns out that it actually takes data per street (sometimes divided into smaller parts). The circle size seems to be only a rough approximation of the actual area covered. Because of this there is some repeats in the data. Where multiple data points contain the same average income and street name. The precis size of the circle also presents some problems. In the case of gathering data from Trelleborg however. The difference between each point is smaller than the estimated range of the circle.

Lastly the data was formatted for use by the simulation. By removing extra information namely street name and coordinates (the order indicates coordinates).

The program contained in "incomeToMaps.hs" when run with this formatted data. Will produce several pictures used as backgrounds in this thesis.

Chapter 7 Results

7.1 Complexities

On goal for this work is that each iteration should have a worst case linear time complexity (O(n)). Since some of the work will need a sorted list this is quite difficult. Sorting algorithms are usually worst case O(n log(n)). [2]

There are three cases that need a sorted list. All related to forming the relationships graph. At each iteration new relations will form and old ones forgotten. To form new relations, people sorted on three different attributes are picked at random in varying numbers. Joined together all these form a new list that is then selected from. This sorting based on culture, profession and position would normally be worst case O(n log(n)). But since the all the attributes are constant in size. It becomes possible to use pigeonhole sort to achieve an O(n) sort.

For the rest of the program it's more straightforward. Divided into three general parts: birth, death and change.

Death is O(n) since it only iterates over the population and does O(1) work per individual. This is were for each individual it determines, based on set probabilities per age, if they are dead or not and sets a flag. Then any one over a set age is removed all at once. This is possible because of the way they are added means that they are in order with respect to age. Since they are stored in a vector this operation is O(n).

Birth is similarly straightforward. It ads new people to the end of the vector. The parents attributes forms the base when creating a new person but this is only two O(1) lookups. Because population growth increases the size from one end, and is only removed from the other. This means that their id can be used as an offset into the vector. Which becomes some additions and a lookup into a vector that is O(1).

Then the biggest one is change. One large subpart of it is forming new relations but that is already covered above. Additionally there is "getAJob" and "getAHome". Which both uses constant lookup time on parents of each individual as described for birth. "getAJob" is just a loop over each person with constant access so it is O(n). "getAHome" is slightly more complicated. In that it also requires information about where people lives and their attributes. This is done by sorting the list with pigeonhole sort as described above to achieve O(n). But this also requires that the map division used is of fixed size. If it's instead resized dynamically. Then the size of the attribute increases and its no longer O(n).

Space is O(n) because, with the exception of copying the population (which is O(n)), no extra memory is used for any of the algorithms.

7.1.1 Conclusion

The O(n) sort possible with pigeonhole sort, and the O(1) lookup time. Created by carefully adding and removing population. So that it remains sorted with respect to their ids. Allows for the total simulation step to be O(n). However this hinges on a lot of things being static. Such as the number of profession not changing during runtime, and the map division grid being of constant size.

7.2 Performance

7.2.1 Specs

All test were run on 64-bit operating system Windows 7 with the following hardware:

processor - Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz

ram - 8 GB

7.2.2 Speed

To measure the time to execution a package for Haskell named Criterion was used. It runs a function with some determined arguments a number of times to determine average time to completion. It was run with the arguments "perf csv perf.csv [list of iterations to test]" and for threading "perf csv perf.csv [list of iterations to test] +RTS -N". 7.1 7.2

Data for the single threaded runs.

Iteration	Time (Mean in s)	Population	Time per person
10	0.02609	862	3.02×10^{-5}
11	0.03489	1066	3.27×10^{-5}
12	0.04263	1322	3.22×10^{-5}
13	0.05203	1636	3.18×10^{-5}
14	0.06207	2026	3.06×10^{-5}
15	0.07435	2499	2.97×10^{-5}
16	0.09161	3088	2.96×10^{-5}
17	0.11041	3753	2.94×10^{-5}
18	0.13316	4593	2.89×10^{-5}
19	0.16119	5666	2.84×10^{-5}
20	0.19615	6932	2.82×10^{-5}
21	0.23807	8498	2.80×10^{-5}
22	0.28704	10384	2.76×10^{-5}
23	0.35306	12710	2.77×10^{-5}
24	0.43187	15508	2.78×10^{-5}
25	0.53335	18944	2.81×10^{-5}
26	0.65890	23151	2.84×10^{-5}
27	0.83739	28307	2.95×10^{-5}
28	1.07612	34484	3.12×10^{-5}
29	1.24300	42166	2.94×10^{-5}
30	1.56582	51567	3.03×10^{-5}
31	1.85017	63154	2.92×10^{-5}
32	2.25006	77346	2.90×10^{-5}
33	2.77446	94357	2.94×10^{-5}
34	3.51735	115506	3.04×10^{-5}
35	4.32108	141036	3.06×10^{-5}

Data for the multi threaded runs.

Iteration	Time (Mean in s)	Population	Time per person
10	0.02286	808	2.82×10^{-5}
11	0.02940	1041	2.82×10^{-5}
12	0.03588	1252	2.86×10^{-5}
13	0.04200	1691	2.48×10^{-5}
14	0.05003	2094	2.38×10^{-5}
15	0.05947	2692	2.20×10^{-5}
16	0.07414	3366	2.20×10^{-5}
17	0.08709	4244	2.05×10^{-5}
18	0.10642	5478	1.94×10^{-5}
19	0.12862	6805	1.89×10^{-5}
20	0.15624	9070	1.72×10^{-5}
21	0.18839	10886	1.73×10^{-5}
22	0.23103	14515	1.59×10^{-5}
23	0.28230	18048	1.56×10^{-5}
24	0.35845	23024	1.55×10^{-5}
25	0.45030	28769	1.56×10^{-5}
26	0.56782	37871	1.49×10^{-5}
27	0.72858	47071	1.54×10^{-5}
28	0.94191	60202	1.56×10^{-5}
29	1.20273	74005	1.62×10^{-5}
30	1.64972	98796	1.66×10^{-5}
31	1.96467	128369	1.53×10^{-5}
32	2.57979	163555	1.57×10^{-5}
33	3.36558	201828	1.66×10^{-5}
34	4.36207	257162	1.69×10^{-5}
35	5.56985	320539	1.73×10^{-5}

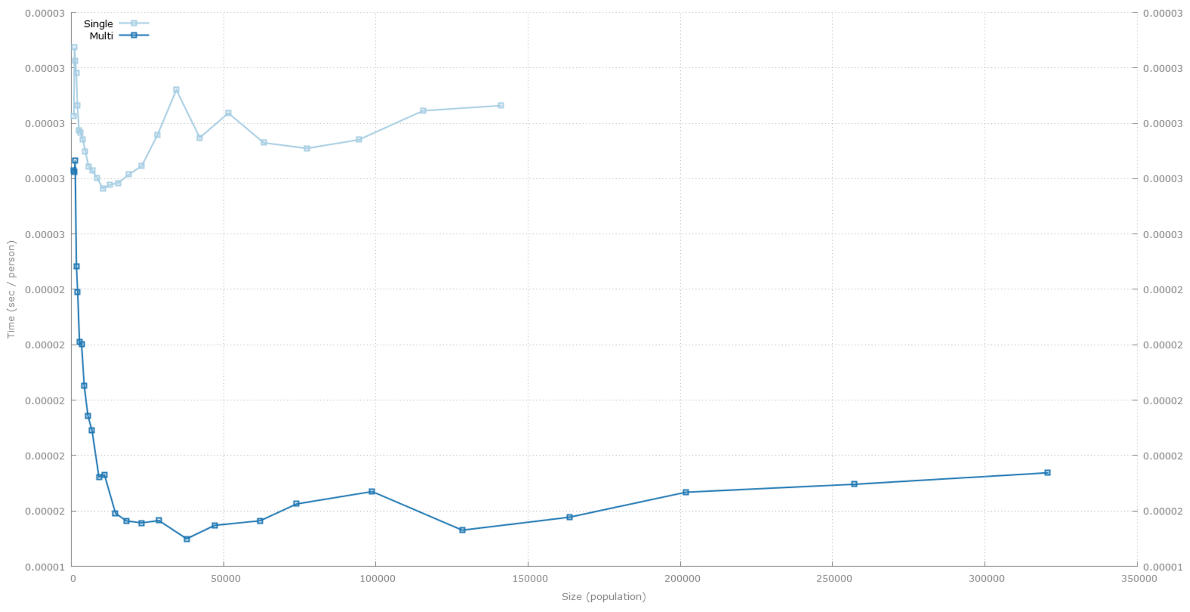

Figure 7.1: Performance of single vs multi threaded based on population size

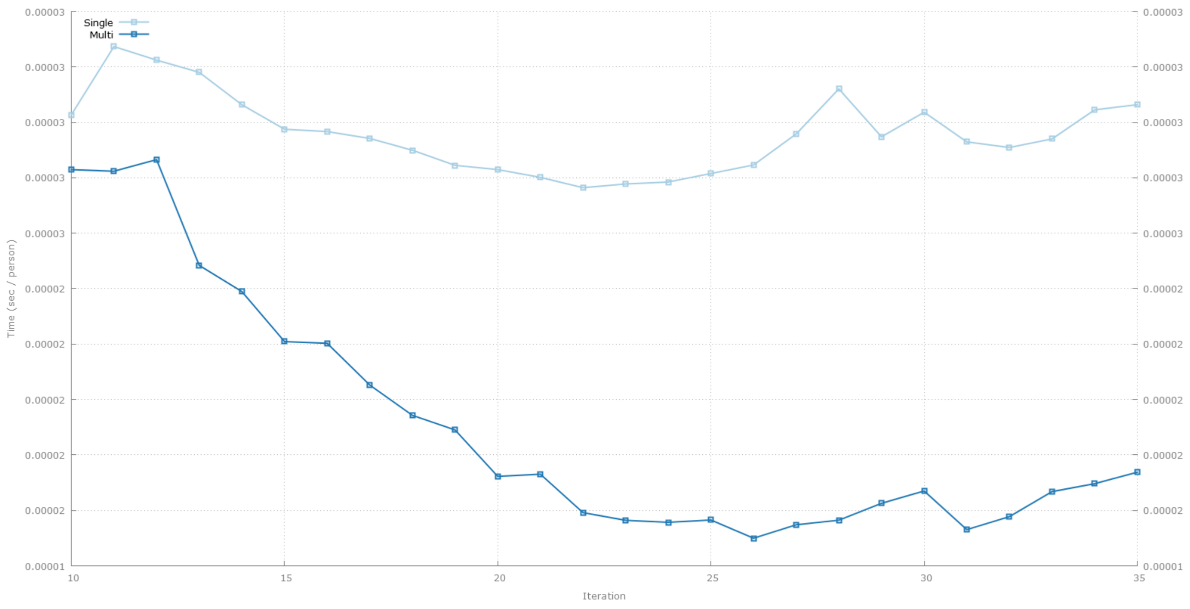

Figure 7.2: Performance of single vs multi threaded based on number of iterations

The population size for the single threaded run is always the same. But for the multi threaded variant it will vary depending on the precise order the threads are scheduled. It is possible to implement it in different ways to achieve much more stable population size. But in all instances where this has been attempted the speed was reduced to a similar level of the single threaded variant. If the final number of iterations are known before hand you can use single threaded for the first iterations. This helpt in preventing cascading fluctuations and produces a more stable population size.

7.2.3 Memory

The program uses garbage collection, because its implemented in Haskell. This makes checking memory usage somewhat tricky. Fortunately their is some built in support to generate graphs over memory usage. By passing arguments "+RTS -hy" it will generate a pdf document with a grap 7.3. That shows size over time with different colours for different memory containers. Here the different memory containers are not of interest. Rather the total amount used and more imporantly the change over time.

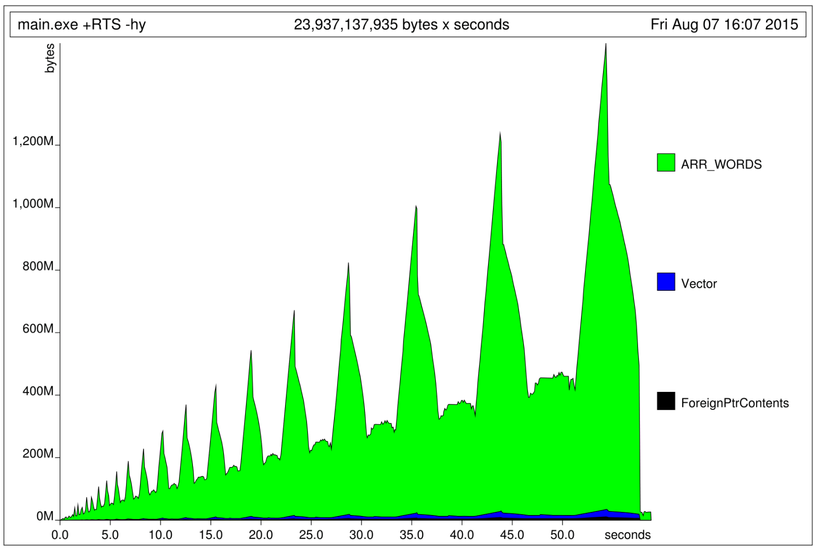

Figure 7.3: memory usage

7.3 Population spread

Since the simulation is probabilistic in nature (but based completely on a seed) reruns will produce the same result but any change of the seed will produce a different result. Although the population spread is the least affected by this. All images here were produced with a seed of 0.

In these images the spread of the population is shown as red squares. Where the size of the squares denotes the amount of people living in that area. The background shows impossible terrain of two types, industrial area (since no ones lives there) and water. Everything else is neutral. The population starts in the x and y coordinates (26, 30) counted from the top left, set by the option "startPosition = 26 30".

The first image can be seen here 7.4.

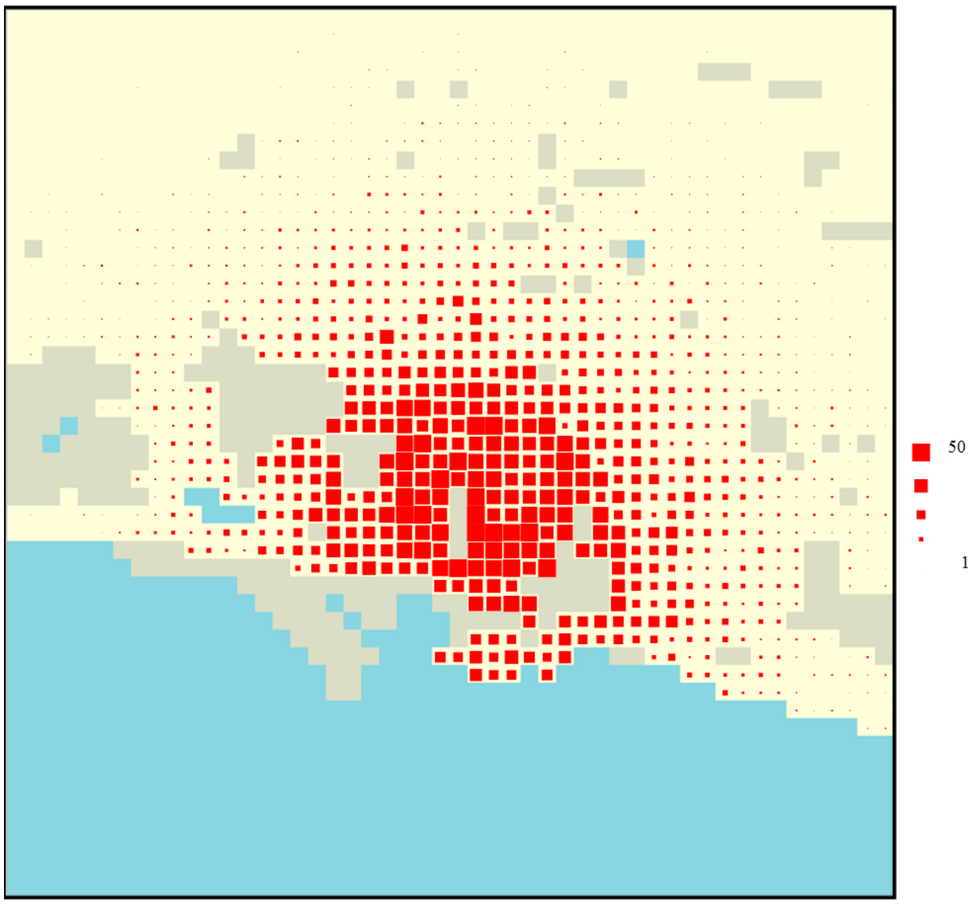

Figure 7.4: Simulated population spread over Trelleborg, blue is water and grey is industrial areas

As can be seen it has more people in the center and then spreads out. With fewer and fewer people the further from the center.

In the next image we add the colour orange to show where houses are. 7.5

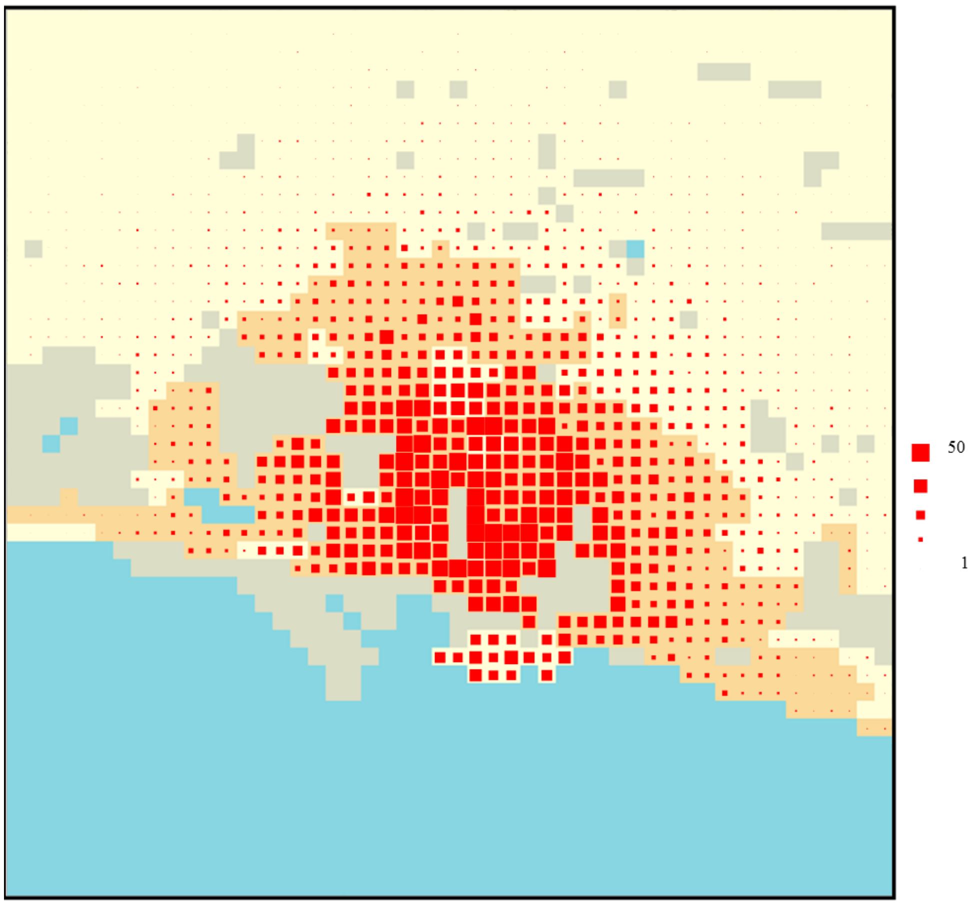

Figure 7.5: Simulated population spread over Trelleborg, orange is areas with a substantial portion of houses, blue is water and grey is industrial areas

If an underlying structure is known before hand. This layer can be taken into consideration by the simulation. Producing the next image. 7.6

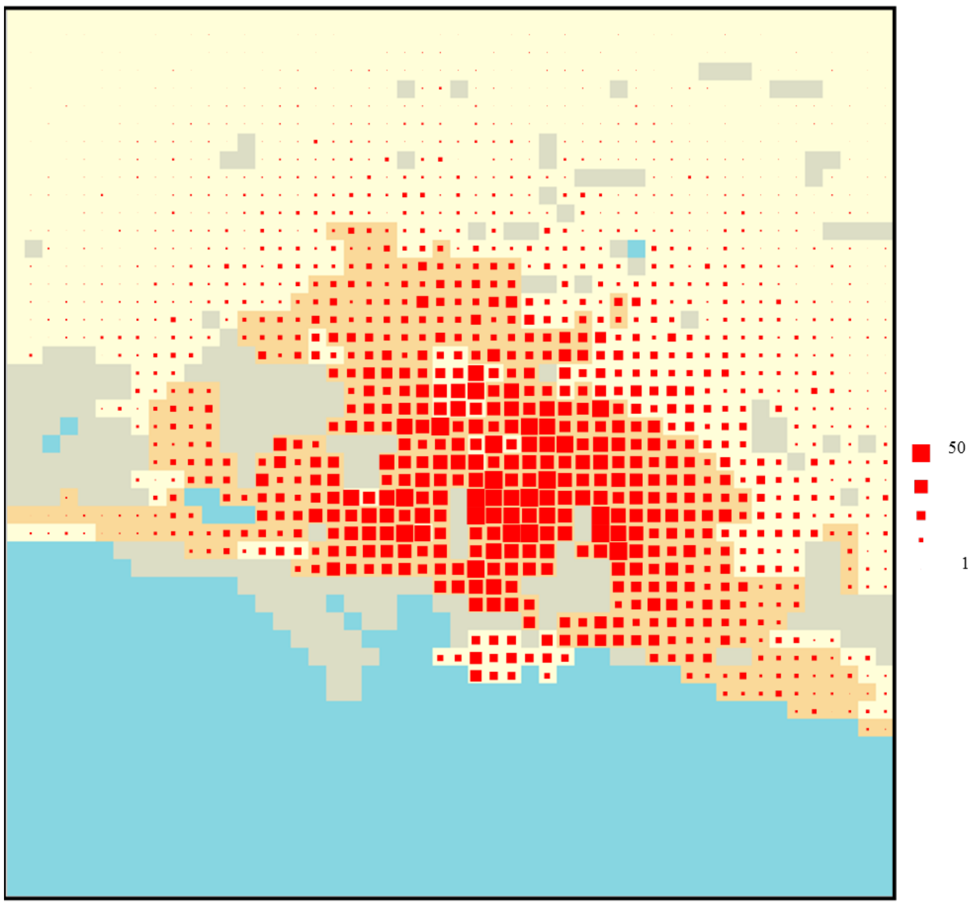

Figure 7.6: Simulated population spread over Trelleborg with fitted value map, orange is areas with a substansial portion of houses, blue is water and grey is industrial areas

For comparison if a random approach is used instead this picture is the result 7.7.

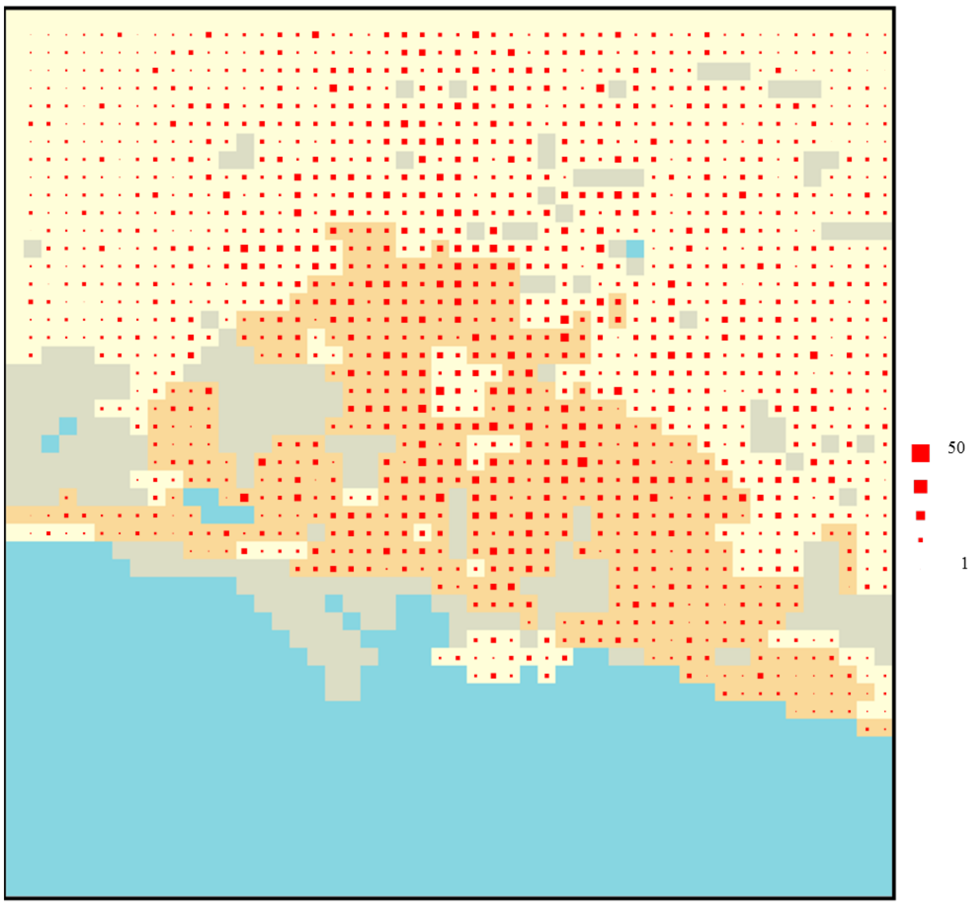

Figure 7.7: Random population spread over Trelleborg with fitted value map, orange is areas with a substansial portion of houses, blue is water and grey is industrial areas

7.4 Socioeconomic Clustering

From the data gathered from "hitta.se" the following image is constructed 7.8.

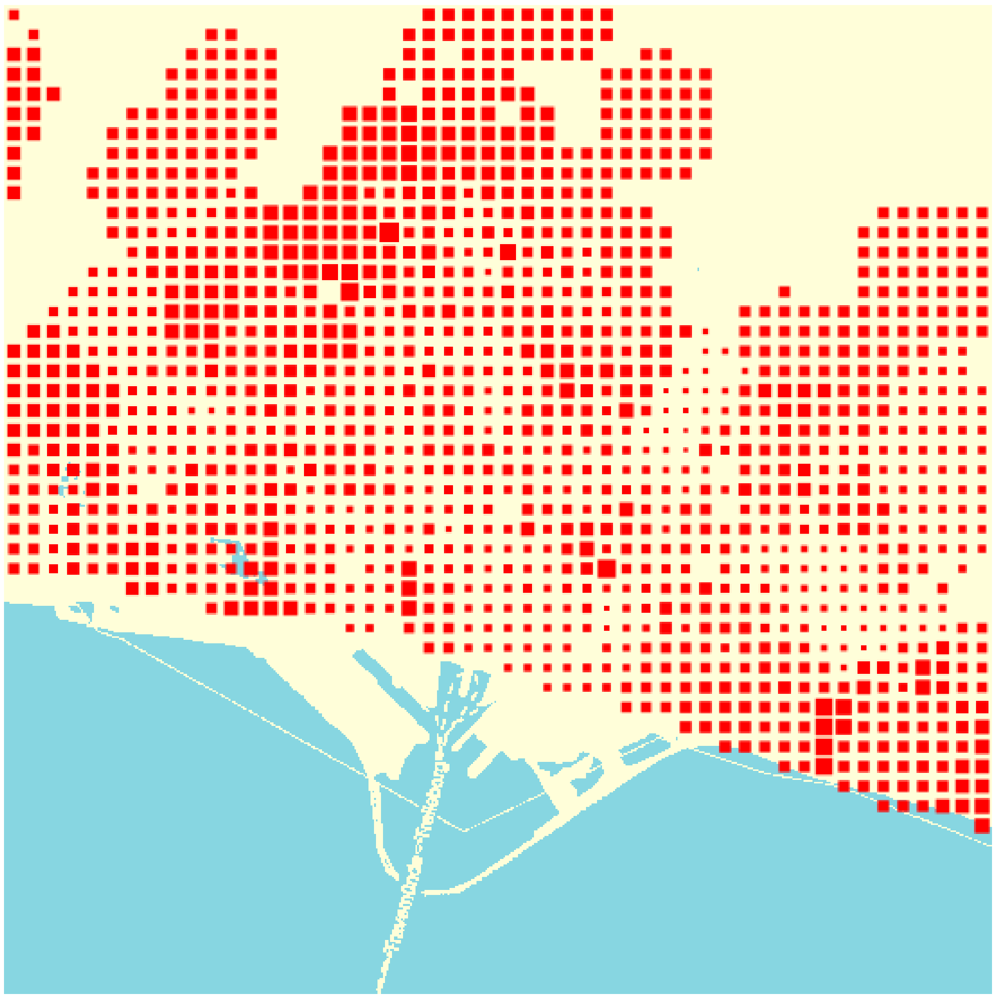

Figure 7.8: income data from "hitta.se" over Trelleborg, size indicates income where smallest is the lowest and the biggest is the highest scaled in proportion to the difference

Using a value translation for each profession the simulation can be shown in a similar way. Where each area is the average value of all the number of professions living there 7.9.

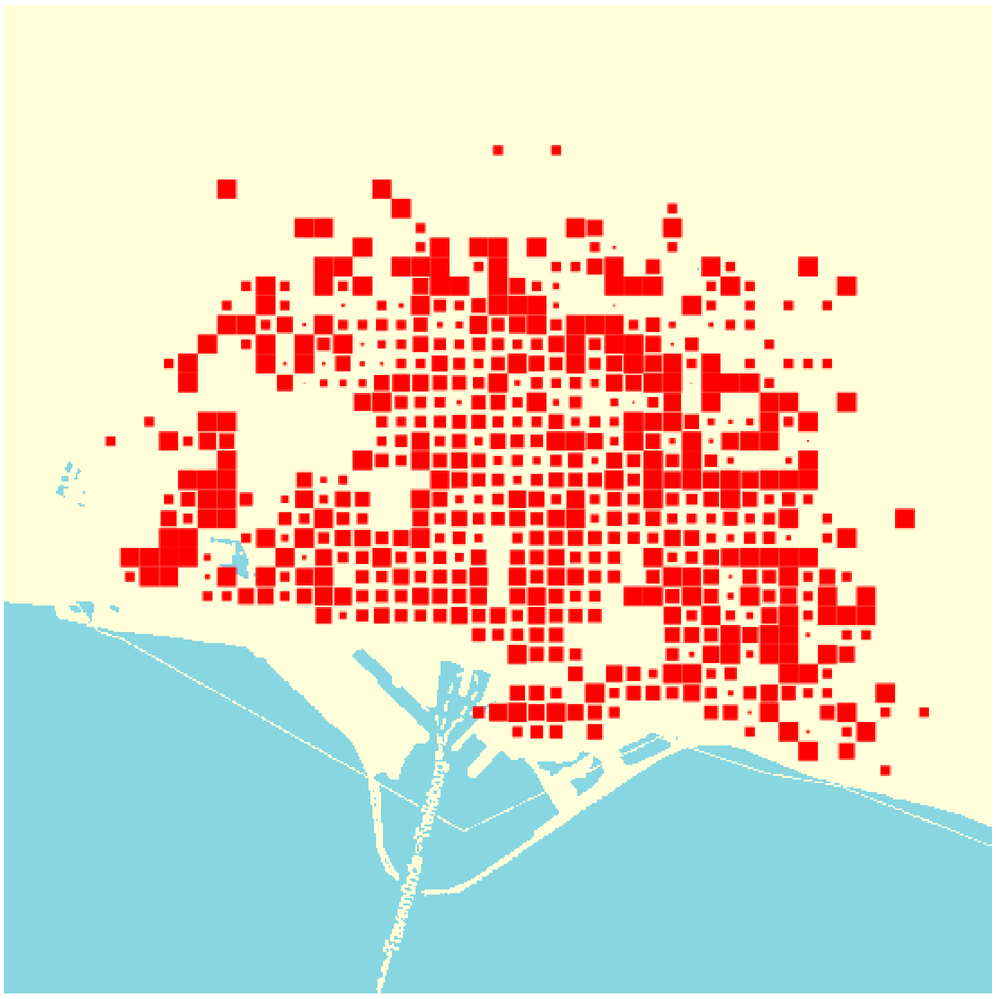

Figure 7.9: data from the simulation where each profession has been given a value, size indicates average value where smallest is the lowest and the biggest is the highest scaled in proportion to the difference

7.5 Social Clustering

7.5.1 Culture spread

First image show the culture simulated for the population spread 7.10.

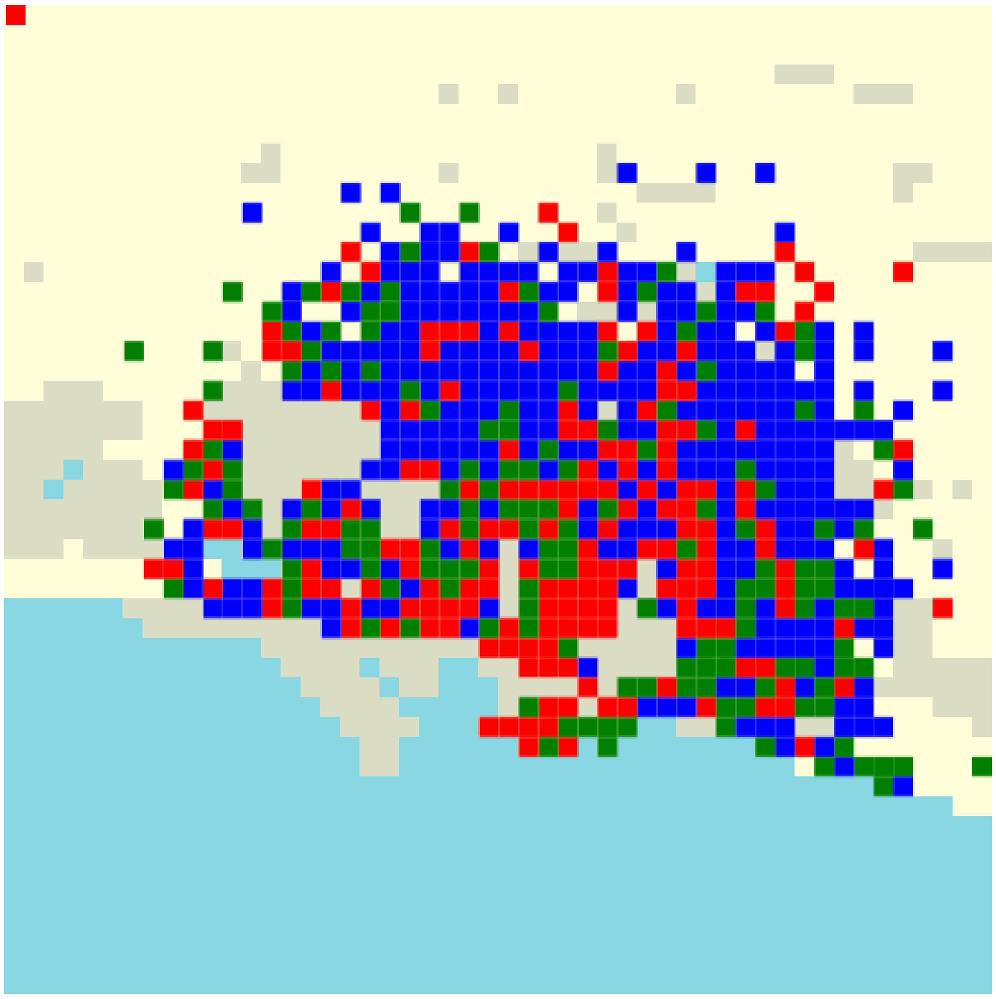

Figure 7.10: each colour shows the dominant culture in that area

Next group of images shows how the map changes depending on seed 7.11 7.12 7.13 7.14. Here a seed of 0 for the first, 1 for the second, 2 for the third and 3 for the fourth. To show how the dynamic feedback can drastically alter the final shape. But all larger patterns still remain.

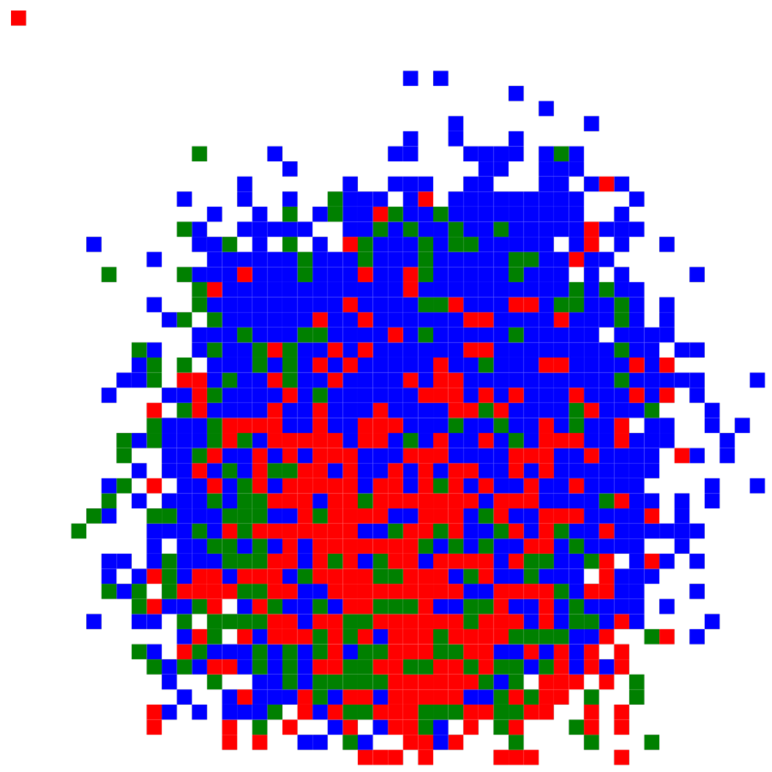

Figure 7.11: each colour shows the dominant culture in that area, seed = 0

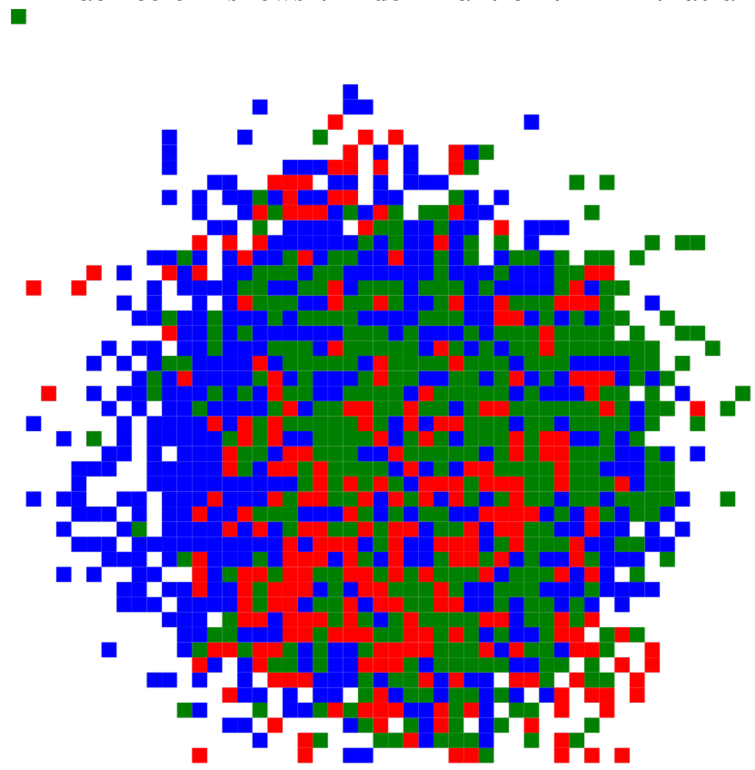

Figure 7.12: each colour shows the dominant culture in that area, seed = 1

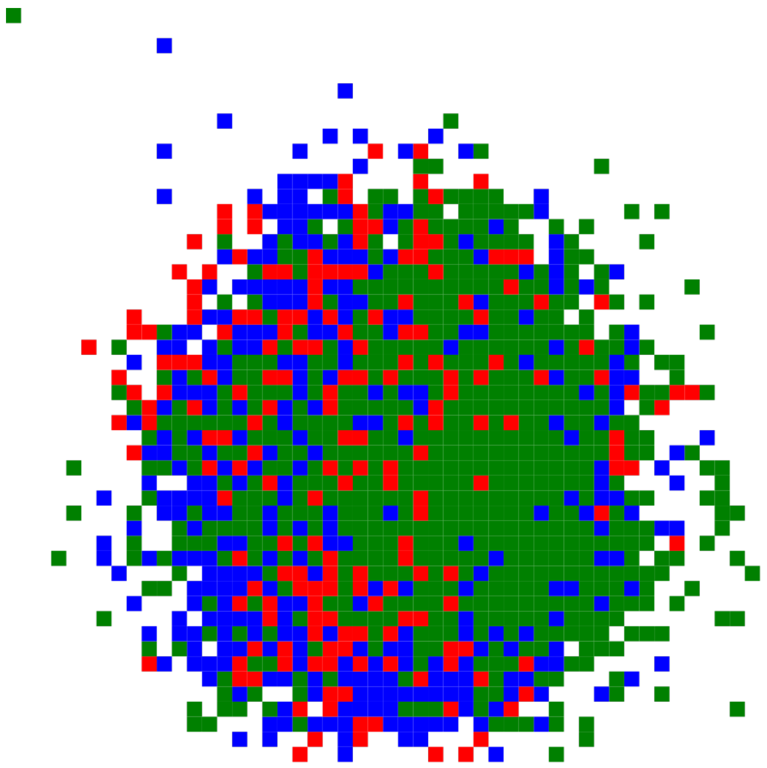

Figure 7.13: each colour shows the dominant culture in that area, seed = 2

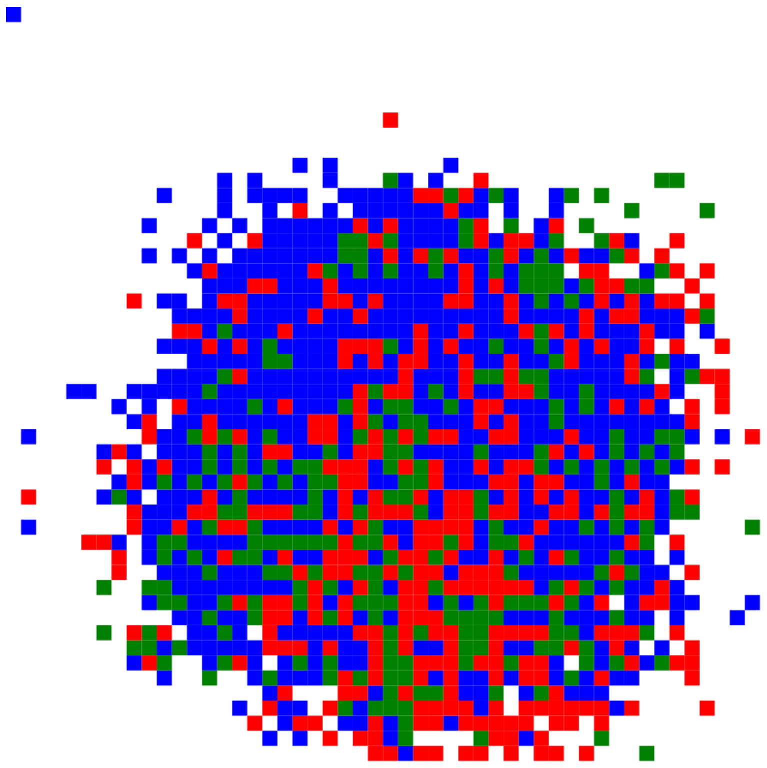

Figure 7.14: each colour shows the dominant culture in that area, seed = 3

7.5.2 Relations

Where different groups relations are focused can be seen by these graphs 7.15 7.16 7.17. Where each person is drawn as a circle and each relations as a line. Relations related to the specific culture is drawn in red and the rest in gray.

One problem here is that the graphs become very large very fast, since each person have a couple of hundred of relations. As a result of this the computer used crashes after a couple of days due to lack of memory. If a graph of about 10k people is attempted to be drawn. Therefore the graphs shown are of about 1k people where only people that have been placed are drawn. This also means that some patterns from the first starting generation remains, as can be seen by the big ball in the middle.

7.6 Relationships graphs

Mentioned previously in subsection Social Clustering is the relations between individuals in the simulation. All these relations together forms a massive graph. This graphs properties compared to other network tells us what kind of network it most resembles.

Two major properties of a network graph is: number of connections per node and clustering coefficient. Where clustering coefficient is the probability that a friends friend is also you friend. [14]

Network	n	Mean degree z	Cc	CC for random graph
Neural network	282	14.0	0.28	0.049
Metabolic network	315	28.3	0.59	0.090
Food web	134	8.7	0.22	0.065
Word co-occurrence	460,902	70.1	0.44	0.00015
Company directors	7,673	14.4	0.59	0.0019
Film actor collaborations	449,913	113.4	0.20	0.00025
Mathematics collaborations	253,339	3.9	0.15	0.000015
Biology collaborations	1,520,251	15.5	0.081	0.000010
Power grid	4,941	2.7	0.080	0.00054
WWW (sites)	153,127	35.2	0.11	0.00023
Internet (AS level)	6,374	3.8	0.24	0.00060

Table 7.1: n is number of connections and Cc is clustering coefficient, source [14]

The above table show a number of different networks and their properties. Calculating the clustering coefficient for this work turned out to be time consuming. Since a naive approach for calculating it is $O(n\hat{2})$. Therefor only a random subset of the entire population determined the clustering coefficient. The simulation toke 15 iterations and totaled 2283 people.

Network	n	Mean degree z	Cc	CC for random graph
This work	158,800	69.5	0.16	0.00013

By comparison its clustering coefficient most closely matches film actor collaborations and mathematics collaborations. The number of connections per node is also high compared to most of the networks. Although lower than film actor collaborations. But this is probably shewed lower by a large number of young people in the simulation. Explained by the rapid growth of about 15% per iteration.

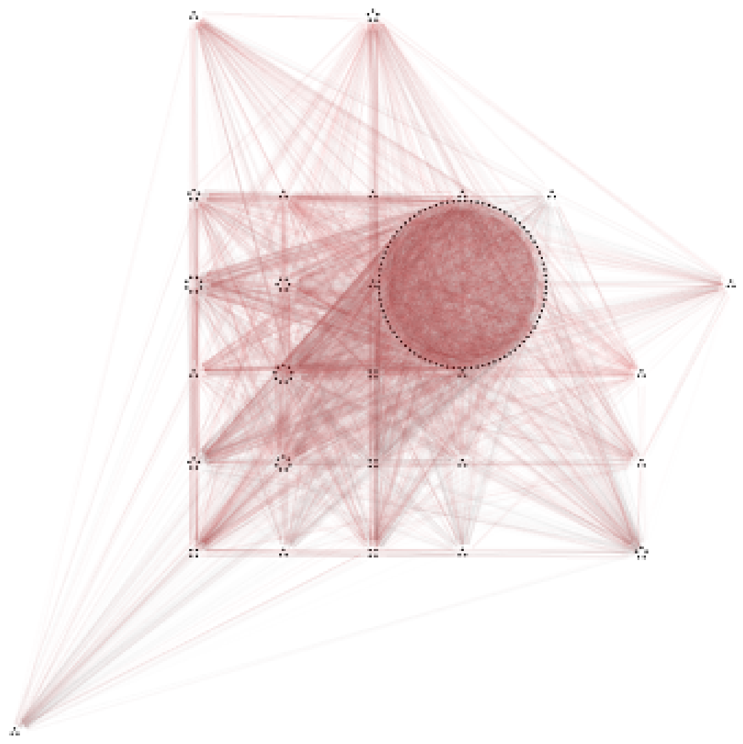

Figure 7.15: Relationship graph with culture one in red

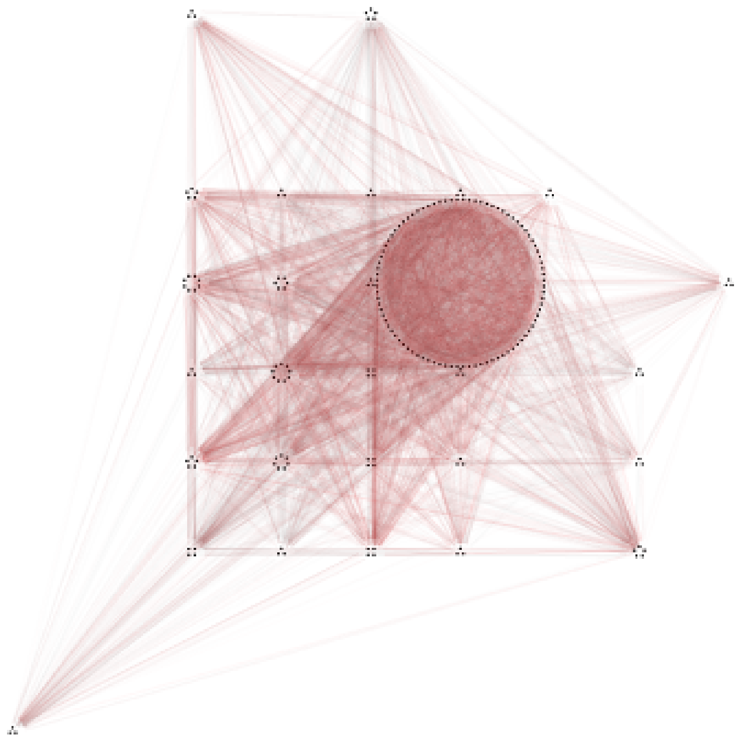

Figure 7.16: Relationship graph with culture two in red

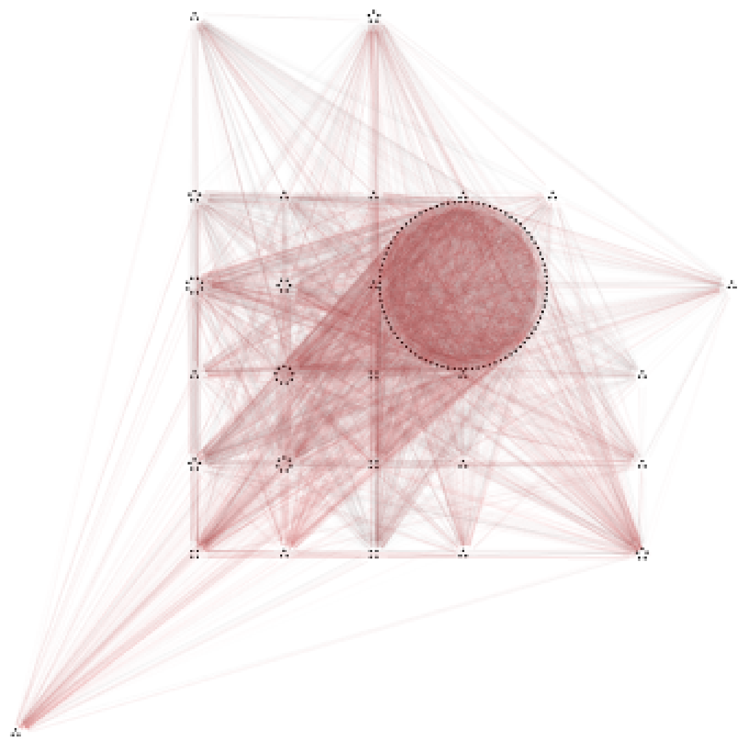

Figure 7.17: Relationship graph with culture three in red

Chapter 8

Software

In this part is descriptions for how to compile and run the program. To be able to generate all the data used in this thesis.

8.1 Build

This build section is specifically for Windows and have not been tested on other platform. Everything used is cross platform and should also work for Linux and Mac but the specific commands might be slightly different.

- Install Haskell Platform found currently on: https://www.haskell.org/platform/. Only version 7.10 is guaranteed to work but should work for all subversion of 7 and possible future version (versions sub 7 will not be able to compile the program without modifications).
- Install Stack by opening a command line (on Windows: press the windows key, type "cmd" and then press enter) then enter "cabal update" and then "cabal install stack".
- Navigate to folder containing the project for this thesis (on Windows: by entering "cd [path]").
- Type "stack build"
- To run the program type "stack exec main [program options]"

8.2 Command Line Options

Typing "help" as the program option will provide all possible options and their format as output.

The format is [Name of command] [Arg1 for command] [Arg2 for command] [etc...] [Name of command2] [Arg1 for command2] [etc..] etc.. In addition there are flags that do not have any arguments so only name is needed.

So for example to generate the population map used in graph 7.4 enter as program options “pop 40 40 optionsFile option.txt”. For general experimentation if no options are provided (this does not include flags) then the general mode will be run that will show this: “Enter number of iteration(s):” Where upon any number entered (plus enter) will be evaluated and general statistics shown.

8.3 Settings

Inside the program is default setting that are used if no options file is specified or if a specific setting is not found. A filed named "options.txt" exists as an example of possible settings to use. In this are examples of typical setting used for most of the experiments.

Chapter 9

Discussion

9.1 Conclusions

9.1.1 Performance

It turns out to be possible to not only do this so the program is O(n) per iteration. But also fast even when the numbers do not approach infinity.

While the analyze of the source code shows that the program takes worst case O(n) per iteration in both time and space. The data is actually slightly better than this. For memory the picture shows a linear increase throughout the programs lifetime. Where each pike corresponds to the copying of the data when its sorted 7.3. Time per person however is even better than expected. As it's almost a linear increase after the first iterations. Also despite the large differences in size. The early iterations have similar performance to the biggest ones. 7.1 7.2

This is better than expected since at each iteration all the previous iterations are also calculated. Each successive step increase the population size at about 15%. Therefor as long as that holds it should be O(n log(n)) with respect to the final population size. In the beginning constants increase the time per person. But as the population grows we can see that it starts to approach this. What is surprising is how close to linear it is while still reaching numbers of hundred of thousands in population size.

Why?

What this speed allows for is direct incorporation into real time systems. So that the variables such as terrain can change and the population can be updated to reflect that change. In addition to this it allows for simulation during run time, reflecting that time has passed and cities change. Or merely to save memory, the amount of relations generated for a population of one million takes an estimated 500 megabyte of data. If several cities needs to be generated in this fashion it will quickly start to take up a lot of memory.

In addition with a time complexity of O(n) it means that the simulations running time wont start to run away as the population size increases.

9.1.2 Social Clustering

Turns out to work really well. Groups relations shows in the data. Where some don't like each other and so do not appear as intemingled with each other. Compared to other that do and so become far more intermingled. A dominant culture can also emerge that pushes the other culture away from the center. 7.11 7.12 7.13 7.14

Apart from group relations there is also the groups relates to themselves. Where people from the same groups prefer positions close to each others. It creates quite interesting patterns that would be useful to draw from. Enabling the creation of interesting game worlds.

There is one aspect however where these patterns are not so clear. Relationships graphs as visualised previously. 7.15 7.16 7.17. These do not clearly show patterns. One hindering aspect here is simply performance. The program outputs these graphs as svg, that is a file format that just describe primitives (such as lines). Converting these to png or similar format have proven difficult for larger populations. Also the time taken for generating the svg files quickly grows. An attempt was made to create on with a couple of thousand people and after four days the computer crashed. One possible solution to this is to write a new rendering back end. But this was deemed to time consuming for this work.

9.1.3 Socioeconomic Clustering

Does not work that well. Although some local patterns can be seen they are too small, and larger patterns where not achievable in the simulation. This presents a clear limit to what i can do in its current state.

Even with the problem with the data it's not enough to explain the difference.

9.1.4 Relations

This is by far the most flexible part of the simulation. The number of connections and clustering coefficient can both be raised or lowers based on what is desired through different variables.

In comparision to other networks it matches social networks such as film actor collaborations very closely. While being quite different from other networks such as neural nets and WWW. This shows it can generate networks whose properties matches those found in real relationships graphs.

9.1.5 Research Questions

RQ1: Can the simulation create patterns that exists in the population of real cities?

RQ2: Will the resulting relationship graph have the same properties as real relationships graphs (in terms of path lengths, clustering and degree distributions [13])?

RQ3: Is it possible to do each iteration in O(n) time and O(n) space (where n is the population size)?

Objectively RQ3 is achieved. As shown in this work the simulation is both fast for relative small values and scales linearly both in size and time.

As for RQ2 the properties of the relationships graph matches closely to those found in social networks. While being markedly different from other networks.

For RQ1 the answer is more subjective. While it has manage to exhibit many patterns that groups would create, 7.11 7.12 7.13 7.14 and the simulated population spread is general enough to approximate a city through use of the underlying value map 7.6. When compared to the data gathered from "hitta.se" the simulated patterns for professions are wrong. While it exhibits somewhat similar patterns as the culture maps they lack larger clustering of high and low values. It seems the current solution is insufficient to create the same patterns as seen in the data gathered, when it comes to income.

9.2 Future Work

There is many avenues to explore for more work on this project. On especially needed is more work on improving professions distribution to take into account larger patterns. This could be done with an additional value map that only influences based on profession. With certain points that attract some and repulse others.

The simulation is quite easily extend able. So to add additional systems to be able to reproduce more patterns would be one possibility.

Larger data set of cities and their population / income spread would also be an improvement. As of right now its a bit limited with only one city.

References

[1] A. Martin, A. Lim, S. Colton, and C. Browne, "Evolving 3d buildings for the prototype video game subversion," vol. 6024, pp. 111–120, 2010.

[2] E. Hirsch, J. Karhumäki, A. Lepistö, and M. Prilutskii, *Computer Science – Theory and Applications: 7th International Computer Science Symposium in Russia, CSR 2012, Niszhny Novgorod, Russia, July 3-7, 2012, Proceedings.* Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012.

[3] S. Legg and M. Hutter, "A collection of definitions of intelligence," *Frontiers in Artificial Intelligence and applications*, vol. 157, p. 17, 2007.

[4] M. A. Bedau, "Weak emergence," *Noûs*, vol. 31, no. s11, pp. 375–399, 1997.

[5] R. M. Smelik, K. J. De Kraker, T. Tutenel, R. Bidarra, and S. A. Groenewegen, "A survey of procedural methods for terrain modelling," pp. 25–34, 2009.

[6] G. Kelly and H. McCabe, "A survey of procedural techniques for city generation," *ITB Journal*, vol. 14, pp. 87–130, 2006.

[7] B. Cullen and C. O'Sullivan, "A caching approach to real-time procedural generation of cities from gis data," 2011.

[8] P. Muller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool, "Procedural modeling of buildings," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 614–623, 2006.

[9] P. Muller, "Procedural modeling of cities," pp. 139–184, 2006.

[10] G. Kelly and H. McCabe, "Citygen: An interactive system for procedural city generation," pp. 8–16, 2007.

[11] J. Barros, "Urban dynamics in latin american cities: An agent based simulation approach," *Environment and Planning B: Planning and Design*, 2004.

[12] R. B. Matthews, N. G. Gilbert, A. Roach, J. G. Polhill, and N. M. Gotts, "Agent-based land-use models: a review of applications," *Landscape Ecology*, vol. 22, no. 10, pp. 1447–1459, 2007.

[13] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications.* Structural Analysis in the Social Sciences, Cambridge University Press, 1994.

[14] M. E. Newman, “The structure and function of complex networks,” *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.

[15] R. Dunbar, “Neocortex size as a constraint on group size in primates,” *Journal of human evolution*, vol. 22, no. 6, pp. 469–493, 1992.