**OUTLINE**

1. **The Problem**

* Explain what gerrymandering is
* Show some examples
* Yes, we are one of them! (both federal legislative and state legislative!)
* Give a brief history of how it is so bad now.
* Show how it’s done
* Population balance, compactness, and contiguity are NOT sufficient. (show naïve example)
* There is some debate about whether or not to use results and/or registration. The answer is clear: it MUST be used. Not doing so produces BAD results. (only reason some results are good is that population is not mixed) (show that)
* However, it is a legitimate concern that a human could use the results to assist in gerrymandering.
* Nonetheless, if an agent – human or computer does NOT have this information – they cannot design even partially fair districts, unless the demographics are poorly mixed (which they are), and even then, only by accident.
* The answer to both of these is that you can – and MUST -- apply the information only at the selection step, and more importantly - only against (and only \_towards\_) just criteria. (e.g. measures of fairness, equity, and practicality.)
* So just test against all possible maps, right?...
* Classical methods are too slow! (It is what’s called “NP-Hard”.)

1. **The Solution**

* Enter iterative refinement….
* Define the model (polygons, features)
* Generate random map
* Zoom in on selection process
* Selection components
* Not on competitiveness – it’s a zero-sum, so have to maximize competitiveness equality (or conversely, minimize inequality)
* Zoom in on crossover and mutation

1. **A Walk-Through of the Program**
   1. Download the app.
   2. Run
   3. Load shapes
   4. Load census
   5. Load demographics or results
   6. Run
   7. Export results
   8. Add steps from beginning to end. – feeding in geo data atoms, connecting atoms, locking atoms. (optional), importing census, importing results, setting # of districts, running, exporting results, post-processing.
2. **What about proportionally representative multi-member districts?**

* Can combine the resulting single-member districts into multi-member districts.
  + The particular combinations don’t really matter all that much – some are slightly better than others, but not by much.
* While there are certainly advantages to multi-member proportional representative districts, they are not without drawbacks. Particularly:
  + The higher number of candidates the voter has to consider means needs they need to be more informed
  + Alternatively picking straight party, ok, then which of the candidates to select?
  + Loss of direct responsiveness? Or increase? (due to multiple choices – but then the candidates also have potentially more people to respond to)
* Though the advantages are important should not be overlooked:
  + Intrinsic protection against gerrymandering, (not complete, but pretty darn good.)
  + Automatically responsive to changes in demographics / voting patterns. (less need for re-districting when populations and/or demographics change.)
* The advantages and disadvantages should be considered together.
* To reiterate the first point: proportionally representative multi-member districts is simply one additional step at the end: combining districts. That is, the two methods can be combined.

1. **Where to get the source code, technologies used, where to get data, etc.**
2. **Sample Results – WI federal (legislative)**
3. **Sample Results – WI state (assembly)**

**THE SOLUTION**

**Iterative Refinement A.K.A. Heuristic Optimization**

**Definition and Motivation**

**Heuristic**

**: using experience to learn and improve**

**:**  involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially [trial-and-error](http://www.merriam-webster.com/dictionary/trial%20and%20error) methods <heuristic techniques><a heuristic assumption>; also **:**  of or relating to exploratory problem-solving techniques that utilize self-educating techniques (as the evaluation of [feedback](http://www.merriam-webster.com/dictionary/feedback)) to improve performance <a heuristic computer program>

*-- Merriam-Webster online.*

**Heuristic Algorithm**

“In [computer science](http://en.wikipedia.org/wiki/Computer_science), [artificial intelligence](http://en.wikipedia.org/wiki/Artificial_intelligence), and [mathematical optimization](http://en.wikipedia.org/wiki/Mathematical_optimization), **a heuristic is a technique designed for**[**solving a problem**](http://en.wikipedia.org/wiki/Problem_solving)**more quickly when classic methods are too slow**, or for finding an approximate solution when classic methods fail to find any exact solution. This is achieved by trading optimality, completeness, [accuracy](http://en.wikipedia.org/wiki/Accuracy_and_precision), or [precision](http://en.wikipedia.org/wiki/Accuracy_and_precision) for speed. In a way, it can be considered a shortcut.

## Definition and motivation

The objective of a heuristic is to produce a solution in a reasonable time frame that is good enough for solving the problem at hand. This solution may not be the best of all the actual solutions to this problem, or it may simply approximate the exact solution. But it is still valuable because **finding it does not require a prohibitively long time.**

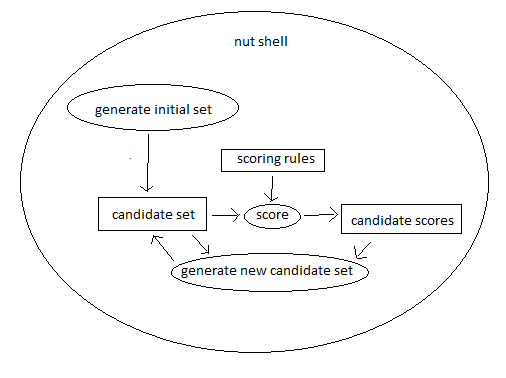
Heuristics may produce results by themselves, or they may be used in conjunction with optimization algorithms to improve their efficiency (e.g., they may be used to generate good seed values).

Results about [**NP-hardness**](http://en.wikipedia.org/wiki/NP-hard) in theoretical computer science make heuristics **the only viable option** for a variety of complex optimization problems that need to be routinely solved in real-world applications.”

*-- Wikipedia (emphasis added)*

**The Algorithm**

The process of “heuristic optimization”, in a nut shell, can be summarized by this graph:



Or, to put it even more succinctly, the process is:

1. Score the candidate set
2. Generate a new candidate set
3. Go back to step 1.

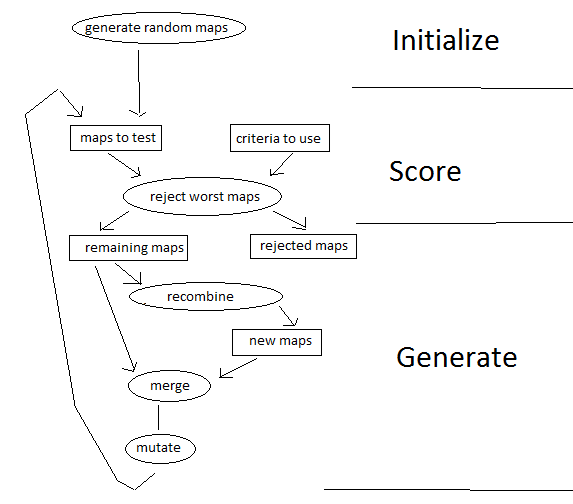
With one very notable detail: In step 2, “Generate a new candidate set”, **the new candidate set is generated by combining information about the previous candidates together with information about how well they meet the objective.**

**Applying it to re-districting**

Enough with definitions. Let’s apply it to re-districting now.

There are many different algorithms we can use for the generation step. We will use selection, recombination, and mutation, because it’s simple, easy to apply, and has good convergence.

The process in detail, applied to the problem of redistricting, can be summarized in this graph:



Information concerning party registration and historical election returns is only used once a plan has been drawn, and only to test the plan for compliance. It is NOT used to generate new maps. It is ONLY used to test the maps for compliance, and to reject the least compliant maps.

* Zoom in on reject worst maps (selection)
  + Include zoom in on criteria
* Zoom in on recombine and mutate (emphasize that recombining is the driving force, and that mutation is just there to keep the diverse, so that it doesn’t settle on a local optima (find better explanation than esoteric “local optima”))

**Scoring in detail**

**Recombination in detail**

**Recombination is the “driving force” in this heuristic algorithm.** It is the part that integrates information from the scoring step to construct a new set of candidates to test. It is the part that “drives” the candidates towards the solution.

Recombination consists of two steps:

1. Randomly select a pair of candidates from the remaining candidates.
2. For each geographic atom, randomly choose either the district from either the first or second candidate of the pair to be the district for the new candidate

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **district** | **district** |  | **district** |
| **precinct** | **candidate a** | **candidate b** | **a or b?** | **new candidate** |
| **1** | 4 | 7 | a | 4 |
| **2** | 2 | 2 | a | 2 |
| **3** | 8 | 6 | b | 6 |

**Mutation in detail**

Unlike recombination, mutation does NOT drive the candidates towards the goal. If anything, mutation drives the candidates AWAY from the goal. At any iteration except for the first, the number of ways a candidate can mutation to a worse solution outnumber the ways it can mutate to a better solution. This means that **a candidate is more likely to mutate to a worse solution than to a better one.** (On the first iteration, the two probabilities are equal.)

So then why have mutation at all? The reason is because that if we *don’t* have it, our candidate set will quickly lose all of its variety. As the candidate sets continue to be recombined and recombined, differences between the candidates will get lost. In short order, all of our candidates will become exactly the same, making the whole recombination step moot.

Recombination becomes essentially cloning, incapable of integrating new information, because it is incapable of gathering new information, because the candidates it tested this iteration were exactly the same as the ones it tested last iteration, as were the results. Without a constant influx of diversity – without new experiments – there is no new information.

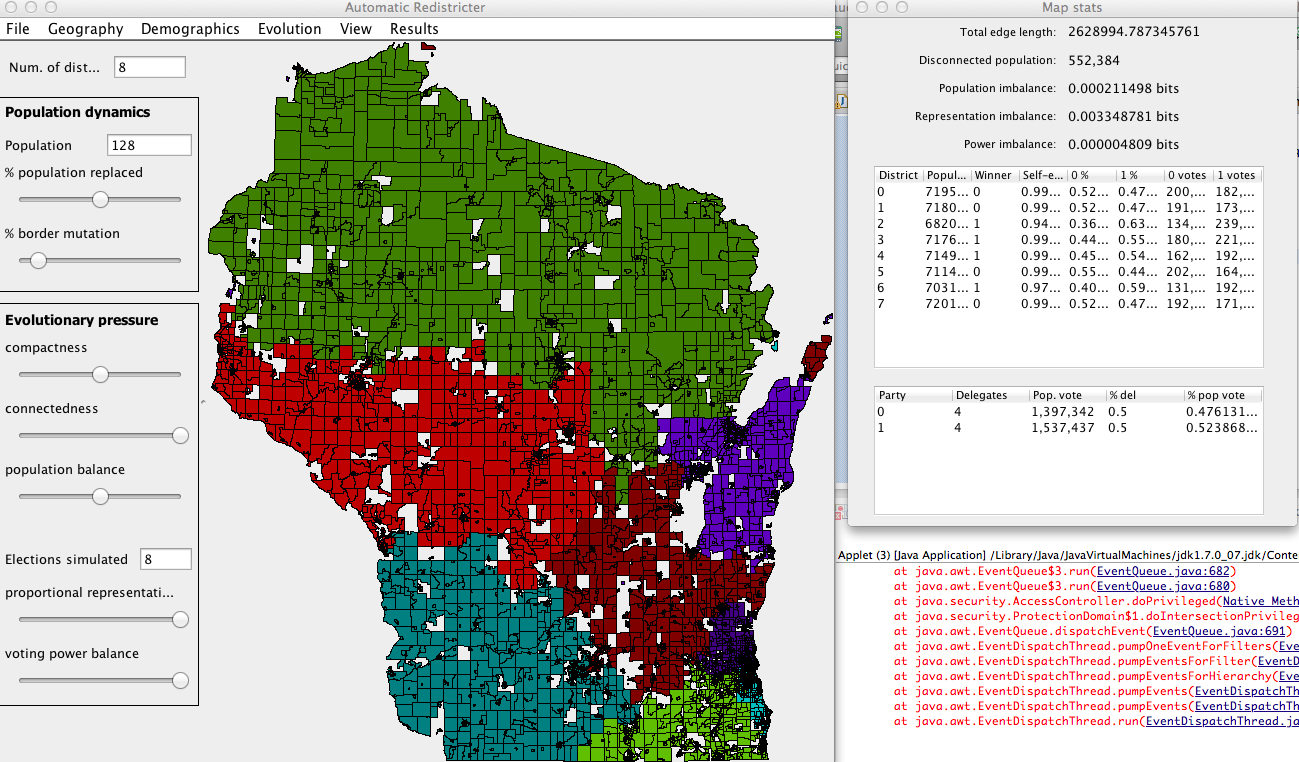
Mutation provides this constant influx of variety. It insures that the test candidates don’t all just become the same map – that new ideas are constantly being explored.

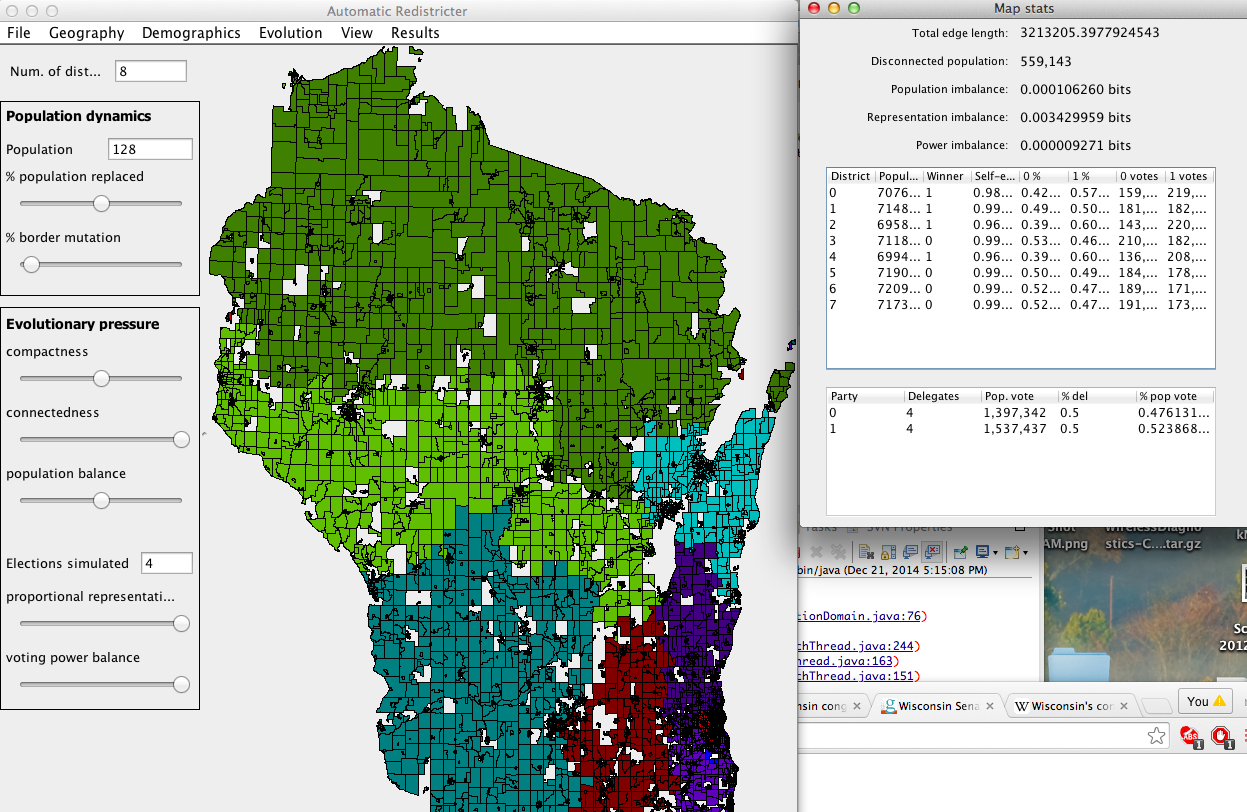
Mutation is done by randomly changing a random set of precincts for a given candidate. In the auto-redistrict program, I restricted the set of possible districts a precinct can mutate to only those that it neighbors on. Consequently only precincts on the perimeter of a district will mutate.

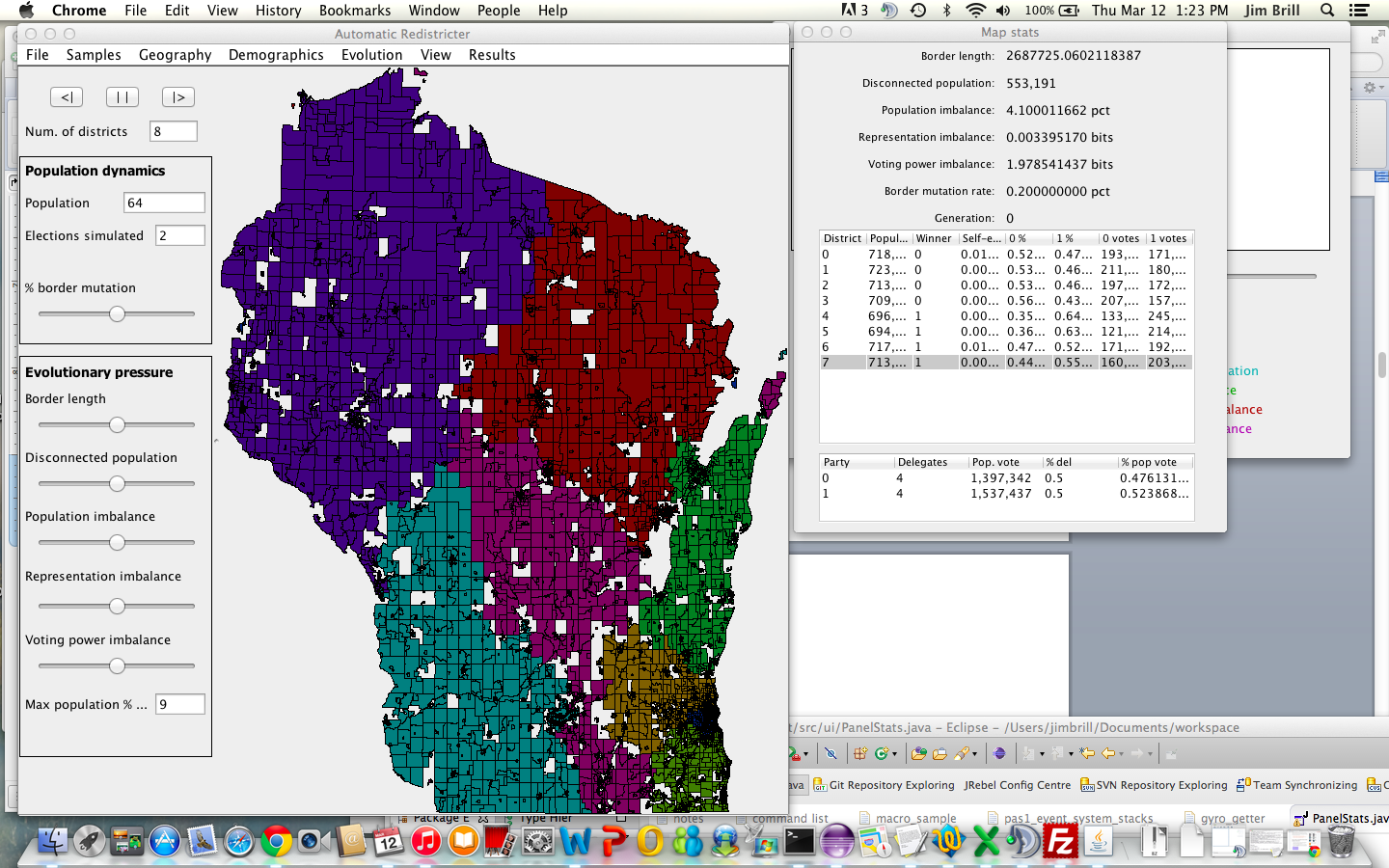
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **district** |  | **random** | **district** |
| **precinct** | **candidate** | **mutate?** | **neighbor** | **mutated candidate** |
| **1** | 4 | yes | 5 | 5 |
| **2** | 2 | no | 1 | 2 |
| **3** | 6 | no | 6 | 6 |

This deserves repeating because it is such a common misconception on such an important point -- mutation does NOT drive optimization. **Recombination** drives optimization. Mutation merely **prevent stagnation**. Mutation on average actually serves to **DECREASE** fitness, but that cost is more than made up for by its ability to keep recombination producing novel results when it would otherwise cease to be productive.

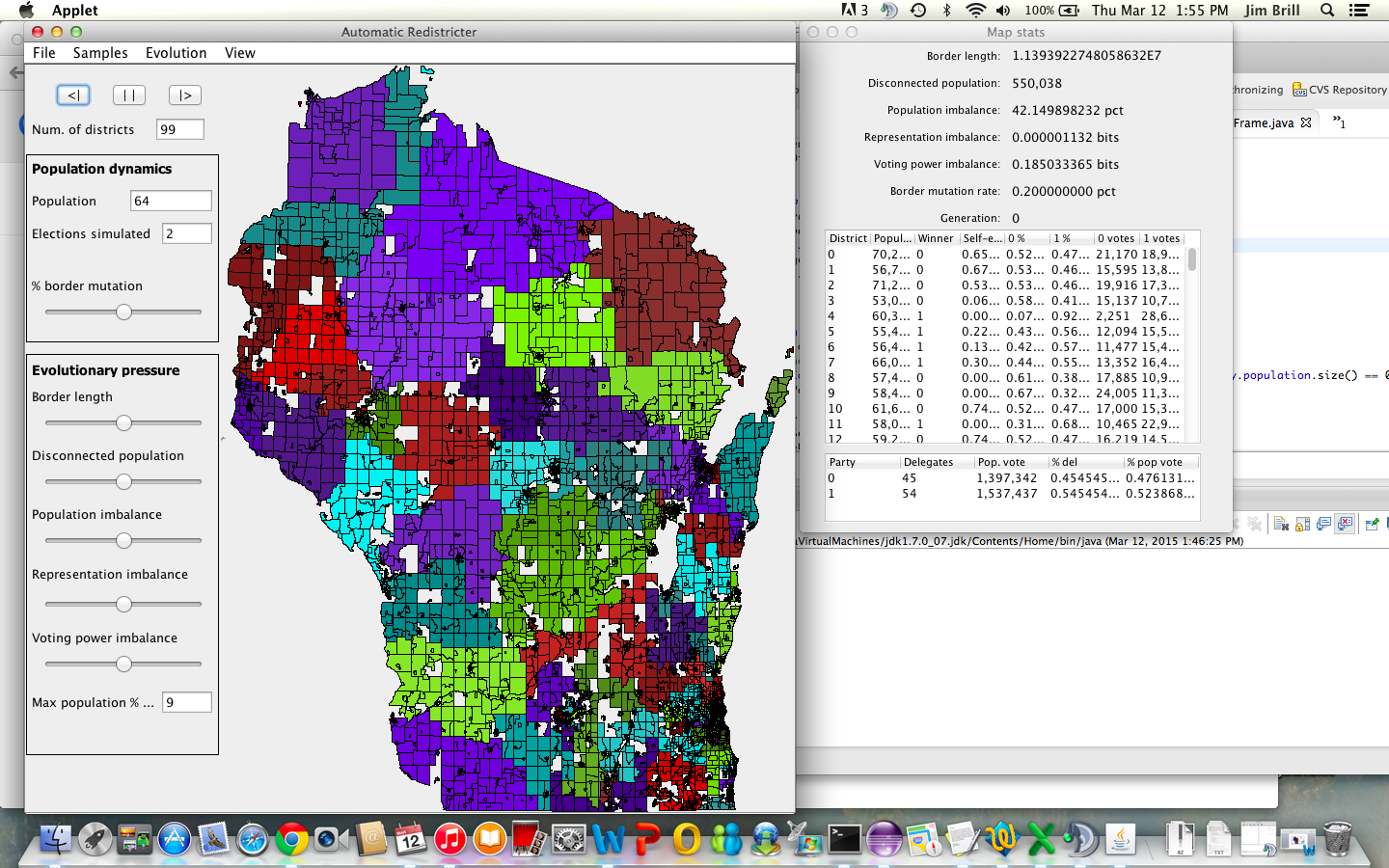
**Appendix C: Sample Results – WI federal (legislative)**

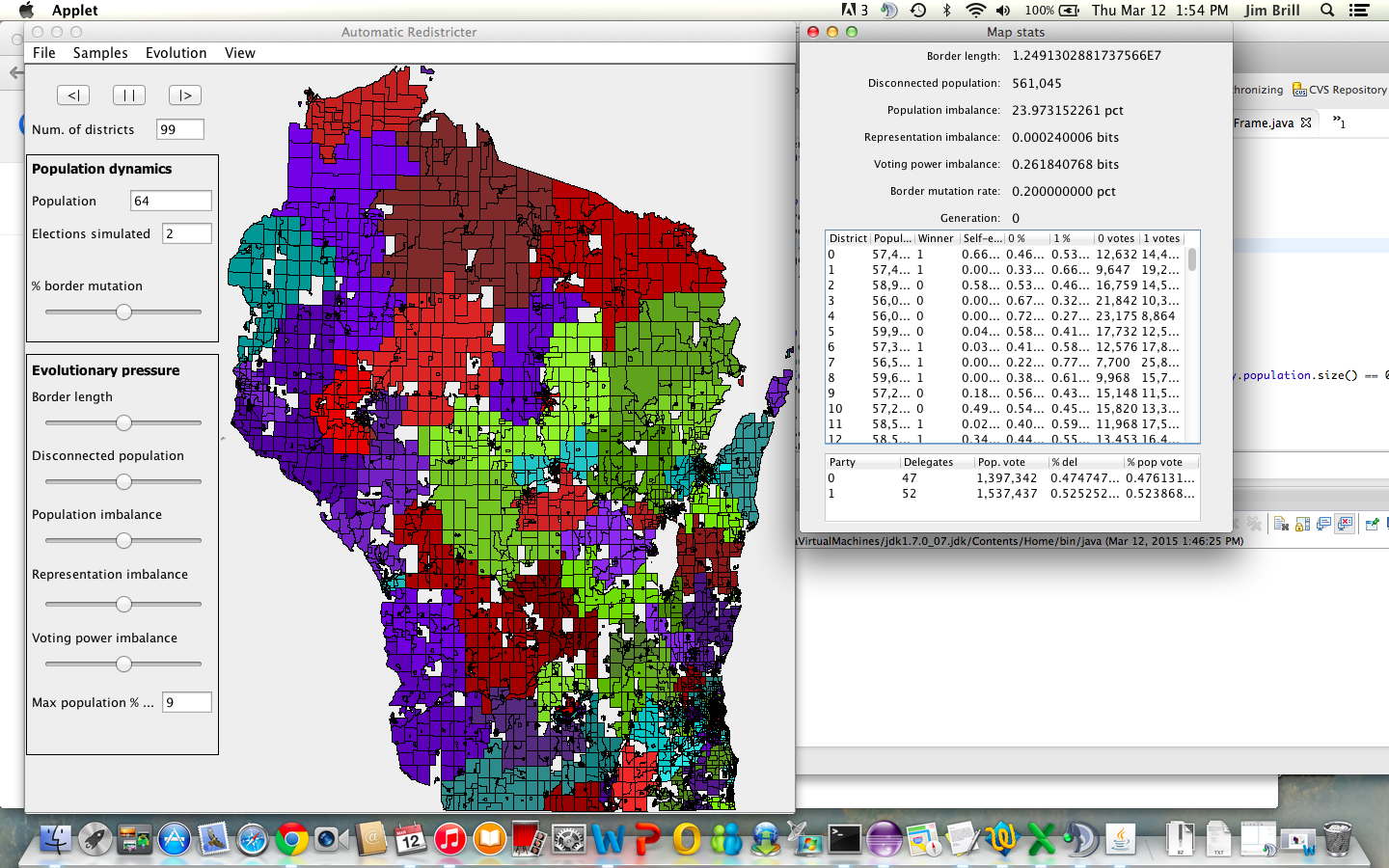
****

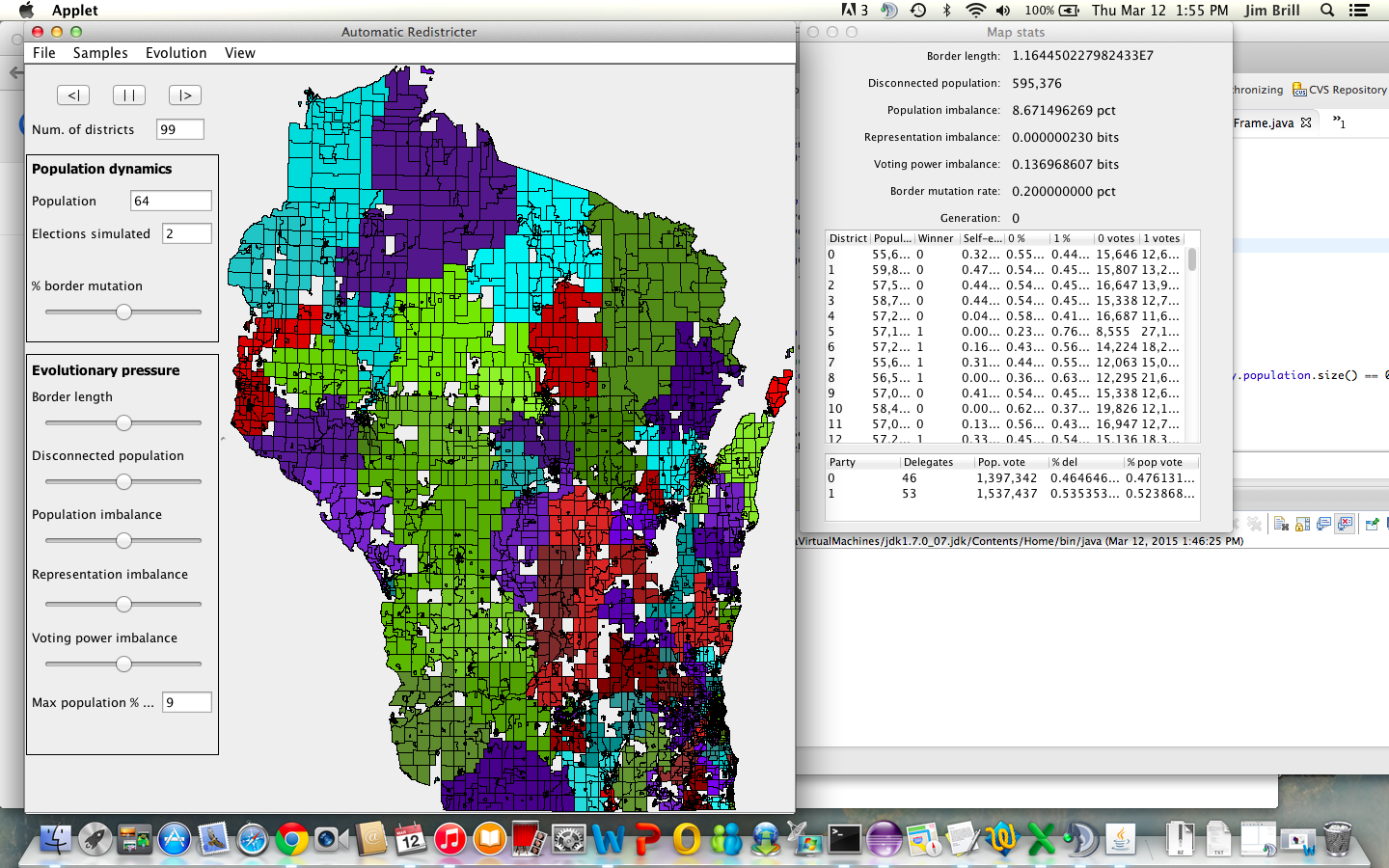
****

****

**Appendix D: Sample Results – WI state (assembly)**



****

****