Automated Congressional Redistricting

HARRY A. LEVIN and SORELLE A. FRIEDLER, Haverford College

Every ten years, when states are forced to redraw their congressional districts, the process is intensely partisan, and the outcome is rarely fair and democratic. In the last few decades, the growing capabilities of computers have offered the promise of objective, computerized redistricting. Unfortunately, the redistricting problem can be shown to be NP-Complete, but there are a number of heuristics that are effective. We specifically define the redistricting problem and analyze several variations of a new divide and conquer algorithm, comparing the compactness and population deviation of our new algorithm to existing algorithms and the actual districts. We offer a comparative component-based analysis that demonstrates the strengths and weaknesses of each algorithm component and the type of input. The comparative analysis shows that there are several ways to produce valid redistricting plans, but each approach has benefits and consequences.

Our new algorithm produces valid results to the redistricting problem in almost every state that undergoes congressional redistricting, offering a new solution to this challenging real world problem. In one version, the algorithm produces plans with the optimal population deviation in 42 out of 43 multi-district states, which is better than most algorithms in the literature. While compactness scores vary, this approach offers new opportunities to improve population deviation. Our output files comply with the accepted format at most government hearings and redistricting competitions, so the results would be compatible with most public participation efforts in 2020.

CCS Concepts: • **Applied computing → Voting / election technologies**; E-government.

Additional Key Words and Phrases: computational geometry, clustering, experimental algorithms, redistricting, gerrymandering

ACM Reference Format:
Harry A. Levin and Sorelle A. Friedler. 2019. Automated Congressional Redistricting. *ACM J. Exp. Algor.* 1, 1 (March 2019), 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Every ten years, state legislatures across the United States redraw the districts for state and federal representatives. The process of redistricting is notoriously partisan and prone to political mischief. Politically motivated redistricting designed to favor a certain group, known as gerrymandering [28], is a persistent problem, and many reformers have sought alternatives to the politicized process [25]. Recently, these have included developing algorithms as a method of reducing the political bias. However, there is considerable debate over what attributes a good district needs to have because there are competing theories on how to define good representation [23]. This has led to a variety of redistricting algorithms with varying results.

Broadly stated, the goal of a redistricting algorithm is to partition a state's population units into a given number of compact, contiguous, and equally sized districts. While redistricting laws differ from state to state, population equality, contiguity, compactness, and communities of interest are

Authors' address: Harry A. Levin, halevin13@gmail.com; Sorelle A. Friedler, sorelle@cs.haverford.edu, Haverford College, 370 Lancaster Avenue, Haverford, PA, 19041.

1084-6654/2019/3-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

almost universal criteria [40]. The version of the redistricting problem that we consider emphasizes population equality, requiring a deviation between districts of at most 0.5% of the average population, which complies with the most recent Supreme Court precedent [49, 52, 53]. Districts must also be contiguous, such that for any two points inside the district there exists a path between the two points that does not cross the district boundary [40]. Contiguity also implicitly requires that a single building, such as houses and apartment buildings, must be in a single district, assuming that an algorithm relies on census blocks for its underlying data.[1] This is consistent with the definition of contiguity in most states [40].

Following H.P. Young's skepticism of compactness measures [68], we exclude compactness from the problem definition, and use it instead to evaluate the quality of redistricting plans. Young demonstrates that each compactness measure fails to capture at least one type of non-compact district, suggesting that all examined compactness measures are flawed [68].

We exclude communities of interest from this analysis because this concept is notoriously vague. A community of interest is a group of people who have similar qualities, such as religion, race, or culture, and live in close proximity [40]. Some reformers suggest that districts should include entire communities of interests to offer like minded voters the same representative [40]. Yet, there is no consensus on which qualities qualify a group as a community of interest because there are flaws with quantitative assessments of communities of interests [43]. State laws suggest that criteria such as socioeconomic status, race, geography, historical interests, culture, traditional neighborhoods, occupations, and lifestyles should factor into the calculation of a community of interest [39], but there is no clear formula for combining the criteria. As a result, the definition of a community of interest and its geographic boundary are subjective [43]. Lacking a precise definition, communities of interest are difficult to incorporate into an algorithm without additional human input, so we do not implement this criterion. The results could be modified to include communities of interest by merging the given population units to reflect a specific community. Even though communities of interest are not explicit features of the algorithms in this analysis, this merge trick works on any algorithm that builds districts from small population units.

Competitiveness and proportionality are other criteria that are scrutinized [40]. Many studies have analyzed the impact of gerrymandering on election competitiveness [2, 14, 27, 29, 33, 42, 62, 67], and recent work has highlighted the preference for proportionality, when the aggregate number of votes for a particular party in a state is proportional to the number of seats that the party wins [5, 60, 65]. Both measures involve predicting the outcome of the next election in the proposed districts based on historical data. The efficiency gap, which measures the number of "wasted votes", has received the most attention, and it has been positioned as a way to measure gerrymandering [13, 60]. However, competitive elections and proportional results are not universally accepted. Election results can be skewed by a number of factors unrelated to the district shape, such as the concentration of Democrats in cities and Republicans in rural ares [20, 21]. Additionally, uncompetitive elections are considered favorable when the legislature tries to increase minority representation through the creation of minority districts [21]. Since competitiveness and proportionality are not yet settled legal requirements, and the interpretation of a good outcome is ambiguous, we exclude election data from our analysis.

1.1 Redistricting Problem Definition

We precisely define the *redistricting problem* and describe the measures that will be used to evaluate it. Redistricting is essentially a set partition problem with additional restrictions. Let the state

[1]Since the census block is the smallest available U.S. Census population unit, we cannot subdivide census blocks, which ensures that any building is in a single district.

graph, $G = (V, E)$, be a connected graph that represents the geography of a given state. For all vertices $v \in V$, v represents a population unit (such as a census block) that has weight $w(v)$ equal to the unit population. There is an edge between vertices $v, w \in V$ if the associated population units' polygonal boundaries are adjacent, sharing at least a vertex. The goal is to partition G into k connected subgraphs each representing a district. These districts, denoted G_i where $G_i = (V_i, E_i)$ and $V_i \subset V, E_i \subset E$, should be created such that the total population of any district $population(G_i) = \sum_{v \in V_i} w(v)$ is equal to the desired population L. The load capacity L is the desired number of people per district under perfectly even representation, i.e., for a state with population $n = \sum_{v \in V} w(v)$, $L = n/k$. The *population deviation* from L is defined as $\varepsilon = (max_i\{|population(G_i) - L|\})/L$.

Definition 1.1. **Congressional Redistricting Problem:** Partition G into k connected subgraphs so as to minimize ε.

We evaluate the effectiveness of an algorithm by the consistency with which it produces plans with low population deviations and high compactness measures. A solution will be considered valid where $\varepsilon \leq \tau$ for $\tau = 0.005$. While population deviation is determined by calculating ε, we evaluate redistricting plan compactness with three standard compactness measures: Convex Hull, Polsby-Popper, and Modified Schwartzberg. *Convex Hull* compactness measures the ratio of the area of the computed district D to the area of the convex hull of the computed district, $area(D)/convexHull(D)$ [47]. *Polsby-Popper* is the ratio of the area of the computed district to the area of a circle which has the same perimeter of the computed district, $\left(4\pi \cdot (area(D)/perimeter(D)^2\right)$ [47]. *Modified Schwartzberg* is the ratio of the circumference of a circle that has the same area as the computed district to the perimeter of this computed district, $\left(2\pi\sqrt{area(D)/\pi}\right)/perimeter(D)$ [47]. Note, there are variations of the Schwartzberg measure, but we choose this one to keep the range between zero and one, which is consistent with our other measures. For all three measures, scores that are closer to zero are considered less compact.

1.2 Results

We introduce a *Divide and Conquer Redistricting Algorithm.* It uses fixed centers at the corners of the state and recursive substates to determine two districts at a time. These districts are determined using a Voronoi approach followed by swapping to ensure contiguity and a small population deviation. Multiple variations of the algorithm are considered, focusing on minimizing population deviation, maximizing compactness, and maximizing compactness subject to a constraint ensuring a valid population deviation. These variations are analyzed experimentally on all 43 multi-district states and the variation maximizing compactness subject to a valid population deviation is the resulting recommended algorithm.

Our new algorithmic results are compared to the existing districts and a simulated annealing approach [55]. In one variation, our algorithm produces redistricting plans with the optimal population deviation of one person in every state except New York. This population deviation is better than the simulated annealing approach although the compactness scores are worse on average [55]. Our new algorithm produces valid results to the redistricting problem in almost every state that undergoes congressional redistricting, offering a new solution to this challenging real world problem.

2 BACKGROUND LITERATURE

There are a large number of possible ways to draw congressional districts. Altman has shown that the redistricting problem is NP-Complete [4]. There are approximation algorithms that are related to redistricting, but they do not satisfy all of the redistricting problem constraints as stated. The load balancing problem has a known 4/3 approximation, which is too large to guarantee the 0.5% population deviation in our problem definition [30]. The set partitioning problem can be solved with the Karmarkar Karp (KK) algorithm, which has a $O(\frac{1}{n})$ approximation bound [38]. Given a large enough value of n, the population bound of KK would satisfy the 0.5% population deviation in our problem definition. However, the algorithm only meets the population deviation constraint and cannot obviously be converted into a version of the redistricting problem when contiguity is required.

There are many existing heuristic redistricting algorithms, which fall into two categories: partitioning algorithms and swapping algorithms. In the following sections, we define the types of algorithms and offer examples from the redistricting literature. There are a number of evaluation methods covering many compactness and competitiveness scores set to different scales. We focus the review on the consistency to which an algorithm produces valid plans, which is more easily comparable between papers. For more detailed information about further redistricting algorithms, see this comprehensive survey [57].

2.1 Partitioning Algorithms

Partitioning algorithms take a geographic region of population units and separate them into k contiguous sets. Partitioning algorithms fall into greedy and recursive approaches each with different variations.

Most of the greedy redistricting algorithms are based on Voronoi approaches [15, 22, 31, 34, 56, 59, 61]. Voronoi algorithms are effective because Voronoi diagrams produce a fixed number of contiguous, compact partitions. Given a set of centers, the Voronoi-based redistricting algorithms assign population units to the closest open center in the set that has not reached a population threshold. The selection of centers has a significant impact on the outcome. Svec *et al.* choose sites based on the k largest population units that are geographically distributed throughout the state and produce a population deviations of 0.74% [61].[2] Ricca *et al.* chooses centers based on minimizing the longest path from a center to the furthest assigned population unit and gets a much worse deviation [56]. While the Svec *et al.* algorithm is one of the more effective algorithms in the literature, it runs the risk of subdividing a single house into separate districts.

A k-means based redistricting algorithm is a variation of a Voronoi approach, in which centers can move after each iteration. Like the Voronoi approach, the initial selection of centers has significant impact on the k-means output. Sparks chooses centers in North Carolina randomly and produces a population deviation greater than 150% [59]. Bottman *et al.* selected centers based on the placement of the actual districts and produced better output with population deviations around 2.5% [15]. The drastic differences in population deviation demonstrate the importance of the initial selection of centers. In addition, the size of the population units may also impact the output. Cohen-Addad *et al.* select centers randomly, similar to Sparks, but districts are produced from census blocks instead of census tracts, producing an optimal deviation of 1 person or less in 6 out of 6 states [22]. While the Cohen-Addad algorithm demonstrates effective results, the districts are not as compact as they appear since the census block geometry is represented by a single point [22].

Moment of inertia algorithms offer another variation of Voronoi techniques [31]. These algorithms alternate between generating a set of districts from fixed centers and generating new centers from a set of districts. Spann *et al.* [31] builds on an approach from Hess *et al.* [34] and implements a moment of inertia algorithm that produces a deviation within 2%. The results from Hess *et al.* are not comparable to modern algorithms because technology limitations restricted their sample size

[2]While this deviation is not valid under our definition of the problem, it is within the acceptable range, according to some court precedent [49, 52, 53].

to less than 300 population units [34]. Spann *et al.* show results improve with more population units and faster technology that can iterate through more plans [31]

While Voronoi algorithms assign units to districts based on the distance to the closest seed, Constrained Polygonal Spatial Clustering (CPSC) uses an objective function to weigh a set of criteria. Joshi *et al.* implement a CPSC algorithm with an objective function, requiring population equality within 1%, contiguity, and maximization of Schwartzberg compactness [35]. They produce multiple redistricting plans with census tracts in Nebraska and Indiana, showing that multiple seed selection approaches result in population deviations within 1% [35].

Recursive algorithms are also effective because they break the redistricting problem into smaller subproblems, which are easier to solve. The shortest split-line algorithm is the most common [12, 37, 64]. Using a divide and conquer method, the goal at every step is to find the shortest line that cuts the state into two pieces such that these pieces contain equal portions of the population. Benn and German show experimentally that a 3% population deviation is achievable [12]. Kalcsics *et al.* use a similar approach on a small sample of German zip codes and produce a population deviation around 15 people [37]. Both algorithms are computationally taxing: Benn and German provide a $O(N^3)$ complexity [12] while Kalcsics *et al.* provide large runtimes [37]. These results suggest that the shortest split-line algorithm may only be feasible in small states. Although the algorithm has the advantage of being simple, it may not produce legal plans since it could subdivide a single house into separate districts.

The Diminishing Halves Algorithm is another divide and conquer approach that uses least squares regression analysis to recursively subdivide districts [31]. Spann *et al.* choose the line perpendicular to this best-fit line to avoid dividing major cities and produces plans with population deviations around 2% [31].

Linear programming and Monte Carlo algorithms are worth mentioning for completeness. Caro *et al.* show that linear programming can effectively limit overcrowding in school redistricting, yet the results were performed on small samples [18]. Linear programming approaches to redistricting do not tend to scale for large problem sizes [3, 18]. Assis *et al.* redistrict at a city, but they solve a different flavor of the redistricting problem with weaker bounds on population deviation that rely on an existing assignment of people to utility districts [7]. Magleby *et al.* implement a redistricting algorithm based on a Monte Carlo simulation but define a large population deviation bound at 1.5% and only show results for 3 states [46].

2.2 Swapping Algorithms

Swapping algorithms improve an existing redistricting plan based on given criteria. The algorithms evaluate population units on the boundary of two districts and swap these boundary units from one district to the other. Swapping algorithms can be categorized as local search swapping, which only allow beneficial swaps [32] and metaheuristic swapping, which allows some detrimental swaps [55]. Both techniques require a complete redistricting plan as an input, and the method attempts to improve the result by reassigning population units to different districts.

A local search swapping algorithm reassigns population units between districts only if the swap improves the plan until there are no more swaps that can improve the output [32]. Kaiser, Nagle, and Hayes implement local search swapping algorithms for counties in various states with varying results [32, 36, 48]. Kaiser produces congressional districts in Illinois with 26.4% population deviation [36] while Nagel produces fewer districts on a smaller subset of Illinois with 4% population deviation [36]. Hayes produces congressional districts in North Carolina with 4.4% population deviation [32]. Local search swapping is limited by the initial districts and a suboptimal initial set can exhaust beneficial swaps before producing a valid plan [32].

Other work has focused on making detrimental swaps to avoid getting stuck in a local optimum plan [32, 44, 45, 55]. Several metaheuristics allow more of the sample space to be explored by allowing these swaps that make the output worse. Simulated annealing introduces non-deterministic random swaps, which can increase population deviation to improve other objective criteria [32]. Hayes compares a simulated annealing approach to a local search swapping approach and improves the output from 4.4% to under 1% [32]. Macmillan and Pierce implement a simulated annealing approach that produces results with good population deviation, but does not scale [45]. While, Macmillan demonstrates a method for speeding up this algorithm, it is not clear whether the population deviation is maintained in larger cases [44] Brian Olson offers another simulated annealing approach and produces population deviations under 1% in all U.S. states. In many cases, the Olson algorithm produces deviations several orders of magnitude lower, such as the 0.02% population deviation in his Pennsylvania plan. We believe this algorithm to be the best in the literature because it is scalable, produces population deviations within or near the legal range in all states, and consistently produces plans with the highest compactness scores compared to other algorithms. (We will compare our algorithm to Olson's in Section 4)

Tabu search is another metaheuristic swapping algorithm that has been applied to redistricting. This approach limits similar outputs by preventing the number of times a search path is taken. The algorithm introduces “tabu" paths that make the result worse initially to explore different areas of the solution set. Bozkaya *et al.* implement a tabu search redistricting algorithm for the city of Edmonton and produces a population deviation around 1% [16]. While this deviation is low, there is no evidence that tabu search algorithms scale to the average U.S. state.

Genetic algorithms, another metaheuristic swapping algorithm, combine features of high scoring plans from an objective function, mimicking the evolutionary process of natural selection. Bacao *et al.* describes a genetic algorithm that is initialized randomly and mutated based on population equality, compactness, and contiguity with 5,000 generations [11]. While Bacao *et al.* do not apply the algorithm to the congressional redistricting problem, Joshi *et al.* implement it for Nebraska and Indiana‘s congressional districts, showing population deviations well over the legal threshold in both states [35]. While Joshi *et al.* use a weighted function to combine criteria in the objective function, Vanneschi *et al.* use a NSGA-II technique to optimize both compactness and population deviation simultaneously, yielding population deviations well under the legal threshold in Nebraska, Indiana, Georgia, and Pennsylvania [63]. Baas implements a similar genetic algorithm with more components in the objective function [10]. While previous algorithms initialize the first generation randomly, Baas initializes both randomly and based on the actual districts [10]. In addition to the standard objective function criteria, population equality, compactness, and contiguity, Baas also includes historical election data, incorporating competitiveness and proportionality [10]. Baas provides a sample result for each state, producing many congressional districts with a population deviation under 1% [10]. However, it is unclear what weights are used in the objective function and how many generations were run.

2.3 Limitations of Existing Approaches

There are a few fundamental gaps in the literature on redistricting algorithm effectiveness. Most of the algorithms in the literature do not show evidence that they can produce valid population deviations for the congressional redistricting problem in all states, but instead only demonstrate a handful [12, 15, 31, 32, 35, 36, 48, 56, 59, 63]. The Olson and Baas algorithms stand out by producing contiguous redistricting plans with population deviations under 0.5% in most of the 43 states that do congressional redistricting [10, 55].

Reproducibility and a full comparative analysis with existing algorithms is another concern in the literature. We suggest that the results in each state should include a block-equivalency file [3] for each redistricting plan that is presented and the code used to generate the results. The block-equivalency file provides enough information to calculate any population or geographic measure, and it is the standard format for government hearings [6, 26, 50, 51] and the redistricting software used in a number of redistricting competitions [9]. By providing source code, the experimental results can be verified and manipulated for further study.

While Olson does not provide the detailed weights and number of executions, he provides the original source code, which is more transparent than most sources [55]. Since this approach shows the best results of any redistricting algorithm, we use it as the main point of comparison for our new algorithm.

3 DIVIDE AND CONQUER REDISTRICTING ALGORITHM

We introduce a new recursive Divide and Conquer Redistricting Algorithm, which combines partitioning and swapping components. The divide and conquer portion of the algorithm allows us to separate the overall district into pieces so that each redistricting step only needs to focus on creating two districts of a given population size. The redistricting step has two components at each recursive iteration: Voronoi and swapping components. To calculate the Voronoi seeds, the algorithm draws the axis-aligned bounding rectangle, called the district envelope,[4] and creates versions of a two district partition from each of the corners of this rectangle. The Voronoi component divides the state in half assigning the next closest population unit to one district until the population is greater than the ideal threshold, L. The remaining units are assigned to another district. The swapping component adjusts for breaks in contiguity and performs local search swapping to improve the population deviation between the two districts. The version of the two district partition with the best population deviation (or, in a variation, the best compactness) is selected and recursively subdivided into more districts.

More specifically, the divide and conquer component (shown in Figure 1) is given the number of remaining partitions to create, k, and the remaining portion of the state that should be divided into these k partitions, subgraph G_s. If the number of remaining partitions is odd, the desired population to achieve in one congressional district in the redistricting step is set to $L = \frac{n}{k}$. The two-district redistricting step will then create one district close to this desired population and another with the remaining population from G_s. If, instead, the number of remaining districts is even, the population threshold will be set to the ideal population of an equally weighted partition, $population(G_s)/2$. The redistricting step will be run recursively until the number of remaining partitions is 1.

The two-district redistricting step (see Figure 2) that we will describe in detail for the remainder of this section has the following components: a Voronoi step where multiple possible source sites are chosen and Voronoi partitions are created, a Contiguity Swapping step where population units are swapped between the two districts in order to create contiguous districts, a Population Swapping step where population units are swapped in order to create a low population deviation while maintaining contiguity, and a Maximization step where the resulting redistricting options, based on different Voronoi site choices, are compared and the best is chosen.

[3]A block-equivalency file is a csv file with two columns: the population unit identifier and the district to which it is assigned.
[4]The envelope of a polygon is defined as the smallest possible, axis-aligned rectangle with a equirectangular projection of the Earth that covers all of the coordinates in the polygon [24].

```
function DivideAndConquer(remainingDistricts, districtToDivide, maxFunction)
    if remainingDistricts is 1 then
        return districtToDivide
    end if
    if remainingDistricts is odd then
        $L \leftarrow n/k$
        remainingDistrictsCall1 ← 1
        remainingDistrictsCall2 ← remainingDistricts −1
    else
        $L \leftarrow$ population(districtToDivide) / 2
        remainingDistrictsCall1 ← remainingDistricts / 2
        remainingDistrictsCall2 ← remainingDistricts / 2
    end if
    leftDistrict, rightDistrict ← RedistrictTwoDistricts(districtToDivide, $L$,
      maxFunction, $false$)
    leftDistrictsList ← DivideAndConquer(remainingDistrictsCall1, leftDistrict,
      maxFunction)
    rightDistrictsList ← DivideAndConquer(remainingDistrictsCall2, rightDistrict,
      maxFunction)
    return concatenateLists(leftDistrictsList, rightDistrictsList)
end function
```

Fig. 1. **Divide and Conquer Redistricting Algorithm**: This is the main function for the redistricting algorithm. The work of dividing the given district in two is done by the redistrictTwoDistricts algorithm that attempts to achieve a district with population close to the given value L. A list of created districts is returned.

3.1 Voronoi Component

The Voronoi component we use modifies the standard Voronoi partitioning to operate with a single site a and a given capacity threshold L, with the goal of partitioning the given region to be districted, $G_s = (V_s, E_s)$, into two regions. (See Figure 3 for pseudocode.) In order of distance from the site's region, we add population units until the site is at capacity. Specifically, letting A be the partition we are creating from a, start by assigning the closest $v \in V_s$ based on the population unit centroid to A. Continue assigning $v \in V_s$ to A in order of proximity until $\sum_{v \in A} w(v) > L$ where L is the given population capacity for the district region. The remaining vertices $V \setminus A$ are assigned to the other partition B. Note that while the method assigns population units that are nearby, it does not guarantee contiguity of A or B since distance is determined geographically and not based on the underlying state graph. Guaranteeing contiguity for A and B will be handled in the next section.

The choice of site to consider is clearly critical to the quality of the resulting districting. Since the algorithm only selects one site for a two district partition, a poor site selection could result in districts that resemble the unusual shapes that we aim to reduce. We select one of the four corners of the envelope of the polygonal boundary of the district region because it limits the amount that one district can surround the other in the two district partition. Since the closest population unit is iteratively added to a district in the Voronoi component, the district has a radius, represented by the grey circle in Figure 4. If we select a site inside the bounding box or on a non-corner edge of the bounding box, the radius may only reach one edge of the bounding box, or none at all. In these

```
function RedistrictTwoDistricts(districtToDivide, $L$, maxFunction, tryingAgain)
    smallestEnclosingRectangle ⟵ envelope of polygonal boundary of districtToDivide
    districtPossibilities ⟵ $\emptyset$
    for each corner of the smallestEnclosingRectangle do
        district1, district2 ⟵ ModifiedVoronoi(corner, $L$, districtToDivide)
        district1, district2 ⟵ ContiguitySwapping(district1, district2)
        district1, district2 ⟵ PopulationSwapping(district1, district2, $L$,
            tryingAgain)
        districtPossibilities ∪ { (district1, district2) }
    end for
    if populationDeviation(ChooseBest(districtPossibilities, MinPop)) > 1 person
            and not tryingAgain and maxFunction is MinPop then
        RedistrictTwoDistricts(districtToDivide, $L$, maxFunction, $true$)
    end if
    return ChooseBest(districtPossibilities, maxFunction)
end function
```

Fig. 2. **Two-district Redistricting**: This method divides a region into two districts, where one has the given population size L and the other has the remaining population. Each corner of the bounding rectangle is tried as a possible site for the modified Voronoi component, and the best resulting districting is returned.

```
function ModifiedVoronoi(site, $L$, districtToDivide)
    district1 ⟵ site
    district2 ⟵ $\emptyset$
    while population(district1) $\leq L$ do
        $u \longleftarrow \arg\min_{u \in \text{districtToDivide} \setminus \text{district1}} \text{distance}(u, \text{site})$
        district1 ⟵ district1 ∪ $u$
    end while
    district2 ⟵ districtToDivide \ district1
    return district1, district2
end function
```

Fig. 3. **Modified Voronoi**: The modified Voronoi component of the redistricting algorithm creates two partitions based on a single given site and a population capacity for that site, L. Population units geographically close to the site are added to its partition until the site is at capacity; the remaining population units are added to the other partition.

scenarios, the second district, represented by the white space inside the bounding box, has a larger perimeter and is unlikely to produce high scores for the Polsby-Popper and Modified Schwartzberg measures. However, the sites at the corners of the bounding box touch two sides of the bounding box, reducing the perimeter of the second district, which will likely improve the compactness scores.

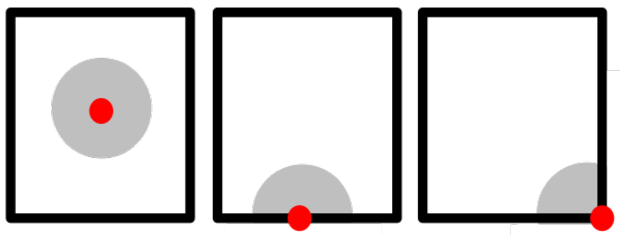


Fig. 4. The selection of sites impacts the amount that one district can surround the other district in the two district partition. The site at the corner of the bounding rectangle limits this, so we only select sites at the corner of the bounding rectangle.

```
function ContiguitySwapping(district1, district2)
    if district1 contains non-contiguous components then
        large ← contiguous component of district1 with most polygonal bounding vertices
        district2 ← district2 ∪ district1 \ large
        district1 ← large
    end if
    if district2 contains non-contiguous components then
        large ← contiguous component of district2 with most polygonal bounding vertices
        district1 ← district1 ∪ district2 \ large
        district2 ← large
    end if
    return district1, district2
end function
```

Fig. 5. **Contiguity Swapping**: The contiguity swapping component fixes breaks in contiguity by reassigning non-contiguous population units to the other district in the two-district partition. Once the non-contiguous units in both districts are reassigned, the two-district partition will have two contiguous districts.

3.2 Contiguity Swapping

The two district partitioning created by the Modified Voronoi component may not create districts that are contiguous; here, we introduce a Contiguity Swapping component to resolve this issue (see Figure 5). The intuition behind this component is that if each district contains a large contiguous component and many smaller disconnected components, each smaller component can be swapped to the other district; since there are only two districts and the state graph is connected, any component not contiguous with one district is contiguous with the other.

Specifically, each district's contiguous components are first identified. The largest contiguous component remains in the district, where size is determined based on the number of vertices on the polygonal boundary of the district. (The number of vertices is used as a proxy for the total population of the component since it is faster to calculate with the data structures used in our implementation and in practice most population units on the district boundary contribute to this vertex count.[5]) The remaining non-contiguous components are swapped to the other district. This set of steps is performed once for each district. Since there are only two districts, the swapped

[5]Future work could consider using the actual total population.

components will be contiguous with the other district. Next we show that the resulting two districts are indeed contiguous.

Lemma 3.1. *The Contiguity Swapping component returns two contiguous districts.*

Proof. The Contiguity Swapping component is given a two district partitioning made up of a set of disjoint contiguous subgraphs. Let `district1` $= \{A_1 \cup A_2 ... \cup A_x\}$ and `district2` $= \{B_1 \cup B_2 ... \cup B_y\}$.

If $x = 1$, `district1` is already contiguous and so the first part of the algorithm is skipped. Otherwise (when $x > 1$), let A_x be the component selected as largest (based on the number of vertices bounding the region) from `district1`. The remaining elements, $\{A_1 \cup A_2 ... \cup A_{x-1}\}$, will be reassigned to `district2` leaving `district1` $= A_x$. Since A_x is contiguous by definition, `district1` is contiguous.

Now `district2` $= \{A_1 \cup A_2 ... \cup A_{x-1} \cup B_1 \cup B_2 ... \cup B_y\}$. Since $A_1, A_2, ... A_x$ are mutually disjoint, the state graph is connected, and there are only two districts, each component $A_1, A_2, ..., A_{x-1}$ must be contiguous with at least one component in B_1, B_2, ... B_y. Let the new disjoint contiguous components of `district2`, now containing some population units that were originally part of `district1`, be $B'_1, B'_2, ..., B'_y$.

If $y = 1$ after these population units are added to `district2`, then `district2` is contiguous and the algorithm finishes. Otherwise (when $y > 1$), let B'_y be the component selected as largest (based on the number of vertices bounding the region) from `district2`. The remaining elements, $\{B'_1 \cup B'_2 ... \cup B'_{y-1}\}$, will be reassigned to `district1`. Since B'_y is contiguous by definition, `district2` is contiguous.

But what about `district1` - is it still contiguous? Now `district1` $= \{A_x \cup B'_1 \cup B'_2 ... \cup B'_{y-1}\}$. Recall that B'_y is not contiguous with B'_i for any i by definition and that B'_i is also not contiguous with any B'_j for any j. Since the state graph is connected, this means that each B'_i is a neighbor of A_x. Therefore, `district1` $= A'_x = \{A_x \cup B'_1 \cup B'_2 ... \cup B'_{y-1}\}$ is contiguous. □

3.3 Population Swapping Component

To improve the population deviation, we introduce another swapping component. This local search component swaps vertices between the two sets that improve the difference in population until there are no more beneficial swaps. The pseudocode for this component is shown in Figure 6. The set of population units in the larger district that neighbor at least one population unit in the other district are called *swappable* population units. The swappable population unit that minimizes the population deviation between the two districts and does not break contiguity is swapped and removed from the swappable set. We continue swapping population units from this set until there are no more swappable population units.

Since the full swapping algorithm takes significant time for a large set of population units, we initially run one level of swaps to determine if the optimal population deviation is achieved with this partial component. We will refer to this as the *partial population swapping component*, in which the swappable population units are only calculated once. This component starts in Figure 2, when *RedistrictTwoDistricts* is recursively called where the flag `tryingAgain` is set to false. In most states, this approach is sufficient for reaching the optimal population deviation. However, in some states we need the full component that recalculates a new set of swappable population units after previous sets are exhausted. This full component is turned on if a perfect deviation cannot be found with the partial component. The full component starts in Figure 2 when *RedistrictTwoDistricts* is recursively called where the flag `tryingAgain` is set to true.

```
function PopulationSwapping(district1, district2, $L$, tryingAgain)
    lastDeviation =$\infty$
    while |population(district1)$-L$| > 1
        and |population(district1)$-L$| < lastDeviation do
        lastDeviation $\leftarrow$ |population(district1)$-L$|
        moreDevDistrict $\leftarrow$ arg $\max_{d \in \{district1, district2\}}$(|population($d$) - $L$|)
        lessDevDistrict $\leftarrow$ district $\in$ {district1, district2 | district $\neq$ moreDevDistrict}
        swappable $\leftarrow$ { $v \in$ moreDevDistrict | $v \in$ neighbors(lessDevDistrict)}
        while swappable $\neq \emptyset$ do
            minDevUnit $\leftarrow$ arg $\min_{u \in \text{swappable}}${ |population(moreDevDistrict)$-L$| |
              IsSwappable($u$, moreDevDistrict, lessDevDistrict) }
            if minDevDistrict = $\emptyset$ then
                break
            end if
            moreDevDistrict $\leftarrow$ moreDevDistrict \ { minDevUnit }
            lessDevDistrict $\leftarrow$ lessDevDistrict $\cup$ { minDevUnit }
            swappable $\leftarrow$ swappable \ { minDevUnit }
        end while
        if not tryingAgain then
            break
        end if
    end while
end function

function IsSwappable($u$, $d1$, $d2$)
    return isContiguous(d1 \ {u}) and isContiguous(d2 $\cup$ {u})
end function
```

Fig. 6. **Population swapping component**: The population swapping component moves units from one district to the other that improve the overall population deviation of the two-district partition. The component continues to swap units until no beneficial swaps exist.

3.4 Maximization Function

Four variations of the two district partition are generated, one from each site based on the rectangular envelope of the state geometry. A maximization function is used to select the best version based on population deviation and compactness. We analyze different maximization functions to determine which one produces the best result. The pseudocode is given in Figure 7. *MinPop* selects the partition with the lowest population deviation, *MaxCompact* selects the partition with the largest average of the Modified Schwartzberg compactness measure, and *ValidCompact* selects the partition with the best average Modified Schwartzberg compactness score with a population deviation less than or equal to 0.5% if available. Otherwise, it selects the same two-district partition that would be chosen in the MaxCompact version. MinPop uses the full population swapping component when run on tracts. The other versions use the partial population swapping component. We analyze the results of these options and the full algorithm in the next section.

```
function chooseBest(districtPossibilities, maxFunction)
    switch maxFunction do
        case MinPop
            return $\arg\min_{\text{districts} \in \text{districtPossibilities}}$ populationDeviation(districts)
        case MaxCompact
            return $\arg\max_{\text{districts} \in \text{districtPossibilities}}$ avgModSchwartz(districts)
        case ValidCompact
            validDistricts ← {districts ∈ districtPossibilities |
               populationDeviation(districts) ≤ 0.005}
            if validDistricts ≠ ∅ then
                return $\arg\max_{\text{districts} \in \text{validDistricts}}$ avgModSchwartz(districts)
            else
                return $\arg\max_{\text{districts} \in \text{districtPossibilities}}$ avgModSchwartz(districts)
            end if
end function
```

Fig. 7. We examine three maximization functions: `MinPop`, `MaxCompact`, and `ValidCompact`. The full population swapping component is only run for `MinPop` on census tracts. We select the two-district partition from the four sites based on this maximization function.

4 EXPERIMENTAL RESULTS

We examined the effectiveness of our new algorithm by comparing the population deviation and compactness scores against the Olson Algorithm and the current congressional districts. The pseudocode descriptions of our algorithm can be found in Section 3 and the full code used to generate these results is available online.[6] Information on the current congressional districts is also available online [8].

We ran the algorithm on all 43 states containing more than one congressional district; these represent large and small states by geography and population. We use the official census tract and census block data for our algorithm [19] and modify the maps to create a connected graph of population units. The following states required modifications to connect islands to the rest of the state: California, Florida, Hawaii, Kentucky, New York, and Rhode Island. While other states have geographic islands, the census population units cover some bodies of water. For example, Maryland has some islands in the Chesapeake Bay, but there are existing population units that cover all of the Chesapeake Bay. To modify a state that is unconnected, we create an extra population unit with zero population that shares at least two coordinates with the unconnected units and the closest connected parts of the state. Unless order is otherwise specified, we use the provided population unit order in the census data in the experiments.

To calculate district compactness, we use the Convex Hull, Polsby-Popper, and Modified Schwartzberg measures, which are based on district shape, area, and perimeter [47]. Section 1.1 describes each measure further. Since there is no universal measure of compactness, we use three measures to show different kinds of compactness [68]. It is possible for a district to have a high Convex Hull compactness score and low Polsby-Popper and Modified Schwartzberg scores (and visa versa).[7]

[6]https://github.com/newspapercentral/automated-congressional-redistricting
[7]The three district measures were calculated using the Vivid Solutions Geometry package, which has built in methods for these calculations [1].

The Polsby-Popper and Modified Schwartzberg measures have similar definitions, but we include both for completeness. For each measure, we calculate the average of a state's computed districts.

4.1 Divide and Conquer Redistricting Algorithm

We test multiple versions of the Divide and Conquer Redistricting Algorithm to understand the impact of each component on the final results. We compare the effects of running the algorithm based on census blocks versus census tracts, as well as the impact of the population swapping component on the algorithm results. We examine all three maximization functions defined in Section 3.4: MinPop, MaxCompact, and ValidCompact.

The results, shown in Figures 10, 11, and 12, give the resulting districting maps[8] as well as the median population deviation, the median compactness scores, and the 25th and 75th percentiles for each compactness score. The column labeled "No Pop Swaps" shows the results without the population swapping component (including the modified Voronoi and contiguity swapping components only) as compared to the results for the full algorithm ("Full Alg"). States with a single district are shown in the map, but were excluded from the calculations since the district geometry is simply the state geometry. (Alaska is a single district and is not shown on the maps.) Note that many bodies of water are covered by the census population units and are assigned to districts.

Figures 8 and 9 provide population deviations and compactness scores for all of our algorithm versions, the Olson Redistricting Algorithm, and the 113th congressional districts. Since the results were run on the same states, we can compare the effectiveness of each algorithm component to the Olson Redistricting Algorithm and actual districts.

As a feasibility check, we calculated the algorithm runtime. The algorithm took the least amount of time for the MaxCompact version run on census tracts in New Hampshire without the swapping component: 5s. The algorithm took the most amount of time for the MinPop version run on census blocks in Texas with the swapping component:1d8h4m8s. The median runtime and standard deviation across all results was 2m42s and 4h0m22s respectively. These runtimes are better than the Olson Algorithm, which sometimes ran for over a week before finding a plan that met the criteria [55]. Each state has more census blocks than tracts, so experiments with census blocks will have larger runtimes. A full list of values is available online.[9] All the experiments were run on a Linux machine with an Intel(R) Xeon(R) processor, CPU E5-1620 v2 at 3.70GHz, and 128GB of RAM.

4.1.1 Census blocks versus tracts. We test the Divide and Conquer Redistricting Algorithm on census tracts and census blocks to demonstrate the impact of different population units. Census tracts tend to have a larger area and a larger population than census blocks. While census tracts typically range from 1,200 to 1,800 people [17], census blocks usually have less than 100 people, and in many cases have zero people. Since census blocks are smaller, there are also more census blocks than census tracts in each state, which increases the size of the congressional redistricting problem.

Running the algorithm with census blocks improves the resulting districting's population deviation in almost every state. This is not surprising, since with smaller units an algorithm is more likely to have possibilities open to it that allow summation to the desired population amount. Using census blocks also improves the Convex Hull compactness measure in most states, while the impact on the Polsby-Popper and Modified Schwartzberg measures are inconsistent. Census blocks are smaller by perimeter and area than census tracts. Adding a new census block to a district will have a minimal impact on the overall district shape, while a new census tract could significantly change

[8]All results are available as equivalency files online: https://bitbucket.org/hlevin/redistricting-results
[9]https://bitbucket.org/hlevin/redistricting-results

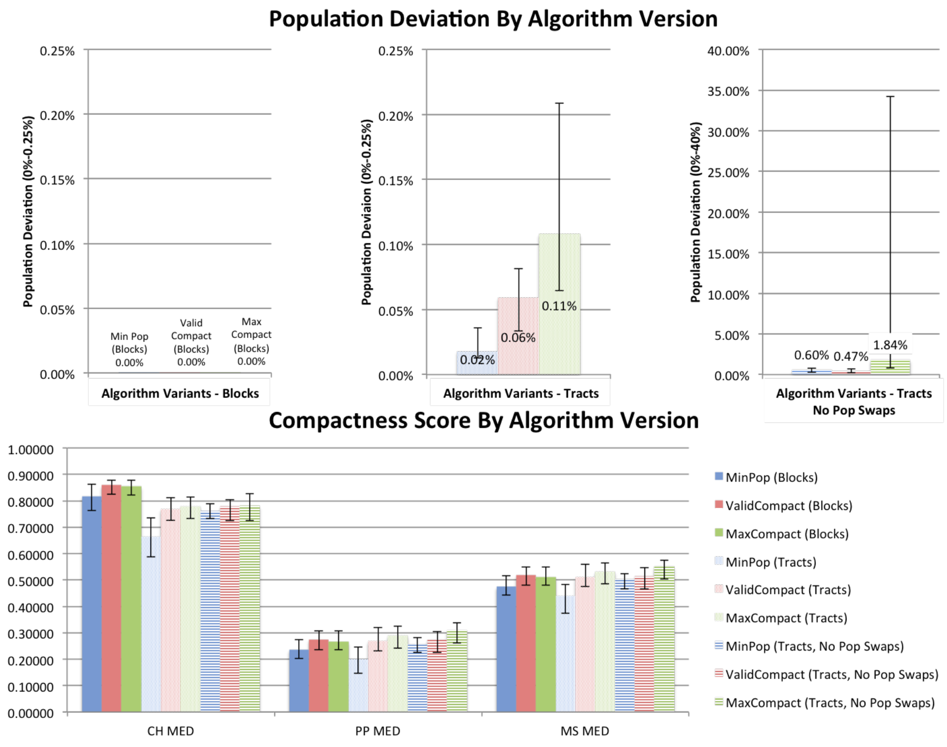


Fig. 8. The changes in population deviation and compactness scores across algorithm versions illustrate the impact of each component. The results show the median with the 25th and 75th percentile error bars. Population deviations closer to 0% are preferred while compactness scores that are closer to 1 are preferred. The population deviations were split into three charts for blocks, tracts, and tracts without population swapping to account for different scales.

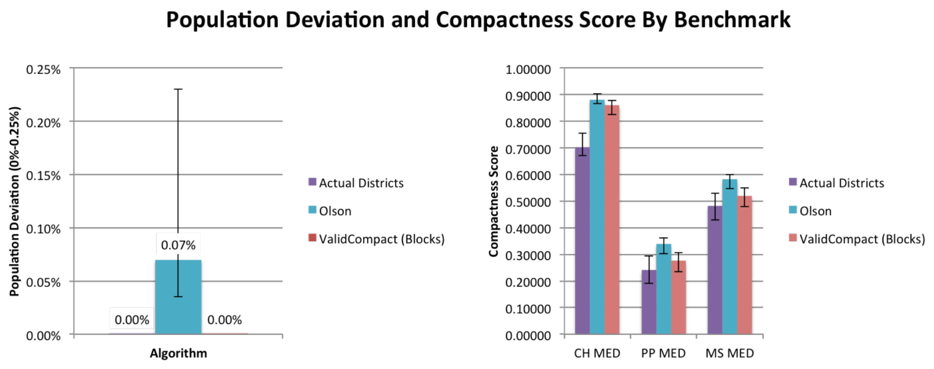
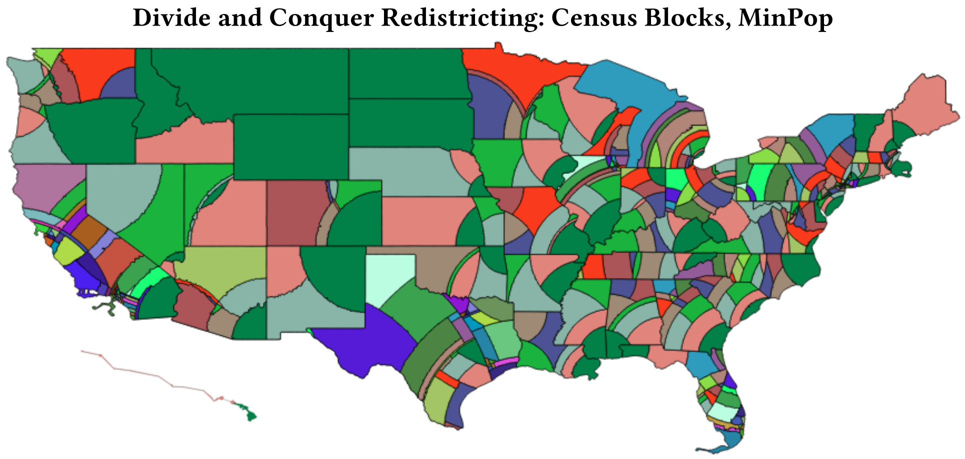
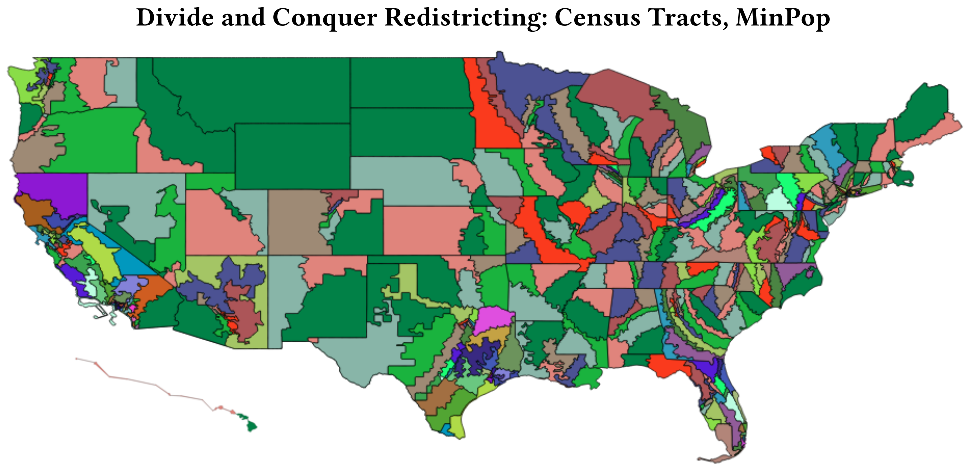


Fig. 9. We compare our results to the Olson Redistricting Algorithm and the actual districts as a benchmark for our algorithm's effectiveness.

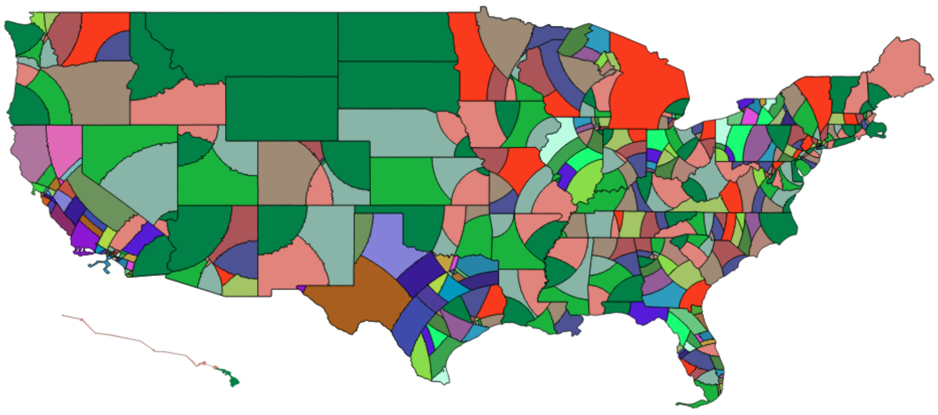




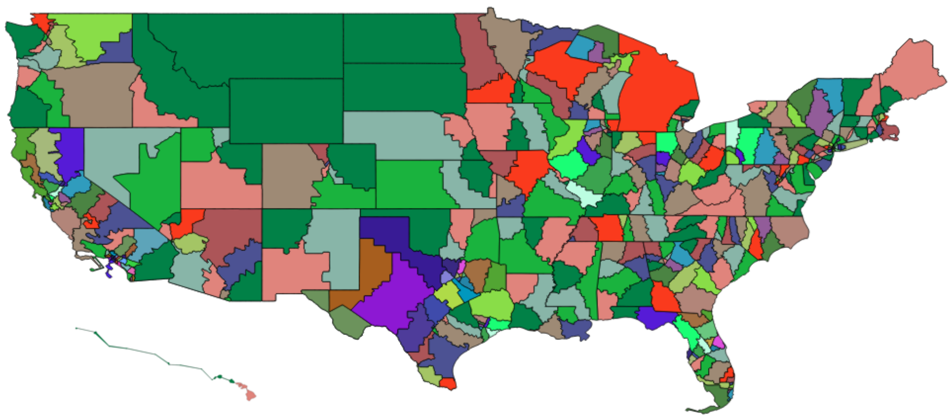
	Census Blocks	Census Tracts	
Measure	**Full Alg**	**No Pop Swap**	**Full Alg**
Valid States (deviation < 0.5%)	42/43	19/43	41/43
Population Deviation Med (%)	0.00%	0.60%	0.02%
Population Deviation Med (people)	1	4,005	127
Convex Hull 25th Percentile	0.76296	0.73297	0.58767
Convex Hull Med	0.81714	0.76072	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

Fig. 10. The results from running the Divide and Conquer Redistricting Algorithm on both census blocks (top) and census tracts (bottom) using the MinPop maximization function.

Divide and Conquer Redistricting: Census Blocks, MaxCompact



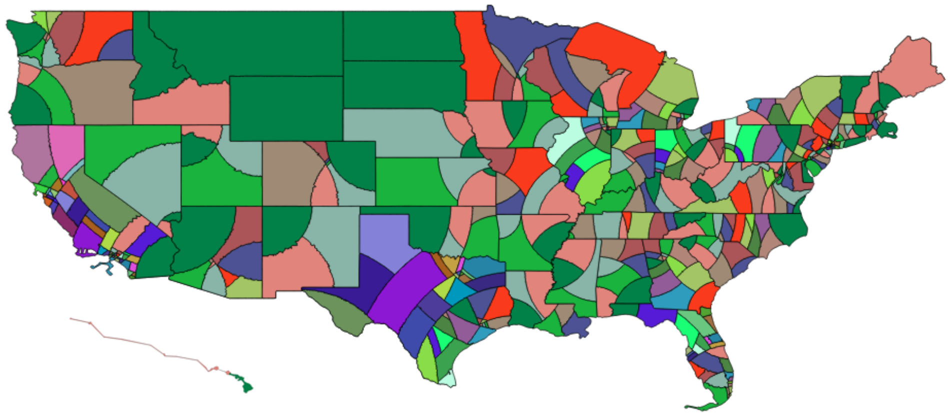
Divide and Conquer Redistricting: Census Tracts, MaxCompact



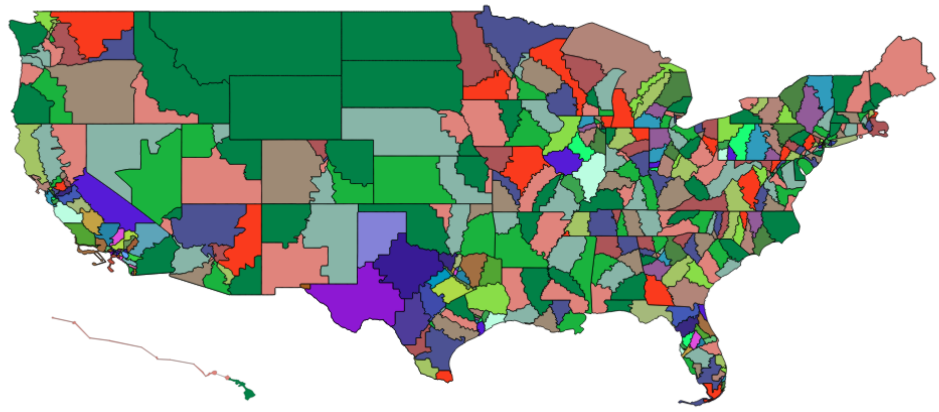
	Census Blocks	Census Tracts	
Measure	**Full Alg**	**No Pop Swap**	**Full Alg**
Valid States (deviation $< 0.5\%$)	36/43	8/43	36/43
Population Deviation Med (%)	0.00%	1.84%	0.11 %
Population Deviation Med (people)	1	13,775	736
Convex Hull 25th Percentile	0.82189	0.72449	0.73293
Convex Hull Med	0.85537	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

Fig. 11. The results from running the Divide and Conquer Redistricting Algorithm on both census blocks (top) and census tracts (bottom) using the MaxCompact maximization function.

Divide and Conquer Redistricting: Census Blocks, ValidCompact



Divide and Conquer Redistricting: Census Tracts, ValidCompact



	Census Blocks	Census Tracts	
Measure	**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

Fig. 12. The results from running the Divide and Conquer Redistricting Algorithm on both census blocks (top) and census tracts (bottom) using the ValidCompact maximization function.

the districts shape. Since Convex Hull compactness measures overall district shape, it is logical that census blocks would produce better Convex Hull scores.

4.1.2 Population swapping component impact. The Population Swapping component significantly reduces the population deviation in every state across all versions. The component was designed to reduce population deviation, so this is logical. The impact on compactness is inconsistent, although negative the majority of the time. Since this component picks the next unit to swap without regard for compactness, this inconsistency also makes sense.

4.1.3 Maximization function variations.

MinPop. The Divide and Conquer Redistricting Algorithm minimized for population deviation produces valid plans in 42 out 43 states when run using census blocks and 41 out of 43 states when run using census tracts (see Figure 10). The census blocks version produces the optimal population deviations (1 person) in every state except New York, which generates results with 3.18%, making this version one of the few algorithms in the literature that has a better population deviation than the current districts in most states. However, the thin arc-shaped districts, most pronounced in Michigan, may not offer a realistic boundary for practical application (the MaxCompact and ValidCompact variants reduce the prevalence of these odd shapes). This version also has lower compactness scores. The difference in compactness scores for the results without the population swapping component compared to the full algorithm demonstrate how the population swapping component significantly reduces compactness scores across all three measures.

MaxCompact. The Divide and Conquer Redistricting Algorithm maximized for Modified Schwartzberg compactness produces valid plans in 36 out of 43 states for both blocks and tracts (see Figure 11). The population deviation for blocks was three orders of magnitude smaller than population deviation for tracts, and the thin arc-shaped districts in the previous version were addressed with the change to the maximization function. This version also has higher compactness scores than any other version of our Divide and Conquer Algorithm. However, this may be due in part to the invalid population deviations that are greater than 50% in some states.

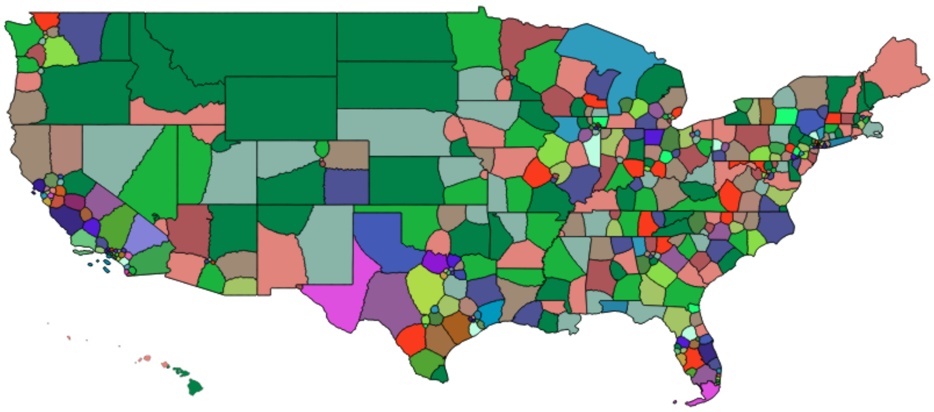
ValidCompact. The Divide and Conquer Redistricting Algorithm maximized for both population deviation and Modified Schwartzberg compactness (ValidCompact) produces valid plans in 42 out of 43 states for blocks and 41 out of 43 states for tracts (see Figure 12). The population deviation for blocks is nearly identical to the MinPop version with a few cases with slightly worse population deviation. Like the MinPop version, the ValidCompact version on blocks does not produce a valid population deviation in New York and generates results with 3.19% population deviation. The algorithm starts on Long Island, and runs out of swaps with the partial population swapping algorithm. Since this part of New York is relatively narrow, there would be a limited number of swappable units in the first level. The compactness scores are roughly identical to the MaxCompact version. Overall, this approach produces the best combination of population deviation and compactness.

4.1.4 Recommended version. Our results demonstrate that near optimal population deviations are only achievable with census blocks, and the ValidCompact version of the maximization function produces strong compactness scores without significant impact to population deviation. Thus, this is the recommended version of the Divide and Conquer Redistricting Algorithm that we will compare to other approaches in the next section.

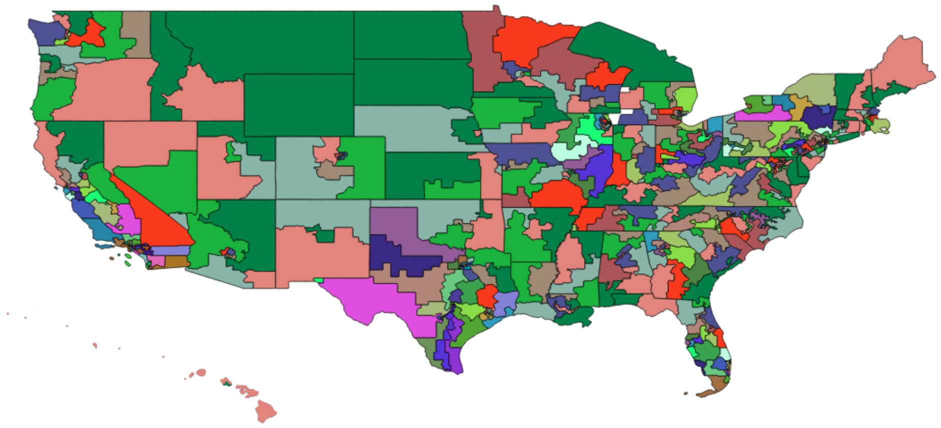
4.2 Comparison with other approaches

As detailed in Section 2, there is only one other algorithm with valid population deviations (less than 0.5%) in most states and block equivalency files for all results, the Olson Redistricting Algorithm.

Comparison Districts: Olson Redistricting Algorithm



Comparison Districts: 113th Congressional Districts



Measure	**Olson**	**113th Districts**	**D&C**
Valid States (deviation $< 0.5\%$)	39/43	40/43	**42/43**
Population Deviation Med (%)	0.07%	**0.00%**	**0.00%**
Population Deviation Med (people)	489	**1**	**1**
Convex Hull Med	**0.88098**	0.70298	0.85929
Convex Hull Stdev	**0.05775**	0.10324	0.07309
Polsby-Popper Med	**0.33778**	0.24088	0.27517
Polsby-Popper Stdev	**0.06265**	0.08992	0.07768
Schwartzberg Med	**0.58080**	0.48120	0.51961
Schwartzberg Stdev	**0.05429**	0.08971	0.07394

Fig. 13. Results for the Olson Redistricting Algorithm (top) and actual 113th congressional districts (bottom). In the actual districts, Maryland, Virginia, and West Virginia are the states with population deviations greater than 0.5%. The Divide and Conquer (D&C) measurements shown are for census blocks using ValidCompact maximization. The best measurements in each row are shown in bold.

Here, we compare the Divide and Conquer Redistricting Algorithm to the Olson algorithm and the 113th real congressional districts.

Olson Redistricting Algorithm. The results for the Olson Redistricting Algorithm were calculated based on the block equivalency files, which can be found online [55]. Using the census block data, we merged the geometry and population data to reproduce the results so that metrics on the population deviation and compactness scores could be calculated. The Olson Redistricting Algorithm has a Vornoi-esque component and a swapping component. Sites are selected randomly and districts are generated by iteratively adding surrounding units. Unlike our algorithm, the swapping component allows swaps that make the population deviation worse. The criteria for a good swap change throughout the execution, allowing swaps that make population deviation worse while improving other criteria. This approach allows the algorithm to explore more possible plans without getting stuck in a local optimum. We only report the final statistics in Figure 13 since we did not have access to intermediate results. We do not know how many executions were examined in each state, nor the weights that were used. It is likely that different weights were used in each state, which may explain the strong results. Out of 43 congressional redistricting plans that were analyzed, 39 have valid population deviations. The worst population deviation is 0.78%, which is slightly above the 0.5% goal. All three compactness scores are the highest among all the versions compared in this analysis. While the Olson Redistricting Algorithm produces districts that are more compact across the three measures, the ValidCompact version of the Divide and Conquer Redistricting Algorithm operating on census blocks produces districts with a median population deviation that is four orders of magnitude smaller.

113th Congressional Districts. To complete the context of the redistricting problem, it is also appropriate to analyze actual congressional districts. A redistricting algorithm would not be useful if it cannot at least match the metrics of the existing districts. Figure 13 provides a summary of the population deviation and compactness scores for the 113th congressional districts. Most states have districts with a population deviation near zero, and many have the optimal deviation, differing only by one person [19]. Only Maryland, Virginia, and West Virginia have population deviations larger than 0.5%.[10] However, compactness scores across all three measures are lower than the Olson Redistricting Algorithm and recommended version of the Divide and Conquer Redistricting Algorithm.

5 CONCLUSION

We offer a new redistricting algorithm that produces an optimal population deviation in most states, demonstrating a valid approach to a challenging real world problem. While the Olson Redistricting Algorithm is consistently the best algorithm if compactness and population deviation are equally weighted criteria, we argue that population deviation should be considered as a more important criteria given that most of the actual districts have optimal or near zero population deviation.

We compared the components of each algorithm to show the tradeoffs of population deviation and compactness. We also compare results with different population units, maximization functions, and sites which suggest a similar trend. It may be impossible to improve both population deviation and compactness, which has implications not only for redistricting algorithm design, but also for electoral policy. Further study of redistricting algorithms should continue our component-based approach to further explore this relationship between population deviation and compactness.

[10]West Virginia has a long tradition of keeping counties in tact, and the U.S. Supreme Court ruled that this tradition justifies the larger population deviation [53]. There has been extensive redistricting litigation in Maryland and Virginia since the 113th Congressional Districts [41].

The selection of sites is a particular area that should be studied further. The literature and our results demonstrate that the selection of sites significantly impacts the final results. Since our algorithm would work with any site choices, different approaches may reveal different qualities of our algorithm.

It is unlikely that a single redistricting algorithm will satisfy the needs of all states. Geography, demographics, and policy preferences differ from state to state, so more variations in redistricting algorithms could be beneficial. While we recommend one of the versions of our Divide and Conquer Redistricting Algorithm, the alternate versions may be useful in cases where our recommendation fails to produce a valid plan. In cases where local political needs necessitate that communities of interest stay together, the population unit vertices making up the state graph can be modified to allow this algorithm to run on merged communities of interest. Similarly, in cases where a few contiguous districts should be redistricted while leaving the rest of the state as is, the algorithm can be run with just the relevant substate graph. Thus, we believe the introduced divide and conquer algorithm can be useful in conjunction with a human understanding of the specific redistricting needs.

Redistricting algorithms, like the one we propose here, can have meaningful impact on policy. In November 2016, a federal court ruled that partisan gerrymanders are unconstitutional and struck down Wisconsin's current districts [54, 66]. This landmark case was the first to strike down districts based on partisan bias, and may mark a turning point in redistricting policy, forcing state governments in several states to redraw their plans with reduced partisan bias [66]. This decision referenced alternative plans as proof that less partisan plans were possible [54]. Clearly, a variety of objective redistricting algorithms can facilitate the generation of alternative plans for this kind of analysis.

It has not yet been determined how many states will be impacted by the growing legal case against partisan gerrymandering, which has been building for some time [58]. If all states are forced to redraw their districts with less partisan bias, there is an opportunity for redistricting algorithms to supplement the debate. A redistricting plan from an automated algorithm may never be chosen as the official districts, but these algorithms can set a better standard by showing mapmakers what is possible.

REFERENCES

[1] 2016. Class Geometry. Retrieved March 1, 2016 from http://www.vividsolutions.com/jts/javadoc/com/vividsolutions/jts/geom/Geometry.html

[2] Alan I. Abramowitz, Brad Alexander, and Matthew Gunning. 2006. Incumbency, Redistricting, and the Decline of Competition in U.S. House Elections. *The Journal of Politics* 68, 1 (2006), 75–88.

[3] Jeroen Aerts, Erwin Eisinger, Gerald Heuvelink, and Theodor Stewart. 2003. Using linear integer programming for multi-site land-use allocation. *Geographical Analysis* 35, 2 (2003), 148–169.

[4] Micah Altman. 1997. Is automation the answer: The computational complexity of automated redistricting. *Rutgers Computer and Law Technology Journal* 23, 1 (1997), 81–142.

[5] Theodore Arrington. 2016. A Practical Procedure for Detecting a Partisan Gerrymander. *Election Law Journal* 15, 4 (2016), 385–402.

[6] North Carolina General Assembly. 2017. House Select Committee on Redistricting. Retrieved January 10, 2018 from https://www.ncleg.net/gascripts/DocumentSites/browseDocSite.asp?nID=356&sFolderName=%5C2017%20House%20Redistricting%20Plan%5CStat%20Pack%20for%20Proposed%20Plan

[7] De Assis, Laura Silva, Paulo Morelato Franca, and Fábio Luiz Usberti. 2014. A redistricting problem applied to meter reading in power distribution networks. *Computers & Operations Research* 41 (2014), 65–75.

[8] Azavea. 2015. 2012 US Congressional Districts. Retrieved January 24, 2015 from https://www.google.com/fusiontables/DataSource?snapid=S506424n-DY

[9] Azavea. 2018. District Builder. Retrieved January 9, 2018 from http://www.districtbuilder.org

[10] Kevin Baas. 2016. Auto-Redistrict. Retrieved June 28, 2016 from http://autoredistrict.org/contact.php

[11] Fernando Bacao, Victor Lobo, and Marco Painho. 2005. Applying genetic algorithms to zone design. *Soft Computing-A Fusion of Foundations, Methodologies and Applications* 9, 5 (2005), 341–348.

[12] Alex Benn and David German. 2008. Unbiased Congressional Districts. Retrieved November 20, 2013 from http://web.cs.swarthmore.edu/~adanner/cs97/s08/papers/benn_german.pdf

[13] Mira Bernstein and Moon Duchin. 2017. A Formula Goes to Court: Partisan Gerrymandering and the Efficiency Gap. *Notices of the AMS* 64, 9 (2017), 1020–1024.

[14] Bill Bishop. 2008. *The Big Sort: Why the Clustering of Like-Minded American is Tearing Us Apart.* Houghton Mifflin Harcourt, Boston.

[15] Nate Bottman, Wes Essig, and Sam Whittle. 2007. Why Weight? A Cluster-Theoretic Approach to Political Districting. *UMAP Journal* 28, 3 (2007), 301–315.

[16] Burcin Bozkaya, Erhan Erkut, and Gilbert Laporte. 2003. A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research* 144, 1 (January 2003), 12–26.

[17] Census Bureau. 2018. Geographic Terms and Concepts - Census Tract. Retrieved March 1, 2016 from http://www.archives.gov/research/census/1940/finding-aids.html#maps

[18] F. Caro, T. Shirabe, M. Guignard, and A. Weintraub. 2004. School redistricting: Embedding GIS tools with integer programming. *Journal of the Operational Research Society* 55, 8 (2004), 836–849.

[19] United States Census. 2015. United States Census. Retrieved January 31, 2015 from http://www.census.gov

[20] Jowei Chen and Jonathan Rodden. 2013. Unintentional Gerrymandering: Political Geography and Electoral Bias in Legislatures. *Quarterly Journal of Political Science* 8, 3 (2013), 239–269.

[21] Jowei Chen and Jonathan Rodden. 2015. Cutting Through the Thicket: Redistricting Simulations and the Detection of Partisan Gerrymanders. *Election Law Journal* 14, 4 (2015), 331–345.

[22] Vincent Cohen-Addad, Philip Klein, and Neal Young. 2018. Balanced power diagrams for redistricting. *arXiv preprint arXiv:1710.03358* (2018).

[23] Suzanne Dovi. 2017. Political Representation. In *The Stanford Encyclopedia of Philosophy* (fall 2017 ed.), Edward N. Zalta (Ed.). Metaphysics Research Lab, Stanford University. https://plato.stanford.edu/archives/win2017/entries/political-representation

[24] ESRI. 2018. GIS Dictionary. Retrieved January 15, 2018 from https://support.esri.com/en/other-resources/gis-dictionary/term/envelope

[25] Brennan Center for Justice. 2018. Citizen-Led Efforts to Reform Redistricting. Retrieved January 4, 2018 from https://www.brennancenter.org/analysis/current-citizen-efforts-reform-redistricting

[26] The United States District Court for the Middle District of North Carolina. 2018. *Common Cause v. Rucho.* Vol. 587. F. Supp. 3d.

[27] Andrew Gelman and Gary King. 1994. Enhancing Democracy through Legislative Redistricting. *American Political Science Review* 88, 3 (1994), 541–559.

[28] Gerrymander. 2019. *Cambridge Academic Content Dictionary.* Cambridge University Press.

[29] David J. Gopoian and Darrell M. West. 1984. Trading Security for Seats: Strategic Considerations in the Redistricting Process. *Journal of Politics* 46, 4 (1984), 1080–1096.

[30] R. L. Graham. 1969. Bounds on Multiprocessing Timing Anomalies. *SIAM J. Appl. Math.* 17, 2 (March 1969), 416–429.

[31] Dan Gulotta, Daniel M. Kane, and Andrew Spann. 2007. Electoral Redistricting with Moment of Inertia and Diminishing Halves Models. *UMAP Journal* 28, 3 (2007), 281–299.

[32] Brian Hayes. 1996. Machine Politics. *American Scientist* 84, 6 (1996), 522–526.

[33] John A. Henderson, Brian T. Hamel, and Aaron Goldzimer. 2017. Gerrymandering Incumbency: Does Non-Partisan Redistricting Increase Electoral Competition? *The Journal of Politics* 80, 3 (2017), 1011–1016.

[34] S. Hess, H. Siegfldt, J. Whelan, and P. Zitlau. 1965. Nonpartisan Political Redistricting by Computer. *Operations Research* 13, 6 (November 1965), 998–1006.

[35] Deepti Joshi, Leen-Kiat Soh, and Ashok Samal. 2012. Redistricting Using Constrained Polygonal Clustering. *IEEE Transactions on Knowledge and Data Engineering* 24, 11 (November 2012), 2065–2079.

[36] Henry Kaiser. 1966. An Objective Method for Establishing Legislative Districts. *Midwest Journal of Political Science* 10, 2 (May 1966), 200–213.

[37] Jorg Kalcsics, Stefan Nickel, and Michael Schroder. 2005. Towards a unified territorial design approach - Applications, algorithms and GIS integration. *Top* 13, 1 (2005), 1–56.

[38] Narenda Karmarkar and Richard M Karp. 1982. The Differencing Method of Set Partitioning. *Technical Report UCB/CSD 81/113, Computer Science Division University of California, Berkeley* (1982).

[39] Justin Levitt. 2008. "Communities of Interest" in State Redistricting Law. Retrieved October 14, 2016 from http://www.brennancenter.org/sites/default/files/legacy/commentary/Communities%20of%20Interest.pdf

[40] Justin Levitt. 2010. *A Citizen's Guide to Redistricting.* Brennan Center for Justice.

[41] Justin Levitt. 2018. All About Redistricting. Retrieved February 11, 2018 from http://redistricting.lls.edu

[42] Michael Lyons and Peter F. Galderisi. 1995. Incumbency, Reapportionment and U.S. House Redistricting. *Political Research Quarterly* 48, 4 (1995), 857–871.

[43] Karin MacDonald and Bruce Cain. 2013. Community of Interest Methodology and Public Testimony. *UC Irvine Law Review* 3 (2013), 609–636.

[44] William Macmillan. 2001. Redistricting in a GIS environment: An optimisation algorithm using switching-points. *Journal of Geographical Systems* 3, 2 (2001), 167–180.

[45] William Macmillan and Todd Pierce. 1994. Optimization modelling in a GIS framework: the problem of political redistricting. In *Spatial analysis and GIS*, Stewart Fotheringham and Peter Rogerson (Eds.). TJ International Ltd, Great Britain, Chapter 11, 221–246. https://plato.stanford.edu/archives/win2017/entries/political-representation

[46] Daniel B. Magleby and Daniel B. Mosesson. 2018. A New Approach for Developing Neutral Redistricting Plans. *Political Analysis* 26, 2 (May 2018), 147–167.

[47] Daniel McGlone. 2016. Measuring District Compactness in PostGIS. Retrieved March 2, 2019 from http://www.newswithnumbers.com/2010/02/04/gerrymandering-and-the-2010-census/#Detail

[48] Stuart Nagel. 1965. Simplified Bipartisan Computer Redistricting. *Stanford Law Review* 17, 5 (May 1965), 863–899.

[49] Chicago-Kent College of Law. 2014. Wesberry v. Sanders. Retrieved April 24, 2014 from http://www.oyez.org/cases/1960-1969/1963/1963_22

[50] State of Minnesota Special Redistricting Panel. 2011. Order Denying Request to Participate as Amicus. In *Congressional Redistricting Plans*. Minnesota Judicial Branch. http://www.mncourts.gov/Documents/0/Public/Court_Information_Office/2011Redistricting/A110152AmendedOrder9.13.11.pdf

[51] Maryland Department of Planning. 2011. 2011 Governor's Redistricting Advisory Committee. Retrieved January 9, 2018 from https://web.archive.org/web/20121115091642/http://www.planning.maryland.gov:80/PDF/redistricting/2010docs/Guidelines3rdpartyplan2011.pdf

[52] The Supreme Court of the United States. 1964. *Wesberry v. Sanders*. Vol. 376. U.S.

[53] The Supreme Court of the United States. 2012. *Tennant v. Jefferson County Commission*. Vol. 576. U.S.

[54] The Supreme Court of the United States. 2016. *Whitford v. Gill*. Vol. 585. U.S.

[55] Brian Olson. 2013. Redistricter. Retrieved October 28, 2013 from http://code.google.com/p/redistricter/

[56] Federica Ricca, Andrea Scozzari, and Bruno Simeone. 2008. Weighted Voronoi region algorithms for political districting. *Mathematical and Computer Modelling* 48 (2008), 1468–1477.

[57] Federica Ricca, Andrea Scozzari, and Bruno Simeone. 2011. Political districting: from classical models to recent approaches. *Annals of Operations Research* 204, 1 (2011), 271–299.

[58] Terry Smith. 2016. Bond V. Floyd And Expressive Proscriptions On The Partisan Gerrymander. *Wisconsin Law Review* (2016), 122–145.

[59] David Sparks. 2016. K-Means Redistricting. Retrieved June 28, 2016 from https://dsparks.wordpress.com/2010/10/18/k-means-redistricting/

[60] Nicholas Stephanopoulos and Eric McGhee. 2015. Partisan gerrymandering and the efficiency gap. *The University of Chicago Law Review* 82 (2015), 831–900.

[61] Lukas Svec, Sam Burden, and Aaron Dilley. 2007. Applying Voronoi Diagrams to the Redistricting Problem. *UMAP Journal* 28, 3 (2007), 315–331.

[62] Edward R. Tufte. 1973. The relationship between seats and votes in two-party systems. *American Political Science Review* 67, 2 (1973), 540–554.

[63] Leonardo Vanneschi and Roberto Henriques Mauro Castelli. 2017. Multi-objective genetic algorithm with variable neighbourhood search for the electoral redistricting problem. *Swarm and Evolutionary Computation* 36 (2017), 37–51.

[64] Range Voting. 2013. Examples of our unbiased district-drawing algorithm in action / comparisons with gerrymandered districts drawn by politicians. Retrieved October 28, 2013 from http://rangevoting.org/GerryExamples.html

[65] Samuel Wang. 2016. Three tests for practical evaluation of partisan gerrymandering. *Stanford Law Review* 68 (June 2016), 1263–1321.

[66] Michael Wines. 2016. Judges Find Wisconsin Redistricting Unfairly Favored Republicans. *The New York Times* (November 2016).

[67] Antoine Yoshinaka and Chad Murphy. 2011. The paradox of redistricting: How partisan mapmakers foster competition but disrupt representation. *Political Research Quarterly* 64, 2 (2011), 435–447.

[68] H. P. Young. 1988. Measuring the Compactness of Legislative Districts. *Legislative Studies Quarterly* 13, 1 (February 1988), 105–115.