Supreme Court Bars Challenges to Partisan Gerrymandering

Assistants run outside the Supreme Court after the court rejected efforts to rein in partisan gerrymandering. Samuel Corum for The New York Times
Gerrymandering

- Manipulating electoral district boundaries to favor one political party over others
- Coined in an 1812 Political cartoon
- Governor Elbridge Gerry signed a bill that redistricted Massachusetts to benefit his Democratic-Republican Party

The Gerrymander
According to the Supreme Court

- Gerrymandering cannot be used to:
  - Disadvantage racial/ethnic/religious groups
- It can be used to:
  - Disadvantage political parties
A loophole

In many cases, rural, predominantly white towns see their population numbers boosted by population counts from prisons disproportionately made up of black and Latinx people.
Virginia Congressional Districts would be drawn by:
• An 16-member panel
• 2 members from each of the two political parties with the most representatives in the Senate and General assembly
• 8 members are citizens selected by a committee of 5 retired circuit court judges
VA 5\textsuperscript{th} District

### 2018 Election

<table>
<thead>
<tr>
<th></th>
<th>Votes</th>
<th>Pct.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>165,339</td>
<td>53.3%</td>
</tr>
<tr>
<td>Blue</td>
<td>145,040</td>
<td>46.7%</td>
</tr>
</tbody>
</table>

### 2020 Election

*It's a toss-up in Virginia's 5th District*

The Deluxe version of our model simulates the election 40,000 times to see who wins most often. The sample of 100 outcomes below gives you a good idea of the range of scenarios the model thinks is possible.

- Bob Good wins \textit{48 in 100}
- Cameron Webb wins \textit{52 in 100}
Gerrymandering Today

Computers make it more effective
How does it work?

- States are broken into precincts
- All precincts have the same size
- We know voting preferences of each precinct
- Group precincts into districts to maximize the number of districts won by my party

Overall: R:217 D:183

<table>
<thead>
<tr>
<th></th>
<th>R:65</th>
<th>R:45</th>
</tr>
</thead>
<tbody>
<tr>
<td>D:35</td>
<td></td>
<td>D:55</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>R:60</th>
<th>R:47</th>
</tr>
</thead>
<tbody>
<tr>
<td>D:40</td>
<td></td>
<td>D:53</td>
</tr>
</tbody>
</table>

vs

Coca-Cola vs Diet Coke
How does it work?

- States are broken into precincts
- All precincts have the same size
- We know voting preferences of each precinct
- Group precincts into districts to maximize the number of districts won by my party

<table>
<thead>
<tr>
<th>Overall: R:217 D:183</th>
</tr>
</thead>
<tbody>
<tr>
<td>R:65</td>
</tr>
<tr>
<td>D:35</td>
</tr>
<tr>
<td>R:45</td>
</tr>
<tr>
<td>D:55</td>
</tr>
<tr>
<td>R:60</td>
</tr>
<tr>
<td>D:40</td>
</tr>
<tr>
<td>R:47</td>
</tr>
<tr>
<td>D:53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R:125</th>
<th>R:92</th>
</tr>
</thead>
<tbody>
<tr>
<td>R:65</td>
<td></td>
</tr>
<tr>
<td>D:35</td>
<td></td>
</tr>
<tr>
<td>R:45</td>
<td></td>
</tr>
<tr>
<td>D:55</td>
<td></td>
</tr>
<tr>
<td>R:60</td>
<td></td>
</tr>
<tr>
<td>D:40</td>
<td></td>
</tr>
<tr>
<td>R:47</td>
<td></td>
</tr>
<tr>
<td>D:53</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R:112</th>
<th>R:105</th>
</tr>
</thead>
<tbody>
<tr>
<td>R:65</td>
<td></td>
</tr>
<tr>
<td>D:35</td>
<td></td>
</tr>
<tr>
<td>R:45</td>
<td></td>
</tr>
<tr>
<td>D:55</td>
<td></td>
</tr>
<tr>
<td>R:60</td>
<td></td>
</tr>
<tr>
<td>D:40</td>
<td></td>
</tr>
<tr>
<td>R:47</td>
<td></td>
</tr>
<tr>
<td>D:53</td>
<td></td>
</tr>
</tbody>
</table>
Gerrymandering Problem Statement

• Given:
  – A list of precincts: \( p_1, p_2, \ldots, p_n \)
  – Each containing \( m \) voters

• Output:
  – Districts \( D_1, D_2 \subseteq \{p_1, p_2, \ldots, p_n\} \)
  – Where \( |D_1| = |D_2| \)
    • \( \frac{n}{2} \) precincts per district (so \( \frac{mn}{2} \) voters)
  – \( R(D_1) > \frac{mn}{4} \) and \( R(D_2) > \frac{mn}{4} \)
    • \( R(D_i) \) gives number of “Regular Party” voters in \( D_i \)
    • \( R(D_i) > \frac{mn}{4} \) means \( D_i \) is majority “Regular Party”
  – “failure” if no such solution is possible

Successful
Gerrymandering!
Idea for the Algorithm

• Given:
  – A list of precincts $v_1, ..., v_n$
  – Each precinct has $m$ voters

• Question:
  – Is there some way to assign $\frac{n}{2}$ precincts to each district such that the number of $R$ voters exceeds $\frac{mn}{4}$ in both districts?

• Generalization:
  – Is there some way to assign some of the precincts to district 1 and some other precincts to district 2 such that each has $x$ and $y$ $R$ voters, respectively.

• Approach:
  – Solve the generalized version, then check for a “yes” where both districts have $\frac{n}{2}$ precincts, and $x$ and $y$ are both over $\frac{mn}{4}$
Dynamic Programming

• Requires **Optimal Substructure**
  – Solution to larger problem contains the solutions to smaller ones

• Idea:
  1. Identify the recursive structure of the problem
     • What is the “last thing” done?
  2. Save the solution to each subproblem in memory
  3. Select a good order for solving subproblems
     • “Top Down”: Solve each recursively
     • “Bottom Up”: Iteratively solve smallest to largest
Dynamic Programming

• Requires **Optimal Substructure**
  – Solution to larger problem contains the solutions to smaller ones

• Idea:
  1. Identify the recursive structure of the problem
     • What is the “last thing” done?
  2. Save the solution to each subproblem in memory
  3. Select a good order for solving subproblems
     • “Top Down”: Solve each recursively
     • “Bottom Up”: Iteratively solve smallest to largest
Consider the last precinct

After assigning the first \(n - 1\) precincts \(p_1, p_2, \ldots, p_{n-1}\)

**World One**

- \(D_1\):
  - \(k\) precincts
  - \(x\) voters for R

- \(D_2\):
  - \(n - k - 1\) precincts
  - \(y\) voters for R

---

If we assign \(p_n\) to \(D_1\)

- \(D_1\):
  - \(k + 1\) precincts
  - \(x + R(p_n)\) voters for R

Valid gerrymandering if:

- \(k + 1 = \frac{n}{2}\)
- \(x + R(p_n), y > \frac{mn}{4}\)

---

If we assign \(p_n\) to \(D_2\)

- \(D_2\):
  - \(n - k\) precincts
  - \(y + R(p_n)\) voters for R

Valid gerrymandering if:

- \(n - k = \frac{n}{2}\)
- \(x, y + R(p_n) > \frac{mn}{4}\)
Define Recursive Structure

\[ S(j, k, x, y) = \text{True if from among the first } j \text{ precincts:} \]
\[ k \text{ are assigned to } D_1 \]
\[ \text{exactly } x \text{ vote for R in } D_1 \]
\[ \text{exactly } y \text{ vote for R in } D_2 \]

4D Dynamic Programming!!!
Two ways to satisfy $S(j, k, x, y)$:

1. **Case 1:**
   - $D_1$: $k - 1$ precincts
   - $x - R(p_j)$ voters for R
   - Then assign $p_j$ to $D_1$

   OR

2. **Case 2:**
   - $D_1$: $k$ precincts
   - $x$ voters for R
   - Then assign $p_j$ to $D_1$

   - $D_2$: $j - k$ precincts
   - $y - R(p_j)$ voters for R
   - Then assign $p_j$ to $D_2$

$S(j, k, x, y) = S(j - 1, k - 1, x - R(p_j), y) \lor S(j - 1, k, x, y - R(p_j))$
Final Algorithm

\[ S(j, k, x, y) = S(j - 1, k - 1, x - R(p_j), y) \lor S(j - 1, k, x, y - R(p_j)) \]

Initialize \( S(0,0,0,0) = \text{True} \)

for \( j = 1, \ldots, n \):
  for \( k = 1, \ldots, \min(j, \frac{n}{2}) \):
    for \( x = 0, \ldots, jm \):
      for \( y = 0, \ldots, jm \):
        \[ S(j, k, x, y) = \]
        \[ S(j - 1, k - 1, x - R(p_j), y) \lor S(j - 1, k, x, y - R(p_j)) \]

Search for True entry at \( S(n, \frac{n}{2}, > \frac{mn}{4}, > \frac{mn}{4}) \)

\( S(j, k, x, y) = \text{True if:} \)
  \( \text{from among the first } j \text{ precincts} \)
  \( k \text{ are assigned to } D_1 \)
  \( \text{exactly } x \text{ vote for } R \text{ in } D_1 \)
  \( \text{exactly } y \text{ vote for } R \text{ in } D_2 \)
Final Algorithm

\[ S(j, k, x, y) = S(j - 1, k - 1, x - R(p_j), y) \lor S(j - 1, k, x, y - R(p_j)) \]

Initialize \( S(0,0,0,0) = \text{True} \)

for \( j = 1, \ldots, n \):
    for \( k = 1, \ldots, \min(j, \frac{n}{2}) \):
        for \( x = 0, \ldots, jm \):
            for \( y = 0, \ldots, jm \):
                \[ S(j, k, x, y) = S(j - 1, k - 1, x - R(p_j), y) \lor S(j - 1, k, x, y - R(p_j)) \]

Search for True entry at \( S(n, \frac{n}{2}, > \frac{mn}{4}, > \frac{mn}{4}) \)

\[ S(j, k, x, y) = \text{True if:} \]

- from among the first \( j \) precincts \( k \) are assigned to \( D_1 \)
- exactly \( x \) vote for R in \( D_1 \)
- exactly \( y \) vote for R in \( D_2 \)
Run Time

\[ S(j, k, x, y) = S(j - 1, k - 1, x - R(p_j), y) \lor S\left(j - 1, k, x, y - R(p_j)\right) \]

- Initialize \( S(0, 0, 0, 0) = \text{True} \)
- for \( j = 1, \ldots, n: \)
  - for \( k = 1, \ldots, \min(j, \frac{n}{2}): \)
    - for \( x = 0, \ldots, jm: \)
      - for \( y = 0, \ldots, jm: \)
        - \( S(j, k, x, y) = S(j - 1, k - 1, x - R(p_j), y) \lor S\left(j - 1, k, x, y - R(p_j)\right) \)

Search for True entry at \( S(n, \frac{n}{2}, \frac{mn}{4}, \frac{mn}{4}) \)

\( \Theta(n^4m^2) \)
\[ \Theta(n^4m^2) \]

- Input: list of precincts (size \( n \)), number of voters (integer \( m \))
- Runtime depends on the value of \( m \), not size of \( m \)
  - Run time is exponential in size of input
  - Input size is \( n + |m| = n + \log m \)
- Note: Gerrymandering is NP-Complete