Congressional District Partitioning: A Two-Stage Approach

Team #2048

COMAP Mathematical Contest in Modeling

February 8-12, 2007

Contents

Summary

We approach the problem of congressional redistricting with a two-level model operating first at the statewide level, and then at the countywide level on some counties. At the statewide elvel, we consider connectivity, total population, and minimization of the number of counties split between districts to be of primary importance in creating "simple" congressional districts. To optimize over these values we use a modified graph partitioning algorithm on a graph representation of the counties. Those counties which must be divided between districts are passed to a county-level model.

In the state-level model, we use a divide and conquer algorithm in which district allocations are handled by recursively splitting the graph into two pieces of about the same population. When no such partition exists, it will split the largest counties along the cut it is making, treat each half of the split counties separately, and continue.

Within counties which must be split we also consider roundness and population density along district lines. We minimize a *badness* function representing a linear combination of inverse values for all criteria we use for simplicity, by using a simulated annealing approach on a square grid of population densities. Initial seeding of the grid into the desired number of districts is done in a greedy manner. Then, grid squares bordering other districts are chosen at random to be given to neighboring districts. Changes which lower the badness are accepted while changes raising the badness are accepted at a rate exponentially decaying with how detrimental their effect is.

We find that our algorithm gives satisfactory results at both model levels. On a map of New York State, the state-level model finds a partition into 29 districts that only divides four counties unnecessarily. Additionally, the districts created by the model tend to be coherent and have relatively high roundness. At the county level, we find that our minimization criteria tend to avoid splitting up cities, and create high roundness shapes as expected. The county model is also faster than any approach found in the literature while considering a finer level of detail.

All in all, our two-stage approach is simple yet effective, and the district plans it creates would be much easier to explain to voters than the current district layout.

1 Problem Restatement

The current rules for partitioning a state into federal congressional districts require only that each district in a state contain approximately the same number of people. This leaves open the possibility of political manipulation in the districting process, often leading to complex district shapes which voters find unsatisfactory. We therefore ask, what makes a simple district? What does an ideal set of "simple" districts within a state look like?

We seek a solution which, in addition to answering these questions, is a satisfactory and utilitarian solution for voters. We seek to define an optimal district in a manner which

- Can be described to voters in a few simple sentences,
- Is simple to describe and implement,
- Can use data at arbitrary levels of detail to produce results of arbitrary precision,
- Creates "simple" district shapes, and
- Is unlikely to cause implicit gerrymandering.

2 Optimal District Criteria

What makes a partition of a state into districts a good one? We list several criteria which are generally considered to be desirable, with the intent of providing specific metrics later.

- **Districts are simply connected wherever possible and don't have any holes**. Mathematically, the districts are homeomorphic to the unit disk. The only exception to this requirement is when the state in question is non-contiguous. In that case, a district may be made up of subsets of non-contiguous parts of the state.
- **Districts contain roughly the same number of voters.** The U.S. Supreme Court considers a deviation of up to 10% between districts as being "essentially equal" [1]. Following this, we require that the populations of each of our districts to be within 5% of the ideal value, which is the total population of the state divided by the number of districts.

- **Few counties are split between multiple districts.** Counties are a natural geographic separator, and can act as an easy way for most people to tell which district they live in.
- **Districts have high roundness.** We wish to find a districting plan such that the component districts are as round as possible.
- **Intra-county district lines fall along less populated areas.** We wish to ensure that no matter where you live, it is likely that your neighbors are in the same district if they live in the same county.

We will have to balance these criteria, and assess which are the more important ones.

3 General Definitions

Before we can define the optimality criteria in detail, some common definitions are needed.

Definition (State). *A* state $R \subset \mathbb{R}^2$ *is the union of one or more simply connected regions and in this problem must be completely divided into districts.*

In most cases, a state is a single contiguous region. Throughout this paper, we will be particularly interested in the case of New York State.

Definition (Population Density). *The* population density *function* $\rho(x, y)$*, takes all of* R *as its domain, and is the measure of the average number of voters per unit area.*

It follows that the total population of a state can be written as

$$P = \int \int_R \rho(x, y) dA \tag{1}$$

The population density in New York State ranges over four orders of magnitude, from 3.1 people per square mile in Hamilton County, to just under 67,000 in the heart of New York County. [2]

Definition (District). *Consider a partition* $\{D_i\}$ *of* R *into* N *mutually disjoint subregions. Each of the* D_i *is called a* district.

Definition (Mean District Population). *The* mean district population *(MDP) is the total population of the state divided by the number of districts desired.*

The number of districts, and thus representatives, per state is determined by a Congressional Appointment algorithm with which we are not concerned. The number of voters per district varies, but is usually in the range of 600,000-700,000 people [3].

The number of voters in a district should be roughly equal to the MDP. The number of people per district can be written as

$$\int\int_{D_i} \rho(x,y)dA \simeq MDP \tag{2}$$

Definition (County). *Each state R has a distinct partition into M mutually disjoint regions C_i. Each is called a* county*.*

The partition of a state into counties is treated as constant for the purposes of this problem, which makes counties a useful geographical identity for voters. In general, there are more counties than districts.

Definition (Gerrymandering). Gerrymandering *is the partitioning of a state into districts in a manner likely to favor the election of a particular party or candidate.*

In most cases, gerrymandering is done to maximize the number of state seats in the House of Representatives for a given party. This can be done by lumping voters that prefer opposing parties into "throwaway" districts, so as to minimize their representative voice. The term is often associated with strangely-shaped districts.

4 Model: A Two Stage Approach

We consider county lines to be a preferable delineation between districts, whenever feasible under the criteria of equal population. It is thus natural to use a two stage modeling approach, in which we allocate counties to or among districts in a State Level Districting Algorithm (SLDA), creating *remainder counties*, which are optimized using a County Level Districting Algorithm (CLDA). The *state badness* is minimized or nearly minimized in the state level optimization, and the *county badness* is minimized or nearly minimized during county level optimization.

4.1 Defining Optimality

We define optimal solutions in terms of *badnesses*, and seek to minimize some combination of them. Some of our metrics for "badness" apply only

at the county level[†], and others apply at both simulation levels.

- **Districts are simply connected.**

  Definition (Connectedness Badness B_{con}). *The* connectedness badness B_{con} *is a measure of a district's connectedness within a county.*

$$B_{con} = \begin{cases} \infty & \textit{if the district is not simply connected;} \\ 0 & \textit{otherwise.} \end{cases} \tag{3}$$

- **Districts contain roughly the same number of voters.**

  Definition (Population Badness B_p). *The* population badness B_p *of a district is an infinite potential well centered on the ideal district population for the state, such that*

$$B_p = \begin{cases} \infty & \textit{if } |\Delta P_{district}| \geq \alpha(MPD); \\ (\Delta P_{district})^2 & \textit{otherwise.} \end{cases} \tag{4}$$

  where ΔP is the deviation of total district population from the MPD, and α is the absolute limit on population deviation (generally taken to be 5%, as previously mentioned).

- **Few counties are split between multiple districts.**

  Definition (Cutting Badness B_{cut}). *The* county-level cutting badness *is defined as the number of districts over which that county must be split. The state-level cutting badness B_{cut} is this value summed over each county.*

- **Districts have high roundness.**

  Definition (Eccentricity Badness B_e). *The* eccentricity badness B_e *of a district within a county is the ratio of district's perimeter to its area.*[‡]

$$B_e(district) = \frac{Perimeter(district)}{Area(district)} \tag{5}$$

- **District lines fall along less populated areas.**

[†]Or to the subset of a district within a county.

[‡]This is similar to *sphericity* in mathematics, and the inverse of the *hydraulic radius* in fluid dynamics, but is not quite equivalent to either.

Definition (Boundary Badness B_b). Boundary badness *acts as an approximation of the number of voters living near a district division line. The boundary badness B_b of a district within a county is the path integral of population density $\rho(x, y)$ over the boundary C of the district,* when that boundary does not fall on a county line.

$$B = \int_C \rho(x, y) ds \tag{6}$$

The **badness** metrics above are appropriate at different times.

- The total **state-level badness** depends only on how many counties are divided between multiple districts, assuming the created districts are contiguous and have the appropriate population level.

  Definition (State-Level Badness B_{state}).

  $$B_{state} = \sum_{districts} (B_{con} + B_p) + \sum_{counties} B_{cut} \tag{7}$$

  In practice, as long as the population is within the given tolerance, and districts are formed by a contiguous set of counties, we are really only interested in minimizing the last term, and can treat the first two as constraints when considering possible divisions on the state level.

- Given a number of districts between which to divide a county, the **county-level badness** depends on the shape, boundary lines and population associated with those districts.

  Definition (County-Level Badness B_{county}).

  $$B_{county} = \sum_{districts} (\hat{C}_b B_b + \hat{C}_p B_p + \hat{C}_e B_e) \tag{8}$$

  The constants $\hat{C}_b$, $\hat{C}_p$, and $\hat{C}_r$ define the relative importance of each badness to the total county badness. We will choose them so that the contributions from each badness are of the same order.

We will tackle these problems one at a time, first dividing up counties into districts while ensuring that (7) is minimized, and then minimizing (8) on the county level.

4.2 County Graph

To divide the state into districts along county lines with a rigorous algorithm, we will need a mathematical description of the counties within the state.

Definition (County Graph). *Consider an undirected graph such that each county corresponds to a node with weight equal to that county's population. An edge exists between two nodes if the counties share a common border. The graph created by this mapping is known as the* county graph*.*

4.3 Population Density

We would like to model the population density $\rho(x, y)$ within a county in a way that is straightforward but captures the salient features of dominating, high-density cities and low-density countryside. Our model depends on the following assumptions.

- **Urban population can be concentrated into radially symmetric cities**. This assumption is supported by common familiarity with large cities.
- **Countryside population is uniformly spread out**. We feel comfortable ignoring the subtle fluctuations in population around small villages and so forth.
- **Urban population center density functions are Gaussian distributions**. In particular, the "radius" of a city varies positively with its population.[†] We will revisit the ramifications of this assumption below.
- **Only large cities matter**. To model this fact, we introduce cities in decreasing order of population, until over half the population of the county is represented. We note that more or less cities can be added, on a county-per-county basis.
- **Urban counties have a constant population density**. In particular, the counties that make up boroughs of New York City are too densely packed and too small to have anything other than a constant population density function.

[†]There are more complicated models in the literature, but "positivelty" is good enough for now.

Applying the above, let the **Gaussian population density function** of a city c be described by a position $\vec{r_c}$, an amplitude A_c, and Gaussian parameter σ_c. Let ρ_b be the background population density in the countryside.

Definition (Population Density Function). *The value of the* population density function *at any point $\vec{r}$ within a county is given by*

$$\rho(\vec{r}) = \rho_b \ + \sum_{c \in \ cities} A_c e^{-(\vec{r}-\vec{r_c})^2/\sigma_c^2} \tag{9}$$

We now confront the correlation of "radius" with population. For ease of algebraic simplicity, let the "intensity" of the Gaussian profile vary as the third root of the population. The total population contribution to a county from a city is given as

$$\int_{\theta=0}^{\theta=2\pi} \int_{r=0}^{r=\infty} A_c r \ e^{-r^2/\sigma_c^2} \ dr d\theta = \sigma_c^2 \pi A_c = P_c \tag{10}$$

If $A_c = K(P_c)^{1/3}$, then algebraic manipulation and sampling of a few cities in New York according to 2000 U.S. Census data [2], gives

$$A_c = K(P_c)^{1/3}, \quad \sigma_c = \frac{(P_c)^{1/3}}{\sqrt{\pi A_c}}, \quad K = 42 \ \frac{\text{people}^{2/3}}{\text{mile}^2} \tag{11}$$

This is enough to fully describe a city in terms of its population alone, and is good enough for our purposes.

5 State-Level Division Algorithm (SLDA)

In the first stage of our solution, we allocate counties to districts such that each district satisfies the population constraint. The output of this step is a list of districts, each of which contains a number of counties or county pieces.

SLDA is a divide-and-conquer algorithm. It keeps on breaking the county graph into two pieces of equal population and assigns half of the districts to each. At some point, it will either get down to one district, which has to contain all of the counties in its graph, or one county, which has to be split into the appropriate number of districts.

At any given step of the algorithm, we have a subset of the counties which form a connected subgraph G of the county graph, and we have a certain number of districts D to divide them into. SLDA attempts to find a subgraph G' of G satisfying the following conditions:

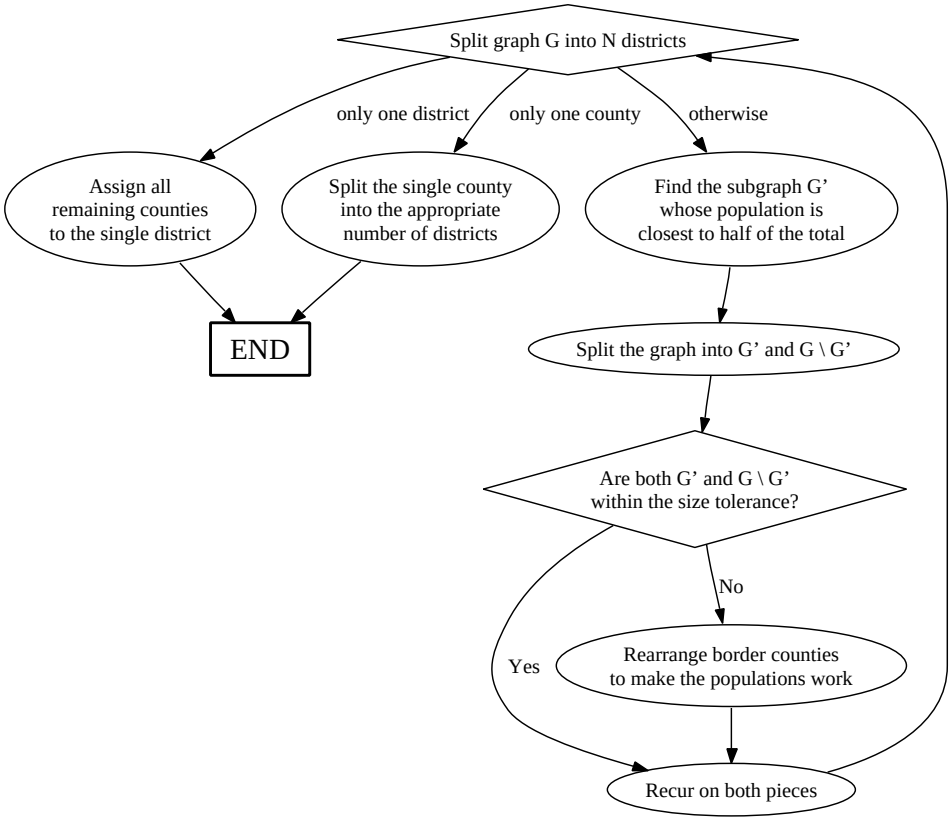

Figure 1: A flowchart representation of SLDA, which is a divide-and-conquer algorithm. The base cases are when there is only one district or only one county. Otherwise, the graph is split into two pieces of approximately equal population. If necessary, border counties are rearranged. Then SLDA is run on both halves recursively.

- G' is connected
- $G \setminus G'$ is connected
- The total population contained in G' falls between 95% and 105% of (MDP * $\lfloor \frac{D}{2} \rfloor$).
- The total population contained in $(G \setminus G')$ falls between 95% and 105% of (MDP * $\lceil \frac{D}{2} \rceil$).

It is not always true that such a split exists. In this case, a county must be split, with parts of it assigned to both halves. The algorithm will find the split that is closest to meeting the population constraints, and determine how much population it needs to gain or shed to have it and its complement meet the constraints. In this situation, the border counties are shuffled and split between the two halves until both satisfy the population conditions. Any counties that are split in this manner are treated as independent counties from then on; they just happen to be part of the same physical county. [†]

Once a split meeting the constraints is constructed, the algorithm is called recursively on each subgraph (G' and $G \setminus G'$), assigning each of them half of the districts.

There are two base cases. If there is only one district to construct, then it must contain all of the counties given to it. Similarly, if there is only one county, then it must be split into the appropriate number of districts. Finally, to construct a solution, the procedure is initially run on the entire county graph for the desired number of districts.

5.1 Valid Splits

The previous discussion has glossed over the issue of finding a valid split (or, if none exist, a split which is closest to meeting the constraints). This is an instance of the graph partitioning problem, which is NP-Complete, so a heuristic-based approximation must be used.

Each county in G is used as a possible "seed" for G'. At each step, one county adjacent to the current G' is added to G', so long as its removal would not disconnect $G \setminus G'$. This process stops when the population of the current G' is too large. The constructed G' whose population is closest to the ideal value (MDP * $\lfloor \frac{D}{2} \rfloor$) is then chosen to be the split.

[†]It is still not necessarily possible to meet the population constraints of both G' and its complement simply by moving around population on the border. Our implementation of SLDA just gives up and reports a failure. More sophisticated algorithms for population transfer could get around this issue.

The state space of this problem is quite large and has a very large branching factor. This branching factor is associated with the selection of a county to add to the current G'. The current implementation solves this problem by selecting any random adjacent county that leaves the $G \setminus G'$ connected. This causes a problem in that solutions found by this algorithm may result in many more (possibly avoidable) county splits. To get around this, multiple possible solutions are generated and the best sent is to the county-level division algorithm for evaluation.

6 SLDA Analysis

The results of SLDA are surprisingly good considering the algorithm's greedy nature. On small test cases (containing ten to twenty counties), the algorithm failed to find a valid solution about 30% of the time. When successful, however, the following trends appeared:

- **The majority of the counties are not split**. In any given districting plan, about half of the districts consist entirely of unsplit counties.
- **Few counties are split more than once**. Except for counties whose populations support more than two districts, only one example of a county being split into more than two pieces arose in tens of test runs. This is most likely because border counties are considered in descending order of size, so a county must be very large to be split twice.

If N is the number of counties and D is the number of districts to split the state into, then the runtime of this implementation of SLDA is $O(N^3 \lg D)$ in the worst-case. The $\lg D$ factor comes from the recursive bisection of the graph, and the N^3 factor comes from the N attempts to form a valid split, which in this implementation takes at most $O(N^2)$ steps each.

6.1 Optimality of Solution

In general, it is difficult to compare the SLDA solution to the optimal solution for any non-trivial case because the solution space is so large that it is infeasible to completely search it. We provide one example of a fairly trivial case in which the SLDA finds the optimal solution. This also acts to demonstrate the strengths and weaknesses of the county graph representation used. In a single test run, the solution which was found, while optimal

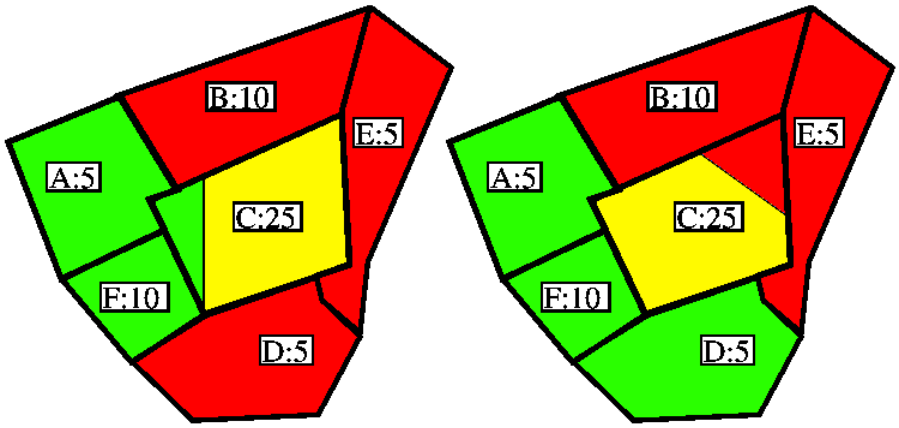

Figure 2: Two optimal solutions for a 6 county system being partitioned into 3 districts. Because it is only based on county connectivity, the SLDA tends to find these solutions with equal probability. It is clear, however, that the solution on the right has a lower eccentricity badness, making it preferable.

when trying to minimize the number of counties to split, it does not take into account eccentricity badness. Another solution which also only splits one county, but has much lower eccentricity badness, is shown in figure 2.

SLDA is an instance of a recursive bisection algorithm, which has been used to solve similar problems, such as the p-way graph partitioning problem. That problem is NP-Complete, but on planar graphs, a recursive bisection approach has a worst-case approximation ratio of $\Theta(\sqrt{n/p})$, where n is the number of nodes in the graph and p is the number of partitions. [4] A similar result holds for the SLDA.

6.2 Strengths

- **Fast.** Each run of the algorithm takes less than one second to complete even on relatively modest hardware. Since the algorithm is non-deterministic, this means that many runs can be completed in a short amount of time and then the best results can be extracted.

- **Low Number of Splits.** The number of counties that are split is relatively low, and most counties that are broken up are split into only two pieces. This means that voters will in general have an easier time

determining which district they belong to, as compared to existing district maps. This is discussed later.

- **Intuitive.** The algorithm is fairly easy to grasp and to explain – the "divide" step of the divide-and-conquer algorithm involves breaking the graph into two equally sized halves, and the "conquer" step is trivial.

6.3 Weaknesses

- **High Failure Rate.** When running this algorithm on small hand-generated test cases, it gave up and reported failure a significant portion of the time. That said, this is by no means a show-stopper because of the extremely fast execution time. If it fails, rerun it until it succeeds.

- **Ignores Geography.** The SLDA does not take into account any geographical properties of the counties or districts, and as such completely ignores the shape of any districts it creates. This means that the districts it creates may be simply connected and ugly at the same time. In practice, districts created by the algorithm tend to be fairly compact even though there is no explicit constraint to this effect.

7 County-Level Division Algorithm (CLDA)

The county level division algorithm (CLDA) begins by discretizing the continuous problem to a rectangular grid. It attempts to arrive at the optimal solution through a **simulated annealing** algorithm. Specifically, it divides a county into some number of **district remainders**, which are fractions of districts allocated from this county during the SLDA. The sum of the badnesses contributed from all district remainders is minimized.

First, a county is divided into equally sized **grid squares** on an n-by-m rectangular grid.[†] Each square is assigned a population value using our population model in Section 4.3, evaluated at the center of its location.

District remainders are represented as a list of points on their borders, and a count of the total population and area in their interior. Each is updated in a differential manner every time the ownership of a grid square changes.

[†]In running time and space estimates, we will take n to mean $max(m, n)$.

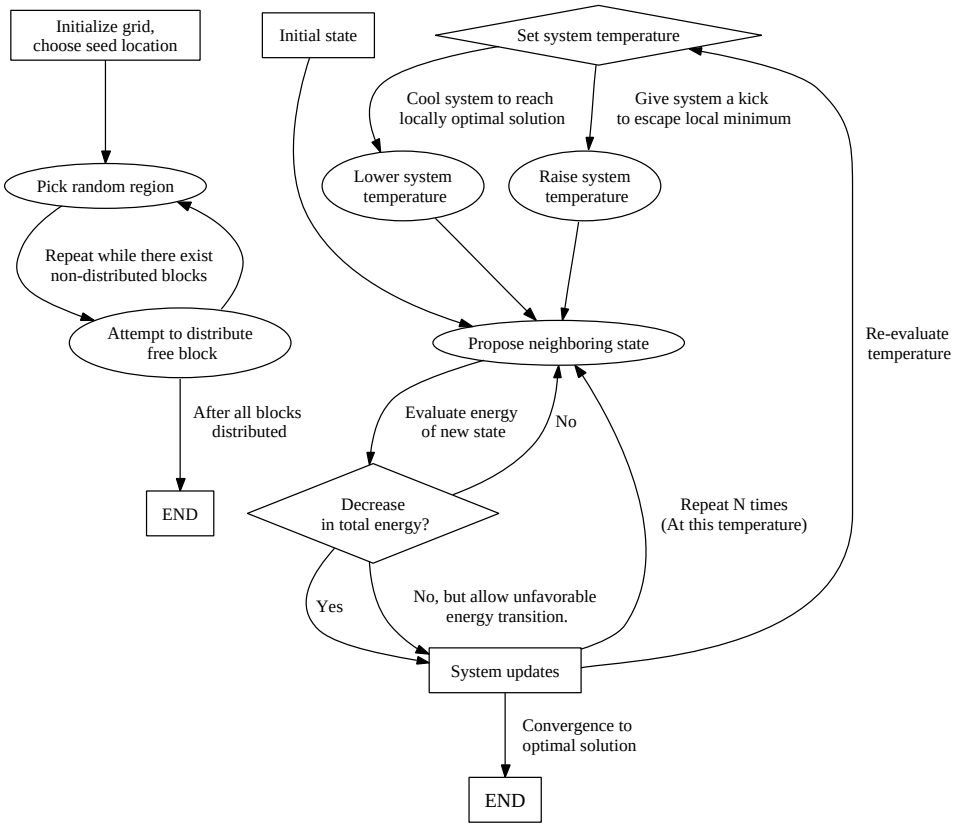

Figure 3: A flowchart representation of CLDA, a simulated-annealing process. Ideally, a solution close to the global optimum is reached.

7.1 Simulated Annealing

The simulated annealing process, described in Ingber [5], proceeds by slowly relaxing the system into a **low badness state**. The entire process is better visualized in Figure 3.

- A **temperature** T is chosen for the system which represents the system's ability to enter high-energy states.
- At each iteration, a district and neighboring square S are chosen at random.
- The change in the sum of the **badness** values of every district in the county ΔB_{county} resulting from assigning S to its new district is calculated.
- This change is accepted with probability

$$p = \begin{cases} 1, & \Delta B_{county} \leq 0 \\ e^{\frac{-\Delta B_{county}}{T}}, & \Delta B_{county} \geq 0 \end{cases} \tag{12}$$

- The process repeats. Over time, T is gradually lowered in an attempt to get the system to settle into a low badness state.

Under simulated annealing, systems will tend towards low badness states. In general, when T is high, the system is given a chance to escape its current potential well. As a final step, running the simulation with $T \simeq 0$ ensures that only changes which strictly improve the county **badness** are made. This greedy approach at the end of a simulation run ensures that the state will move towards the bottom of any well it is in.

7.2 Initial Division

An initial division of the grid into variable-population districts is accomplished with an organic growth approach, as diagrammed in Figure 3.

- For each district, a single grid square is designated the **entry point** into the county.
- A district is chosen at random to gain a neighboring grid square. Which district is chosen is weighed towards the proportion of this county's population that the district needs.

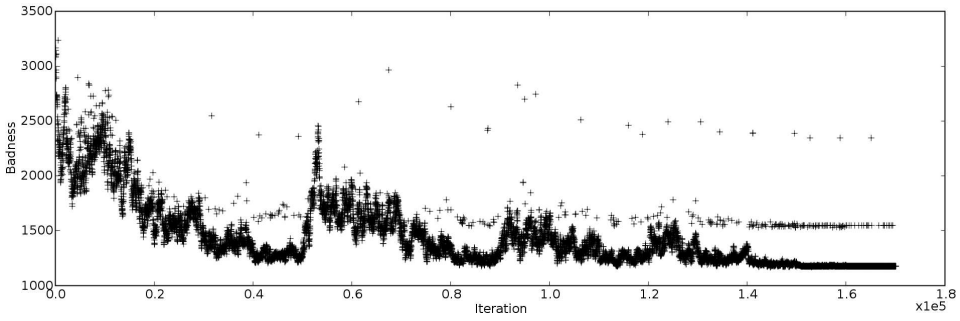

Figure 4: Plot of *badness* metric versus number of iterations during a typical run of the CLDA. Every thousand iterations, T either decreases incrementally or gets a "kick". The lowest *badness* is generally reached at the end of the process.

- If the grid square is not owned by any other district, it is assigned to this one. The district internally updates its area, perimeter and various associated **badnesses**.
- The process iterates until all grid squares have been claimed.

Since the process is random, each district seems to grow organically outwards from its **entry point**, meeting other districts somewhere in the middle (as dictated by their relative population needs). Since every grid square is reachable by some chain of neighboring squares, the probability that the grid is not filled goes to zero as the number of iterations is increased.

7.3 Temperature Schedule

We allow the system to remain at a given temperature for $O(n^2)$ terms so that, on average, every possible exchanging of border squares is proposed $O(n)$ times. This allows each border square to move enough times to travel across the screen $O(1)$ times.

According to Hajek [6], the **temperature schedule** required for **weak ergodicity** is given by

$$T(I) = \frac{T_0}{\log(1+I)} \tag{13}$$

This method is meant to model the natural cooling process of an object, with T_0 setting the scale of typical changes in **badness**. In particular, T_0

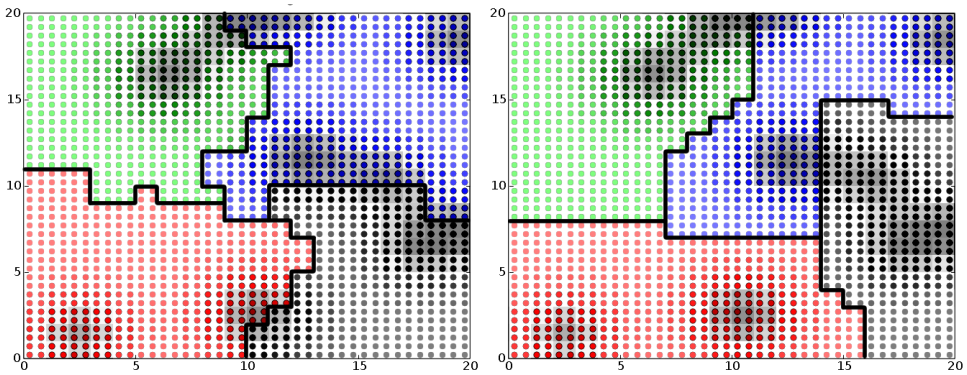

Figure 5: Typical result of the CLDA. The initial seeded state and the eventual annealed state are the left and right diagrams, respectively. Note the prevalence of straighter lines and decreased divisions of highly populated areas (cities).

must be high enough to explore all regions of the **state space** towards the beginning of the simulation. This implies that changes in T_0 between annealing simulations, when T_0 is sufficiently high, will have little noticable effect.

In our model, we used a change in temperature over time $\frac{\Delta T}{\Delta t}$ approximating the one proposed in (13), but we did not explore the optimal value of T_0.

8 CLDA Analysis

CLDA attempts to find a minimum which is not susceptible to improvement by large or small deviations. In practice, it does this accurately and quickly, with marked decreases in the **county badness** (a quantitative metric) and marked improvements in the visual shape of the counties (a qualitative metric). Note that:

- At no point during our simulation testing did the CLDA fail to decrease the **county badness** by a significant amount. An example of the rate at which badness decreases is shown in Figure 4.
- The divisions produced by iteratively running the CLDA results in **intuitively correct districts**. They tend to avoid cutting through

cities, preferring straight lines to jagged, winding paths and preserving the relative population. An example on a simulated six-city county is shown in Figure 5.

Our algorithm benefits significantly from design choices made during implementation. In particular, we maintain a list of border squares and are able to select from them at random without considering region interiors. Every border square therefore has the chance to be swapped into a different region on the order of every $O(n)$ iterations. Since the furthest a border square can travel to reach an optimal solution is n grid squares, our algorithm can make the solution settle to a minimum in $O(n^2)$ iterations.

We know of a similar attempt to use an energy-minimization approach to county districting. Chou [7] uses a Potts-inspired model to district Taipei city over a grid. However, the model he uses must iterate through $O(n^2)$ grid squares at random in order to swap border squares an average of once each, and thus requires $O(n^3)$ iterations before reproducing the optimal solution.

8.1 Optimality of Solution

We make no attempt to compare this model to the optimal solution found through complete enumeration of the solution space. Instead, we claim that the CLDA produces "close to optimal" results through a probabilistic argument.

As the time spent cooling increased, the chances of enumerating all meaningful states approaches one, making the system weakly ergodic. Theoretically, the number of iterations required to claim that the model explores a large fraction of the state-space is quite large, but because of the speed with which our model runs (about a thousand iterations per second for a thirty-by-thirty grid) and the concave-up nature of the badness curves over time, we can say that there is a high probability we reach a near optimal badness.

An excellent example of an optimal solution can be seen in Figure 6. The boundary lines, after simulated annealing, are the optimal solution *within the grain-size of the problem.*

8.2 Strengths

- **Efficient**. As mentioned above, our algorithm requires $O(n^2)$ iterations to reach equilibrium. The closest competitor from the literature requires $O(n^3)$. This is a significant improvement.

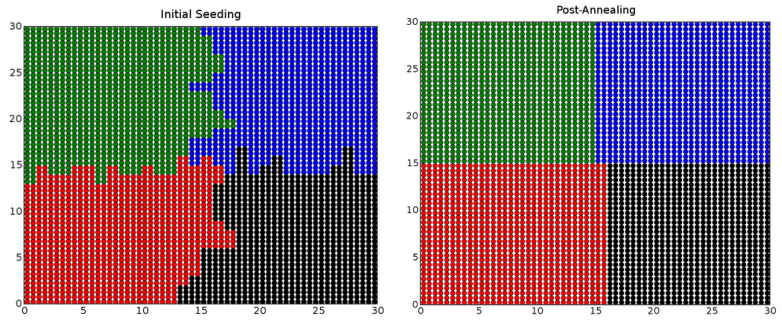

Figure 6: An example of our simulated annealing process for four districts over a constant population density county. Ten thousand iterations were performed, with the system "temperature" decreasing from 2.0 to .5.

- **Quick Badness Computation**. Badness is updated differentially. Any time a change in districting is made by swapping ownership of a square, the differences incurred on B_{county} are incurred on the total. This makes each iteration run in $O(1)$ time.
- **Roughly Ergodic**. By design, our system explores significant portions of the state space, and can later revisit optimal state points.
- **Easy to Augment**. It is easy to add more or different badnesses.

8.3 Weaknesses

- **Complex to code**. While the algorithm is quite simple to describe, the specific implementation is quite complex, taking over 1000 lines of code to implement. In contrast, a Potts model implementation is much simpler.
- **Anisometric**. By using a square grid, we have made certain directions more favorable than others. On the bright side, this helps districts follow latitude and longitude lines.
- **Statewide Badness Not Considered**. Though we seeded initial divisions in locations close to county lines, we made no effort to make sure districts were contiguous on the *state level* (though they certainly were on the *county level*).

9 Implementation

9.1 State Level Division

The state-level division algorithm was implemented in C++, using the features of the C++ Standard Template Library to ease the implementation while still ensuring a fast runtime. The code was written using a straightforward doubly recursive approach: one level was for the main partitioning algorithm and the lower level was for finding the best split, used as a subroutine of the first level.

When compiled with maximum optimization, each execution of the algorithm on the New York State data set took about a quarter of a second to run, whether successful or not. The code was run on a 2 GHz Athlon 64 machine with 2 GB of RAM.

9.2 County Level Division

We implemented the county-level algorithm in Python, using the Scientific Python and PyLab libraries as a computational heart and display utility, respectively. We wrote some thousand lines of code with an object-oriented design philosophy. As regions grew and collapsed, they modified their internal representation to keep track of changes in the three types of *badness*, which depended on the perimeter, area, fractional population and border population density.

The runtime was quick: a ten-thousand iteration annealing routine performed on four districts fighting over a grid of granularity thirty-by-thirty took just a few seconds to run on a 1280 MB RAM, 1.7 GHz Pentium M machine running Ubuntu Linux 6.06. An appendix of the more important parts of our code has been included at the rear of this document.

10 New York State Results

10.1 State-Level

For the purposes of the state-level algorithm, the only data necessary are the population of each county, adjacency lists showing which counties border each other, and the number of desired districts. Population data was taken from the 2000 US Census and border data was entered from a New York State county map.

SLDA was run several times on the New York State data set until the first success, which is roughly shown in Figure 7. The majority of counties

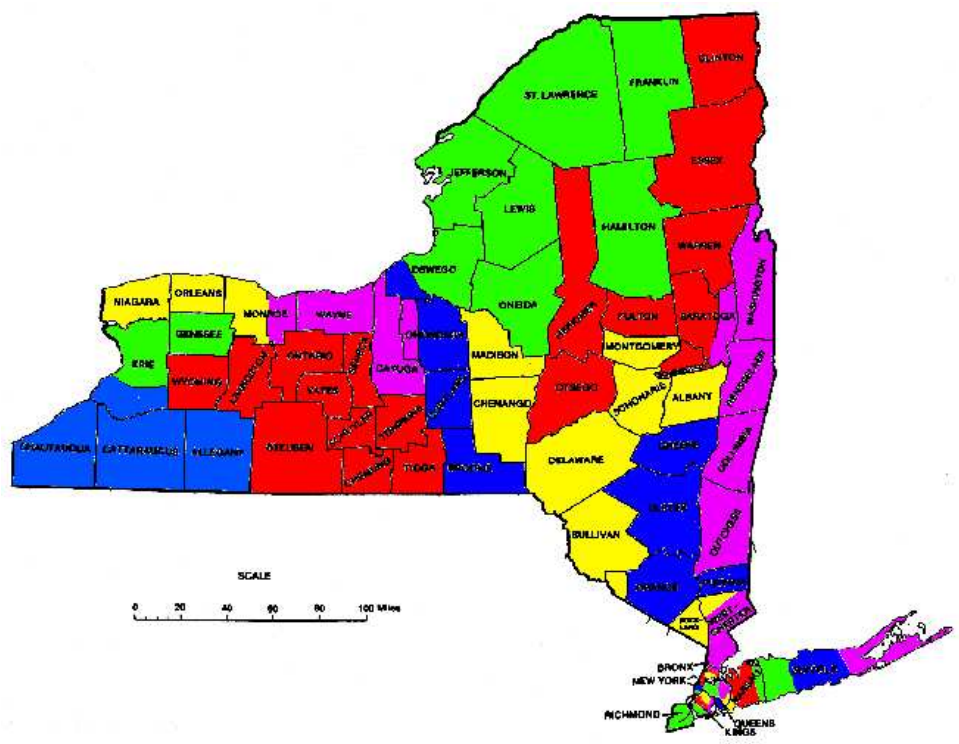

Figure 7: The map initially produced by the SLDA is shown here. Adjacent counties that have the same color are part of the same district. Counties that are split have more than one color; the area on each side of the split roughly indicates the proportion of the population assigned to each district.

were not split at all and therefore did not need any additional processing. The counties that were split were sent to the county-level algorithm along with the relative sizes of the pieces.

10.2 County-Level

The state-level solution described above split thirteen counties between districts. The number of districts ranged from two to four. To model the county population density distribution, we projected the county onto a rectangle of equal aspect ratio and equal surface area. We used [8] to measure the rough height and width of each county. Then, starting with the biggest cities, we placed Gaussian distributions on the rectangular maps until more than half

Figure 8: The current districting in New York State. We see that our proposed redistricting offers a large improvement.

of the county's population was represented. The rest became background population density ρ_b and was evenly distributed in the county. We used [9] to obtain total county population, city population and county area data. An example of the simulation is given in Figure 9.

We used the seeding procedure described above to come up with an initial division, starting each seed closest to the county line to which the district in question was connected. Then, we used the simulated-annealing algorithm to optimize the *badness* of the divisions, with several "kicks" along the way. An example of the resulting division is given in Figure 10.

11 Sensitivity Analysis

11.1 SLDA Sensitivity to Population Deviation

We can map both the failure rate of the algorithm and the number of split counties against the allowed deviation in population between districts. The results are shown in Figure 11.

As the tolerance increases, the failure rate and number of split counties required both decrease, which makes sense. The average number of split

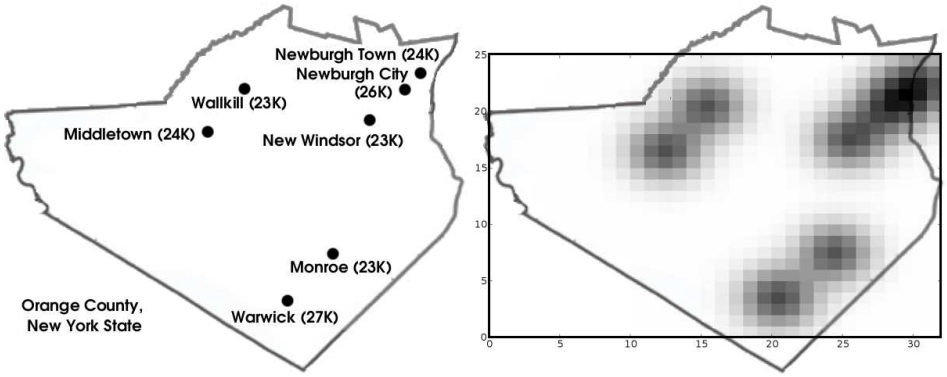

Figure 9: The seven most populated towns in Orange County are identified, and turned into Gaussian distributions along a square grid of the same area.

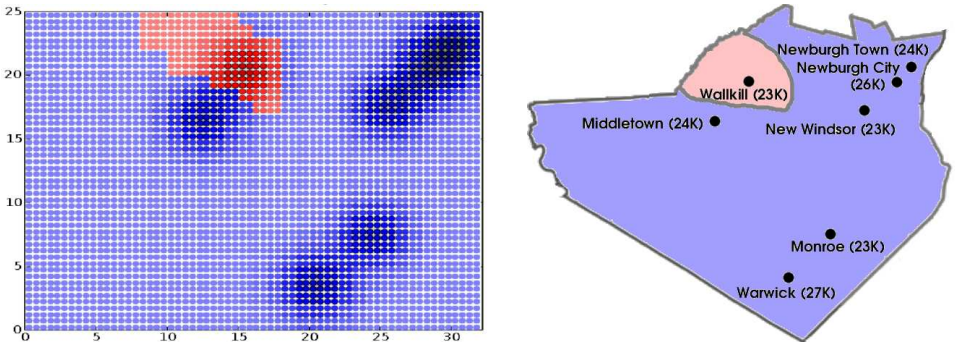

Figure 10: A relatively unequal (due to the population needs) distribution is created through seeding and simulated annealing. The result of the CLDA is then qualitatively translated into a district division on the original map.

Tolerance	Failure Rate	Split Counties	Forced Splits
2.5%	56.5%	12.63	9
5%	43.9%	12.41	9
10%	39.9%	10.86	9
20%	31.4%	9.10	8
30%	25.2%	8.51	8

Figure 11: The percent tolerance for deviations in population between counties, compared to the average failure rate of the algorithm, and the average number of counties split over a large number of successful runs. Forced splits are those made in counties with a population larger than the ideal district population. The "default" threshold is 5%.

counties, however, is somewhat misleading; some of the counties need to be split because they are too large to fit entirely in one district. The number of counties that need to be split is in the last column. Subtracting this from the previous column indicates that the number of counties that need to be split is extremely small compared to the number of counties overall, even in the case where the tolerance is more restrictive than required.

The failure rate is an artifact of the SLDA implementation used here, which elects to fail instead of performing more and more complex operations to ensure some valid solution is constructed.

11.2 CLDA Sensitivity to Arbitrary Parameters

We have two parameters defining the importance of the three different badnesses incorporated into county badness (the third merely defines the scale of temperature), and any number of parameters defining the initial temperature, number of steps, etc, for the annealing process.

We set the values of parameters $\hat{C}_b$, $\hat{C}_p$, and $\hat{C}_r$ in a manner which creates district shapes we find pleasing to the eye, and which are very close to the ideal district population.

The parameters for the annealing process are easier to set. We use a three temperature model which we run at varying temperatures, with the number of iterations at each level being $I \gg n^2$. This is the range of I for which each boundary position can trace a path with path length greater than n with high probability. With a sufficiently high starting temperature and large number of iterations, we found that various annealing runs converged to similar optimal badnesses with little sensitivity to the particular values

of the parameters.

12 Possible Improvements

12.1 SLDA

There are several obvious improvements that could be made to the current implementation of SLDA:

- A **more sophisticated county rearrangement algorithm** would ensure that a valid solution to the statewide problem could be constructed in all cases.
- When growing G', **prioritizing counties** by their **number of existing connections** to G' would likely make the districts more compact and therefore rounder.

12.2 CLDA

While our county-level solutions seem quite good, there are a number of improvements which could be made easily, with little bearing on the effectiveness of the model.

- We currently do not use a **diagonal distance measure**, instead opting for sum of side lengths of every block included.
- The amount of **population data** included is **relatively small**. Including more cities and towns will yield both more accurate and more interesting solutions.
- Use of a **hexagonal grid** would also reduce anisotropy.
- A **ergodic temperature schedule** set by algorithm as opposed to experience might lead to better results. Making a careful estimation of T_0 and using the schedule outlined in (13) would cause our system to be **weakly ergodic**.
- Suppress inferior local minima using **stochastic tunneling**, described below.
- Include a badness associated with district lines off of **city lines and other existing boundaries**, to further simplify for people what district they live in.

12.2.1 Stochastic Tunneling

In general, simulated annealing processes are adversely sensitive to the prevalence and depth of local (i.e. non-global) minima. Wenzel and Hamacher [10] propose a way to overcome this obstacle called **stochastic tunneling**.

Let $E(X)$ be the energy function of state X. After the annealing process has converged to a suspected local minima with energy E_0, we update the energy function as follows:

$$E'(X) = 1 - e^{-\gamma(E(X)-E_0)} \tag{14}$$

This new function will **suppress all local minima of a higher energy** than the current one, and **accentuate local minima of a lower energy**. Above, γ is a **tunneling parameter** that magnifies the transformed energy of potential wells deeper than the current one. We then seed a new starting position and begin the simulation again.

13 Conclusion

We have formed an unambiguous metric for what makes a simple and effective districting plan within a state. We have designed a two-level algorithm which attempts to create an optimal partition of a state into districts. We have implemented this solution in Python and C++, and used it to propose a redistricting of New York State.

Our state-level simulation has demonstrated a high level of effectiveness in avoiding county divisions between districts whenever possible. Even though it does not explicitly consider geometric criteria, the resulting district maps show a high degree of compactness.

Our county-level simulation has demonstrated an ability to reach minimal or near-minimal badness solutions. It provides an improvement over the Chou solution discussed earlier in terms of running time, while considering a finer (sub-county) level of detail.

Both solutions are simple in theory, and relatively simple to program and use. We believe that, if the data about towns and cities were improved in quality and quantity, most voters would find our model's plans quite acceptable for New York's congressional districting needs.

Finally, we can express our districting method and plans to the layman easily as follows:

> *To avoid gerrymandering, we have proposed a redistricting of your state which*
>
> 1. *Avoids splitting any one county between multiple districts whenever possible, and*
> 2. *Tries to make the district lines as simple as possible while not cutting up cities and towns, when a county must be split.*

References

[1] John R. Birge. Redistricting to maximize the preservation of political boundaries. Available at `http://deepblue.lib.umich.edu/bitstream/2027.42/3637/4/bam7063.0001.001%.txt`.

[2] New york by county: Population, housing units, area, and density: Census 2000. Available at `http://factfinder.census.gov`.

[3] Some districts are more equal than others. Available at `http://www.thirty-thousand.org/pages/section_IV.htm`.

[4] Horst D. Simon and Shang-Hua Teng. How good is recursive bisection? *SIAM Journal on Scientific Computing*, Volume 18(5):1436–1445, 1997.

[5] Lester Ingber. Simulated annealing: Practice versus theory. *Math. Comput. Model.*, 18(11):29–57, 1993.

[6] Bruce Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2), 1988.

[7] Chung-I Chou and S.P. Li. Taming the gerrymander- statistical physics approach to political districting problem. *Physica A*, 369, 2006.

[8] Google maps. Available at `http://maps.google.com/`.

[9] New york state census data. Available at `http://www.empire.state.ny.us/nysdc/`.

[10] W. Wenzel and K. Hamacher. Stochastic tunneling approach for global minimization of complex potential energy landscapes. *Phys. Rev.*, 82(15):3003–7, 1999.