An improved approach to explore spatial district aggregation

**Abstract**: Districts redistribution problem is an important research subject in census geography in terms of its potential ability to facilitate the organization and management of census while providing customized area for the user to use [1]. In many situations, it is often useful to group a large number of spatial objects such as census tracts into smaller ones whose results will form a more comprehensive spatial extend [2]. This procedure is called district aggregation. In order to deal with district aggregation problem, both performance and computational time should be well considered. This paper uses an improved approach to solve district aggregation problem by using GAT (Give and Take) algorithm [3] combined with Merge and Combine method and Parallel computing technology. This approach, not only solves the problem that GAT’s performance cannot be further improved when reaching to a certain number of iteration, but can also generate multiple optimized results at the same time. The approach and experiment are implemented and analysis are made in this paper.

**Key words:** District Aggregation, Evolutionary Algorithm, GAT, Parallel Computing, Census Geography

**Introduction:**

**Political redistribution problem, also called zone design problem, is very common in census geography. In general, it is to find a method to aggregate n units into k zones (k<n) such that some objective function values are optimized while some spatial constraints are maintained [4]. One of the most common applications in political redistribution is to aggregate administrator units into a predefined number of regions such that each new region is internally contiguous and the sum of population in each region is as similar as possible [4]. In many situations, by solving this problem, the user is able to get customized regions which will form a more comprehensive spatial extend.**

**Redistribution problem can be considered as geographic optimization problem. In general, geographic optimization problem can be categorized as four types [5]: First, selection problem without spatial constraints. What is does it to find the subset of spatial entities to satisfy one or more goals. One typical example of this problem includes p-median problem which PJ Densham, G Rushton had done some optimized method research on this area [6].**

**The second type is selection problem with spatial constraints. In addition to the selection procedures that are similar with first type, it also need to maintain the spatial constraints. One common example of this is site selection problem. In 2000,** [**TJ Cova**](https://scholar.google.com/citations?user=bnnYw7AAAAAJ&hl=en&oi=sra)**,**[**RL Church**](https://scholar.google.com/citations?user=t4-JG2IAAAAJ&hl=en&oi=sra) **raised multiple spatial constraints for a single region site search problem [7] while in 2002, H Önal, RA Briers incorporated spatial constraints and used integer programming method to solve this problem [8]. In 2005, Takeshi Shirabe purposed a new formulation of contiguity that can be used to describe spatial constraints and be incorporated into any complex integer programming model [9]. In 2009, he himself purposed a new integer programming approach to solve site selection problem [10].**

**The third type of problem is the partition problem without spatial constraints. Each spatial object is assigned a value and we need to find out the combination value of those spatial entities to satisfy some objective. Bennett, Xiao, and Armstrong has done deep research in this area by using Evolutionary algorithm [11].**

**The last type of this problem which we will also address specifically in this paper refers to partition problems with spatial constraints. In addition to the process of the third problem, spatial constraints must be satisfied. Political redistribution problem is exactly such a problem which space needs to be rearranged while spatial constraints must be maintained as well.**

**Solving this problem is challenge for the following reasons: first, it is hard to optimize the objective function value while at the same time maintains spatial constraints like contiguity. Second, these problems are often computational expensive because the number of feasible solutions will increase exponentially with the input size [5]. In this case, exact method may not be suitable for solving this problem although it can reach to global optimized result. Instead, a compromised method, called heuristic method is used to produce results in an acceptable amount of time while achieving near optimal solutions.**

**Many methods have been raised to solve this problem so far. Back in 1991, S Openshaw and L Rao tried to solve Redistribution problem by using AZP algorithm [12] and applied this method in the field of social economic units. This method though achieves good result, is computational expensive. Improvement to AZP lead to zone design system in 1995 and in 2002, Alvanides used simulated annealing to develop an alternative implementation of AZP** [13]. In 2004, [F Bacao](https://scholar.google.com/citations?user=sFoh1WAAAAAJ&hl=en&oi=sra), [V Lobo](https://scholar.google.com/citations?user=z3xr3NQAAAAJ&hl=en&oi=sra), [M Painho](https://scholar.google.com/citations?user=L1PHpI8AAAAJ&hl=en&oi=sra) used generic algorithm to zone design while in 2005 RM Assunção, MC Neves etc. used Minimum Spanning Tree (MST) to regionalize social economic units in a more efficient way. **In 2008, N, Xiao proposed a unified framework which solves geographic optimization problem by using graph theory and evolutionary algorithm. After that, in 2011, Myung Jin Kim utilized Xiao’s thinking and raises a new and efficient algorithm called Give and Take (GAT). This algorithm performs quite effectively and efficiently in terms of its ability to solve population equality problem.**

**However, this algorithm does has some limitations. First, its final result cannot be further improved when the total number of iteration reaches to a certain point. Second, this algorithm can only generate one good result at a time. If user wants to have more than one result, he/she should run this algorithm multiple times. This is not a big issue when dealing small dataset, but the factors of saving computational time becomes more critical when the input dataset becomes larger. This paper solves these two limitations by utilizing the thinking of evolutionary algorithm and the parallel computing technology. Methods will be elaborated and experiment and results will be shown in the following chapters.**

**The paper is organized in this way: Chapter one is introduction which has an overview of this topic’s research condition. Chapter two gives detailed explanation of theory and methods. Chapter three is the experiment, result and relevant analysis. Chapter four is the conclusion and the future work that needs to be done.**

**Methods:**

**This paper implements GAT (Give and Take Algorithm), one of the heuristic method to solve political redistribution problem while utilizing Evolutionary Algorithm and parallel computing technique to solve the two limitations I have discussed of GAT algorithm. This chapter will give a detailed description and explanation of the methods that are used. First, the paper will talk about graph theory and how it is used to represent spatial unities. And then it describe details of GAT. Thirdly it describes the recombining methods which utilizes the thinking of Evolutionary Algorithm to solve the first limitation that has been mentioned in the introduction. Finally, it introduces parallel computing technology to generate multiple good solutions at a time.**

**Graph theory and its representation**

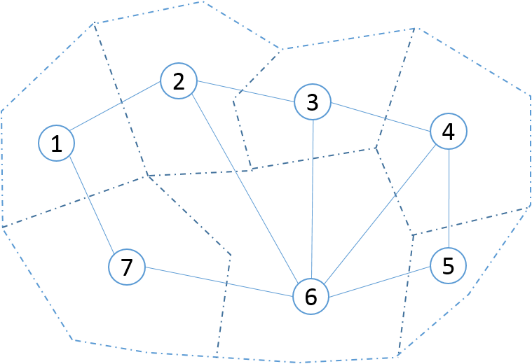
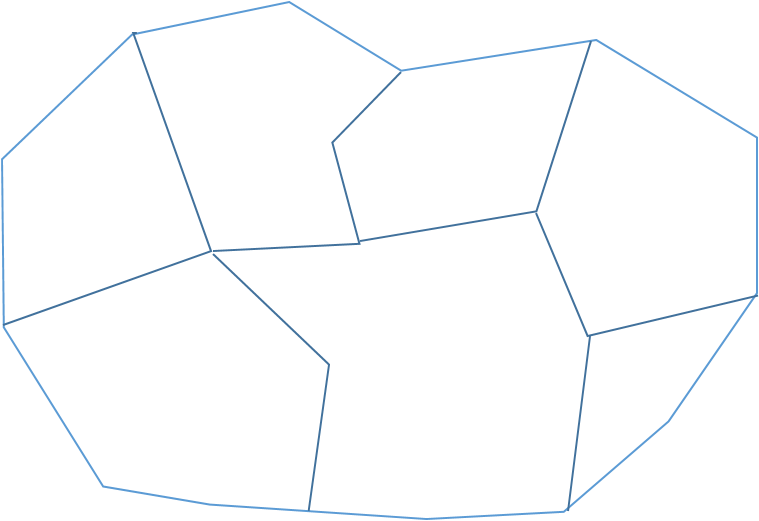
**A computer graph is defined as follows: where is called vertex or nodes and is a set of all edges that may connect one vertex to another. For example, can be represented an edge between vertex and vertex, thus.**

**Graph can be categorized as directed graph and undirected graph. Directed graph has direction attribute for each edge which uses an arrow to represent while undirected graph does not have such attribute. Figure 1a shows the example of this difference.**



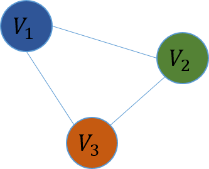
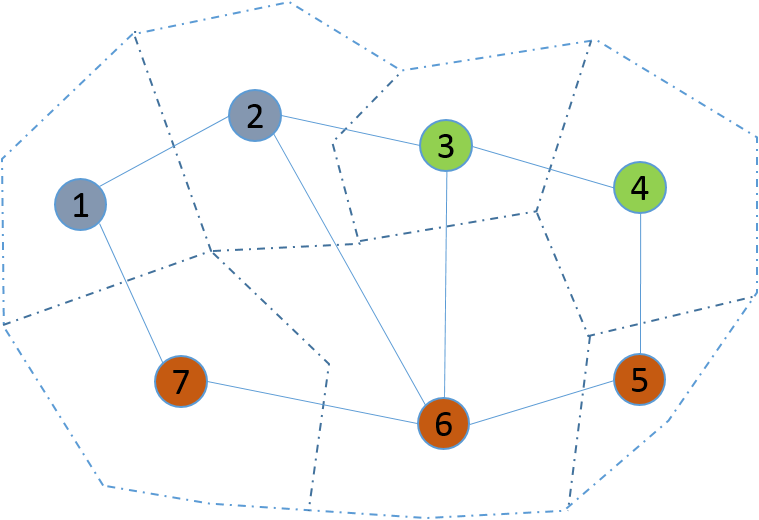
***Figure 1a: Left part is directed graph which edge has arrow to represent direction. Right part is undirected graph which edge does not have arrow.***

**In this paper, computer graph is used to represent spatial. Each census tract or block in a map is represented as a vertex or node in a graph. Since edge between each vertex in a graph is used to represent the contiguity between each region and no direction attribute is used, will use undirected graph. The relationship between geographic map and its corresponding graph representation is shown in figure 1b.**



***Figure 1b: Left part is geographic map in real world and right part is its corresponding graph representation. Here V={1,2,3,4,5,6,7} and E= {(1,2),(1,7),(2,3),(2,6),(3,4),(3.6),(4,5),(4,6)}.***

**Suppose we use some algorithms to aggregate the graph above, the result will have several new groups (say three). For each new group, it can be considered as a new ‘super node’ which contains more than one ‘sub node’ in original graph and I can use ‘super edge’ to represent the contiguity between each group. Then in this case, I can create a new super graph which simply the graph structure while maintains the core aggregation results and the spatial relationship between each other. One example of the result using graph representation is shown in figure 1c.**



***Figure 1c: Left part is one partition result example. The graph is divided into three groups. The right part figure is the simplified super graph which only maintains the grouping result and the contiguity relationship between each group. Here, V= {}, E = {(), (), ()} while.***

**This concept is very important in this method and will be used in solution recombination. More details will be covered in that part later.**

**GAT Algorithm**

**This algorithm is first developed by Kim, Myung Jin in her PhD dissertation. It is a new heuristic method as well as greedy algorithm that can be used effectively in district redistribution problem.**

Suppose we want to aggregate census tract data to generate new regions such that each region has almost same population. The GAT algorithm will be explained based on this goal.

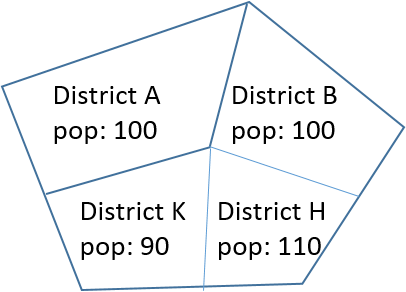
The algorithm consists of two parts: Initialization and Optimization.

* **Initialization**

The first step is to roughly generate a solution by randomly assigning census tracts to its adjacent group. We keep doing this until there is no census tract left. Since it just randomly assign census tract to its nearby group, it is hard to guarantee population equality based on initialization. Figure 2a displays one example result of initialization.

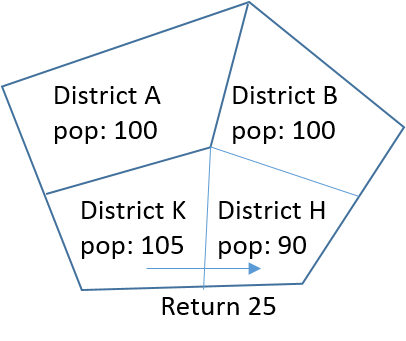
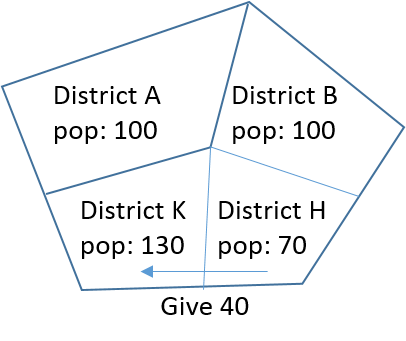
* **Optimization**

After having initial solution, Give and Take is applied to optimize this result by swapping census tracts that belong to different districts but are adjacent with each. First, it randomly picks two districts whose total population are not equal with each other. I assume we pick district c and district d and calculate the population difference between each other. Then we randomly pick a group of census units from high regions (here is H) and move these units into low regions (here is K). The total population of these selected regions should be larger than the population difference such that after moving, the population of region H now is less than the population of region K (see figure 2b). Then we repeated take single census unit back from K to H to make complement of that population gap until the total population that returned is greater than initial population difference (see figure 2c).



***Figure 2a: Initial result with District K and District H having different population.***

***Differences= pop (H)-pop (K)***



***Figure 2b: give 40 from H to K. Figure 2c: return 25 from K to H***

From this example, we can see that the population gap between district K and H has decreased. In GAT, it iterates this process until no more improvement can be made or it reaches the maximum number of iterations that user sets for it.

After exchanging units between districts, GAT will check whether this swapping violates the spatial contiguity constraints. According to **Kim, Myung Jin, it follows three steps:**

1. Randomly select a unit among those swapping units as a seed.
2. Based on this seed, keep adding adjacent units into this seed lists until no units can be added.
3. Calculate the total number of units that has been included into that seed lists (including the seed itself). If its number equals the total number units in that district, then this seed can be swapped.

Finally, objective function is used to evaluate the effectiveness of this algorithm. The objective function used in this paper is as follows:

Where P is the total population, n is the number of district that user want to generate, is census unit’s population and is the ideal population which can be calculated by *P* divided by *n*. Obviously the optimized process is to make the value of objective function as small as possible.

.

The detailed pseudo-code of this GAT algorithm is as follows:

**Input:** A Graph which represent the real world census map; number of regions that need to be aggregated; number of iterations that needs to be run.

**Output:** Aggregated results and objective function value.

Initialization ()

For each iterated solution :( # of iterations):

For each district,put all the district whose pop< target into the list\_low:

Randomly pick one from the list\_low, called LOW\_D:

Put all the districts which adjacent to LOW\_D and whose pop>target into the lists\_high

If there is no such district, return

Else

Randomly pick one from the lists,called HIGH\_D

Calculate the High pop from HIGH\_D

Calculate the Low pop from LOW\_D

Calculate the diff( called diff1)

Find the edges on border(units can be swapped)

uH=Find the border nodes belong to HIGH\_D

uL=Find the border nodes belong to LOW\_D

Swap=FALSE

While (! swap OR uH is not empty) do:

uH\_selected=randomly pick many from the uH

Remove the picked from uH

Calculate the pop from uH\_selected(actuall pop to be added)

While (returned pop<differ1 OR uL is not empty) do:

u=randomly pick one from uL

Remove the picked one from uL

Calculate the gap(gap=returned pop-differ1)

If gap>0:

Add the u into the uL\_selected

If gap<0:

Swap=DONE

If swap==DONE:

Add nodes uL\_selected to uH

Add nodes uH\_selected to uL

Remove nodes uL\_selected to uL

Remove nodes uH\_selected to uH

Reset graph

Valid\_solution()

If validated:

return result

Else

return NONE

***Figure 3d:pseudo-code of GAT algorithm***

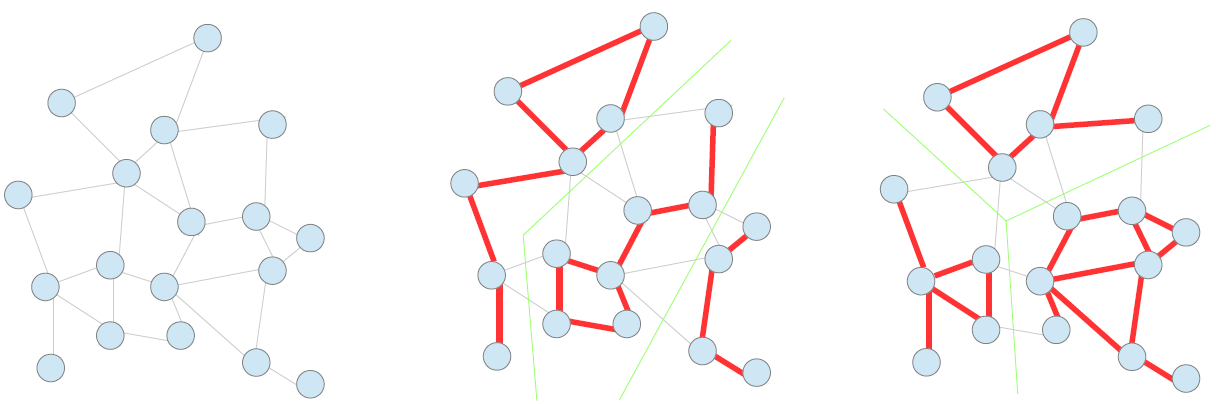
**Solution recombination**

AlthoughGAT algorithm has achieved relatively good results in dealing with redistribution problem, it also has its limitations. Normally, with the number of iteration increasing, the value of objective function will decreases. But it only happens within a certain small range of iterations. When the total number of iteration reaches a certain level, the final result cannot be further improved no matter how many iterations will be added. Sometimes, due to the randomly selection process, the result may even be worse by constantly increasing its number of iteration. That’s one of its limitation. Solution recombination method is designed to solve this problem.

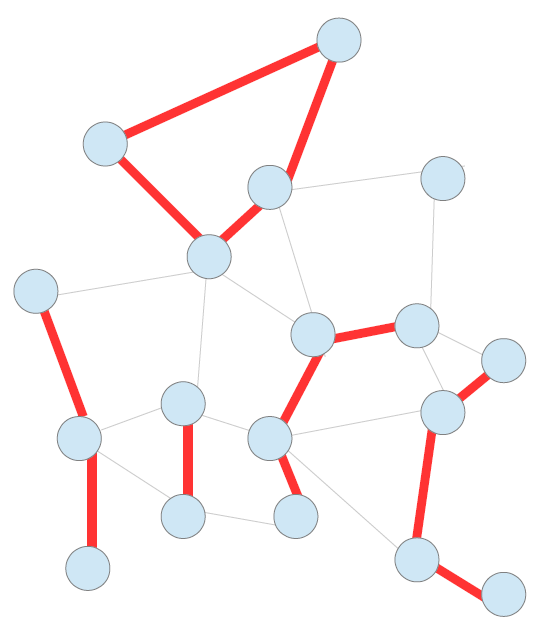
The main reason why the result cannot be further improved after its iteration reaching a certain level is that the overall structure of that aggregation has already reached its local optimal point and there are not much useful units to be swapped between each district. In other words, after reaching to a certain amount of iteration, the district structure has already be stable and any additional iterations won’t change much of its structure.

Just like gene evolution, it cannot be evolved into even better gene without incorporating different types of other gene, this aggregated structure cannot be further improved without taking advantages of different other optimal solutions. Solution recombination is such method which simulates evolutionary process and try to combine two already existing solutions into a new one which is better than by purely increasing the number of iterations.

Suppose we have two solutions for the same census map (see figure 3a) and we want to combine these two solutions into one and apply GAT again to produce an even better solution which GAT itself cannot do. What I did is to find common edges of these two solutions and these edges are kept while the rest of them are discarded. After this, I have a new graph whose number of subgraphs is greater than the number of districts we want to aggregate (see figure 3b). To construct a valid solution, we always merge the subgraph with the smallest total population to its neighbouring subgraph with the smallest population. After this procedure, the new graph has incorporated useful information both from itself and outside graph. Then GAT is applied to this merged solution to produce an even better solution. Detailed experiment and analysis will be elaborated in chapter 3.



***Figure 3a: Initial graph and its two different solutions, both of which have three subgraph. Each of consecutive part is considered as one subgraph.***



***Figure 3b: Subgraph that after finding common edges. Now it has five subgraph which is larger than three. Need to repeatedly find two subgraphs with least total population to merge until the total number of subgraph is equal to three. Then GAT can be applied to this valid merge solution.***

**Multi-solutions and Parallel computing technology**

The methods used in this part is to solve another limitation of GAT that it can only produce one good solution at a time. In order to produce many other good solutions, user has to run the program multiple times. This is not a big issue when dealing with small dataset, but the negative effect of high computation time becomes more obvious when dealing with large dataset. Multi-solution combined with parallel computing technology can be used to generate many good solutions at a time which can greatly reduce computation time.

Before digging into details of this idea, I will first introduce some prerequisites. First, a concept of Pool is introduced. Pool is a container that includes many solutions (can either be initial solutions or optimal solutions) at the same time. The GAT algorithm as well as solution recombination will be applied directly to this pool. Second, in order to let parallel computing technology work, the program has to be run in a multicore computer.

There are a few advantages to use pool to hold multiple solutions and let program to deal with pool directly instead of dealing with a single solution like traditional GAT. First, since pool can hold many solutions at the same time, when applying GAT algorithm combined with parallel computing to optimize it, user can get a pool of good results instead of just one good result. Besides, since we use parallel technique, many solutions can be generated and optimized in a parallel way such that the total amount of time it takes to generate many results is almost equal to the previously time that generates one result. Of course, time will sometimes be longer due to parallel computing communication and the limitation on number of cores a computer has. Thirdly, since we have a pool of solutions, information on different solutions is abundant. When applying to solution recombination, there are a lot of new candidate solutions can be selected to recombine, thus getting more chance of generating even better result. Of course, this also take the risk of generating bad result but in this case we can discard them. The later sensitive experiment will show that occurrence of good result occupies 60 to 80 percent of all results compared with bad ones which we don’t need to discard them and recalculate. Thus overall, this procedure still produce optimized pool of result while saving much computational time.

In summary, this process is a good simulation of evolution process: each solution is considered as single unity of the same species. The quality of them can either be good or bad. But a variety of them with different kind provide abundant source to generate all kinds of offspring which good quality remains while bad quality unity will be eliminated. In this case, after a certain period of time, only species with good quality remains.

The detailed workflow for combining all these three methods is described as follows: First, I generate a pool of initial random solutions using parallel computing technique. Then solutions in this pool are randomly chosen to be recombined and then GAT is used to improve the recombined solutions. Solutions in the pool will be replaced by better solutions which are generated through these process. Note that parallel computing is also used in this part which means multiple recombined process as well as GAT improvement can be done at the same time. The figure 4a demonstrate the whole process.



***Figure 4a: One process from beginning to end.***

**Experiment and analysis**

[1] Martin, David. "Geography for the 2001 Census in England and Wales."*Population Trends* 108 (2002): 7-15.

[2] Assunção, Renato M., et al. "Efficient regionalization techniques for socio‐economic geographical units using minimum spanning trees." *International Journal of Geographical Information Science* 20.7 (2006): 797-811.

[3] Kim, Myung Jin. "Optimization Approaches to Political Redistricting Problems." PhD diss., *The Ohio State University*, 2011.

[4] Bacao, Fernando, Victor Lobo, and Marco Painho. "Applying genetic algorithms to zone design." *Soft Computing* 9.5 (2005): 341-348.

[5] Xiao, Ningchuan. "A unified conceptual framework for geographical optimization using evolutionary algorithms." *Annals of the Association of American Geographers* 98.4 (2008): 795-817.

[6] Densham, Paul J., and Gerard Rushton. "A more efficient heuristic for solving largep-median problems." *Papers in Regional Science* 71.3 (1992): 307-329.

[7] Cova, Thomas J., and Richard L. Church. "Contiguity constraints for single‐region site search problems." *Geographical Analysis* 32.4 (2000): 306-329.

[8] Önal, Hayri, and Robert A. Briers. "Incorporating spatial criteria in optimum reserve network selection." *Proceedings of the Royal Society of London B: Biological Sciences* 269.1508 (2002): 2437-2441.

[9] Shirabe, Takeshi. "A model of contiguity for spatial unit allocation."*Geographical Analysis* 37.1 (2005): 2-16.

[10] Shirabe, Takeshi. "Districting modeling with exact contiguity constraints."*Environment and planning. B, Planning & design* 36.6 (2009): 1053.

[11] Bennett, David A., Ningchuan Xiao, and Marc P. Armstrong. "Exploring the geographic consequences of public policies using evolutionary algorithms."*Annals of the Association of American Geographers* 94.4 (2004): 827-847.

[12] Openshaw, Stan, and Liang Rao. "Algorithms for reengineering 1991 Census geography." *Environment and planning A* 27.3 (1995): 425-446.

[13] ALVANIDES, S., OPENSHAW, S. and REES, P., 2002, Designing your own geographies. *In The Census Data System*, P. Rees, D. Martin and P. Williamson (Eds), pp. 47–65 (Chichester, UK: Wiley).