Fighting Gerrymandering by Automating Congressional Redistricting

Jacob Jenness

Computer Science Senior Seminar
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

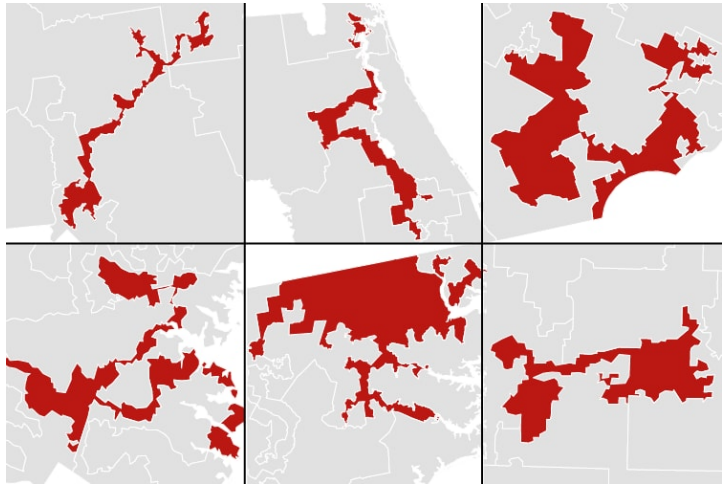
November 13, 2021

Outline

Outline

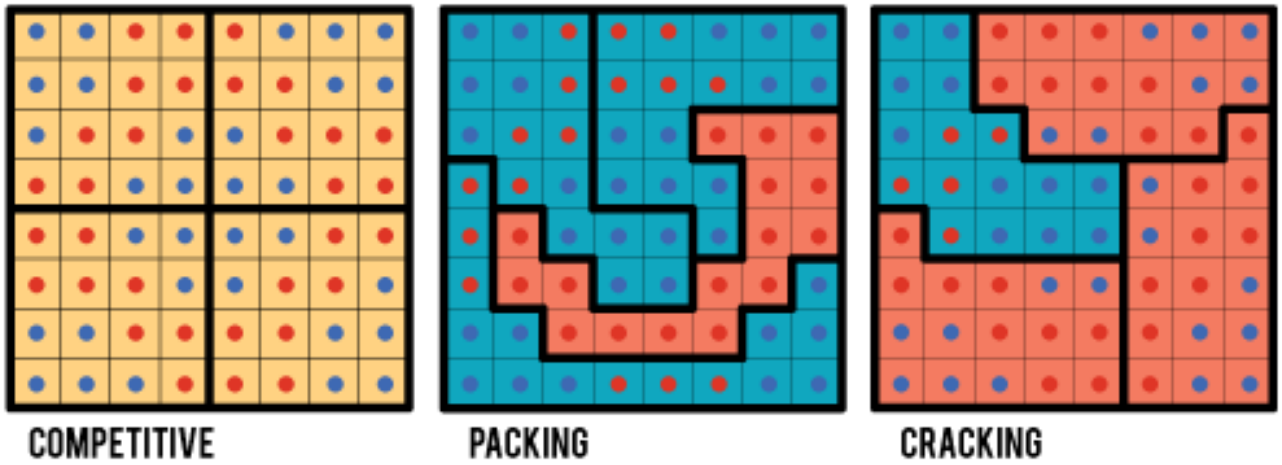
Introduction

- Congressional redistricting takes place every 10 years
- Partisan gerrymandering presents problems
- Algorithms could solve this



Examples of gerrymandered districts [3]

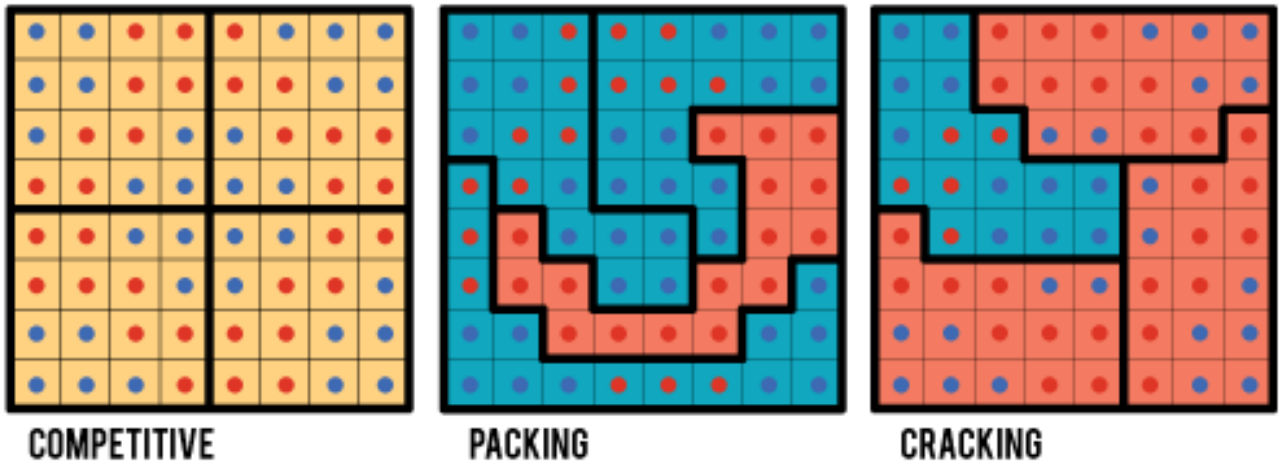
What is Gerrymandering?



A figure of competitive, packed, and cracked districts [5]

- Gerrymandering is when districts are drawn to favor a specific group
- A packed district clumps similar voters together in one district

What is Gerrymandering? Cont.



A figure of competitive, packed, and cracked districts [5]

- A cracked district splits similar voters across districts

How are districts drawn currently?

- Census collects data every 10 years
- Congress apportions representatives to states
- States must drawn district lines
  - Advisory committees
  - Independent committees

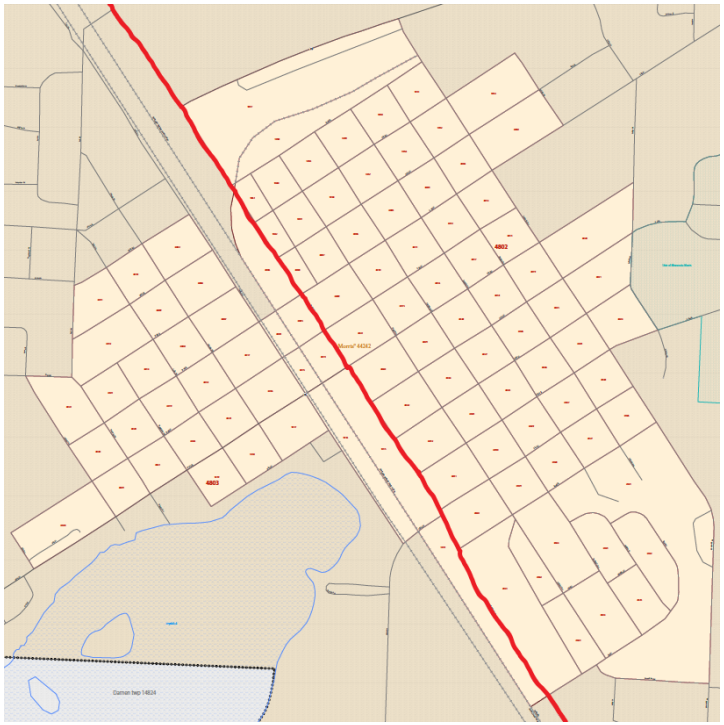
Outline

Different metrics

- There's no consensus on what makes a "good" district
- Different metrics are used to measure different goals
  - Party registration for competitiveness
  - Racial composition for majority-minority districts
  - Various definitions of compactness

Census Data

- Census *blocks* are the smallest population units
- Census *tracts* are made of census blocks
- District lines cannot cut through census blocks



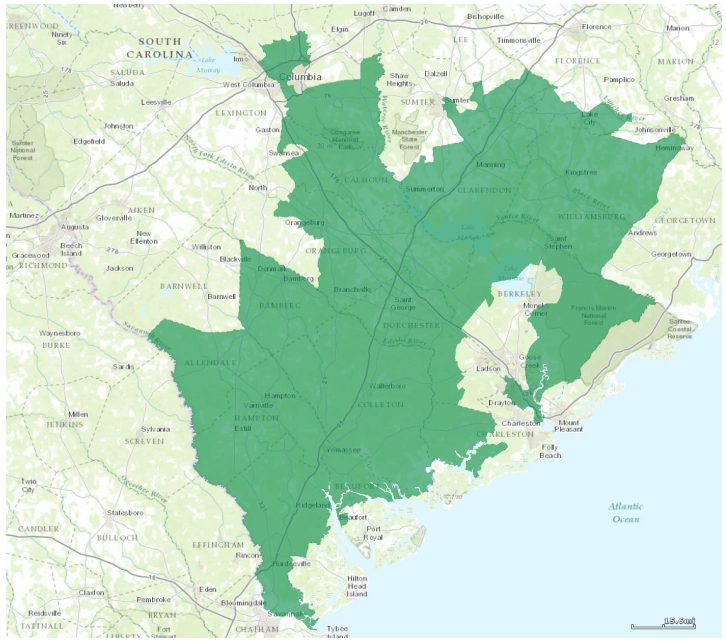
The 2020 census block and tract lines for Morris, MN [1]

Communities of Interest

- Could be demographic, or could be something people share in common
- Very difficult to define reliably
- Nearly impossible to quantify
- Excluded from analysis

Racial Minorities

- Voting Rights Act of 1965
- Majority-Minority districts
- Can be incorporated into the algorithm

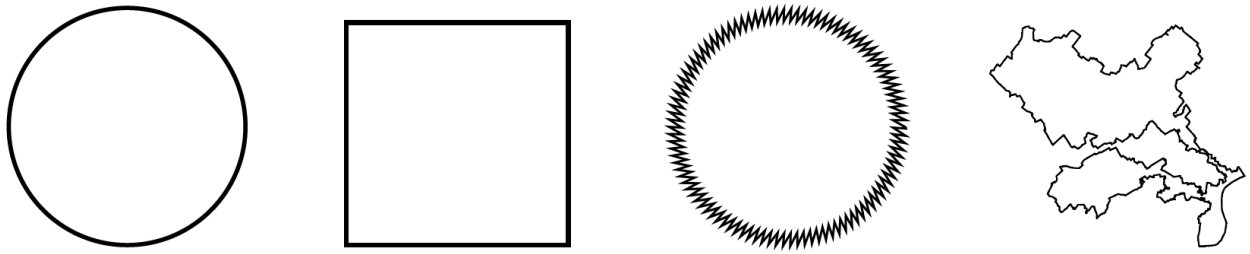


South Carolina District 6 is a majority-minority district that is 57% African-American [2]

Population Deviation

- Measures population differences between districts
- Ideal population deviation is Population(State)/Num. of Districts
- Primary focus of Levin and Friedler [4]
- Must be less than 0.5%
- Lower scores are better

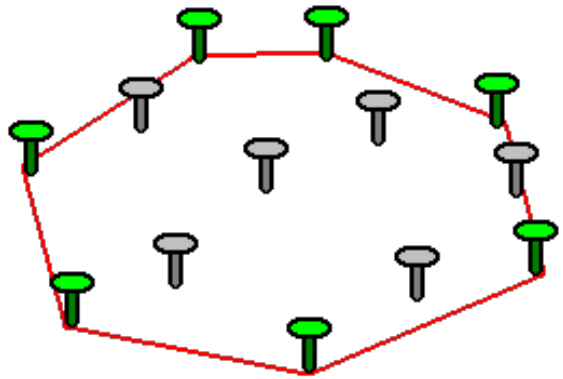
Compactness



Starting from the left, high levels of compactness that decrease with more complexity [6]

- No consensus on definition
- Three kinds used in analysis
- Higher scores are better

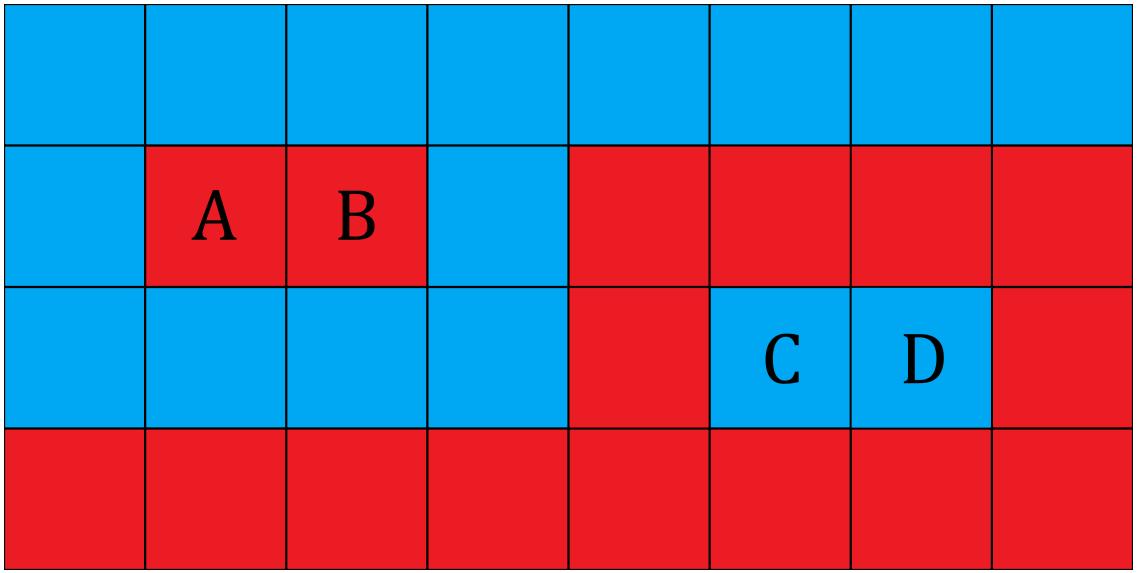
Compactness Cont.



An example of a convex hull. The set of green nails are the vertices of the Convex Hull that surrounds all of the nails

- D stands for District and $p(D)$ stands for perimeter
- Convex Hull: $area(D)/convexHull(D)$
- Polsby-Popper: $(4\pi \cdot area(D))/p(D)^2$
- Modified Schwartzberg: $\left(2\pi\sqrt{area(D)/\pi}\right)/p(D)$

Contiguity



An example of non-contiguous districts. Blocks A,B,C, and D are non-contiguous

- Necessary for all congressional districts
- Every point must connect to every other point without crossing a district line

Outline

Proposed Redistricting Algorithm

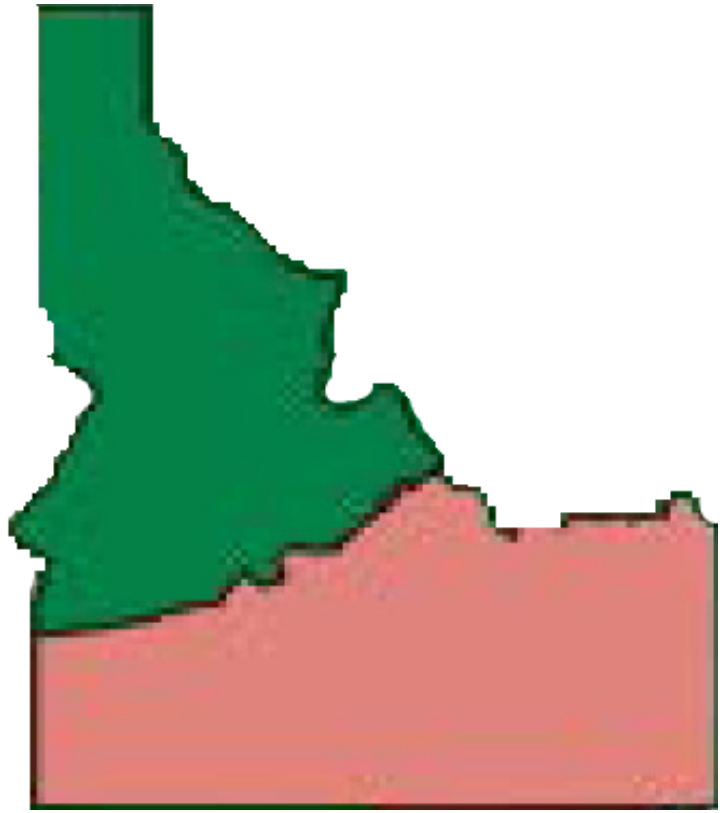
- Recursive Divide and Conquer Redistricting Algorithm, by Harry Levin and Sorelle Friedler [4]
- Utilizes components to show importance of each part

Divide and Conquer

- Main function that calls components
- Starts with a state's geography and population
- Creates a two-district partition
- Recursively sub-divides those two-district partitions

Example 1

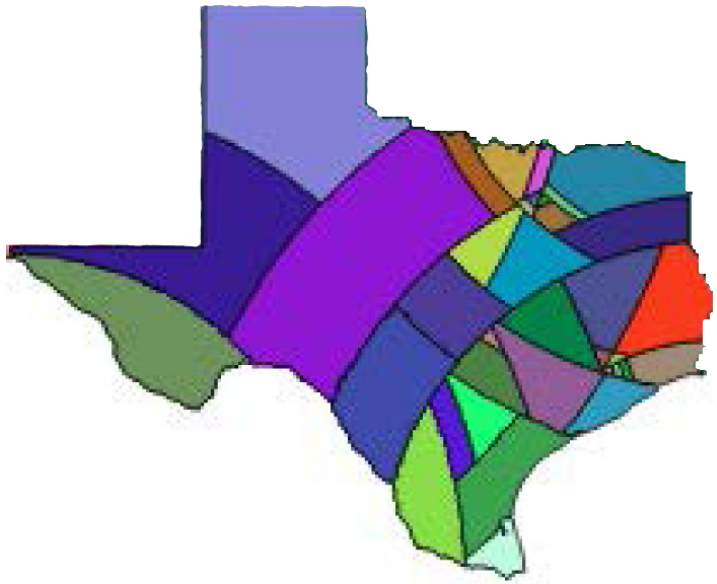
- For a state with two districts, this is easy
- Sub-division is not needed



Idaho, which only has 2 districts [4]

Example 2

- For a state with more districts, this becomes more complicated
- Sub-division happens recursively



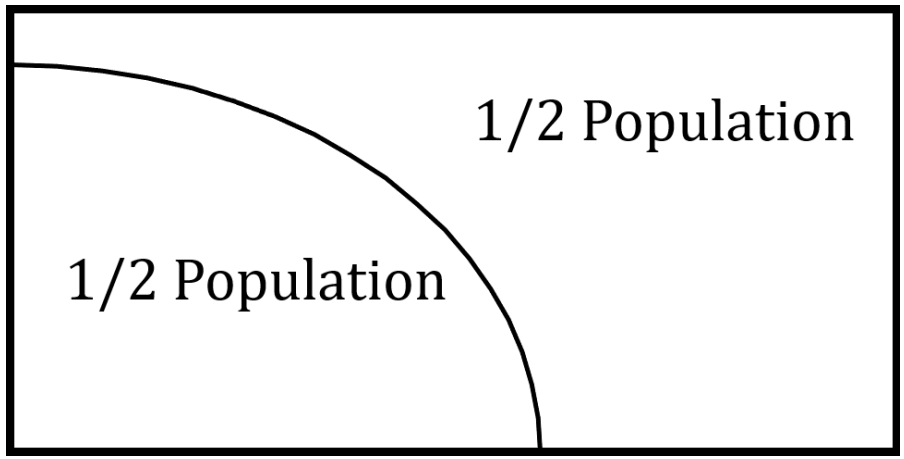
Texas, which has 36 districts in this example [4]

Voronoi Component

- Starts with a rectangle that encompasses the part of the state we want to divide
- This rectangle is called a sub-graph
- Create a two-district partition from this sub-graph
- This process is done for each corner of the rectangle, and is analyzed later

Voronoi Component Cont.

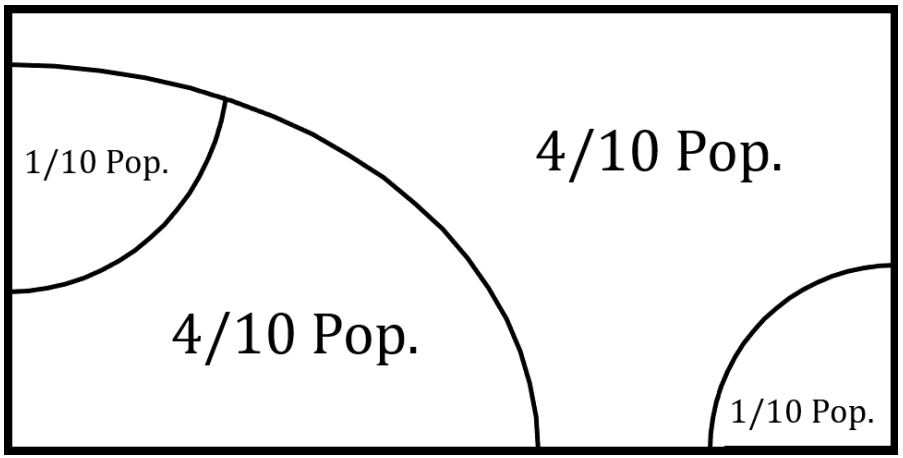
- The diagram on the right shows a state that needs 10 districts being divided
- We start with a district in the corner
- This district will grown until it reaches the desired population
- Because the num. of districts we want is *even*, ideal population is half the sub-graph's population



A two-district partition split into two equal halves

Voronoi Component Cont.

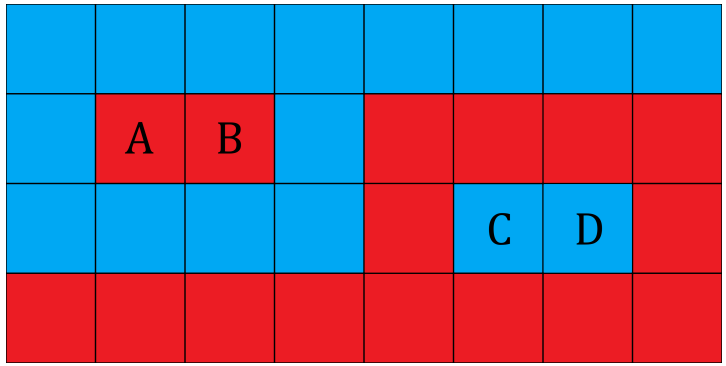
- We still need districts, so both districts become sub-graphs
- The num. of districts needed per sub-graph is 5
- Because 5 is *odd*, ideal population is the population of the sub-graph / num. of districts for the sub-graph
- We will end up with a district that has 1/10th the population of the state, as $1/2 \cdot 1/5 = 1/10$



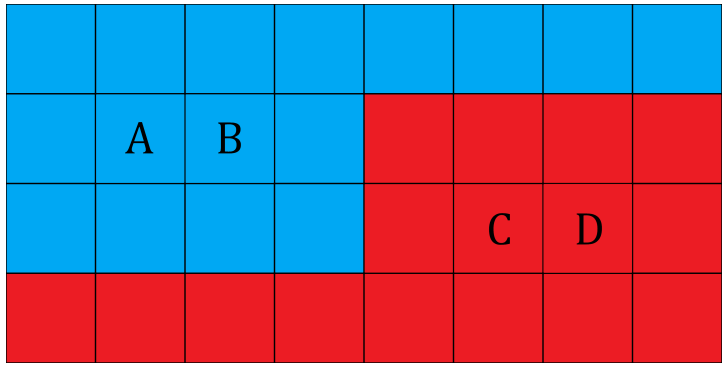
Each sub-graph needs 5 districts, so we create a small district before we cut in half again

Contiguity Swapping

- Both districts must be contiguous
- Find largest collection of contiguous units
- Swap all non-contiguous units to the other district



A,B,C, and D are non-contiguous



A,B,C, and D were swapped, and became contiguous

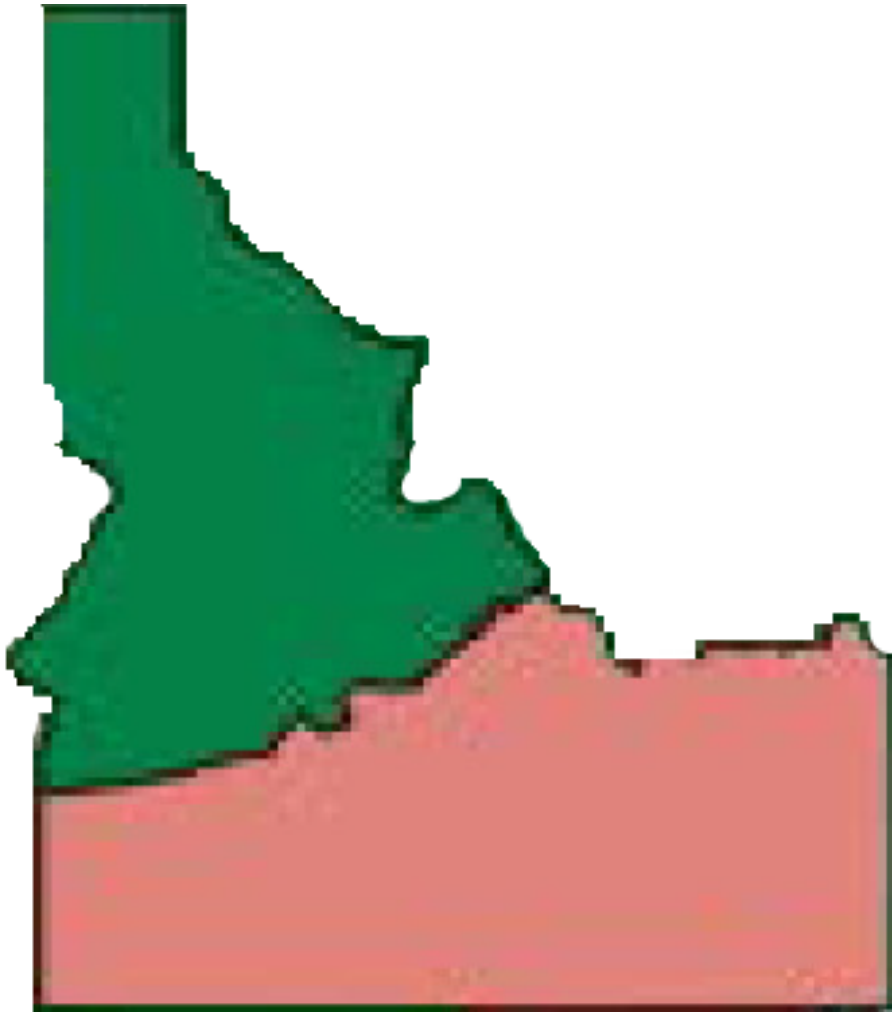
Population Swapping

- Similar to contiguity swapping
- Try and make population deviation as low as possible
- Create a collection of population units (census blocks) that border the edge of the two districts
- Swap population units if it's beneficial

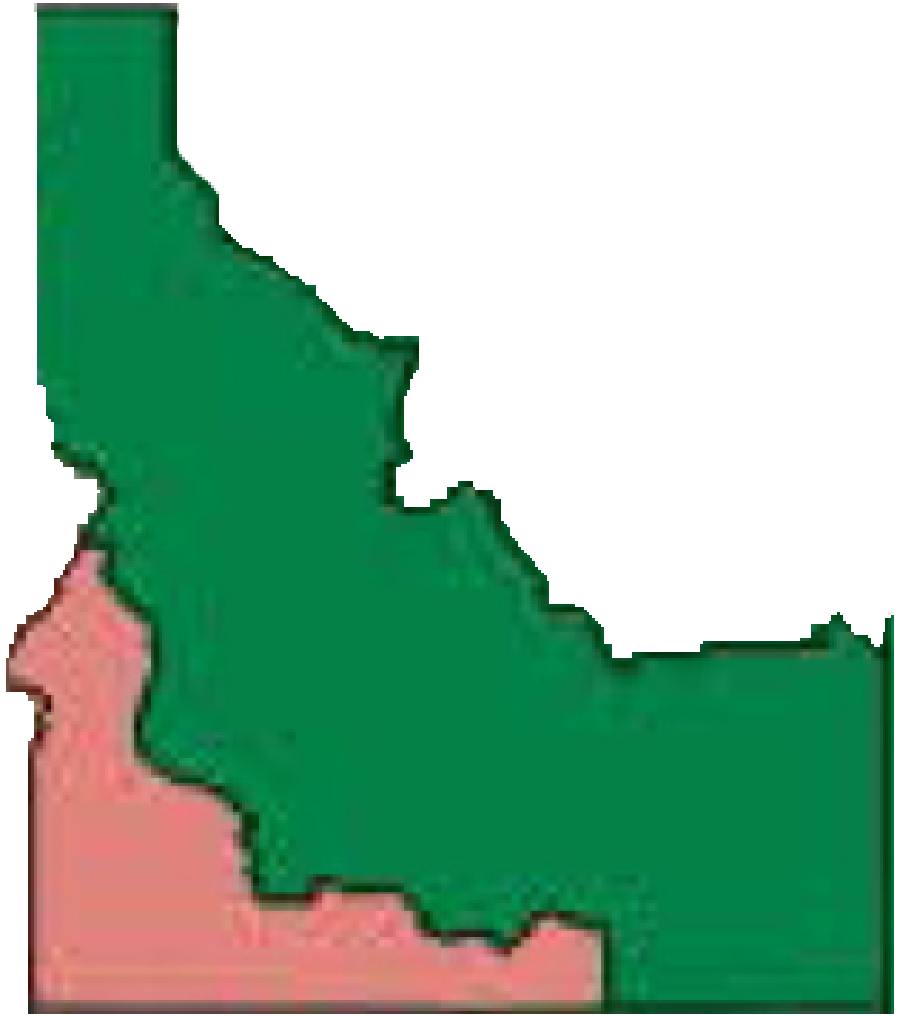
Maximization

- Chooses the best of the four corners in the Voronoi component
- Can do this three ways
  - MinPop: Minimal population deviation
  - MaxCompact: Maximum compactness
  - ValidCompact: Combination of MaxCompact and MinPop (if possible)

Maximization Example



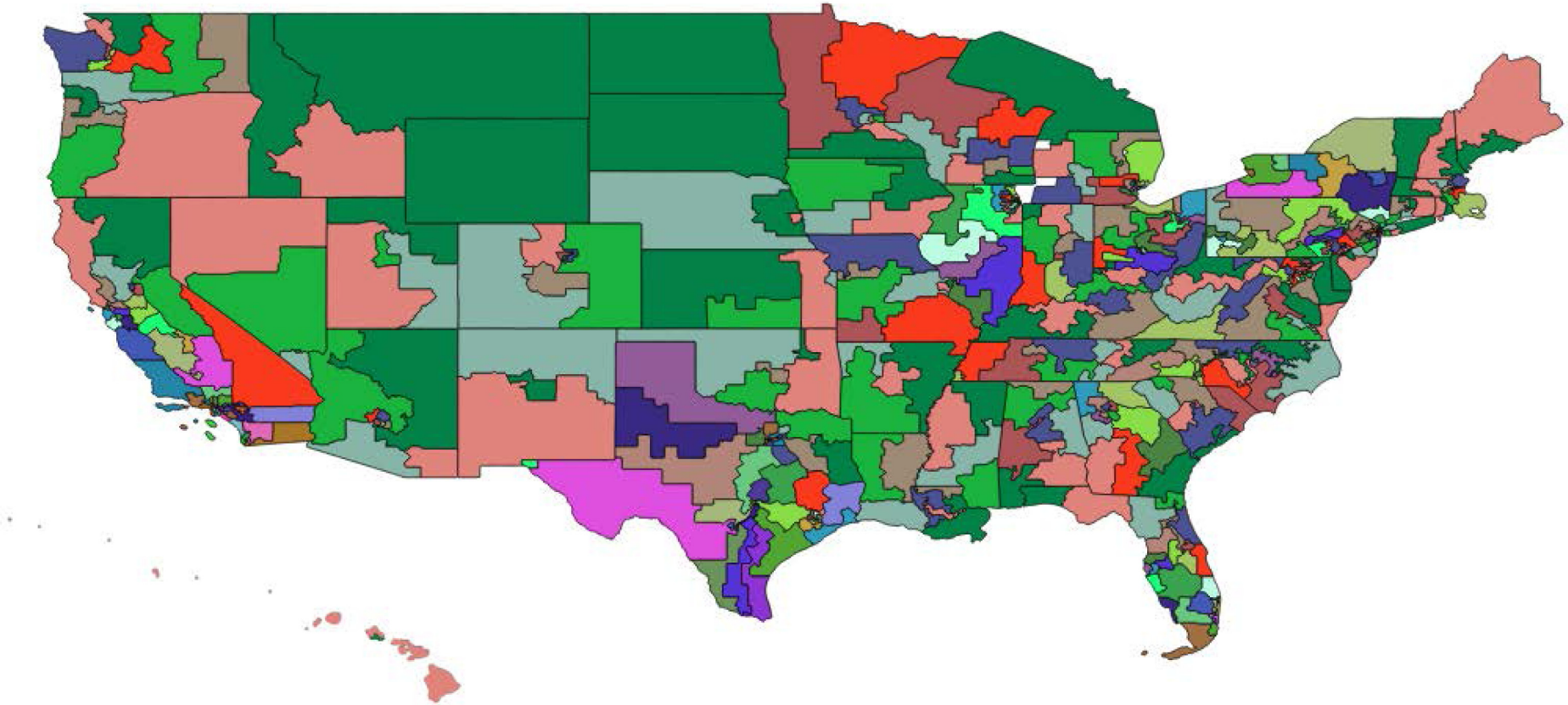
A two-district partition chosen under blocks [4]



The same two-district partition chosen under tracts [4]

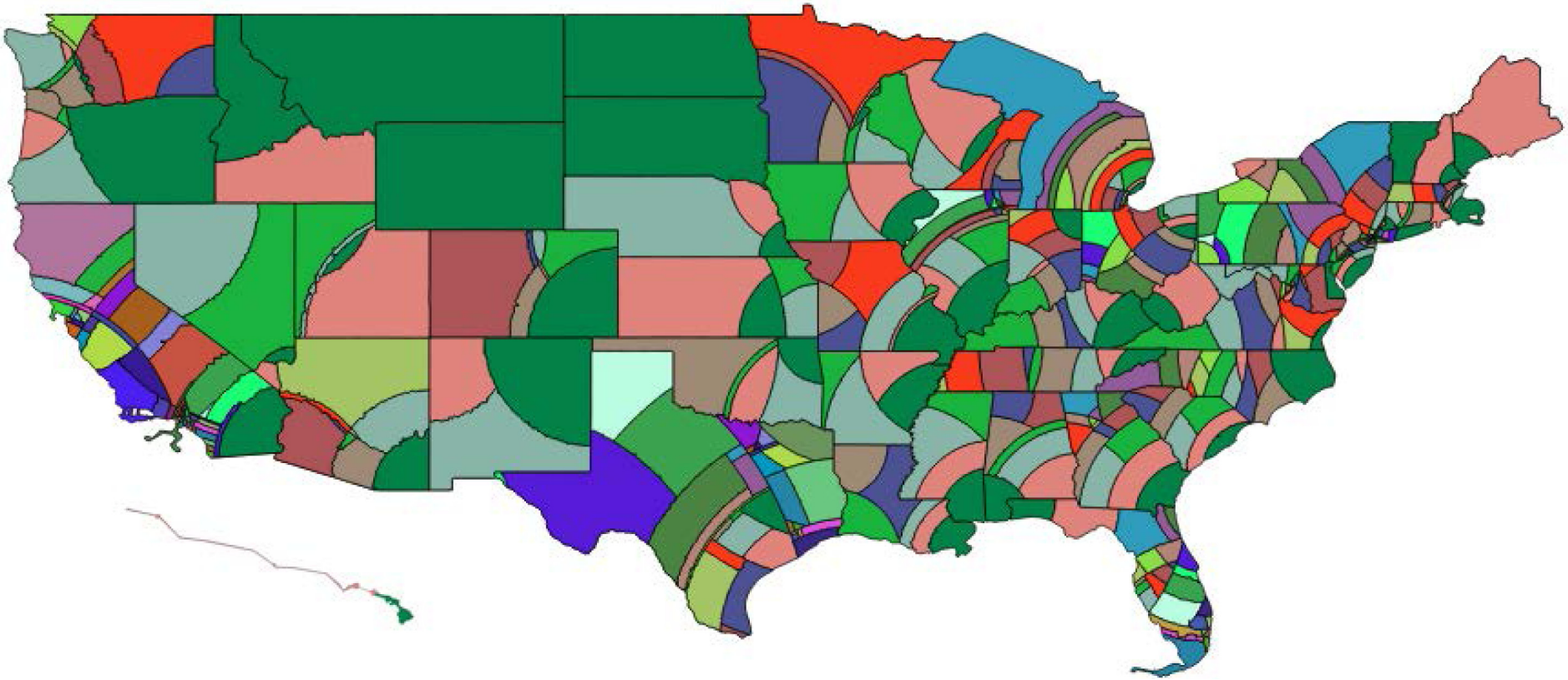
Outline

Actual 2010 Districts



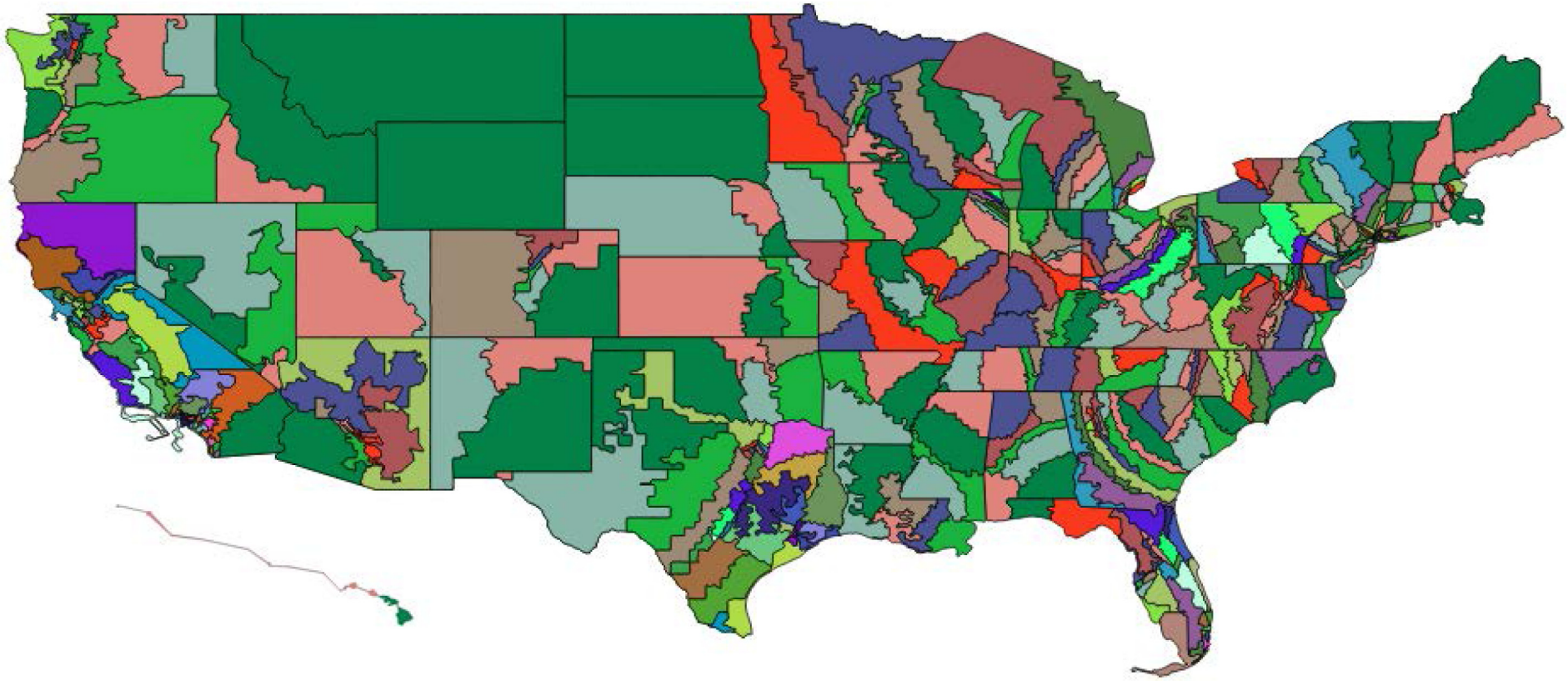
The 113th congressional districts based on the 2010 census data [4]

MinPop run on census blocks



MinPop being run on census blocks [4]

MinPop run on census tracts



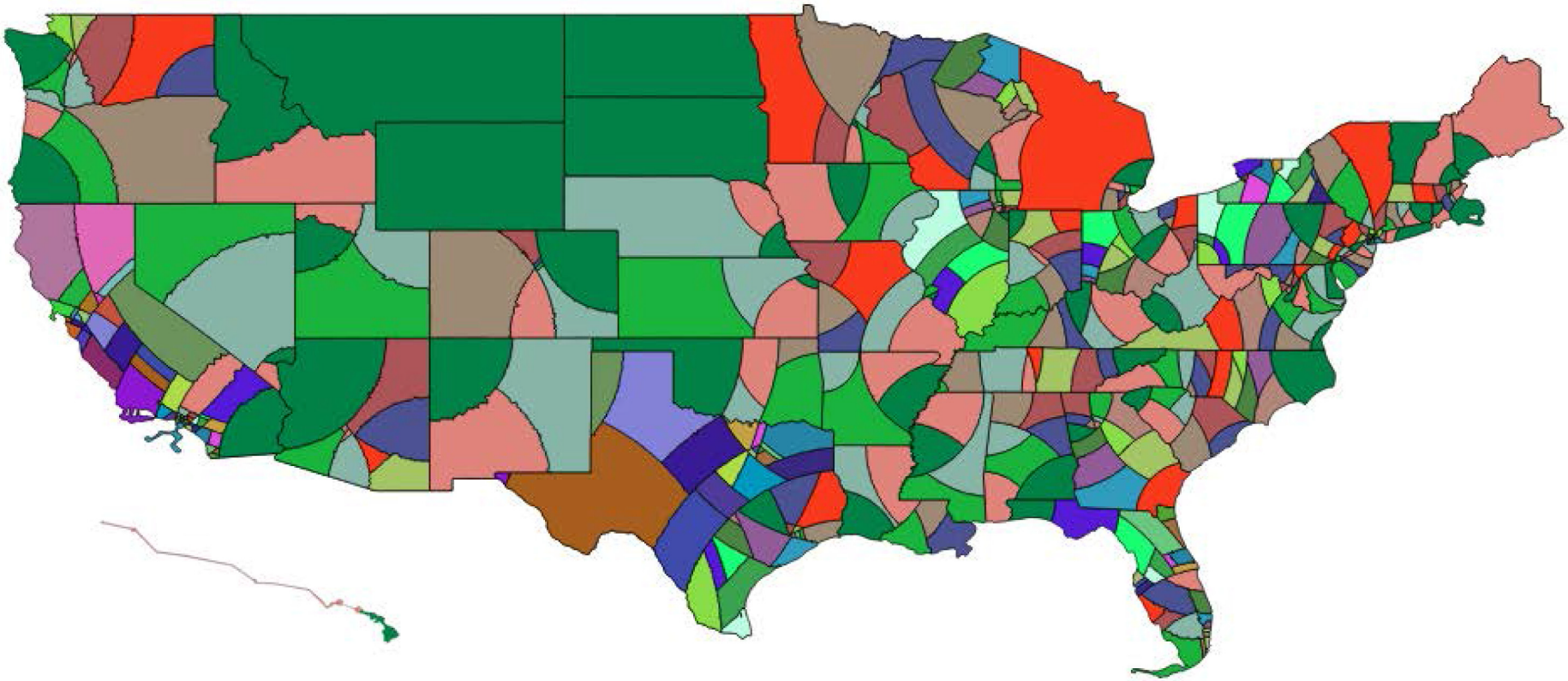
MinPop being run on census tracts [4]

Data Results of MinPop

Measure	Census Blocks Full Alg	Census Tracts No Pop Swap	Census Tracts Full Alg
Valid States (deviation < 0.5%)	42/43	19/43	41/43
Population Deviation Med (%)	0.00%	0.60%	0.02%
Population Deviation Med (people)	1	4,005	127
Convex Hull 25th Percentile	0.76296	0.73297	0.58767
Convex Hull Med	0.81714	0.76072	0.66362
Convex Hull 75th Percentile	0.86209	0.78885	0.73640
Polsby-Popper 25th Percentile	0.20278	0.22542	0.14579
Polsby-Popper Med	0.23686	0.25676	0.20019
Polsby-Popper 75th Percentile	0.27402	0.28161	0.24559
Schwartzberg 25th Percentile	0.44288	0.46605	0.37398
Schwartzberg Med	0.47477	0.50227	0.43874
Schwartzberg 75th Percentile	0.51713	0.52496	0.48407

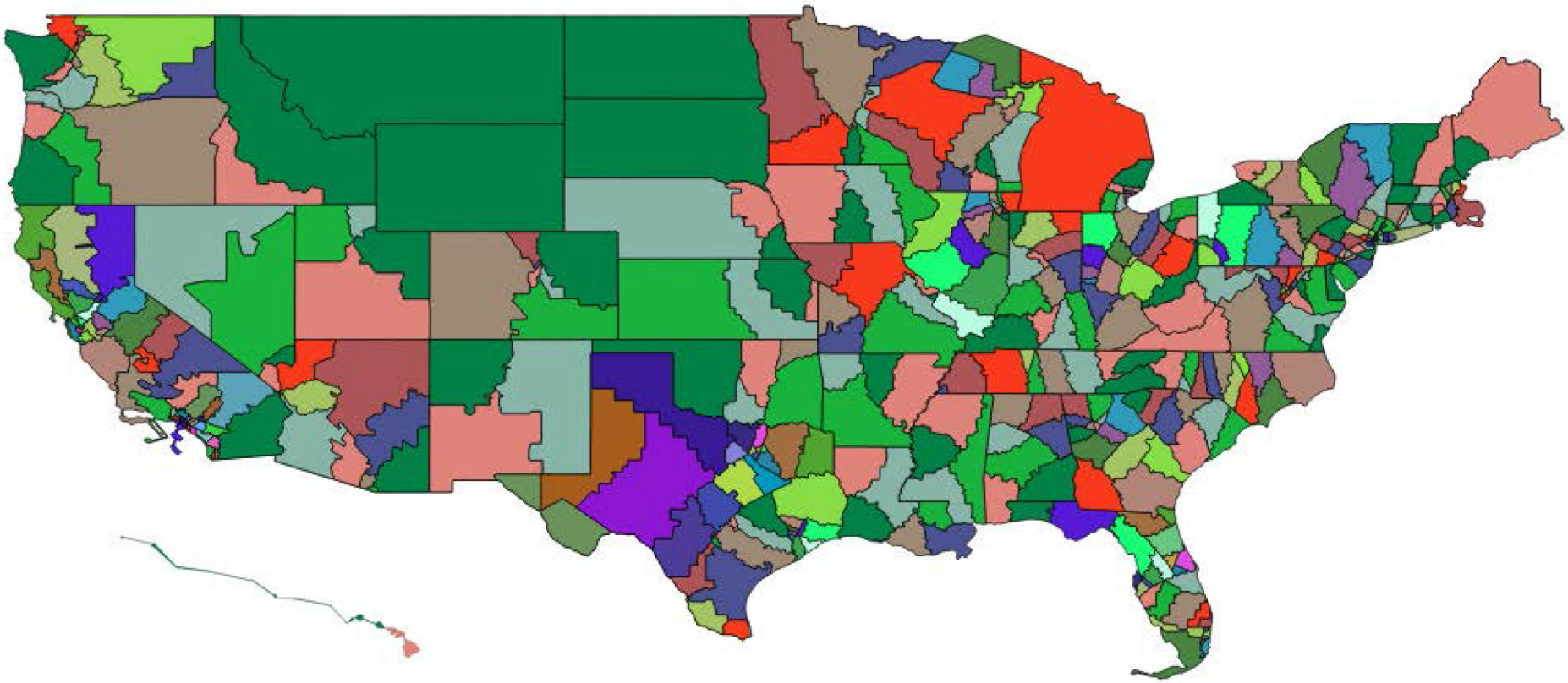
Data Results of MinPop [4]

MaxCompact run on census blocks



MaxCompact being run on census blocks [4]

MaxCompact run on census tracts



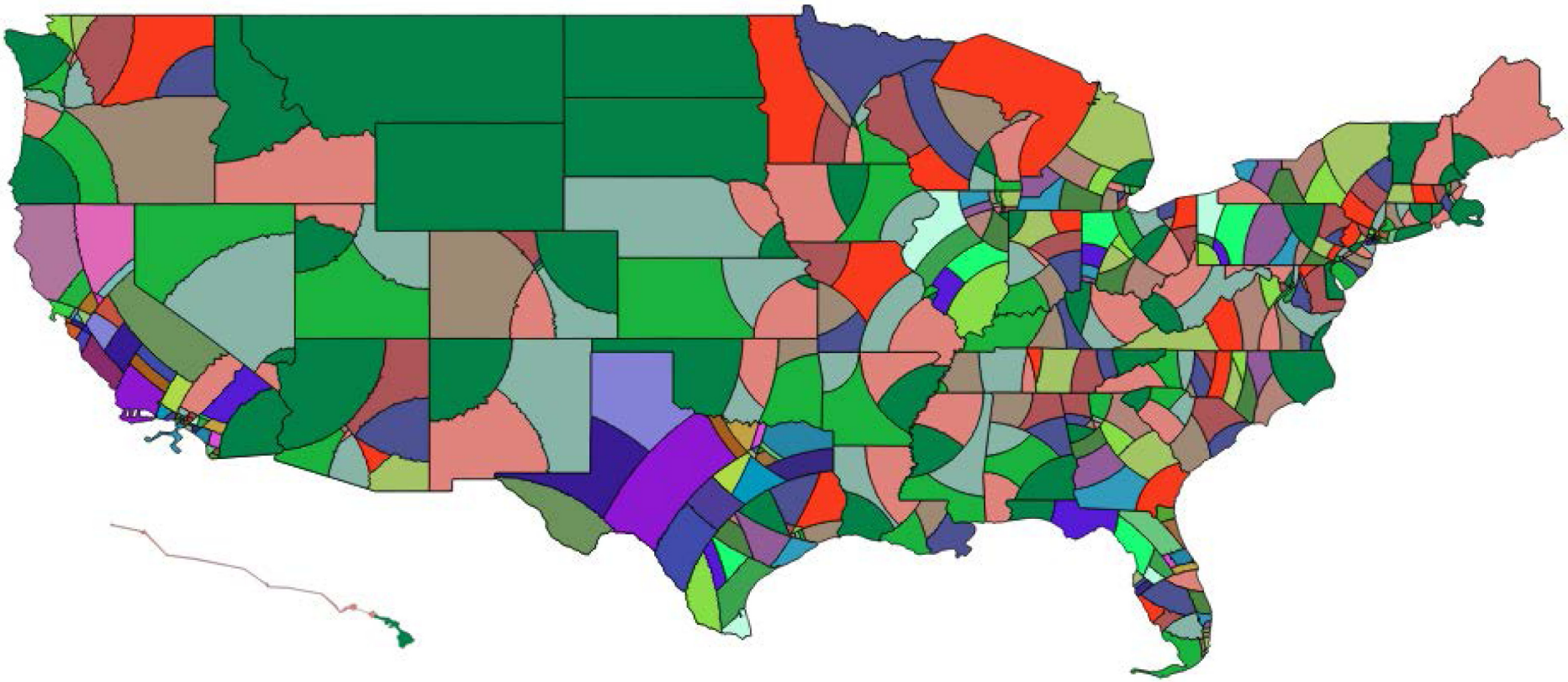
MaxCompact being run on census tracts [4]

Data Results of MaxCompact

Measure	Census Blocks Full Alg	Census Tracts No Pop Swap	Census Tracts Full Alg
Valid States (deviation < 0.5%)	36/43	8/43	36/43
Population Deviation Med (%)	0.00%	1.84%	0.11 %
Population Deviation Med (people)	1	13,775	736
Convex Hull 25th Percentile	0.82189	0.72449	0.73293
Convex Hull Med	0.85537	0.78063	0.77731
Convex Hull 75th Percentile	0.87740	0.82690	0.81376
Polsby-Popper 25th Percentile	0.23476	0.26090	0.24178
Polsby-Popper Med	0.26598	0.30871	0.28837
Polsby-Popper 75 Percentile	0.30701	0.33692	0.32579
Schwartzberg 25th Percentile	0.47934	0.50370	0.48475
Schwartzberg Med	0.51140	0.55189	0.53229
Schwartzberg 75th Percentile	0.54979	0.57534	0.56395

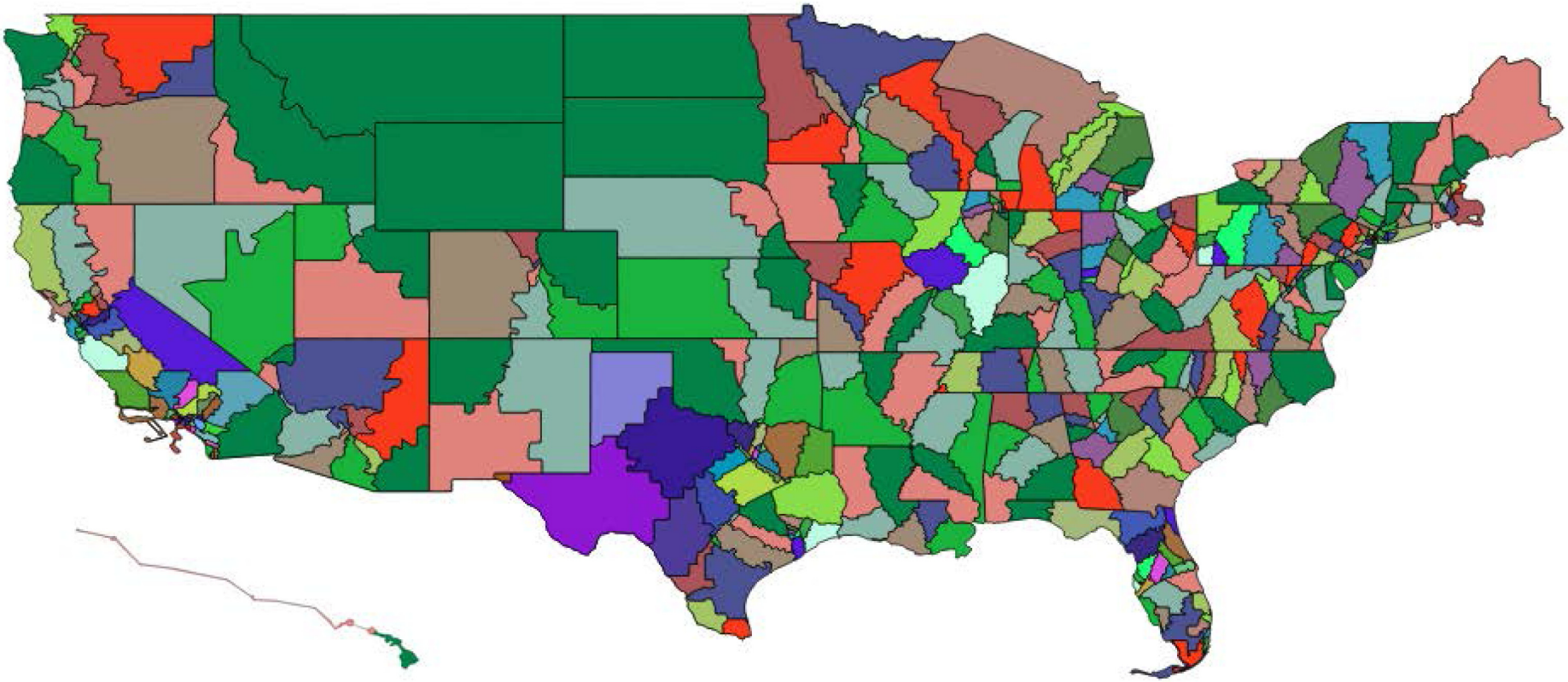
Data Results of MaxCompact [4]

ValidCompact run on census blocks



ValidCompact being run on census blocks [4]

ValidCompact run on census tracts



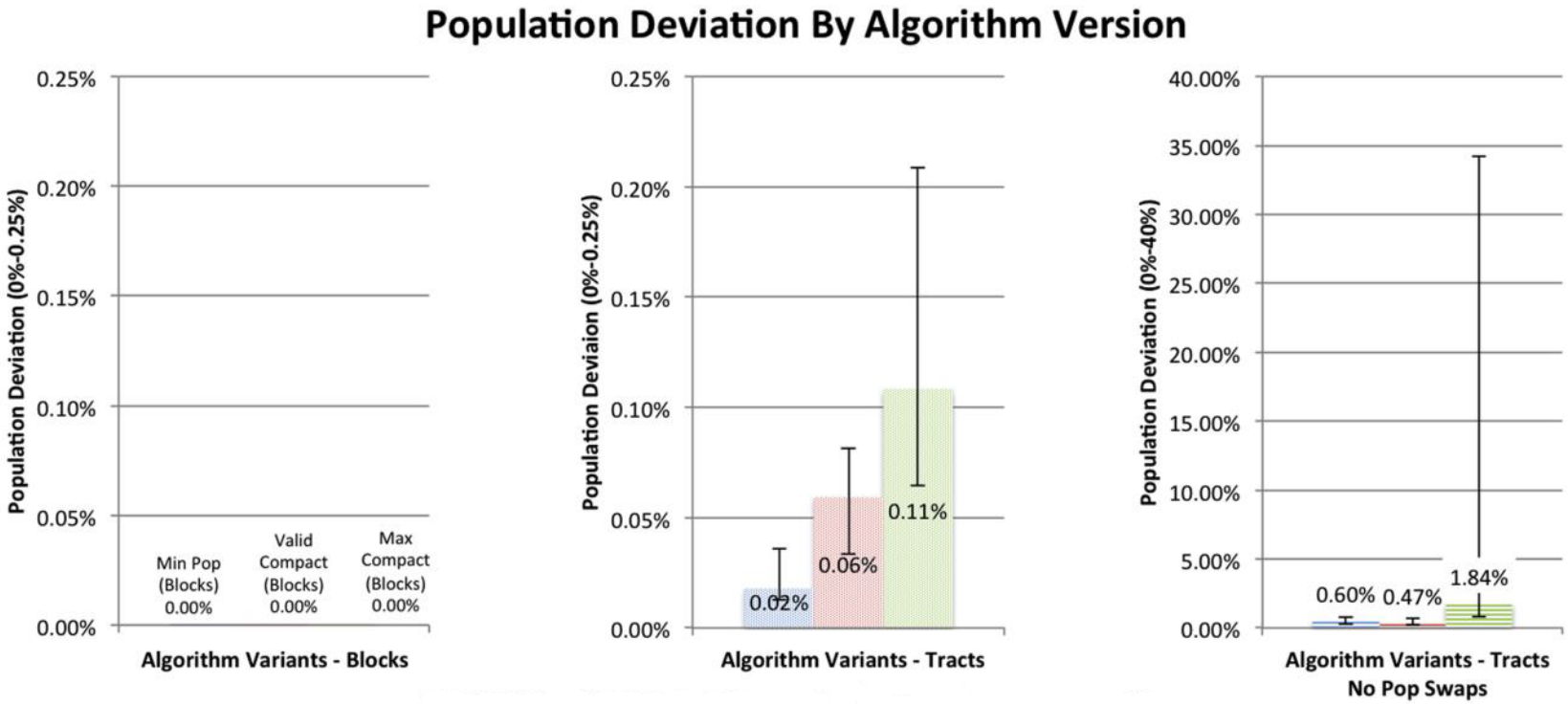
ValidCompact being run on census tracts [4]

Data Results of ValidCompact

Measure	Census Blocks	Census Tracts	
	Full Alg	No Pop Swap	Full Alg
Valid States (deviation < 0.5%)	42/43	23/43	41/43
Population Deviation Med (%)	0.00%	0.47%	0.06%
Population Deviation Med (people)	1	3,291	449
Convex Hull 25th Percentile	0.82489	0.72429	0.72583
Convex Hull Med	0.85929	0.77753	0.76874
Convex Hull 75th Percentile	0.87740	0.80296	0.81066
Polsby-Popper 25th Percentile	0.23476	0.22563	0.23141
Polsby-Popper Med	0.27517	0.27422	0.26915
Polsby-Popper 75th Percentile	0.30701	0.30421	0.31924
Schwartzberg 25th Percentile	0.47934	0.46613	0.47507
Schwartzberg Med	0.51961	0.51365	0.51129
Schwartzberg 75th Percentile	0.54979	0.54638	0.56034

Data Results of ValidCompact [4]

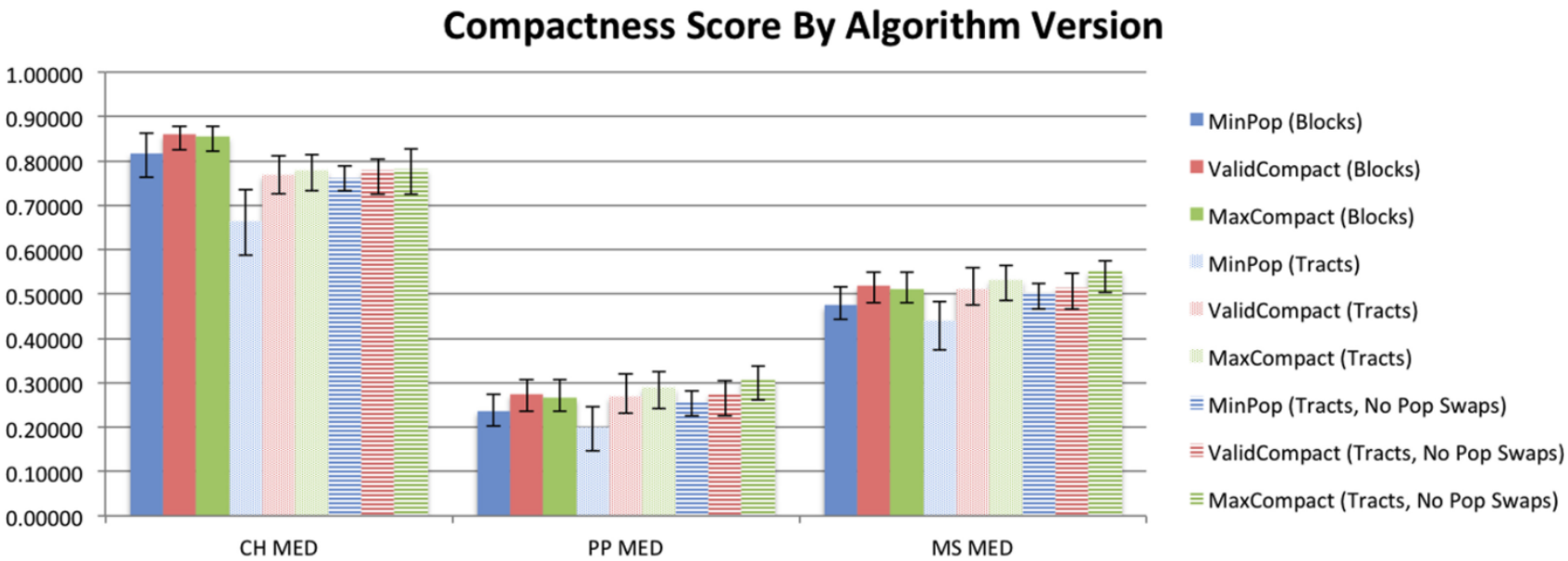
Population Deviation Results



Population Deviation scores on every version of the algorithm [4]

- Scores reflect median values with 25th and 75th percentile error bars
- Scores get worse when using tracts
- Scores also get worse when population swapping is removed

Compactness Results



Compactness scores on every version of the algorithm [4]

- Scores reflect median values with 25th and 75th percentile error bars
- Scores mostly improve when using tracts
- Scores also slightly improve when population swapping is removed

Outline

Closing Remarks

- Source code is available online!
- Not every state will want the same thing
- Human input is still needed for communities of interest
- Majority-minority merge trick can be done by combining population units

Thank you!

Questions?

Bibliography I

https://github.com/newspapercentral/automated-congressional-redistricting


BUREAU, U. C.
2020 census - census block maps, Oct 2021.


COMMONS, W.
File:south carolina us congressional district 6 (since 2013).tif — wikimedia commons, the free media repository, 2020.
[Online; accessed 11-November-2021].


INGRAHAM, C.
America's most gerrymandered congressional districts, Apr 2019.


LEVIN, H. A., AND FRIEDLER, S. A.
Automated congressional redistricting.
ACM J. Exp. Algorithmics 24 (Apr. 2019).

Bibliography II

McGlone, D.
Exploring pennsylvania's gerrymandered congressional districts, Jan 2018.

Schutzman, Z.
Trade-offs in fair redistricting.
In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society* (New York, NY, USA, 2020), AIES '20, Association for Computing Machinery, pp. 159–165.