**Finding the single best re-districting map is not computationally feasible.**

To find the single best (fairest, etc.) re-districting map, one has to score every possible map. If we want to split a map with, say, 10 precincts, into 2 different districts, we can count the number of possible combinations this way:

1. The first precinct could be in either the first district or the second. So that’s 2.
2. Independent of that, the second precinct could be in either the first district or the second. So that’s 2 times 2 = 4.
3. Independent of both of those, the third precinct could be in either the first district or the second. So that’s 2 times 2 times 2 = 8.
4. And so on and so forth…

At the end we’ll be multiplying by 2, 10 times. We’ll have 2^10 possibilities. (That comes out to 1,024, if you’re curious. And even if you’re not curious… it still comes out to that number.)

This simple relation holds true for any combination of number of precincts and number of districts. The number of possible maps is always [number of districts] to the power of [number of precincts].

In a more realistic scenario, we’d have something more like 6,000 precincts, and maybe 50 or so legislative districts. So that’s 6,000^50. A quick way to approximate that is round 6,000 down to 1,000 – 1,000^50 – which is the same as 10^(50\*3=150), or 1 with 150 zeros after it. Almost literally a google times a google. (“a google” being 10^100.)

It is quite simply not computationally feasible to score all of these maps.

We need to somehow find a way to score a LOT fewer maps, while still making sure we find a good – and ideally *very* good – solution.

Enter a concept in computer science known as “heuristic optimization”…

.

**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.**

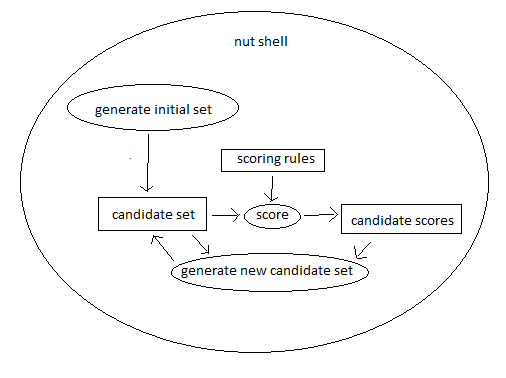
Heuristics may produce results by themselves, or they may be used in conjunction with optimization algorithms to improve their efficiency (e.g., they may be used to generate good seed values).

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.

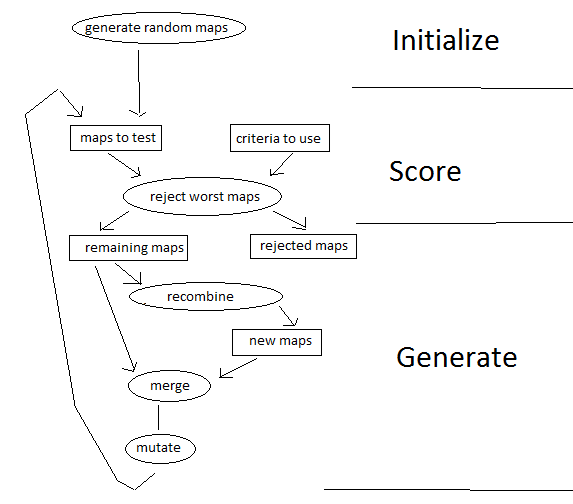
With one very notable detail: In step 2, “Generate a new candidate set”, **the new candidate set is generated by combining information about the previous candidates together with information about how well they meet the objective.5**

**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.

**Scoring in detail**

The scoring part scores 5 separate factors. 3 practical measures, and 2 fairness measures:

Practical

* Equal population – the percent difference between the highest population district and the lowest.
* Compactness– minimize the total length of all borders between districts. This ensures compact districts
* Contingency – the total count of population that is not connected to the largest region. The map starts off totally random, to help it search all possibilities without bias. This helps it slowly de-randomize without causing it to favor certain maps that meet the goal over others.

Fairness

* Proportional representation
* Equal voting power

The results are then ranked individually (smallest to largest), and then the ranks are added together. For example, the map with the 5th smallest border length, 10th least disconnected population, 7th least population imbalance, 4th lowest representation imbalance, and 20th most least voting power imbalance, will have a final score of 5+10+7+4+20 = 46.

The highest scoring maps are thrown away, and the remaining are recombined to make new maps.

You can change the relative weights of the scores by adjusting the sliders on the left. Increasing the weight of a score will cause that trait to be more strongly selected for.

**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.

**The Scoring Criteria in detail – Practicality Criteria**

**Equal population**

To score the maps, a set of potential maps is ranked, least to greatest, by the standard deviation of the populations of the districts.

Furthermore, if a map contains any districts that differ in population by more than the “Minimum % population diff”, that map is penalized by 100 ranks.

**Contiguous**

To score the maps, a set of potential maps is ranked, least to greatest, by the total amount of population not connected to the largest connected region of the district, over all districts.

Thus, the connected area with the largest population is always favored over the other potential “accretion” areas.

**Compact**

The program measures compactness by the area divided by the square of the perimeter. This is known as the “**Isoperimetric quotient**” (though some publications refer to this as the “**PolsbyPopper and Schwartzberg method**.”)

This is hands-down the best method for measuring compactness. Other methods are deliberately not included because they are objectively inferior. For more detail you can read: “Legislative Redistricting -- Compactness and Population Density Fairness” by Kathy Dopp. <http://electionmathematics.org/em-redistricting/LegislativeRedistricting2.pdf>

More precisely, the program adds together, for each district, the perimeter squared divided by the area. It then takes this total, divides by the number of districts, and then takes the reciprocal of that to get the final district-weighted average isoperimetric quotient. (The graph shows the value before the reciprocal, the stats panel shows after the reciprocal.)

**What’s missing: Why are political subdivision splits not a criteria?**

Avoiding splitting political subdivisions has a *negative* impact on both proportional representation and voting power equality. In particular, **it dilutes the votes of people in areas of high population density**.

Since districts have approximately equal population, a political subdivision is more likely to be split – and split more often - if it has a higher population density.

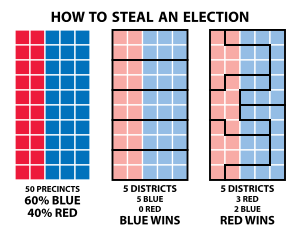
Avoiding “splitting” the districts prevents mixing the demographics. It concentrates members of a group in a single district or handful of districts, thereby allowing the other party to win the remainder of the districts. That is, it has the effect of “**packing**”. This effect disproportionally affects high population density political subdivisions, such as cities.

For this reason it is deliberately not included as a criteria, since it is incompatible – it conflicts with – the primary objectives of this redistricting software – namely, to create fair districts. Whereas **minimizing political subdivision splits results in systematically biased packing.**

Thus, **minimizing political subdivision splits**

**violates Section 2 of the Voting Rights Act.**

**Why these criteria do NOT lead to fair districts**

****

**(Image credit: Steven Nass – License: CC-BY-SA-4)**

Note the second graphic – the one in the middle – represents districts that are:

1. Equal population
2. Contiguous
3. Compact

And yet a whopping **40%** of the population have effectively **ZERO** representation.

The second graphic actually represents the most common outcome, given these 3 criteria. That is, if we were to use just these three criteria, and randomly select a map that meets them, we’d be far more likely than not to pick a map very much resembling this second graphic – where the majority party has *way* more representatives then the minority party, even though the popular vote only *slightly* favors them.

In sum, if only these three criteria, equal population, contiguous, and compact, are used, the result will **systematically exaggerate the vote totals.** In other words, they **systematically dilute votes for minority parties.**

Thus, **using only these three criteria**

**violates Section 2 of the Voting Rights Act.**

**More technical explanation**

If we were, for example, to only use the criteria of equal population, and then select a map at random from all the possible permutations, and then if it doesn't meet that throw it out and re-pick, until we get one that meets that, the map that we end up with will, due to the law of large numbers, have most districts be proportioned in about the same way as the entire state.

That is, if the total popular vote is about 60% for one party, then the individual districts will all be about 60% for that party, too. (Since as the sample size N gets larger, it regresses towards the population mean, with smaller and smaller variance, and since we're counting people, N is very large.)

So in almost all districts, the majority vote will be for the party that has the majority in the total population.  Which means that party will get almost 100% of the seats.  Even though they only got 60% of the popular vote.

**The Scoring Criteria in detail – Fairness Criteria**

**Proportional Representation**

(the composition of those elected should mirror that of the voters)

proportional repr. minimizes: the kl-divergence of the proportion of elected officials of each party from the proportion of votes for each party.

more weight on proportional representation will make the number of members elected of each party more closely match the proportion of citizens who voted that party.

This measures the kullback-leibler divergence of the election results, by party from the popular vote.

**Equal voting power (competitiveness)**

(each person's vote should have about the same impact on the final composition of the elected body)

voting power balance: minimizes kl-divergence of the self-entropy of the districts, weighted by population - meaning it makes it so each citizen has about the same chance of being a deciding vote - the same "voting power" - that their vote counts the same amount.

more weight on voting power balance will make it draw maps where each person has about the same likilihood of affecting the outcome, regardless of what district they're in.

This attempts to balance out how much each person's vote counts.

it calculates the self-entropy of the election results for each district. this represents how much information about the outcome the district create on average.

the ideal distribution of that information is that each citizen has control over the same number of bits.

So this ideal distribution of bits is compared against the actual the actual distribution, via the kullback-leibler divergence.

(measuring how much the actual diverges from the ideal)