MINORITY VOTER DISTRIBUTIONS AND PARTISAN GERRYMANDERING

JIAHUA CHEN, ANEESHA MANNE, REBECCA MENDUM, POONAM SAHOO, AND ALICIA YANG,
MENTORED BY: DIANA DAVIS

ABSTRACT. Many people believe that it is disadvantageous for members aligning with a minority party to cluster in cities, as this makes it easier for the majority party to *gerrymander* district boundaries to diminish the representation of the minority. We examine this effect by exhaustively computing the average representation for every possible 5×5 grid of population placement and district boundaries. We show that, in fact, it is *advantageous* for the minority to arrange themselves in clusters, as it is positively correlated with representation. We extend this result to more general cases by considering the dual graph of districts, and we also propose and analyze metaheuristic algorithms that allow us to find strong lower bounds for maximum expected representation.

1. INTRODUCTION AND MOTIVATION

Elections are integral to the operation of electoral democracies. In the legislative branch of the United States government, *legislative bodies* such as the House of Representatives, as well as state legislatures, all consist of groups of representatives from various *electoral districts*. Plurality elections are used to determine individual representatives from each electoral district. These electoral districts are partitioned through a *districting plan* that assigns each *census block*, the smallest geographic unit of population, to a district. Geometrically, this is the partitioning of a set of census blocks into districting plans. In partisan democracies such as the United States, the political makeup of such legislative bodies heavily dictates the overall political stance, causing the entire assembly to vote in favor of a specific *political party*. Therefore, opposing political parties, especially the incumbent party, with the power and authority to create districting plans, often *gerrymander* the districts to create biased districting plans that favor themselves.

This current system does not yield *proportional representation*, which means that the percentage of voters for a given political party does not yield a commensurate number of seats. In the state of Massachusetts, for example, it has been rigorously proved that it would be nearly impossible to create a districting plan that deviates from the current 9 Democrat, 0 Republican plan, although Republican candidates often receive 30% to 40% of the total votes [3]. Gerrymandering exploits the fact that districting does not necessarily yield proportional representation and is used by incumbent political parties to maximize the number of seats they win.

There are several types of gerrymandering: *partisan gerrymandering*, which benefits one party at the expense of another; *racial gerrymandering*, which dilutes the political representation of racial minorities; and *incumbent gerrymandering*, which protects incumbents from both parties and creates *safe seats*—districts in which the incumbent is guaranteed a victory. While our research is focused on partisan gerrymandering, it has

implications that extend to other forms of gerrymandering of a similar nature (specifically, gerrymandering that diminishes some binary indicator, such as political affiliation).
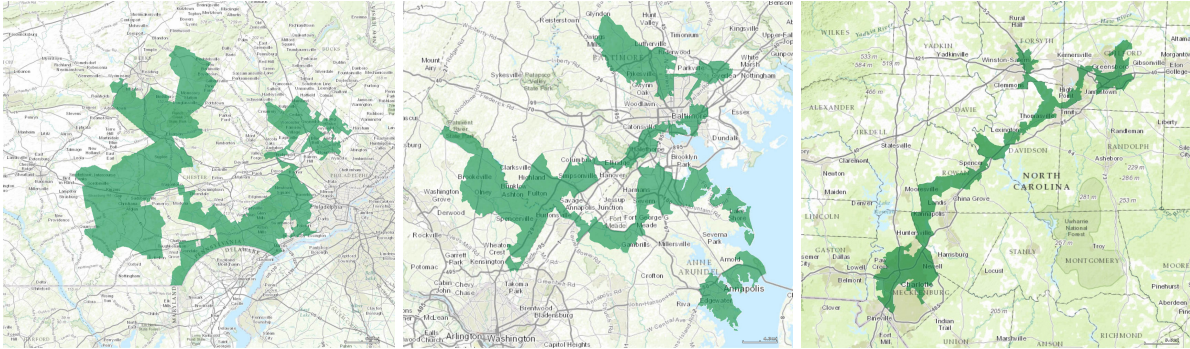

FIGURE 1. The congressional districts of PA-7 (prior), MD-3, and NC-12 (prior) demonstrate one of the geometrical consequences of gerrymandering—oddly shaped districts. The districting plans of Maryland and North Carolina have recently reached the Supreme Court.

There are several methods for quantifying gerrymandering and 'fairness' in elections. Recently, the Supreme Court of the United States saw the landmark case *Rucho* v. *Common Cause* (and *Benisek* v. *Lamone*) over the partisan gerrymander of North Carolina and Maryland's congressional districts. Although the Supreme Court came to the conclusion that "partisan gerrymandering claims present political questions beyond the reach of the federal courts," leaving the case of partisan gerrymandering still hugely undecided, the court does however propose several methods for determining a fair districting plan [6].

One method proposed by the Supreme Court (and commonly investigated in the mathematical community), is to determine the mean or median result out of a wide range of different districting plans. Because it is computationally difficult to enumerate through the space of all districting plans, such analysis is achieved by sampling the space of all districting plans using a stochastic Markov Chain Monte-Carlo method (also known as a Metropolis-Hastings algorithm). By constructing a *metagraph* with nodes that are distinct districting plans and edges adjoining two similar districting plans, the algorithm makes a random walk on the graph to sample the space of districting plans [2]. Each *voter distribution* paired with a districting plan yields a different outcome (which is the number of seats for a given party). Using the algorithm, we can estimate the expected outcome for a given voter distribution over **all** the districting plans by evaluating the expected number of seats on a sample of all the districting plans. This is the measure of 'fair' representation that courts and mathematicians propose—the mean result yielded by all the plans.

This measure has significant implications. As the mean of the distributions of varying districting plans, it is an indicator for the overall distribution itself. A higher mean signifies that a districting plan yielding a high number of seats is more likely, while a lower mean signifies that a plan yielding a similar number of seats is more difficult to create and statistically unlikely. Thus, it is in the best interest of every political party to maximize this metric for themselves. In other words, every political party wants a voter distribution that yields the highest mean over all districting plans. This *robust* voter distribution gives that party the best shot at gaining a higher amount of representation (seats) in the House or other legislative bodies.

Our research is motivated by the following questions, which are closely related to this metric:

1. *How do certain attributes of a given voter distribution, if any, affect this measure?*
2. *Are there specific ways of arranging a limited number of voters on a map that yields the best result for a certain political party?*
3. *How can we compute the optimal voter distribution that yields the best result?*

Specifically, we are interested in cases where the specific party is the minority party in a given state, as minority parties are often disadvantaged the most by malicious gerrymandering (such as the case in North Carolina in 2018).

Chen and Rodden propose that one key characteristic affecting a particular voter distribution's expected representation is *clustering*—a factor of human geography that affects the concentration of certain voters in cities or specific regions. By analyzing maps and election data, they come to the conclusion that Democrats disadvantage themselves by clustering in cities [1].

Whereas Chen and Rodden analyzed real-world data, we examine this effect on a theoretical level using a model that simulates the geographical landscape of a state: a 5×5 grid population of voters. By exhaustively computing every possible combination of voter placement on this grid and determining the average representation (the above metric) for each of these voter placements, we are able to eliminate many confounding complexities that are associated with human geography in order to come to stronger conclusions regarding the correlation between clustering and expected representation. This is detailed in Section 3.

We show that, in fact, it is advantageous for a minority party to arrange themselves in clusters, as it is positively correlated with expected representation. By representing an area of land as dual graphs, we are then able to extend this result to the general case, which we propose as a conjecture in Section 3.

Conjecture. (Clustering Relation). For minority voter populations, there is a positive correlation between a measure of clustering and their expected representation.

Exploiting this relation, we are able to construct metaheuristic algorithms that allow us to optimize voter distributions to yield the maximum expected representation by using a greedy algorithm. We compare such an algorithm against other discrete metaheuristic optimization techniques in Section 4.

2. Basic Definitions and Terminology

We first define some basic definitions and terminology that we use throughout the paper.

Definition 2.1. We model population using a *dual graph*, a planar, undirected graph $G = (V, E)$ consisting of k vertices $V = \{b_1, b_2, \ldots, b_k\}$, which we call blocks (since the vertices represent census blocks of equal population). If b_i and b_j are connected by an edge, they are *adjacent* blocks. If b_k is on the boundary of the unbounded face of the graph, we say b_k is a *border*. If two census blocks touch at a corner, they are not considered as sharing an edge, which ensures we get a planar graph.

For each dual graph, we can create *districting plans* that assign each block to a unique district. Many of the 'rules' regarding districting arise from the Voting Rights Act of 1965, which placed restrictions on districting in the United States [4].
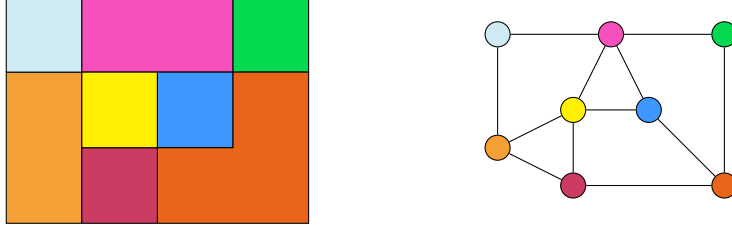
FIGURE 2. A region of 8 elementary census blocks (left) is represented as a dual graph (right) of 8 vertices (each representing an elementary census block), sharing an edge when two corresponding blocks are adjacent.

Definition 2.2. A *districting plan* $\mathcal{D}$ partitions a dual graph G into n *districts* $\{d_1, d_2, \ldots, d_n\}$. Then, a *legal* districting plan is one that satisfies the following:

1. (Population) Each district d contains the same number of blocks. Since blocks represent census blocks of equal population, our districts will also have equal population. The number of blocks is the *size m* of a district.
2. (Contiguity) For all census blocks b_x, b_y in a district d_j, b_x and b_y are connected by a series of edges with vertices all in d_j. In addition, if b_α is a block not in district d_j, there exists a series of edges connecting b_α to a border b_λ in which none of the vertices pass through d_j.

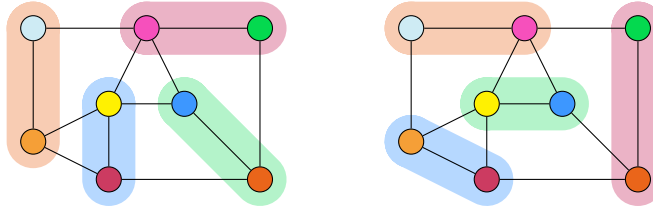The set of all districting plans is denoted by $\mathbb{D} = \{\mathcal{D} \mid \mathcal{D} \text{ legal}\}$.

FIGURE 3. Two *legal* districting plans of $m = 2$ for the dual graph in Figure 2.

While the dual graph and districting plans are a representation of the geographic distribution of populations and districts, they do not contain information regarding the voter preferences of the people living in each individual block.

Definition 2.3. We define a *discrete voter distribution* as a function $\Delta : V \to \{0, 1\}$, mapping each vertex from the set of vertices of our dual graph to a binary measure 0 or 1. A measure of 1 means that the block leans toward the given party, and a measure of 0 means that the block leans away from the given party. Naturally, we can find the proportion of voters for a given party over the whole dual graph G:

$$\mathsf{Avg}_G(\Delta) = \frac{1}{k} \sum_{i=1}^{k} \Delta(b_i). \tag{2.1}$$

Similarly, the proportion of voters for a given party in a given district d_j is:

$$\mathsf{Avg}_{d_j}(\Delta) = \frac{1}{m} \sum_{b_i \in d_j} \Delta(b_i); \quad m = |d_j|. \tag{2.2}$$

In a bipartisanship, not leaning toward the party of interest implies a lean toward the opposing party. For the purposes of this paper, we let the party of interest (the 'given party') be the ● party. The set of all possible voter distributions is denoted by set $\{\Delta\}$.
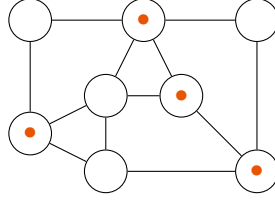

FIGURE 4. A discrete voter distribution for the dual graph in Figure 2, where the party of interest is the ● party, and the opposing party is an empty node.

For each voter distribution, we need a measure to quantify *clustering* of a given party, which is an important indicator in our later analysis.

Definition 2.4. *Clustering* Clus $: \{\Delta\} \to [0,1]$ of a voter distribution is the proportion of edges that join two vertices voting for the same party:

$$\mathsf{Clus}(\Delta) = \frac{\#\Big(\Delta(b_i) = \Delta(b_j)\Big)}{\#E}; \quad (b_i, b_j) \in E. \tag{2.3}$$

We can also define a similar measure, *partisan clustering* ClusP for, which comes with an additional rule to only count **directed** connections containing a block voting for our given party (using our definition above, means that $\Delta(b) = 1$):

$$\mathsf{ClusP}(\Delta) = \frac{\#\Big(\Delta(b_i) = \Delta(b_j) = 1\Big)}{\#\{(b_i, b_j) \in E \mid \Delta(b_i) = 1\}}. \tag{2.4}$$

Here, # simply denotes the *count of*.

To compute Clus, we enumerate all the edges on the graph and divide the number of like edges by the total number of edges. Similarly, to compute ClusP, we enumerate all directed edges that start at a ● block and divide the number of edges connecting a ● block to another ● block over the total number of edges containing ● blocks.

Example 2.5. For the voter distribution Δ in Figure 4,

$$\mathsf{Clus}(\Delta) = \frac{3}{12} \quad \text{and} \quad \mathsf{ClusP}(\Delta) = \frac{4}{13}.$$

Note that for ClusP, we are double-counting each edge connecting a ● to a ● as we are concerned with directed edges. This emphasizes the clustering of the ● party, which creates a more representative measure of clustering, especially when ● is the minority party.

We now want to be able to evaluate the *representation* of a district to determine which party wins the seat of that district.

Definition 2.6. The *representation* of a district d_i is the outcome of the plurality election held in the district. It is a function $R : \mathcal{D} \times \{\Delta\} \to \{0, 0.5, 1\}$ that takes a voter distribution and a given district in our districting plan and assigns the district a value of 0, 0.5, or 1. R evaluates to 0 if the • party loses the seat in said district, R evaluates to 1 if the • party wins the seat in said district, and R evaluates to 0.5 if there is an equal number of blocks in the district voting for both parties.

$$R_\Delta(d_i) = \begin{cases} 0 & \text{if } \mathsf{Avg}_{d_j}(\Delta) < 0.5 \\ 0.5 & \text{if } \mathsf{Avg}_{d_j}(\Delta) = 0.5 \\ 1 & \text{if } \mathsf{Avg}_{d_j}(\Delta) > 0.5 \end{cases} . \tag{2.5}$$

The *total representation* $\mathsf{Rep} : \mathbb{D} \times \{\Delta\} \to [1, m]$ is the number of seats that a given party wins over **all** m districts for a given voter distribution Δ and districting plan $\mathcal{D}$:

$$\mathsf{Rep}_\Delta(\mathcal{D}) = \sum_{d_i \in \mathcal{D}} R(d_i, \Delta). \tag{2.6}$$

We need a measure for each voter distribution Δ that does not depend on which districting plan we use. In other words, we need a measure that encapsulates the representation of a given voter distribution over **all** possible districting plans. We can achieve this by treating Rep_Δ as a random variable.

Definition 2.7. Another way we interpret Rep_Δ is as a discrete random variable. Consider a sample space $\Omega = \mathbb{D}$ of all the districting plans where an event is selecting a particular districting plan $\mathcal{D}$ from our sample space uniformly. $\mathsf{Rep}_\Delta : \Omega \to \mathbb{R}$ is then our discrete random variable, which is the representation corresponding to a voter distribution. This allows us to do the following:

1. We can create a probability density function for Rep_Δ, which shows how likely each level of representation is for our given party.
2. We can compute the *expected amount of representation* for our given party:

$$\mathbb{E}(\mathsf{Rep}_\Delta) = \frac{1}{|\mathbb{D}|} \sum_{\mathcal{D} \in \mathbb{D}} \mathsf{Rep}_\Delta(\mathcal{D}). \tag{2.7}$$

3. We can compute the *variance* of representation $\mathsf{Var}(\mathsf{Rep}_\Delta)$.
4. We can compute the *minimum* and *maximum* representation that could be won: $\min(\mathsf{Rep}_\Delta)$ and $\max(\mathsf{Rep}_\Delta)$.

Remark 2.8. The expected amount of representation in equation 2.7 becomes a key measure in our analysis as it provides a precise quantification of how well each districting plan Δ performs. However, $\mathbb{D}$ is usually very large, so it is computationally hard to compute this expected value. This also becomes a main focus and rate limiting factor in our optimization.

3. Grid

In the United States, laws often restrict the process of redistricting to complying with county or town boundaries. Especially in the Midwest, land is often divided into grid-like *precincts*, which are the fundamental units of voting districts. We can easily use a dual graph representing a square grid, which resembles the often square or rectangular regions of land.
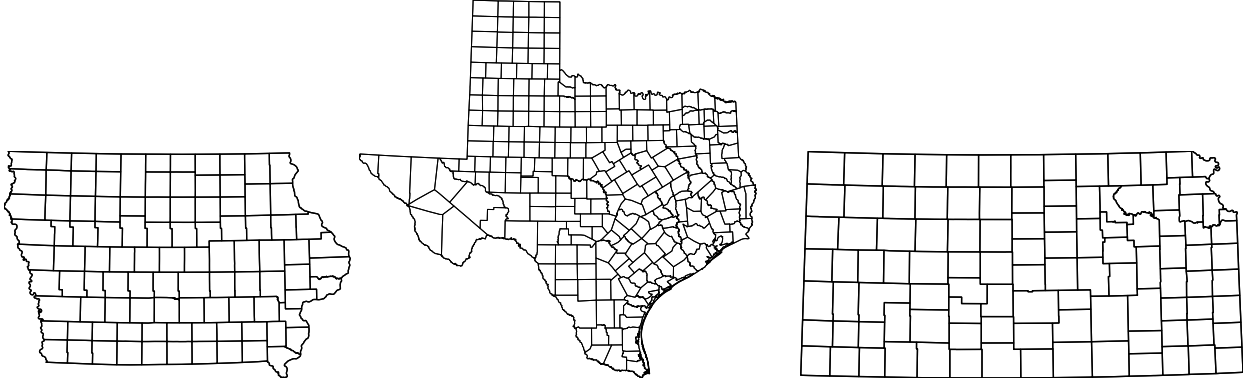
FIGURE 5. Maps of the counties of Iowa, Texas, and Kansas. These states in the Midwest have counties that resemble a rectangular grid.

While many states do not follow this geographical pattern, square grids are a computationally accessible model that gives a surprising amount of insight into the clustering problem. Additionally, the total number of possible legal districting plans, which we denote by $\mathbb{D}$, can be easily computed for small cases.

Each square of the grid is a block of equal population. Following Definition 2.3, we can also have voter distributions for our grids.
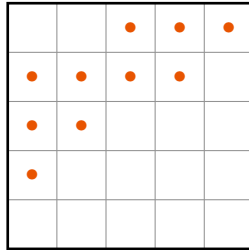

FIGURE 6. A 5×5 *square grid* with a voter distribution yielding 10 • blocks.

For a square grid, all legal districting plans are formed by *polyominoes*, which are plane geometric figures formed by joining one or more equal squares edge to edge. The number of distinct polyomino partitions of square grids is well known for small dimensions, and the algorithms for finding such partitions are also optimized [5]. For a 5×5 grid of pentomino tilings (5 districts of 5 blocks each), there are a total of 4006 possible districting plans.

Remark 3.1. Even with efficient algorithms to find pentomino tilings, this problem of tiling is still NP-complete. The rate at which the number of possible partitions increases exceeds $O(e^n)$. The known numbers of these tilings are listed in Table 1.

Figure 7 details two possible districting plans for a 5×5 grid. It also demonstrates that different districting plans can yield varying amounts of representation. The • party secures one seat with the plan on the left, as opposed to 3 seats with the plan on the right. *Proportional representation* would call for 2 seats won by the • party as there are 10 • blocks.

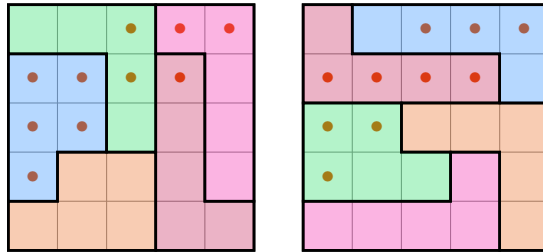| n	n-omino tilings
1	1
2	2
3	10
4	117
5	4006
6	451206
7	158753814
8	187497290034
9	706152947468301

TABLE 1. Number of n-omino tilings on a $n \times n$ square grid (OEIS: A172477) [5].

FIGURE 7. Two districting plans of 5 districts with 5 blocks each applied to the voter distribution from Figure 6.

3.1. **Methodology.** Harris provides an efficient algorithm for determining the various population distributions on an $n \times n$ square grid given sufficiently small dimensions [5]. We use this algorithm to generate an exhaustive list of all 4006 possible districting plans for a 5×5 square grid, $\mathbb{D}$. There are a total of 2^{25} possible voter distributions Δ on a 5×5 grid, with 2 choices for each of the 25 blocks. Then, taking the quotient by the dihedral group D_4 to account for symmetries of a square, we obtain $2^{23} \approx 8 \times 10^6$ possible districting plans. We were able to compute this exhaustive search over every districting plan in several hours on a standard laptop. The 6×6 square grid case increases the computational complexity by a factor of over $200,000 < \frac{451206}{4006} \cdot \frac{2^{36}}{2^{35}}$, which exceeds the computational capacity available to us and most institutions.

For each voter distribution that we iterate through, we record the following pieces of information: the number of $\bullet$ blocks, the clustering score (Clus and ClusP), the distribution of the number of seats this voter distribution yields over all districting plans, and the expected (average) seats this voter distribution yields over all districting plans.

Definition 3.2. We assume there are two parties, Dot ($\bullet$) and Blank. Define Num as the number of dot-voting blocks in a given voting distribution Δ.

3.2. **Observations.** We use our exhaustive data from the 5×5 grid search to set a certain number of $\bullet$ blocks as a constant and retrieve all such voter distributions Δ with that constant number of $\bullet$ blocks. Using these, we examine the correlation between clustering $\mathsf{ClusP}(\Delta)$ and expected representation $\mathbb{E}(\mathsf{Rep}_\Delta)$ for that

specific Num. Additionally, we quantify this correlation, which we find to be approximately linear, using a linear regression test.
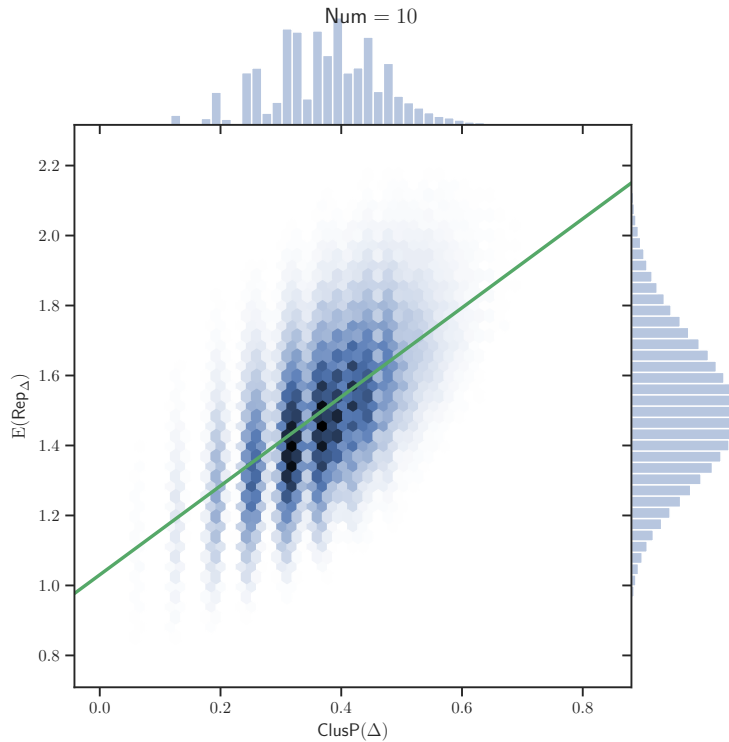

FIGURE 8. ClusP vs. $\mathbb{E}(\text{Rep})$ for Num $= 10$.

For instance, Figure 8 shows plot of $\text{ClusP}(\Delta)$ against $\mathbb{E}(\text{Rep}_\Delta)$ with a linear regression line of best fit superimposed. There are $\binom{25}{\text{Num}}$ possible ways to arrange Num on a 5×5 grid, so a scatter plot obfuscates the relative densities of data points. Hence, we use a a heatmap-bin plot to display the data, with histograms of each data set, $\text{ClusP}(\Delta)$ and $\mathbb{E}(\text{Rep}_\Delta)$. The darker shades of the plot indicate higher relative densities.

Furthermore, we can vary Num to extend these results, which each gives us a scatterplot for each Num. We can find the slope of the linear regression line for each of these cases, and we can compare this slope with each corresponding Num value. Note that when Num $= 0$, there is no possible Δ to calculate $\text{ClusP}(\Delta)$ from. When Num $= 1$ or 2, the state space of possible voter distributions is so small that the linear regression test results in a slope of 0. See Table 2 for the slope of every linear regression line for when Num < 13. Plots for other Num can be found in Appendix B, which produces the results seen in Table 2.

Then, we take the slope from each linear regression test and plot Num against these slope values. We display the results of this scatter plot in Figure 9. Figure 9 shows that when Dots are in the minority such that $3 \leq$ Num ≤ 12, the slope is positive. Thus, there is a positive correlation between increased clustering, according to a higher clustering score, and increased representation. Therefore, if the minority desires maximal representation, then clustering is a better strategy. This correlation peaks at Num $= 8$ before decreasing in correlation as Num increases. Furthermore, when Dots are in the majority such that $13 \leq$ Num ≤ 22, the slope is negative. This means that when the majority clusters itself, the expected

Num	Slope
1	0
2	0
3	0.2993106942
4	0.6704477756
5	1.040404652
6	1.350056768
7	1.553744171
8	1.619139053
9	1.527148112
10	1.271799538
11	0.8600922591
12	0.3117844548

TABLE 2. Results of linear regression test for $\mathsf{Num} = 3$ to 12

representation reduces, so the majority's best strategy is to stop clustering and disperse evenly instead. For $\mathsf{Num} = 23, 24$, the cases are symmetric to $\mathsf{Num} = 1, 2$, with a very limited number of possible Δ and a linear regression test of slope 0.
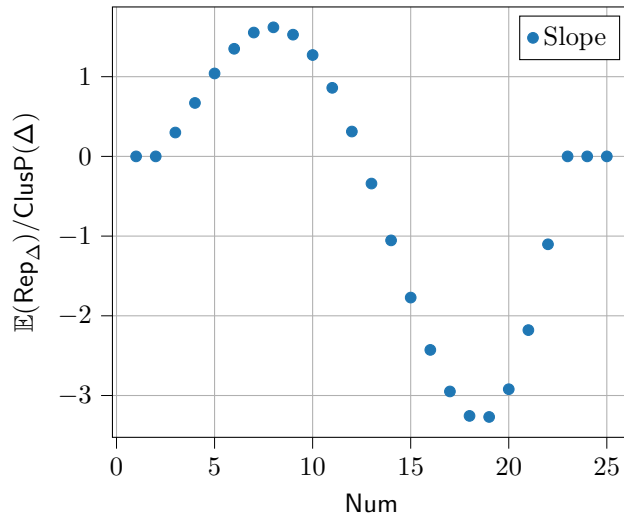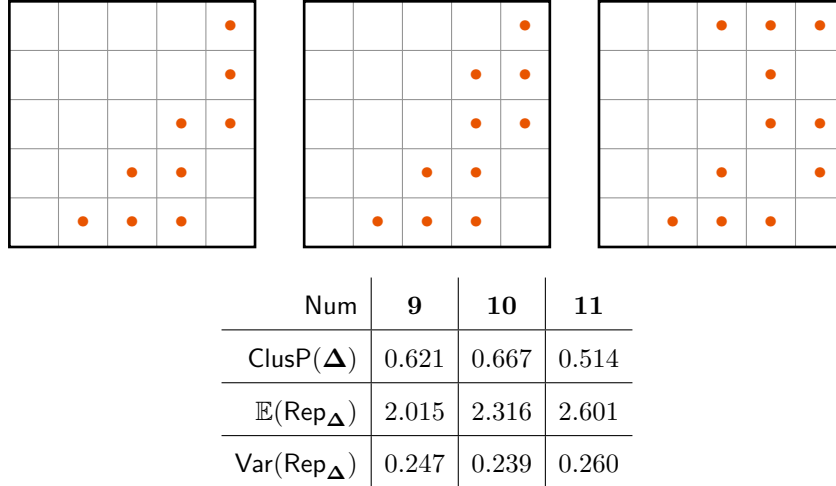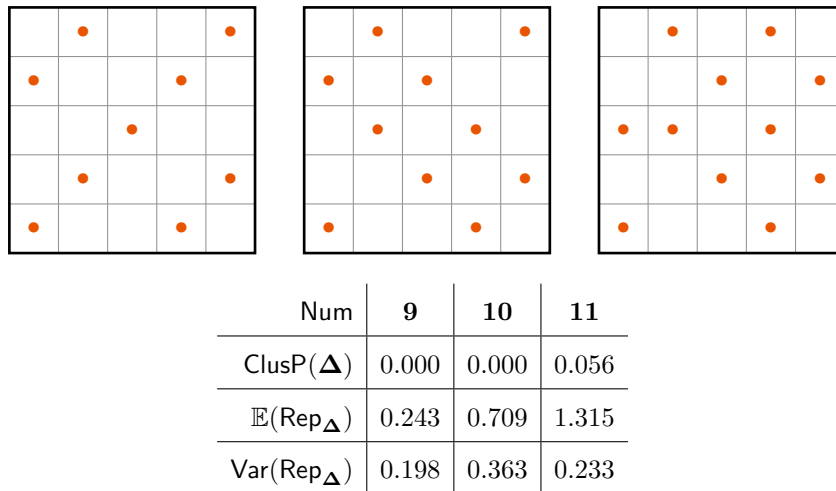

FIGURE 9. This is a graph of the extended table of Table 2, which plots Num against the ClusP and $\mathbb{E}(\mathsf{Rep}_\Delta)$ score. This gives quantitative evidence that clustering is beneficial for minorities and detrimental to majorities.

Using our data, we can also determine the **absolute** best and worst voter distributions that yield the highest and lowest amounts of representation averaged over all the districting plans. We can make observations regarding these best and worst performing voter distributions.

Num	9	10	11
ClusP($\mathbf{\Delta}$)	0.621	0.667	0.514
$\mathbb{E}(\mathsf{Rep}_{\mathbf{\Delta}})$	2.015	2.316	2.601
$\mathsf{Var}(\mathsf{Rep}_{\mathbf{\Delta}})$	0.247	0.239	0.260

FIGURE 10. 5×5 grids of optimal voter distribution with $\mathsf{Num} = 9, 10, 11$, left to right.

Num	9	10	11
ClusP($\mathbf{\Delta}$)	0.000	0.000	0.056
$\mathbb{E}(\mathsf{Rep}_{\mathbf{\Delta}})$	0.243	0.709	1.315
$\mathsf{Var}(\mathsf{Rep}_{\mathbf{\Delta}})$	0.198	0.363	0.233

FIGURE 11. 5×5 grids of worst voter distribution with $\mathsf{Num} = 9, 10, 11$, left to right.

Clustering leads to packing for the minority party, which tends to yield higher expected representation across all possible districting plans. We observe trends across all outlier cases that maximize the expected representation. In Figure 10, we show the representation-maximizing voter distributions for Dot vote share of 9, 10, and 11 out of 25. Note that although a higher clustering score generally correlates with higher expected representation, the districting plan that maximizes expected representation does not actually have the highest clustering score. An important observation is that these optimal voter distributions all contain *enclaves* of blocks of the opposing party. These enclaves mean that the opposing party is the local minority and is far away from the majority of their blocks, which is sub-optimal for the opposing party, as it is difficult for a district contain enough blocks of the opposing party to win representation. This means that the votes secured for the opposing party by these 'enclaves' are almost always wasted.

We also observe trends across all outliers that minimize expected representation. Figure 11 displays the representation-minimizing voter distributions for Dot vote share of 9, 10, and 11 out of 25. The least successful minority voter distributions have extremely low cluster scores, with each minority block adjacent

to mostly or only majority blocks. The lack of minority clustering allows the opposing (majority) party to easily crack minority votes and secure maximal representation.

This leads us to make the following conjecture:

Conjecture 3.3. (Clustering Relation). For minority Num, there is a positive correlation between ClusP and $\mathbb{E}(\mathsf{Rep}_{\Delta})$.

Figure 12 plots voter distribution, clustering score for Dots, and expected representation together on a three-dimensional plot. Figure 12 shows how expected representation varies with both clustering and vote share. It is clear from the plot that both increased clustering and increased vote share increase expected representation.
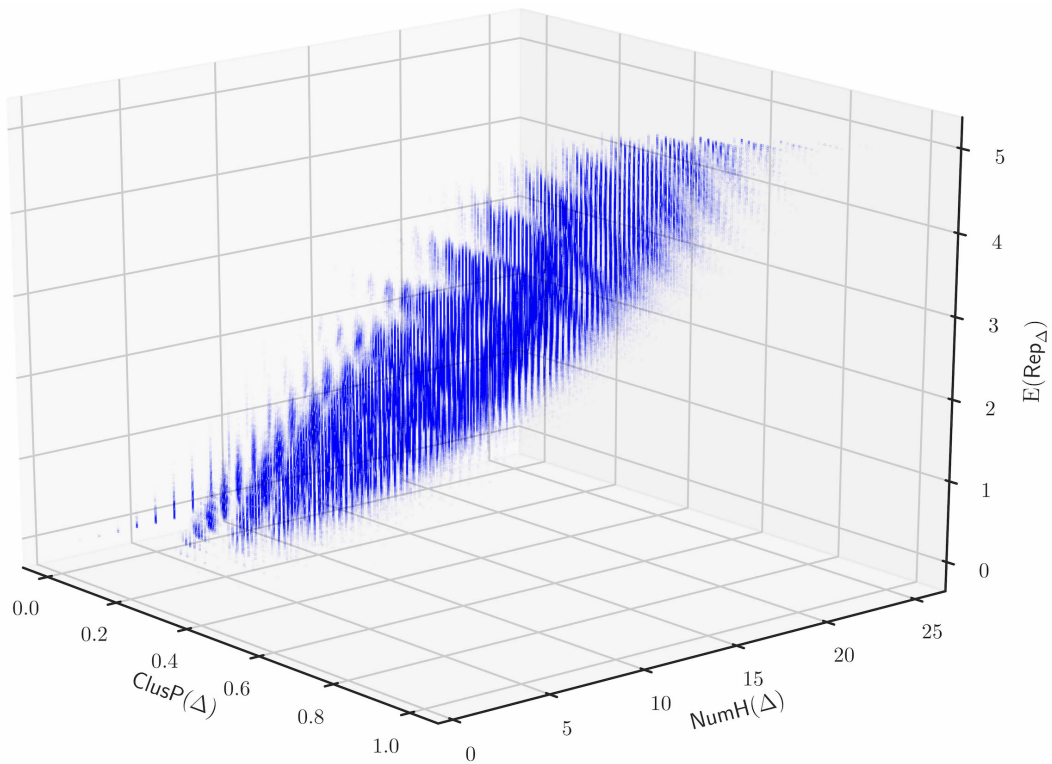

FIGURE 12. Num vs. ClusP(Δ) vs. $\mathbb{E}(\mathsf{Rep}_{\Delta})$.

4. Metaheuristics for Optimization

Recall that our goal is to find or estimate the best voter distribution that maximizes the representation of a certain political party within a geographic region split up into districts. The problem is two-fold: first, there must be a way to assign a score to a voter distribution; second, there must be an algorithm to efficiently find such a maximum within the large and discrete set of all voter distributions.

4.1. **Evaluative function.** We tackle the first problem using a known algorithm. Our score for each voter distribution is $\mathbb{E}(\mathsf{Rep}_\Delta)$, which gives the expected representation over all possible districting plans for a voter distribution. A higher score represents a voter distribution that is robust against gerrymandering, in the sense that it is difficult and improbable to create a districting plan that grossly disadvantages the party in question. This measure also aligns with the 'mean test' that the supreme court proposes. However, it is hard to compute this measure through an enumeration of all districting plans. Even for an $n \times n$ grid of squares, exhaustive enumeration of, or even counting, all possible partitions into n equal-sized districts is computationally infeasible above $n = 9$ as demonstrated in Table 1. Duchin gives and demonstrates a Metropolis-Hastings algorithm (also known as Markov Chain Monte Carlo) to stratify and uniformly sample over all districting plans [2]. We denote this stochastic algorithm as the evaluative function $\mathsf{Eval}(\Delta)$ which approximates $\mathbb{E}(\mathsf{Rep}_\Delta)$. However optimized this algorithm is, note that it is still the time-limiting step in our proposed algorithms.

Hence, we turn to the second problem, to find a voter distribution yielding maximal representation (or one which is good enough) within a large and discrete space of all voter distributions. A crude method to achieve this would simply be to generate random voter distribution, record down the respective representation that each yields, and have the algorithm return the voter distribution that yielded the highest representation after a set amount of tries. This is the *Random algorithm*, and we use it as a benchmark to evaluate our proposed algorithms. One strategy we use to tackle this problem is through a quasi-greedy algorithm that mutates a voter distribution at every step instead of generating a completely new and random one.

4.2. **Cellular automata.** *Cellular automata* are discrete models in which the state of each cell affects its neighboring cells. We apply a cellular automata algorithm to our 5×5 grid to simulate clustering.

First, we define a condition for each block that determines if it is *happy* or *unhappy*.

Definition 4.1. A block $b \in V$ is *happy* when the proportion of its like neighbors over its total number of neighbors (i.e. those it shares an edge with) is above a set threshold θ. Likewise, the block b is *unhappy* when the proportion of its like neighbors over its total number of neighbors is less than θ.

The algorithm proposes an evolution of our voter distribution Δ by taking the set of all the unhappy blocks (which could be both empty and • party blocks) and shuffling the blocks (in essence, permuting the individual blocks). Notably, this preserves Num and achieves an effect of increasing the clustering score. This is due to the fact that the blocks we are perturbing contribute to a lower clustering score (as they have a lower number of like neighbors), so we are decreasing the number of 'unlike' connections. Its implementation is given in Algorithm 1.

Algorithm 1 Cellular automata evolution

Require:

Dual graph $G = (V, E)$, discrete voter distribution function $\Delta : V \to \{0, 1\}$, and threshold $\theta \in [0, 1]$.

1: **procedure** EVOLVE(Δ)   ▷ Initial voter distribution Δ

2:      $S \leftarrow \{\}$ ▷ Set of blocks b with proportion $< \theta$ of similar neighbors

3:      $\{\Delta\}_{\text{swap}} \leftarrow \{\}$ ▷ $\{\Delta\}_{\text{swap}} = \{\Delta(b) \mid b \in S\}$

4:      **for** b in V **do**

5:         $conn_{total} \leftarrow 0$

6:         $conn_{\text{same}} \leftarrow 0$

7:         **for** b_n in $\{b_n \mid (b, b_n) \in E\}$ **do**   ▷ For all blocks adjacent to b

8:            **if** $\Delta(b) = \Delta(b_n)$ **then** $conn_{\text{same}} \leftarrow conn_{\text{same}} + 1$ ▷ # of similar connections

9:            $conn_{\text{total}} \leftarrow conn_{\text{total}} + 1$ ▷ # of total connections

10:        **end for**   ▷ After enumeration of b_n

11:        **if** $conn_{\text{same}}/conn_{\text{total}} < \theta$ **then**

12:           $S \leftarrow S \cup \{b\}$

13:           **append** $\Delta(b) \to \{\Delta\}_{\text{swap}}$, preserving order

14:        **end if**

15:     **end for**   ▷ After enumeration of b in V

16:     **shuffle** $\{\Delta\}_{\text{swap}}$ ▷ Permutes indices of $\Delta_i \in \{\Delta\}_{\text{swap}}$

17:     **define** $\Delta'(b) = \begin{cases} \Delta_i \text{ in } \{\Delta\}_{\text{swap}} & \textbf{for } b = b_i \in S \\ \Delta(b) & \textbf{otherwise} \text{ (i.e. } b \notin S) \end{cases}$

18:     **return** Δ' ▷ Returns mutated voter distribution function Δ'

19: **end procedure**

The progression below gives a visual example of what the cellular automata algorithm does in each iteration.
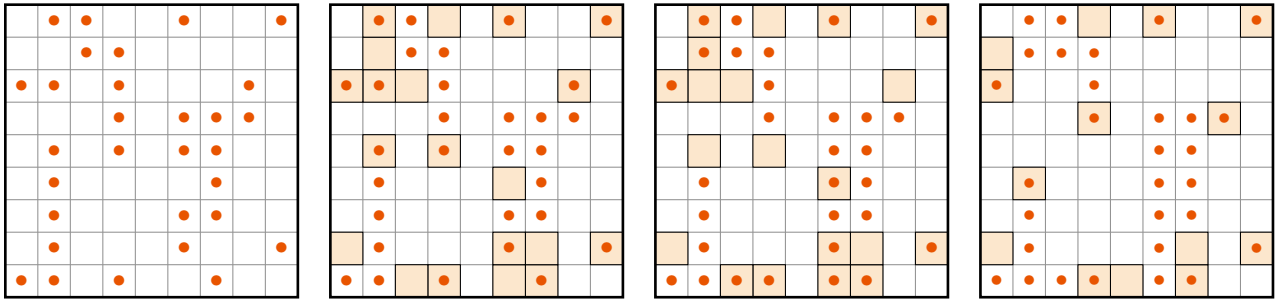

FIGURE 13. A 9×9 square grid with $\mathsf{Num} = 30$ demonstrating the respective steps of the cellular automata evolution, with $\theta = 0.4$.

The first grid on the left of Figure 4.2 is our initial voter distribution Δ. We can identify all the unhappy tiles on our grid by enumerating the tiles where the proportion of like tiles connected to it is less than 0.4. In our case, this means a tile with less than 2 similar edges out of 4 or 3 total edges, or 0 similar edges if it

has 2 total outgoing edges. Unhappy tiles are highlighted on the second grid. The algorithm then shuffles all the unhappy tiles around, leaving the happy tiles as is. This produces the diagram in the third grid, which still highlights the unhappy tile locations after being shuffled. Those blocks that are not highlighted have not been changed. The last grid demonstrates what re-evaluating the unhappy tiles in our new grid would look like. As you can see, the number of unhappy tiles has decreased.

Relying on our conclusion in Conjecture 3.3 and the fact that cellular automata generally gives a more clustered voter distribution with every evolution, we can say that the cellular automata algorithm also improves expected representation $\mathbb{E}(\mathsf{Rep}_\Delta)$ for the minority. Hence, we can use the cellular automata algorithm to approximate a quasi-greedy algorithm, which attempts to find similar voter distributions with better expected representation every time. This becomes useful for Algorithms 2 and 3 detailed below, both of which rely on the fact that we can easily find or generate an optimal 'neighbor' given some voter distribution Δ. This is non-trivial, given that $\{\Delta\}$ is discrete and so large.

4.3. **Random-restart iterated local search algorithm (RRILS).** Using the cellular automata algorithm as a quasi-greedy algorithm allows us to apply this algorithm repeatedly to generate increasingly better voter distributions (hence the 'iterated local search'). Of course, this algorithm is succeptible to terminating at a local maxima when we are trying to find the global maxima instead. We can randomly restart the algorithm at some given point any time we reach a local maxima, and this gives a relatively straightforward method for finding a stronger maxima.

In words, the algorithm executes as follows: first, we set the number of trials as $k_{\max}$. The algorithm will then generate a random voter distribution, Δ. It then approximates the $\mathbb{E}(\mathsf{Rep})$ of this Δ using the evaluative function Eval detailed in subsection 4.1. Then using the greedy algorithm, which shuffles unhappy blocks by swapping them, RRILS can propose a greedy evolution using cellular automata. If there are no new evolutions to be proposed, another random initial voter distribution will be generated. This process repeats, and after iterating $k_{\max}$ times, the algorithm return the Δ with maximum $\mathbb{E}(\mathsf{Rep}_\Delta)$ that it has found.

Algorithm 2 Random-restart iterated local search

Require:

    Evaluative function $\mathsf{Eval} : \{\Delta\} \to \mathbb{R}$ which approximates $\mathbb{E}(\mathsf{Rep}_\Delta)$.

1: **procedure** RANDOMRESTART($k_{\max}$)

2:      $k \leftarrow 0$

3:      **while** $k < k_{\max}$ **do**

4:          $\Delta \leftarrow \mathrm{rand}(\Delta)$ ▷ Generates a random initial voter distribution

5:          $\Delta_{\mathrm{new}} \leftarrow \mathrm{null}$ ▷ Always accepts initial Δ

6:          **while** $\Delta \neq \Delta_{\mathrm{new}}$ **do**

7:              $\mathrm{Trials}[\Delta] \leftarrow \mathsf{Eval}(\Delta)$ ▷ Appends $\mathsf{Eval}(\Delta)$ to dictionary <u>Trials</u> with key Δ

8:              $\Delta_{\mathrm{new}} \leftarrow \mathrm{EVOLVE}(\Delta)$ ▷ Proposes an evolution of Δ

9:              $k \leftarrow k+1$

10:         **end while**

11:      **end while**

12:      $\Delta_{\max} \leftarrow \kappa$ such that $\mathrm{Trials}[\kappa]$ maximal

13:      **return** $\Delta_{\max}$ ▷ Returns Δ that gives maximal $\mathsf{Eval}(\Delta)$

14: **end procedure**

4.4. **Simulated annealing.** *Simulated Annealing* is another metaheuristic designed for optimization, meant for cases where the search space is discrete, such as the space of all voter distributions $\{\Delta\}$.

The algorithm tries to achieve a balance between exploiting (descending a gradient to reach local extrema) such as an iterated local search algorithm, as well as exploring (sampling the search space completely randomly for optimal voter distributions). The algorithm begins with an initial temperature (parameter) of T_0. This is the maximum temperature it will ever be at and represents a tendency for the algorithm to explore rather than exploit, which means it readily accepts worse states of proposed voter distributions. With every iteration, the temperature T becomes $T \cdot \alpha$ due to a cooling schedule α, which lowers the temperature. As

the temperature decreases, the likelihood of accepting a worse state also decreases (defined by a function Prob).

This first version of simulated annealing we present is modified to take into account a known greedy algorithm. Similar to the RRILS, we evaluate Δ's and propose an evolution of Δ. The algorithm then accepts this proposed state with a probability $\mathsf{Prob}(\delta_{score}, T)$ such that $\mathsf{Prob}(\delta_{score}, T) = e^{\delta_{\mathrm{score}}/T}$. The algorithm then decreases the temperature (if it chooses to accept the proposed state), or accepts the random state with a probability of θ_r which is a predefined constant. This process repeats, and after iterating $k_{\max}$ times, the algorithm returns the Δ which maximizes $\mathbb{E}(\mathsf{Rep}_\Delta)$.

Algorithm 3 Simulated annealing

Require:

Evaluative function $\mathsf{Eval} : \{\Delta\} \to \mathbb{R}$ which approximates $\mathbb{E}(\mathsf{Rep}_\Delta)$, acceptance probability function $\mathsf{Prob} : \{\Delta\} \times \mathbb{R} \to [0,1] \in \mathbb{R}$, and following hyper-parameters: initial temperature T_0, cooling schedule $\alpha \in [0,1]$, threshold θ (of EVOLVE), and constant probability of accepting random state θ_r.

1: **procedure** SIMULATEDANNEAL($k_{\max}$)
2:     $k \leftarrow 0$
3:     $T \leftarrow T_0$ ▷ Begins at initial temperature T_0
4:     $\Delta \leftarrow \mathrm{rand}(\Delta)$
5:     $score \leftarrow \mathsf{Eval}(\Delta)$
6:     **while** $k < k_{\max}$ **do**
7:        $\mathrm{Trials}[\Delta] \leftarrow score$ ▷ Appends $\mathsf{Eval}(\Delta)$ to dictionary <u>Trials</u> with key Δ
8:        $\Delta_{\mathrm{new}} \leftarrow \mathrm{EVOLVE}(\Delta)$ ▷ Proposes an evolution of Δ
9:        $score_{\mathrm{new}} \leftarrow \mathsf{Eval}(\Delta_{\mathrm{new}})$
10:       $\delta_{\mathrm{score}} \leftarrow score_{\mathrm{new}} - score$
11:       **if** $\mathsf{Prob}(\delta_{\mathrm{score}}, T) > \mathrm{rand}(0,1)$ **then**   ▷ Accepts proposed state with $\mathsf{Prob}(\delta_{\mathrm{score}}, T)$
12:          $\Delta \leftarrow \Delta_{\mathrm{new}}$
13:          $score \leftarrow score_{\mathrm{new}}$
14:          $T \leftarrow T \cdot \alpha$ ▷ Decreases temperature
15:       **else if** $\theta_r > \mathrm{rand}(0,1)$ **then**   ▷ Otherwise accepts random state with probability θ_r
16:          $\Delta \leftarrow \mathrm{rand}(\Delta)$
17:          $score \leftarrow \mathsf{Eval}(\Delta)$
18:       **end if**
19:       $k \leftarrow k + 1$
20:     **end while**
21:     $\Delta_{\max} \leftarrow \kappa$ such that $\mathrm{Trials}[\kappa]$ maximal
22:     **return** $\Delta_{\max}$ ▷ Returns Δ that gives maximal $\mathsf{Eval}(\Delta)$
23: **end procedure**

4.5. **Random Simulated Annealing.** *Random Simulated Annealing* works largely the same way as the aforementioned simulated annealing in Algorithm 3. However, the steps that this algorithm takes are no longer the greedy steps prescribed by the cellular automata evolution algorithm. Where Algorithm 3 would

call EVOLVE on line 8, Random Simulated Annealing calls STEP. STEP takes a parameter n, the number of blocks to shuffle, randomly selects n blocks, and shuffles their political preferences in place on the dual graph. This is very similar to what our cellular automata algorithm EVOLVE does, but instead of selecting a set of blocks that fits a certain criteria to swap (in EVOLVE we selected 'unhappy' blocks), STEP just selects n random blocks to swap. From our tests, we have found that it is best that the random threshold θ_r be set to 0, which means that the algorithm will never select a totally random state. This is because the steps that the algorithm is making is already random and is not susceptible to being trapped at a local extrema point.

One benefit of the random variant of simulated annealing is that it does not rely on the Clustering Conjecture, and hence, we can use this algorithm for both the minority and majority parties.

4.6. **Benchmark Algorithm.** We use a naive benchmark algorithm to test the performance of the proposed algorithms. The benchmark algorithm simply generates new voter distributions of a set Num and evaluates them at every step. It returns the voter distribution that has yielded the largest expected representation for a given Num.

4.7. **Results.** Using the 3 algorithms detailed in Subsections 4.3, 4.4, and 4.5, we can evaluate the computational improvements that they provide over our benchmark algorithm (and the next best naive alternative), the random sampling in Subsection 4.6.

In these trials, we use the dual graph of the 5×5 square grid as described in Section 3. Since we have exhaustively analyzed this case, we know the maximum outcomes for each Num, which gives us yet another benchmark to compare the results to. We run simulations by stochastically sampling the performance of each algorithm a large number of times (here, we run 10,000 trials) and cap each $k_{\max}$ to a certain value. For each $k_{\max}$ value, we can determine the average outcome (the optimized maximum expected representation) of each algorithm. As $k_{\max}$ is the number of iterations our program makes (or the computation time of our program), the algorithm also produces results that are closer to the global maximum as it increases.

Plotting out each of these results gives Figure 14, where the rates of each algorithm are visually displayed for Num = 6 and up to a $k_{\max}$ of 1000. For the case of Num = 6, the random-restart iterated local search algorithm performed similarly to simulated annealing, both of which performed better than the random variation of simulated annealing. Notably, all three algorithms asymptotically approached the absolute maximum significantly faster than random selection.

However, when we change Num to equal 10, comparing the algorithms gives us different results. Figure 15 is a variation on Figure 14, where Num = 6. In this case, the random variation of simulated annealing performs better than both random-researt iterated local search and the original greedy variant of simulated annealing.

Unfortunately, we can only conclude that our algorithms differ on a case-by-case basis and that it requires additional evaluation to determine the best algorithm for such an optimization.
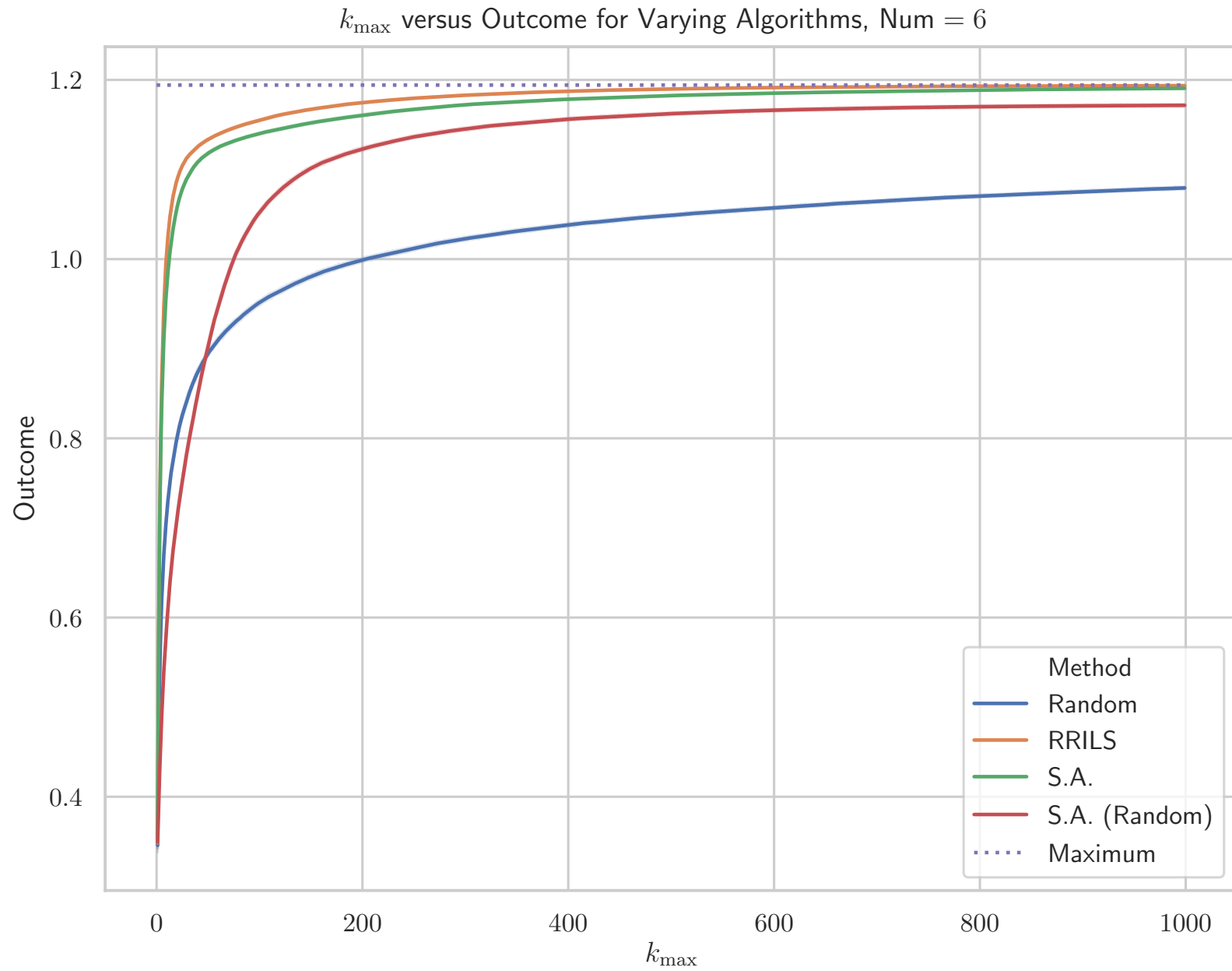
FIGURE 14. The rates at which each algorithm approaches the absolute known maximum for a 5×5 grid of Num $= 6$ for differing $k_{\max}$.
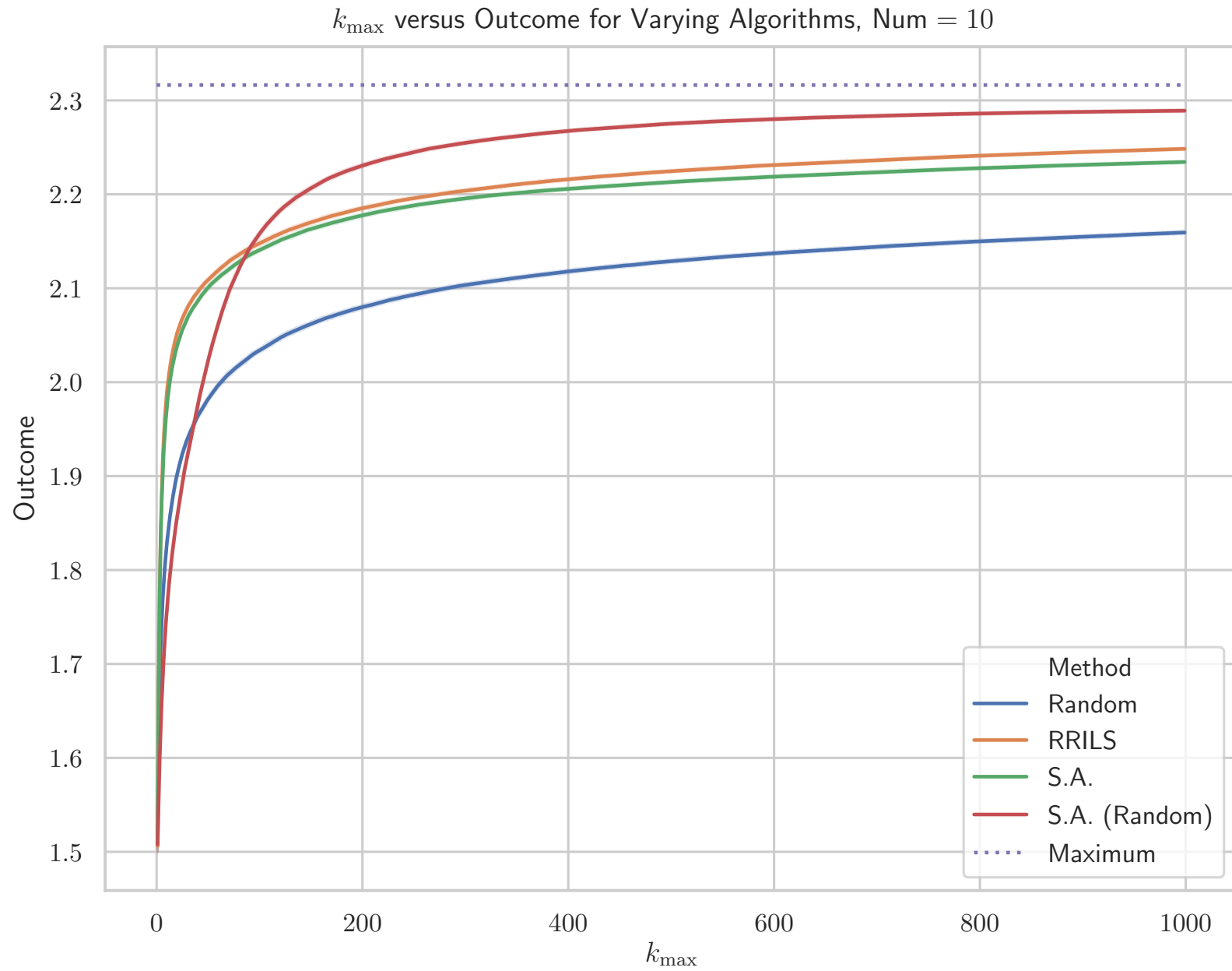
FIGURE 15. The rates at which each algorithm approaches the absolute known maximum for a 5×5 grid of Num $= 10$ for differing $k_{\max}$.

5. Conclusion and Further Research

By performing an exhaustive search all possible voter distributions and districting plans of a 5×5 grid, we were able to observe that a positive correlation exists between clustering and expected representation, giving us statistical evidence that clustering is beneficial for the minority. We also analyzed outlier cases of voter distribution that give maximum and minimum representation for a certain Dot vote share, discovering trends that indicate clustering generally leads to a larger amount of expected representation. This observation specifically extends and disproves, for certain cases, the observations of Chen, Rodden [1]. We use this result to implement the cellular automata, random-restart iterated local search, and simulated annealing algorithms to optimize the voter distribution for the highest expected representation.

For large n, starting with $n > 5$, it becomes much harder to exhaustively search all districting plans and voter distributions. We hope to generalize many of our theorems and findings to larger grids, as well as other uniform geometries such as triangular and hexagonal tilings, as well as general dual graphs.

An important direction of future gerrymandering research may be to determine methods which give representative random samples of a voter distribution given a particular districting plan or voter distribution. While MCMC already allows mathematicians to search the metagraph of districting plans using a random walk, there is no equivalent method to random walk on the space of all voter distributions in a way that provides a representative sample (the search space of all voter distributions is much larger than the search space of all districting plans).

Extensive evidence also shows that the distribution of seats won over varying districting plans for a given population distribution is approximately normal, and we believe the central limit theorem could be leveraged to prove this observation. This allows us to use implications of normal distributions to our advantage when analyzing the effectiveness or result of a given voter distribution.

We hope that our research serves as insight into the benefits of clustering for minority parties and as a basis for future research into exhaustive searches on other naive grids. We also hope that it gives insight into patterns and quantifications of specific voter distributions that do not rely on stratified MCMC searches to estimate the seats distribution.
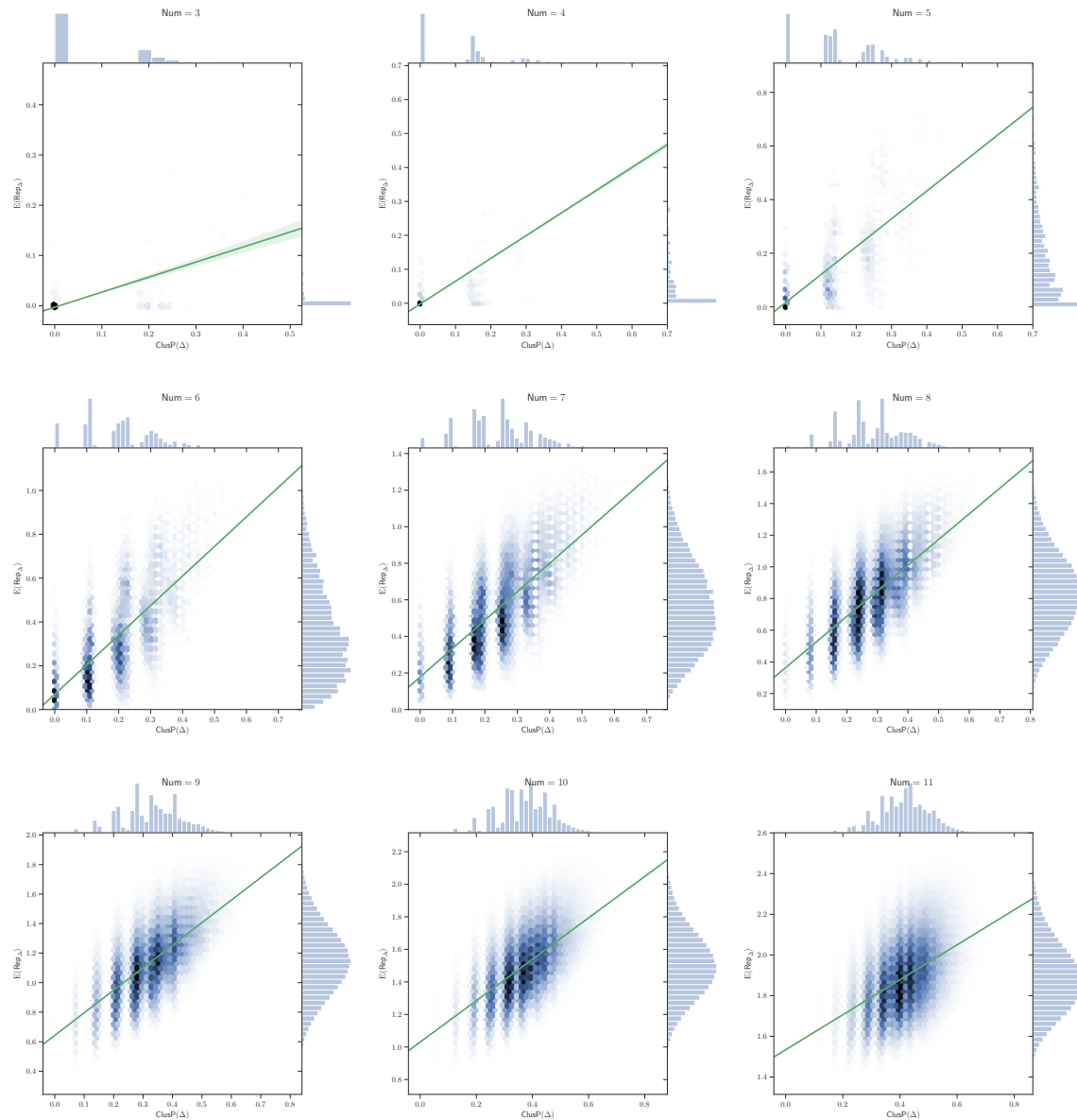
REFERENCES

[1] Jowei Chen. "Unintentional Gerrymandering: Political Geography and Electoral Bias in Legislatures". In: *Quarterly Journal of Political Science* 8.3 (2013), pp. 239–269. ISSN: 15540634. DOI: 10.1561/100. 00012033. URL: http://www.nowpublishers.com/article/Details/QJPS-12033.

[2] Moon Duchin. "Gerrymandering metrics: How to measure? What's the baseline?" In: (2018).

[3] Moon Duchin et al. "Locating the representational baseline: Republicans in Massachusetts". In: (2018). URL: http://arxiv.org/abs/1810.09051.

[4] Raymond Gavins and Raymond Gavins. "Voting Rights Act of 1965". In: *The Cambridge Guide to African American History*. 2016. DOI: 10.1017/cbo9781316216453.299.

[5] Bob Harris. *Counting Nonomino Tilings and Other Things of that Ilk*. Tech. rep. 2010.

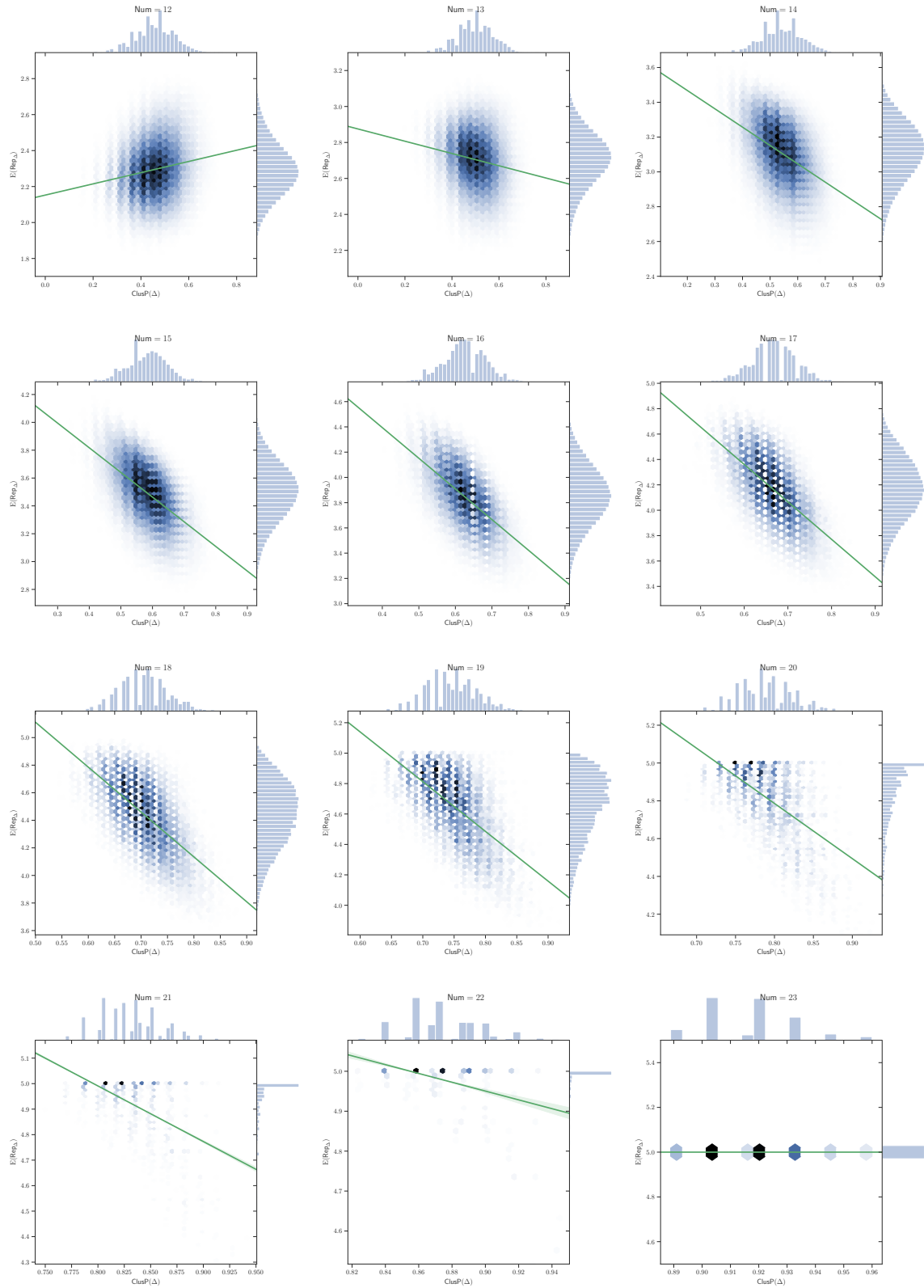[6] *Rucho et al. v. Common Cause et al.* 2019.

APPENDIX A. COMPUTATION

The code used to perform our search, as well as the code for our metaheuristic algorithms (and their evaluations), can be found at https://github.com/jiahuac/GerryGrid.

APPENDIX B. ClusP AGAINST $\mathbb{E}(\mathsf{Rep})$ GIVEN Num

We include the ClusP vs. $\mathbb{E}(\mathsf{Rep})$ graphs for varying Num for a 5×5 grid.