2024

USING BOARD GAMES FOR RESEARCH IN MATHEMATICAL REDISTRICTING

Ventsislav Gotov
University of Rhode Island, vgotov@gmail.com

USING BOARD GAMES FOR RESEARCH IN MATHEMATICAL REDISTRICTING

BY

VENTSISLAV GOTOV

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

UNIVERSITY OF RHODE ISLAND

2024

MASTER OF SCIENCE THESIS

OF

VENTSISLAV GOTOV

APPROVED:

Thesis Committee:

Major Professor Edmund Lamagna

Noah Daniels

Lubos Thoma

Brenton DeBoef

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2024

ABSTRACT

Gerrymandering is the manipulation of electoral district borders in a way that gives one party or group an unfair advantage over another in elections. In recent years, the problem has received renewed attention from computer scientists and mathematicians seeking to offer methods for the fair redistricting of electoral maps. In this work we demonstrate how board games can be used as a tool for conducting research in mathematical redistricting. By studying the issue on a small scale, we can draw insights applicable to the larger, computationally difficult problem of real world redistricting. Specifically, we develop a measurement that quantifies the susceptibility of a party's vote to the gerrymandering practice of "packing". We then assess the reliability of this measurement. Finally, we leverage the new measurement to test a hypothesis in order to investigate the effect of low "packability" scores on election outcomes.

ACKNOWLEDGMENTS

I would like to express my appreciation to my research advisor, Dr. Edmund Lamagna, for his role in guiding the selection of my thesis topic and for his support and insight during the research process. I am also grateful for the opportunity he provided me to present my work at MAA MathFest.

I would also like to extend my thanks to my thesis committee members, Dr. Noah Daniels, Dr. Lubos Thoma, and Dr. William Kinnersley, for their time and effort in reviewing this work and for their valuable feedback.

Lastly, I am deeply thankful to my family and friends for their continuous love and support during my work.

DEDICATION

To my father, Valentin Gotov, who passed away during my graduate studies.

Love you Dad.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGMENTS iii

DEDICATION iv

TABLE OF CONTENTS v

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER

1 Introduction 1

1.1 Political redistricting and gerrymandering 1

1.2 Packing and cracking 1

1.3 The Distrix board game 2

2 Measuring the Susceptibility to Packing 6

2.1 Expected value benchmark score 6

2.1.1 Recursive backtracking tiling algorithm 7

2.1.2 Algorithm complexity 12

2.1.3 Speeding up the tiling algorithm 13

2.1.4 Distributions of seats won 15

2.1.5 Expected Value score calculation 16

2.2 Packability score design 18

2.2.1 Sliding kernel algorithm 18

Page

2.2.2 Parameterizing the algorithm 22

3 Evaluation of Score Reliability 27

3.1 Experimental design 27

3.1.1 Sampling random boards 27

3.1.2 Choosing algorithm parameters 29

3.1.3 Evaluating score reliability using correlation 29

3.2 Experimental results 30

3.2.1 Scoring results 30

3.2.2 Score deficiencies 34

4 Hypotheses Testing 35

4.1 Low P_B scores and tied district plans 35

4.2 The TTP_B benchmark score 36

4.3 Correlation analysis 38

4.4 The hypothesis test 38

4.5 Empirical observations 39

5 Future work 41

5.1 Logistic regression for detecting biased boards 41

5.2 A measure for the susceptibility to cracking 41

5.3 Sliding kernel algorithm parameters 42

6 Summary 44

LIST OF REFERENCES 46

APPENDIX

Page

A Kernels Used in the Experiment 48

A.1 W kernels 48

A.2 V kernels 49

A.3 T_{19} kernels 50

B Another Redistricting Game 52

BIBLIOGRAPHY 56

LIST OF TABLES

Table		Page
1	The number of tetromino placements for each tetromino	9
2	Descriptive statistics of the sample of 1500 random boards	28
3	P_B scores for all kernels	29
4	Correlation coefficient results for *all kernels/EV*	31
5	Ranges of strength for statistical correlation	32
6	$P_B(W_{2\times4}, K_s = 1)$ scores descriptive statistics	33
7	TTP_B scores descriptive statistics	36
8	Mean $\lvert P_B(W_{2\times4}, K_s = 1)\rvert$ and TTP_B scores	40

LIST OF FIGURES

Figure		Page
1	Packing and cracking	3
2	Distrix puzzle-book example	4
3	Tetrominoes	5
4	Generating the list of tetromino placements	8
5	Speeding up the recursive backtracking algorithm	14
6	A random Distrix board	15
7	Distribution of seats won	15
8	An example 3x4 kernel	18
9	Sliding kernel algorithm	19
10	$W_{3\times4}$	22
11	$V_{3\times4}$	23
12	S tetromino	24
13	EV scores distribution	28
14	$P_B(W_{2\times4}, K_s = 1)$ scores distribution	33
15	Tied Distrix plan	36
16	Tied plans data	37
A.1	W kernels	48
A.2	V kernels	49
A.3	T_{19} kernels	50
B.1	Mapmaker	52
B.2	Free polyhexes	55

CHAPTER 1

Introduction

1.1 Political redistricting and gerrymandering

In majority vote representative democracies, gerrymandering[1] refers to the practice of redrawing electoral district boundaries in a way that gives a political party an unfair advantage over its opponents in elections. The issue is most often discussed in the context of the U.S. House of Representatives election, where state voters elect representatives from their state's congressional districts to be sent to the lower chamber of Congress. Following the U.S. Census, which takes place every 10 years, states are mandated to redraw the borders of their districts to account for population migration and to ensure equal representation at the federal level. It is now widely recognized that in a "winner-takes-all" voting system, whoever controls the redistricting process has the power to give one party or group an advantage in future elections.

1.2 Packing and cracking

The two primary strategies in gerrymandering are "packing" and "cracking". They are frequently employed together.

Packing concentrates as many of the opponent's voters as possible into as few districts as possible. In a majority vote electoral system, a candidate (or a party) needs to secure a simple majority to win a district. This means that in two candidate or party elections the minimum share of the total vote that ensures a win is 51%. Any votes above this threshold are "wasted" as they don't contribute to additional victories. Yet these votes can be decisive in other districts

[1]The term comes from a combination of the last name of Elbridge Gerry, a 19th century American politician, and "salamander", which was used to describe the shape of a notoriously redrawn electoral district in Massachusetts in 1812, under Gerry's governorship.

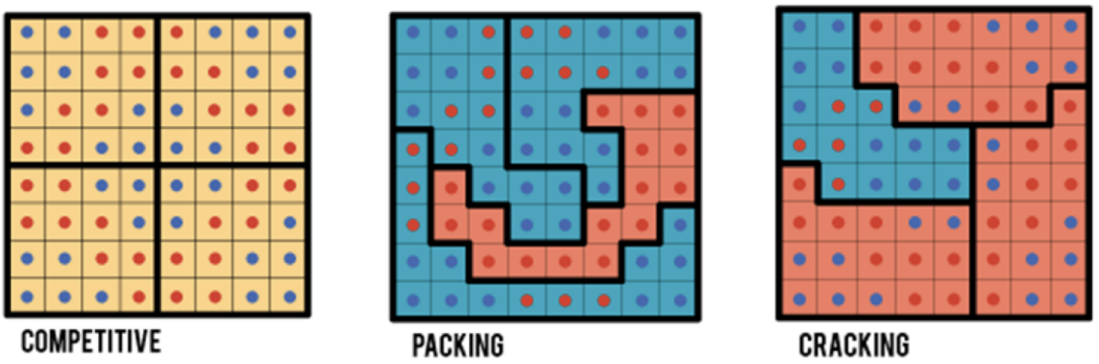
where the races are tighter. Thus, by concentrating the opponent's voters in a few districts won by a large margin (e.g. 70%, 80%), the gerrymandering party gains an advantage. As a result it can secure more seats than would be proportional to its overall vote share across all districts.

In contrast cracking spreads the voters of the opponent party across multiple districts, in a way that prevents them from forming a majority and securing a victory. By strategically dispersing these voters among several districts, the gerrymandering party dilutes the voters' voting power and makes it challenging for them to win seats. The tactic ensures that even if the opposition has significant support, it is not concentrated enough to translate into electoral wins. As with packing, this allows the gerrymandering party to achieve disproportionately greater representation.

The sequence of images in Figure 1 provides a visual representation of the gerrymandering strategies of packing and cracking, illustrating how district boundaries are manipulated to influence electoral outcomes[1]. Figure 1a shows a competitive map representing a state with two parties where 50% of voters favor the Red party, and the other 50% favor the Blue party. Figure 1b illustrates how, on the same map, Blue can win three districts by packing the Red voters into one district where Red wins 100% of the votes. Figure 1c demonstrates how, again on the same map, Red can prevent Blue from winning more than one district by diluting Blue's votes through cracking.

1.3 The Distrix board game

This research is an opportunity to explore mathematically the relationship between board games and the real-world problem of redistricting. The board game can be considered a small-scale, laboratory version of the problem, where experi-



(a) A map representing a state with two equally popular parties.

(b) Blue wins 3 districts by packing Red into 1 district.

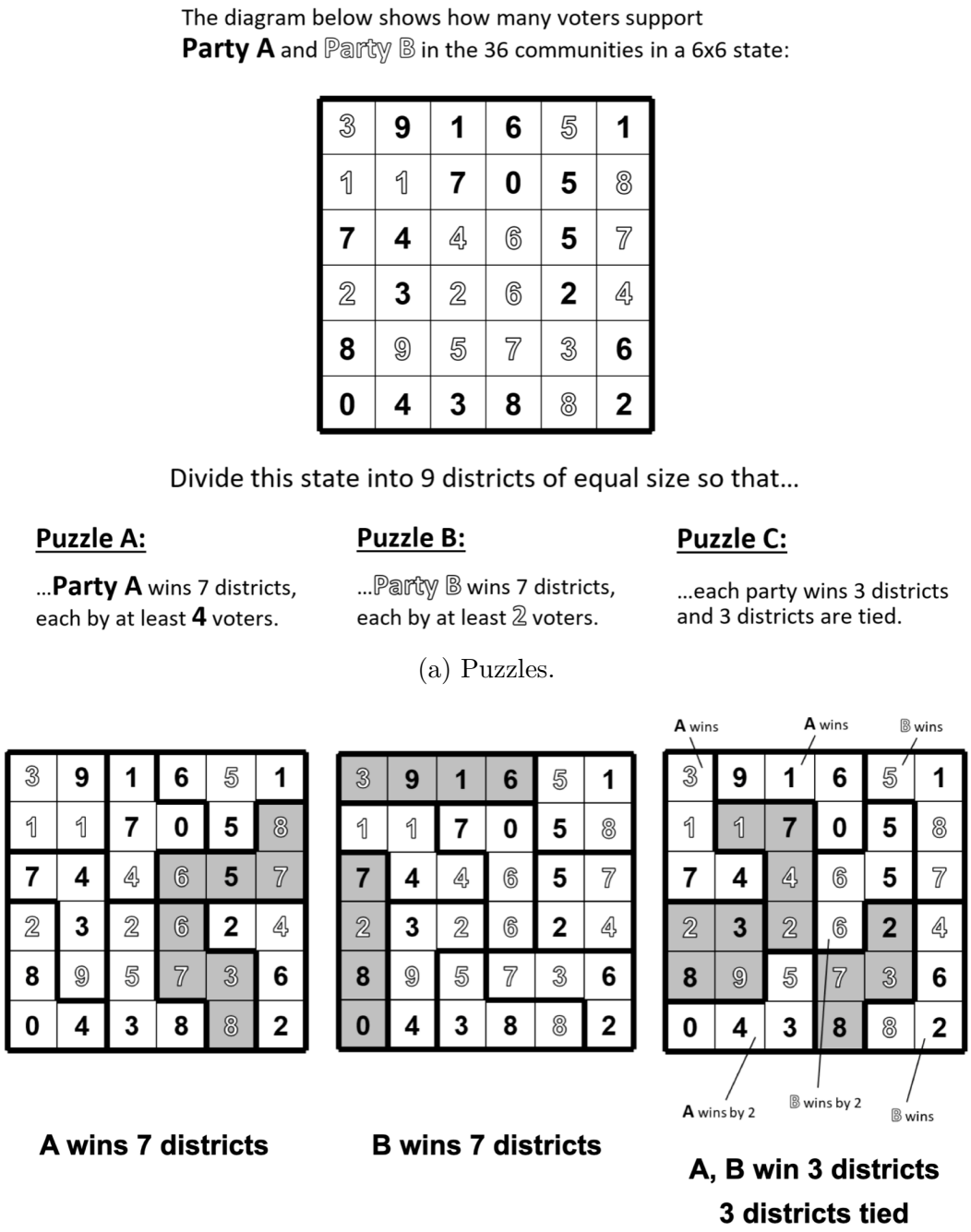
(c) Red wins 3 districts by employing cracking to dilute Blue's vote.

Figure 1: A demonstration how packing and cracking strategies can skew a balanced board to favor one party[1].

ments can be conducted and hypotheses tested. The game used in this research is the solitaire, puzzle-book version of Distrix developed by Matt Petering[2].

The Distrix puzzle-book board is a 6x6 matrix, Figure 2. Each cell in the matrix represents a voting unit, i.e. a voting precinct. The value in each cell represents the vote advantage that the given party holds in the unit based on historic data. For example, a white 3 means that historically this precinct has been dominated by party B by a margin of 3 votes. For the rest of this work the vote advantage value will be referred to as simply "vote(s)". The sum of all votes for one party is 81, with the values 1-8 each occurring twice per party, and 0 and 9 each occurring once per party.

Four orthogonally connected voting units create a district. A district can take the shape of one of the 19 fixed tetrominoes. A tetromino is a type of polyomino composed of 4 squares. There are 5 "free" tetrominoes most of which can undergo rotations, reflections or both. These transformations result in a total of 19 distinct "fixed" tetrominoes, Figures 3a and 3b. There are $\frac{6\times6}{4} = 9$ districts on the board. A player is required to apply the gerrymandering strategies of packing and cracking

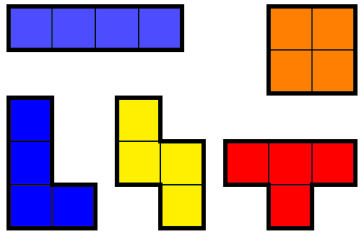


(a) Puzzles.

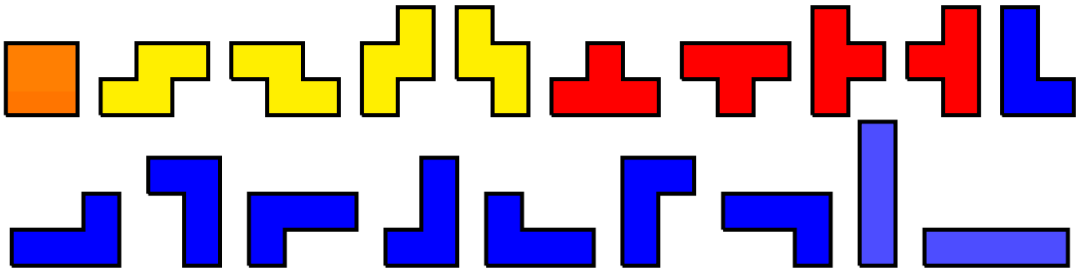
(b) Solutions.

Figure 2: An example puzzle from the Distrix puzzle-book manual[3].

to give a winning advantage to a specific party, or to create a district plan which causes the game to end in a tie.



(a) The 5 free tetrominoes[4].



(b) The 19 fixed tetrominoes[5].

Figure 3: Tetrominoes.

No research publications on Distrix can be found as of this writing. However, mathematical techniques that can be applied to rectangular grid puzzles have been studied in depth in the past. In [6] we find general information on polyominoes and their properties, information on tiling rectangular grids, and problems and solutions that involve polyforms. [7] gives us insight on how combinatorial puzzles can be created by using the regular polygons as building blocks. Papers like [8] offer novel approaches to the tiling problem.

CHAPTER 2

Measuring the Susceptibility to Packing

In this chapter we propose a measure for quantifying the susceptibility of a party's vote to the gerrymandering practice of packing.

2.1 Expected value benchmark score

A benchmark score needs to be established for the purpose of evaluating the reliability and accuracy of the proposed packability score. This benchmark should be based on the long-term average number of seats that each party is expected to win on a randomly configured board. The statistical measure of an *expected value* provides this average for each party[9]. From these averages, a score is calculated to indicate the overall bias of a board towards one party over another. We term this score the Expected Value score (EV score).

The EV score is adopted as a benchmark throughout this work. It is derived from an exhaustive search of all possible end-game scenarios (district plans), which is feasible on a 6x6 board but becomes computationally intractable as the board size increases. For larger boards and real-world applications, a sample that approximates the set of all possible district plans can be generated using methods such as Markov Chain Monte Carlo algorithms. A family of such algorithms specifically designed for the redistricting problem is described in [10]. However, because the benchmark score is intended to be used as a definitive measure for evaluating our proposed packability approximation score, it is more appropriate to derive it from an exhaustive search rather than another approximation.

2.1.1 Recursive backtracking tiling algorithm

To calculate the expected value of seats won for each party, the set of all end-game scenarios for Distrix needs to be discovered first. This is done by generating all possible 178,939 tilings for the 6x6 Distrix board[5]. A recursive, backtracking algorithm was designed for this purpose.

The list of tetromino placements

The input for the backtracking algorithm is the list of all possible positions of the 19 fixed tetrominoes on the board that needs to be tiled. We call the combination of a tetromino and its position on the board a *tetromino placement*. It is denoted by $TP^{(x,y)}$ where TP is a fixed tetromino per Figure 3b (matrix implementations are provided in A.3), and $^{(x,y)}$ are the row and column coordinates of its upper left corner on the Distrix board. Tetromino placements which cross any of the edges of the board are not allowed. An example of a tetromino placement for the S tetromino placed in the upper left corner of the Distrix board $(0, 0)$ is $S^{(0,0)}$, Figure 4a. To generate a list of all tetromino placements for the 6x6 board, the board is scanned with each one of the 19 fixed tetrominoes in row-major order, as illustrated with the S-tetromino on Figure 4.

As each tetromino slides, its coordinates at each position on the board are recorded. The pseudo-code for the scan is provided in Algorithm 1, where:

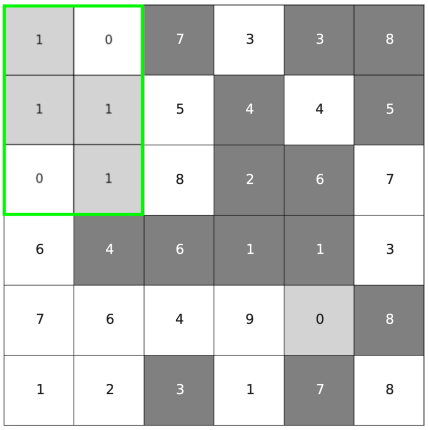
$\boldsymbol{T_{19}} \rightarrow$ a list of the 19 fixed tetrominoes $\boldsymbol{T_f}$.

$\boldsymbol{B_{p\times q}} \rightarrow$ a board of p-rows and q-columns.

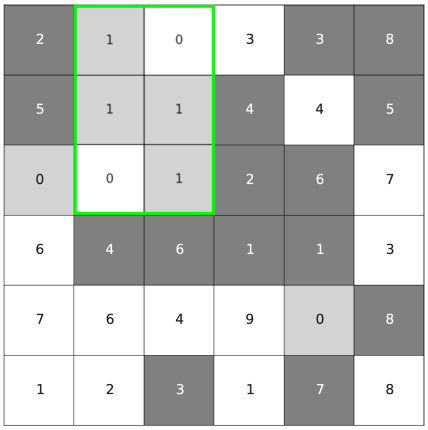
$\boldsymbol{TP} \rightarrow$ a tetromino placement.

$\boldsymbol{L_{TP}} \rightarrow$ a storage list for all tetromino placements.

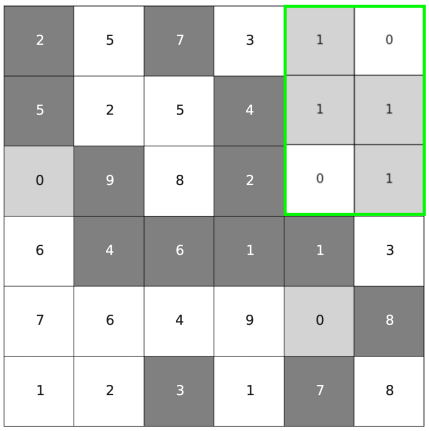
$\boldsymbol{Rows()} \rightarrow$ a sub-procedure for determining the rows of board / tetromino.



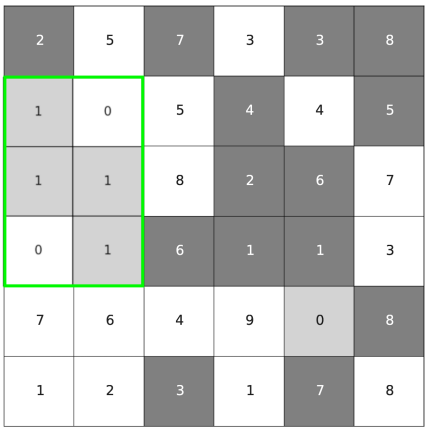
(a) The $S^{(0,0)}$ tetromino placement.



(b) The $S^{(0,1)}$ tetromino placement.



(c) The $S^{(0,4)}$ tetromino placement.



(d) The $S^{(1,0)}$ tetromino placement.

Figure 4: Generating the list of all tetromino placements by sliding each of the 19 fixed tetrominoes across the board. An example is shown with the S-tetromino (positions (0,3) and (0,4) are omitted for brevity). This process is repeated for the remaining 18 fixed tetrominoes.

Cols() → a sub-procedure for determining the cols of board / tetromino.

RecordTP() → a sub-procedure for creating a TP object.

Algorithm 1 Scanning the board

```
procedure ScanBoard(T_19, B_{p×q})
    A procedure for scanning the board with tetrominoes.
    Input: T_19 – A list of the 19 fixed tetrominoes.
    Input: B_{p×q} – A board of p-rows and q-cols.
    Output: L_TP – A storage list for all tetromino placements.
    for all T_f ∈ T_19 do
        for row ← 0, Rows(B_{p×q}) − Rows(T_f) do
            for col ← 0, Cols(B_{p×q}) − Cols(T_f) do
                TP ← RecordTP(row, col, T_f)        ▷ Create the TP object.
                L_TP.append(TP)
            end for
        end for
    end for
    Return L_TP
end procedure
```

Table 1 shows the number of tetromino placements for each tetromino on the board. The total number of tetromino placements for a 6x6 board is 381.

(a) Part 1

Tetromino	I	I_{90}	J	J_{90}	J_{180}	J_{270}	L	L_{90}	L_{180}	L_{270}
# Placements	18	18	20	20	20	20	20	20	20	20

(b) Part 2

Tetromino	O	S	S_{90}	T	T_{90}	T_{180}	T_{270}	Z	Z_{90}	Total
# Placements	25	20	20	20	20	20	20	20	20	381

Table 1: The number of tetromino placements for each tetromino on the board.

It may seem that continuously pulling 9 random tetromino placements from the list of 381 and checking if they result in a tiling is an acceptable way of tiling the 6x6 board. This, however, is computationally intractable as the number of combinations without repetition that need to be checked in this case is $\binom{381}{9} = 4.23861 \times 10^{17}$. Therefore another approach is needed.

The recursive backtracking procedure

This is the recursive backtracking part of the algorithm. To tile the board an exhaustive depth-first search is performed through all possible combinations of tetromino placements, without repetition. The pseudo-code for the tiling procedure is provided in Algorithm 2. In addition to the symbols for the $ScanBoard()$ procedure, the following symbols are defined:

$\boldsymbol{L_{TL}} \rightarrow$ a storage list for all board tilings.

$\boldsymbol{BoardTiled()} \rightarrow$ a sub-procedure for checking whether a board is tiled.

$\boldsymbol{StoreBoard()} \rightarrow$ a sub-procedure for storing a board to L_{TL}.

$\boldsymbol{OverlapCheck()} \rightarrow$ a sub-procedure for checking whether TP overlaps another TP on the board.

$\boldsymbol{PlaceTP()} \rightarrow$ a sub-procedure for placing a TP on the board.

$\boldsymbol{Reduce()} \rightarrow$ a sub-procedure for reducing L_{TP} by removing TPs that have been skipped due to overlap, or have already been placed on the board.

$\boldsymbol{RemoveTP()} \rightarrow$ a sub-procedure for removing a TP from the board.

Details on the main components of Algorithm 2 are provided below:

1. Base case.

   For its base case, the algorithm checks whether the board is already tiled. If so, the combination of tetromino placements that has made up the tiling is recorded to L_{TL} and the algorithm proceeds with 2. If the board is not tiled, the algorithm proceeds directly with 2.

Algorithm 2 Tiling the board

An updated procedure for tiling the board.

procedure TileBoard($L_{TP}, B_{p\times q}$)
  Input: L_{TP} – A list of all tetromino placements.
  Input: $B_{p\times q}$ – A board to tile.
  Output: L_{TL} – Storage for all tilings.
  for all $TP \in L_{TP}$ **do**
    if $BoardTiled(B_{p\times q})$ **then** ▷ Base case.
      $StoreBoard(B_{p\times q}, L_{TL})$
      Return
    else if $OverlapCheck(TP, B_{p\times q}) = False$ **then**
      $PlaceTP(TP, B_{p\times q})$
      $RL_{TP} \leftarrow Reduce(L_{TP})$
      $TileBoard(RL_{TP}, B_{p\times q})$ ▷ Recursive call.
      $RemoveTP(TP, B_{p\times q})$ ▷ Backtrack step.
    end if
  end for
  Return L_{TL}
end procedure

2. Iterate over L_{TP}.

   For each TP:

   (a) call the $OverlapCheck()$ procedure to check whether placing the current TP on the board would partially overlap any TP already on the board.

   (b) If $OverlapCheck() = True$ the algorithm backtracks and continues the iteration over L_{TP}.

   (c) If $OverlapCheck() = False$, $PlaceTP()$ is called and the TP is placed on the board. At this point all TPs that have been skipped due to overlap, and all TPs that have been placed on the board are removed from L_{TP}, resulting in a reduced list RL_{TP}. A recursive call of $TileBoard()$ is made on the reduced RL_{TP}.

3. Backtracking.

   The algorithm backtracks when:

   (a) a TP cannot be placed on the board due to overlap, and a new TP is picked from L_{TP}.

   (b) a board is tiled (the procedure enters its base case) after which a TP is removed from the board by a call to $RemoveTP()$.

   (c) the end of L_{TP} is reached during iteration, after which $RemoveTP()$ is called.

2.1.2 Algorithm complexity

The backtracking step triggered by an overlap or a completion of a successful tiling, depends on runtime checks and cannot be predicted in advance. This makes assessing the complexity of the algorithm challenging. A loose upper bound on the time complexity can be defined based on a simplification: we assume that *OverlapCheck()* and *BoardTiled()* always return *False*. Under this assumption the algorithm never backtracks early[1], which eliminates any form of pruning that could reduce the number of explored search paths. As a result, all possible subsets of TPs from the set L_{TP} are generated. This is the power set of L_{TP}, denoted by $\mathcal{P}(L_{TP})$[2]. The number of subsets in $\mathcal{P}(L_{TP})$, or its cardinality, is 2^k[11], where k is the cardinality of the original set L_{TP}. This can be summarized as:

$$k = |L_{TP}| \tag{1}$$

$$|\mathcal{P}(L_{TP})| = 2^k \tag{2}$$

Generating $\mathcal{P}(L_{TP})$ involves a binary decision, where a choice is made whether to include a TP in a subset or not include it. Since this simplified model of Algo-

[1]The algorithm still backtracks when it reaches the end of an RL_{TP} at each recursive level.
[2]The power set of a set S is the set of all subsets of S, including the empty set and S itself.

rithm 2 has no pruning mechanism, a new subset from the power set is generated each time a tetromino is placed on the board. This occurs at every iteration step, at each recursive level of the algorithm. Consequently, the asymptotic upper bound on the time complexity of Algorithm 2 is determined by Eq. 2, where k is the number of tetromino placements for a given board size

$$\mathcal{O}(2^k) \tag{3}$$

It is crucial to note that the presence of the *OverlapCheck()* and *BoardTiled()* checks (i.e. if the assumption described in this section is removed) improves the performance of the algorithm significantly, compared to the worst case scenario given in Eq. 3. This makes the further tightening of the upper bound on the time complexity of Algorithm 2 an interesting problem to investigate. However, it is beyond the scope of this work.

2.1.3 Speeding up the tiling algorithm

The performance of Algorithm 2 can be improved considerably by implementing an additional trigger for early backtracking. For all regions on the board which are still not covered by tetrominoes, if the total number of tiles nt, in any region, is not divisible by 4 (i.e., $nt \mod 4 \neq 0$), the algorithm will backtrack. This is shown in Figure 5 which depicts a tiling in progress. The tiles covered by a tetromino have a value of 1, and any tiles that remain to be covered have a value of 0. The placement of the green $T^{(4,0)}$ during the tiling process creates the two red regions that can never be tiled with a tetromino. This dead-end path is detected by the algorithm and it backtracks early instead of continuing to try other TPs from the list L_{TP}. The theoretical background of this approach is described in [12].

In our experience, implementing this additional trigger for early backtracking lead to a near 84% reduction in the time the algorithm took to generate the set of

all possible tilings for a 6x6 board. The pseudo-code for the updated algorithm is given in Algorithm 3. The only addition is the $CheckOddRegions()$ sub-procedure:

> ***CheckOddRegions*($B_{p\times q}$)** $\rightarrow$ a base case sub-procedure which returns $True$ if any non-tiled board regions are of size not divisible by 4.

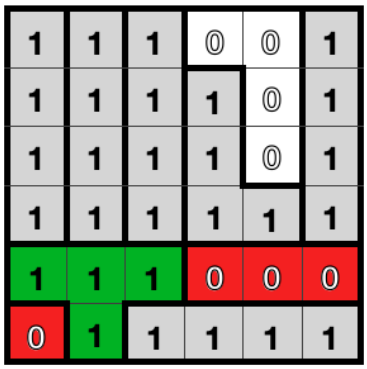


Figure 5: Tiling in progress: the placement of a $T^{(4,0)}$ creates two regions that can never be tiled, shown in red. This causes the updated algorithm to backtrack early.

Algorithm 3 Improving the performance of the backtracking algorithm.

An updated procedure for tiling the board.
procedure TileBoard($L_{TP}, B_{p\times q}$)
  Input: L_{TP} – A list of all tetromino placements.
  Input: $B_{p\times q}$ – A board to tile.
  Output: L_{TL} – Storage for all tilings.
  for all $TP \in L_{TP}$ **do**
    if $CheckOddRegions(B_{p\times q})$ **then** ▷ Base case.
      Return
    else if $BoardTiled(B_{p\times q})$ **then** ▷ Base case.
      $StoreBoard(B_{p\times q}, L_{TL})$
      Return
    else if $OverlapCheck(TP, B_{p\times q}) = False$ **then**
      $PlaceTP(TP, B_{p\times q})$
      $RL_{TP} \leftarrow Reduce(L_{TP})$
      $TileBoard(RL_{TP}, B_{p\times q})$ ▷ Recursive call.
      $RemoveTP(TP, B_{p\times q})$ ▷ Backtrack step.
    end if
  end for
  Return L_{TL}
end procedure

2.1.4 Distributions of seats won

With the set of 178,939 tilings available, frequency distributions of the seats (districts) won by each party can be generated for any given random board. Figure 6 introduces the random board that will be used in examples throughout this study. Figure 7 displays the distributions of seats won for this random board.

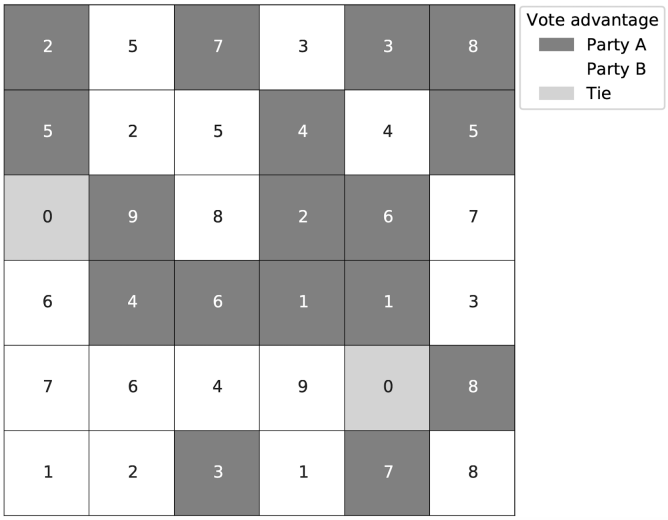


Figure 6: A random Districx board

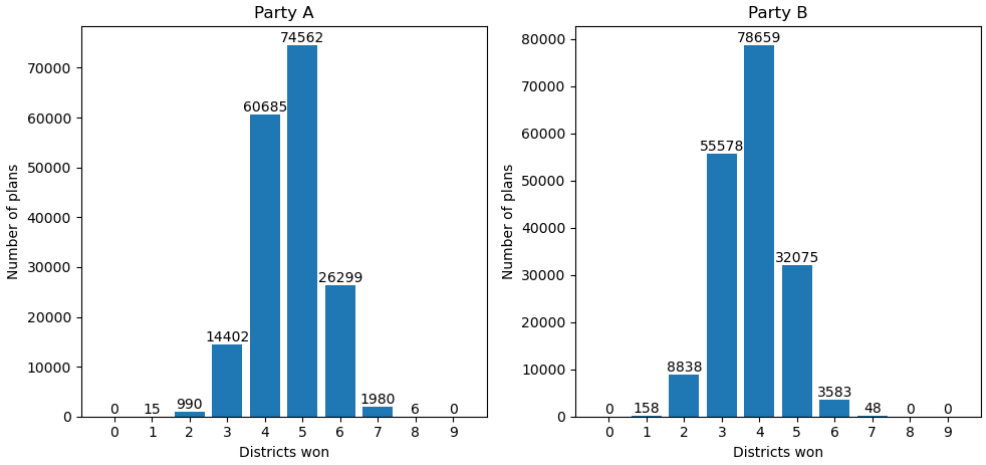


Figure 7: Distributions of the districts won by party A and party B for the board on Figure 6, across all 178,939 district plans. The numbers of plans giving 5,6,7, or 8 seats to Party A are significantly larger then those for Party B. This shows that the vote distribution from Figure 6 clearly favors party A.

The tallest bar for party A shows that in 74,562 out of the total of 178,939 plans, party A wins 5 seats. In comparison, party B wins 5 seats in only 32,075 plans. Furthermore, a significantly larger number of plans give party A 6,7, or 8 seats when compared to party B. This is mirrored by a proportionately greater number of plans giving party B only 1,2,3 or 4 seats as compared to party A[3]. This clearly indicates that the vote distribution from Figure 6 favors party A, making it a lot easier for A to win the game/election by at least 5 out of 9 districts.

2.1.5 Expected Value score calculation

The benchmark EV score is calculated as follows. Let EV_A and EV_B be the expected number of seats won by party A and party B, respectively. EV is the overall expected value benchmark score for the board. d is the categorical variable for the number of seats won. For the distributions on Figure 7:

$$\begin{aligned} EV_A &= \sum_{i=0}^{9} d_i P(d_i) \\ &= 0\frac{0}{178939} + 1\frac{15}{178939} + 2\frac{990}{178939} + 3\frac{14402}{178939} + 4\frac{60685}{178939} \\ &+ 5\frac{74562}{178939} + 6\frac{26299}{178939} + 7\frac{1980}{178939} + 8\frac{6}{178939} + 9\frac{0}{178939} \\ &= 4.65 \end{aligned} \tag{4}$$

$$\begin{aligned} EV_B &= \sum_{i=0}^{9} d_i P(d_i) \\ &= 0\frac{0}{178939} + 1\frac{158}{178939} + 2\frac{8838}{178939} + 3\frac{55578}{178939} + 4\frac{78659}{178939} \\ &+ 5\frac{32075}{178939} + 6\frac{3583}{178939} + 7\frac{48}{178939} + 8\frac{0}{178939} + 9\frac{0}{178939} \\ &= 3.81 \end{aligned} \tag{5}$$

[3]No plans give any of the two parties either 0 or 9 seats.

$$\begin{aligned} EV &= EV_B - EV_A \\ &= 3.81 - 4.65 \\ &= -0.84 \end{aligned} \tag{6}$$

A negative EV score indicates that party A has an advantage in terms of the average number of seats won across all 178,939 plans. Conversely, a positive EV score suggests an advantage for party B.

2.2 Packability score design

2.2.1 Sliding kernel algorithm

To identify opportunities to pack an opponent, a human player typically inspects the board visually for areas of high concentration of opponent votes where packing can be attempted. Following the same approach, we scan the board in search of clusters of votes for a given party by sliding a small window matrix - a kernel - across the board. The kernel has p rows and q columns, and weights w_{ij} as its elements. The scan starts at the upper left corner of the board and proceeds left to right, top to bottom. It is similar to how kernel convolution with a stride of 1 is performed in image processing, with a few differences. First, instead of the vector dot product, the Frobenius inner product is taken[4][13]. Second, no kernel flipping is involved. Finally, the kernel never extends beyond the edges of the board - instead once its right side reaches the edge, it moves down to the next row. Figure 8 shows an example kernel. The scan process using this kernel on our example board is illustrated in the sequence on Figure 9.

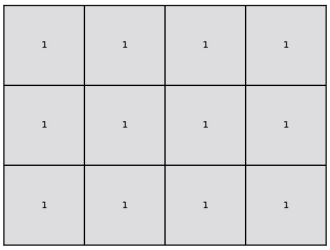
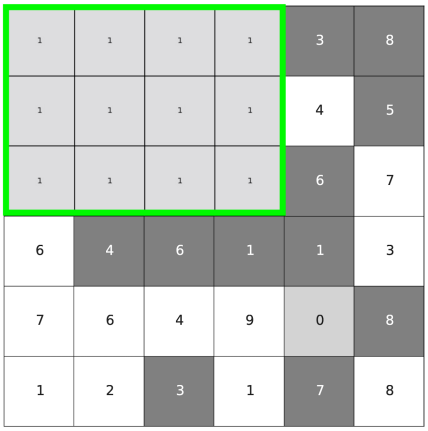


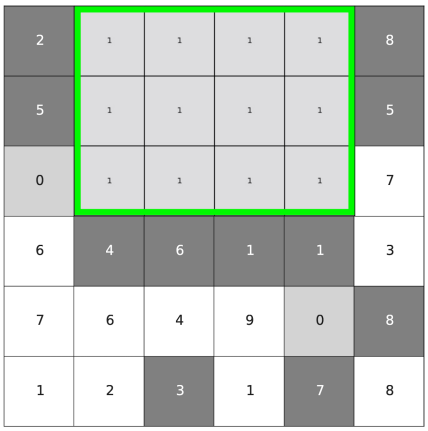
Figure 8: An example 3x4 kernel.

At each position of the sliding kernel, a local score is calculated and stored to a list. This is done as follows. Let W be the sliding kernel and B the board. B is split into two complementary sparse matrices B_A and B_B such that B_A retains only the votes for party A with all votes for party B set to 0. Correspondingly,

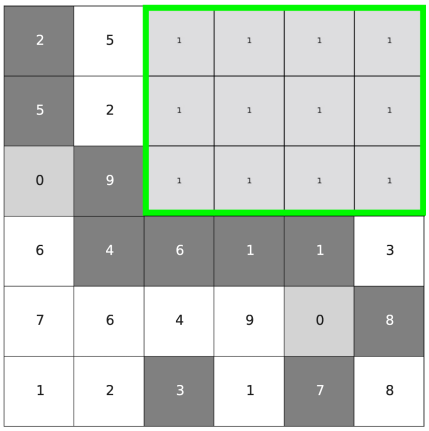
[4]The Frobenius inner product between two matrices of the same dimensions is the element-wise product of the elements of the two matrices followed by a sum of this product. It is a generalization of the vector dot product to matrices.



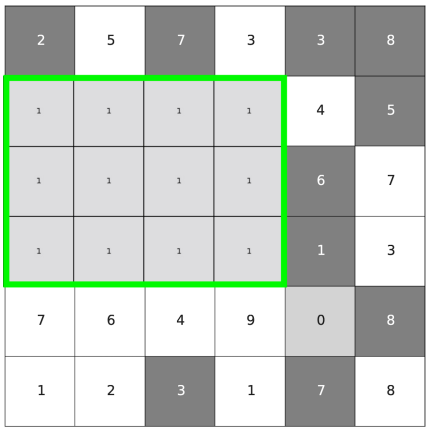
(a) sliding kernel at coordinates (0,0)



(b) sliding kernel at coordinates (0,1)



(c) sliding kernel at coordinates (0,2)



(d) sliding kernel at coordinates (1,0)

Figure 9: The first four positions of a 3x4 sliding kernel. The kernel never goes over-board during sliding.

B_B only retains the votes for party B with all votes for party A set to 0. For each position i, j of the sliding kernel, the Frobenius inner product between W and the sub-matrices $B_A[i : i + p - 1, j : j + q - 1]$ and $B_B[i : i + p - 1, j : j + q - 1]$ is calculated. The results are two non-negative integers s_{Aij} (Eq. 7) and s_{Bij} (Eq. 8), which reflect the total votes for each party at the current W_{ij}. s_{Aij} is then subtracted from s_{Bij} to come to a single number s_{ij} for the local vote concentration score at the current position of the kernel (Eq. 9).

$$s_{Aij} = \langle B_A[i : i + p - 1, j : j + q - 1], W \rangle_F \tag{7}$$

$$s_{Bij} = \langle B_B[i : i + p - 1, j : j + q - 1], W \rangle_F \tag{8}$$

$$s_{ij} = s_{Bij} - s_{Aij} \tag{9}$$

The sign of s_{ij} indicates which party has a vote advantage at position $_{ij}$. A negative s_{ij} indicates an advantage for party A, while a positive sign indicates a vote advantage for party B. As W slides across B, the local scores are stored to a list L_s. An example of this list for the scan from Figure 9 is shown in Eq. 10. The length of the list L_s varies with the size and shape of the kernel in use, which determine the number of valid positions for the sliding kernel on the board.

$$L_s = [-6, -4, -8, -10, -14, 2, 18, -2, 7, 22, 0, -1] \tag{10}$$

At the next step of the scoring procedure, all negative integers from L_s are stored in sub-list L_{s_A} which holds all local scores for party A. All positive integers are stored in sub-list L_{s_B}.

$$L_{s_A} = [-6, -4, -8, -10, -14, -2, -1]$$

$$L_{s_B} = [2, 18, 7, 22]$$

The elements of L_{s_A} are then summed to produce an integer P_{PA} which is the party A component of the overall packability score for the board. The sum of the

elements in L_{s_B} is the party B component of the board packability score.

$$P_{PA} = \sum L_{s_A} = -45,$$
$$P_{PB} = \sum L_{s_B} = 49.$$

Finally, P_{PA} and P_{PB} are added together to arrive at a single signed integer P_B, which is the overall packability score for the board.

$$\begin{aligned} P_B &= P_{PA} + P_{PB} \\ &= -45 + 49 = \\ &= 4 \end{aligned}$$

A high packability score indicates that the party is more vulnerable to packing as opposed to its opponent, due to the distribution of its votes on the board. A high negative P_B indicates a board bias against party A, while a high positive P_B indicates a bias against party B. A threshold for when a score is considered "high" is provided in Section 3.2.

The upper bound on the time complexity of the sliding kernel algorithm is

$$\mathcal{O}(k)$$

where k represents the number of valid positions for the kernel as it slides across the board. For a 3x4 kernel on a 6x6 board, $k = 12$. Thus, in contrast to the EV score, which requires an exhaustive search accomplished by an $\mathcal{O}(2^k)$ algorithm, the P_B score is an approximation based on a computationally cost-effective $\mathcal{O}(k)$ algorithm. This efficiency allows P_B scoring to be applied effectively to boards larger than a 6x6.

2.2.2 Parameterizing the algorithm

The parameters described in this section influence the packability score produced by the sliding kernel algorithm. In Chapter 3 we conduct an experimental investigation under various configurations of these parameters.

Kernel shapes and weights

Our score reliability experiment is conducted with 10 kernels of different shapes and weights. They fall under three categories:

a. **Rectangular kernels W, with all weights set to 1.**

These are the simplest sliding kernels. An example 3x4 kernel was already shown in Figure 8 and is provided here for reference. We denote a rectangular kernel with p rows and q columns by $\boldsymbol{W_{p\times q}}$.

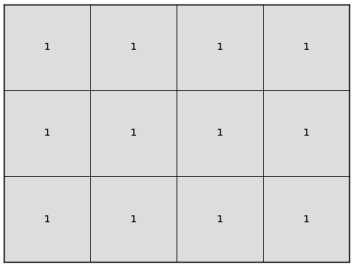


Figure 10: A $W_{3\times 4}$ kernel with all weights set to 1

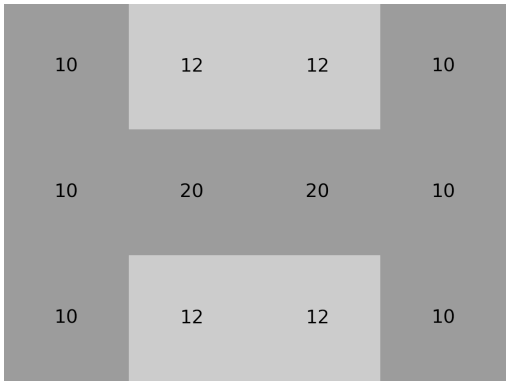
b. **Rectangular kernels V, with variable weights.**

The weights in these kernels are based on the number of unique tetromino placements that cover each tile across the entire set of possible tetromino tilings for the kernel matrix. For example, from [14] we know that there are 23 unique ways to tile a 3x4 kernel. By scanning the 3x4 kernel with each of the 19 fixed tetrominoes, we find that there is a total of 65 tetromino

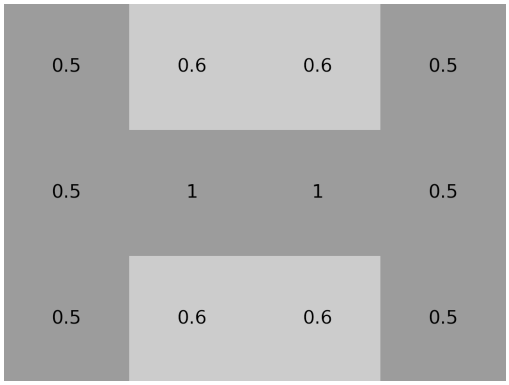
placements for this kernel size. We then record the number of unique tetromino placements that cover each tile in the 3x4 kernel, across all 23 tilings. We call the result the **tetromino saturation score** t_{ij} of a tile. Figure 11a shows the tetromino saturation scores for each tile of a 3x4 kernel. The weight value w_{ij} of each tile is then calculated in Eq. 11 where $\max t$ is the maximum saturation score value anywhere on the board. The $\max t$ value in Figure 11a is 20. Figure 11b shows the resulting weights for each tile in the 3x4 kernel.

$$w_{ij} = \frac{t_{ij}}{\max t} \tag{11}$$

The purpose of designing a kernel with variable weights, is to find out whether a higher likelihood of a unique tetromino covering a tile results in a more accurate packability score. We denote a rectangular kernel with p rows and q columns by $\boldsymbol{V_{p\times q}}$.



(a) The tetromino saturation score for each tile in a $V_{3\times4}$ kernel.



(b) The calculated weights for each tile in a $V_{3\times4}$ kernel.

Figure 11: Calculating the variable weights for a $V_{3\times4}$ kernel.

c. **The set T, of multiple tetromino kernels.**

Scoring the board with all of the specific shapes that a district can take is expected to produce P_B score results closer to the benchmark EV score,

when compared to scoring with simpler kernel shapes. However, the former requires that we have all possible shapes known in advance. For a 6x6 board where the districts are limited to exactly 4 orthogonally connected tiles, we know that the possible shapes are the 19 fixed tetrominoes. For larger size districts however, this number grows quickly. If the district size is set to 5 tiles, its possible shapes are the 63 fixed pentominoes. For 6-tile districts, the possible shapes are the 216 fixed hexominoes[15]. The problem becomes computationally intractable if we try to enumerate all the possible shapes of real-world districts that may have thousands of voting units (e.g. the state of New York has over 5,000 voting units[16]). Therefore the method of scoring with all possible shapes is not scalable and a using simpler shapes is preferred. Nevertheless, it is important to examine the P_B scores based on the set of 19 fixed tetrominoes. Stronger correlation of such P_B scores with the EV score will be evidence that the sliding kernel algorithm functions as expected.

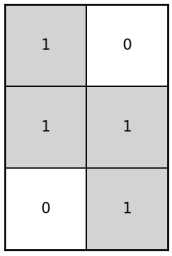


Figure 12: Example S tetromino kernel.

The 19 tetromino kernels used for scoring are the same ones used in the recursive backtracking algorithm from Section 2.1.1. To improve score accuracy, the scanning process with tetrominoes omits kernel positions (TPs) which result in the "cornering" of regions with size less than 4 tiles. A configuration like this does not allow for a board to be tiled. By omitting such kernel

positions, no local scores will be recorded from tetrominoes that will never be part of a tiling. In addition, unlike with rectangular kernels, the scanning is done with the array of all 19 tetrominoes sequentially instead of scoring with a single tetromino. This is because the unique and specific shapes of the tetrominoes, do not allow for a proper generalization of the score. Therefore, in the tetromino kernel experiments we scan with all 19 fixed tetrominoes sequentially and, at the end of the scan, the list L_s consists of 357 local scores (based on 381 total TPs, less 24 TPs which "corner" regions of less than 4 tiles). The set of 19 tetromino kernels is denoted by $\boldsymbol{T_{19}}$.

Maximum number K_s of local scores to keep in L_s

The list L_s from Section 2.2.1 stores all local scores s for every position visited by the kernel. For a 3x4 kernel, the list holds 12 scores at the end of the scan.

$$L_s = [-6, -4, -8, -10, -14, 2, 18, -2, 7, 22, 0, -1]$$

Since packing needs to be concentrated in as few districts as possible (see Section 1.2), only the highest local scores should be kept in L_s to prevent dilution of the final packability score for the board. As shown in Section 2.2.1, and provided here for reference, the final packability score for the board is calculated as:

$$P_{PA} = \sum L_{s_A}$$
$$P_{PB} = \sum L_{s_B}$$
$$P_B = P_{PA} + P_{PB}$$

From the goals of the puzzles in Distrix, we know that on a 9-district board, it is reasonable to attempt to pack an opponent in 1, 2 or at most 3 districts. Therefore, only the top 1, 2, or 3 scores should be kept in the list L_s. The number of highest local scores to keep in L_s is denoted by K_s, so for our 9-district board

$K_s \in \{1, 2, 3\}$. Eq. 12 shows how the list L_s is reduced to include only the number of local scores specified by the K_s value. First the list is sorted in descending order based on the absolute values of its elements. It is then reduced to only include 1, 2 or 3 scores depending on the value of the K_s parameter passed to the algorithm.

$$\begin{aligned} L_s &= sort(L_s) = [22, 18, -14, -10, -8, 7, -6, -4, -2, 2, -1, 0] \\ K_s &= 1 \rightarrow L_s = [22] \\ K_s &= 2 \rightarrow L_s = [22, 18] \\ K_s &= 3 \rightarrow L_s = [22, 18, -14] \end{aligned} \tag{12}$$

In the experiment conducted in Chapter 3 we perform calculations using all three K_s values $K_s \in \{1, 2, 3\}$, to demonstrate a complete methodology for selecting algorithm parameters that produce the most reliable scores. However, for our gerrymandering hypothesis testing in Chapter 4, we exclusively use scores generated with

$$K_s = 1$$

The reason for this, is a current limitation of the sliding kernel algorithm, which requires an additional mechanism in order to generate satisfactory scores at $K_s = 2$ or $K_s = 3$. Such mechanism would ensure that only local scores covering distinctly different regions of the board are added to the list L_s. For instance, if a $W_{2\times4}$ kernel covers several neighboring tiles with a high vote concentration for one party and then slides to its next position, it will likely still be covering the same high concentration cluster. This could lead to the same cluster being counted multiple times in L_s, and at $K_s = 2$ or $K_s = 3$ it might prevent the inclusion of other significant vote concentration areas in the final score calculation. A method for preventing this double counting of clusters is proposed for future work in Chapter 5.

CHAPTER 3

Evaluation of Score Reliability

3.1 Experimental design

In this chapter, we conduct an experimental investigation to determine the combination of algorithm parameters described in Section 2.2.2 that results in the most reliable packability score. A reliable score is one that properly reflects the susceptibility of a party's vote to packing, and therefore can be used in hypotheses testing and statistical inference.

3.1.1 Sampling random boards

For the purpose of testing the sliding kernel algorithm, we create a sample of 1,500 randomly generated boards. The goal is to evaluate the consistency and reliability of the scores that the algorithm produces over many boards. Each board in the sample is scored on its bias towards a party by calculating the board's EV score as detailed in Section 2.1.5. Descriptive statistics for the EV scores for the boards in the random sample are provided in Table 2. The table reveals that on average, each party wins approximately 4.3 seats across all boards. Figure 13 shows the distribution of EV scores for the 1,500 boards. The normal distribution is centered around $EV = 0$ which is evidence that the majority of the boards in the sample are not inherently biased towards either of the two parties.

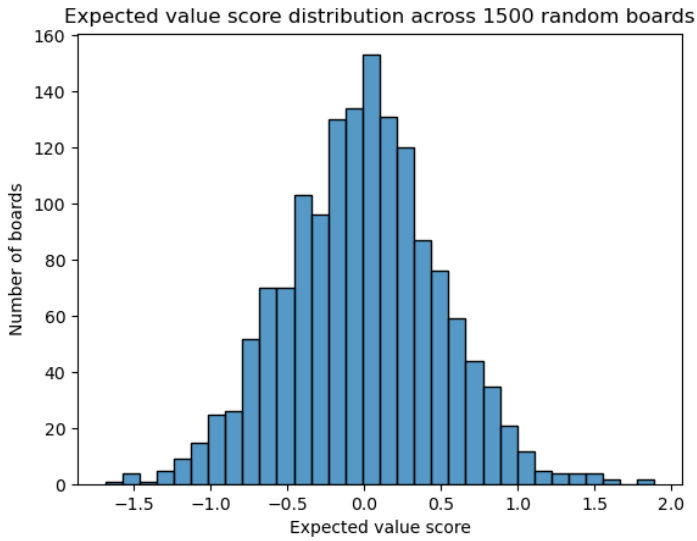


Figure 13: Distribution of the Expected Value (EV) scores across the sample of 1,500 random boards. The negative values on the left-hand side represent the EV scores for Party A. The positive values on the right represent the EV scores for party B. A higher absolute value of the EV score indicates a greater number of seats won by a party on average, compared to their opponent.

	EV A	EV B	EV Score
count	1500	1500	1500
mean	4.33	4.32	-0.01
std	0.26	0.26	0.51
min	3.40	3.41	-1.69
25%	4.16	4.15	-0.35
50%	4.32	4.32	-0.01
75%	4.50	4.48	0.31
max	5.18	5.33	1.89

Table 2: Descriptive statistics of the sample of 1500 random boards. The mean Expected Value score is close to 0 due to both parties winning similar number of seats on average (around 4.3 seats).

3.1.2 Choosing algorithm parameters

In our experiment, the 1,500 boards in the random sample are scored using different combinations of the parameters described in Section 2.2.2. We use the W, V and T kernels from Appendix A and $K_s \in \{1, 2, 3\}$. This results in 30 different scores calculated for each board in the sample. Table 3 shows the score results for the example board from Figure 6.

Kernel	$K_s = 1$	$K_s = 2$	$K_s = 3$
$W_{2\times4}$	27.0	6.0	27.0
$W_{4\times2}$	13.0	3.0	13.0
$W_{3\times4}$	22.0	40.0	26.0
$W_{4\times3}$	14.0	1.0	13.0
$W_{4\times4}$	19.0	3.0	19.0
$V_{3\times4}$	14.9	9.1	14.7
$V_{4\times3}$	8.1	0.2	-6.7
$V_{4\times4}$	8.55	0.55	4.6
$V_{6\times6}$	1.42	1.42	1.42
T_{19}	148.0	9.0	169.0

Table 3: The 30 different packability scores P_B resulting from scanning the example board from Figure 6 with the kernels from A.1, A.2, and A.3 for the 3 different values of the parameter K_s. For reference, the EV score for this board is -0.84.

3.1.3 Evaluating score reliability using correlation

It is expected that a party with a high packability score will win fewer seats on average, compared to its opponent. This is why we consider a negative correlation between P_B and EV an appropriate indicator of P_B's reliability as a measure of packability. As the parameters of the sliding kernel algorithm are varied, we track the correlation between the P_B scores that they produce and the EV score. The score with the strongest and statistically significant negative correlation is considered the most reliable for our hypotheses testing needs.

A Pearson correlation test is conducted to quantify the strength of the relationship, measured by the correlation coefficient, and to determine its statistical

significance, ensuring the observed correlation is not due to random chance.

The Pearson correlation test evaluates the linear relationship between two continuous variables by calculating the Pearson correlation coefficient denoted by r. This coefficient ranges from -1 to 1, indicating perfect negative correlation at -1, no correlation at 0, and perfect positive correlation at 1. To test the statistical significance of r, the hypothesis $H_0 : r = 0$ (no linear correlation), is tested using a t-statistic calculated as $t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$, where n is the number of data points. This t-statistic follows a t-distribution with $n-2$ degrees of freedom. A significant result rejects H_0, indicating a statistically significant correlation between the variables.

The statistical significance of the correlation is indicated by the p-value of the Pearson correlation test. We choose a significance level of $\alpha = 0.05$ with a p-value below this level indicating a statistically significant correlation.

At the end of the parameter evaluation process we select the kernel and the K_s value parameters that produce a P_B score that we consider most reliable for use in hypotheses testing related to gerrymandering.

3.2 Experimental results

3.2.1 Scoring results

The boards in our random sample are scored and each board receives 30 P_B scores. The correlation coefficients of each of the 30 P_B scores and the EV score is computed. The coefficient results are shown in Table 4. Almost all of the values have a negative sign indicating that the scoring algorithm is capturing the expected negative correlation between P_B and EV. As expected, the strongest correlation results come from the T_{19}-based scores. However, the T_{19}-based scores should not be used in gerrymandering hypotheses testing as explained in Section 2.2.2. From the results it is also evident that the use of variable weight kernels V provides

a boost to the reliability of the score when compared to using W kernels. This stands out most when comparing the r values for $W_{3\times4}$ and $V_{3\times4}$.

	$K_s = 1$	$K_s = 2$	$K_s = 3$
$W_{2\times4}$	-0.21	-0.21	-0.17
$W_{4\times2}$	-0.17	-0.14	-0.13
$W_{3\times4}$	-0.10	-0.07	-0.04
$W_{4\times3}$	-0.11	-0.05	-0.02
$W_{4\times4}$	-0.02	-0.03	0.03
$V_{3\times4}$	-0.14	-0.09	-0.07
$V_{4\times3}$	-0.11	-0.07	-0.05
$V_{4\times4}$	-0.05	-0.02	0.0
$V_{6\times6}$	-0.05	-0.05	0.05
T_{19}	-0.33	-0.28	-0.22

Table 4: Correlation coefficient results for the W, V and T_{19} kernels at $K_s \in \{1, 2, 3\}$, for the sample of 1,500 boards. The top 3 results are highlighted in gray.

The results in Table 4 show that kernel $W_{2\times4}$ produces the P_B scores with the second strongest negative correlation to the benchmark EV score. Due to its simple shape, scores derived from this kernel are suitable for practical applications. As noted in Section 2.2.2, correlation coefficients using scores based on $K_s = 2$ and $K_s = 3$ are included here primarily to demonstrate the process of selecting the optimal K_s parameter value. For this reason, the $W_{2\times4}$ kernel and $K_s = 1$ are chosen as the sliding kernel algorithm parameters to produce P_B scores used in hypotheses testing related to gerrymandering. The P_B score based on these parameters is denoted by

$$P_B(W_{2\times4}, K_s = 1)$$

The correlation coefficient between $P_B(W_{2\times4}, K_s = 1)$ and EV across the sample of 1,500 boards is

$$r = -0.21$$

With a p-value below our threshold of $\alpha = 0.05$ this results is statistically significant

$$p = 9 \times 10^{-7}$$

The thresholds for categorizing the strength of the correlation r as "weak", "medium" or "strong" are not strictly defined in statistics and can very depending on the context of the research. General guidelines are offered in Table 5[17].

Weak	$\lvert r \rvert \leq 0.3$
Moderate	$0.3 < \lvert r \rvert \leq 0.6$
Strong	$\lvert r \rvert > 0.6$

Table 5: Ranges of strength for statistical correlation.

Although in Table 5 this correlation is classified as weak, it is noteworthy because it signifies a consistent negative relationship between the approximation P_B score and the benchmark EV score. The finding underscores the relevance of the P_B score in reflecting the patterns captured by the EV score, even in the presence of only a modest correlation. The correlation coefficient results obtained in this experiment can serve as a reference point in future studies aimed at improving the packability scoring algorithm.

Figure 14 shows the distribution of the 1,500 $P_B(W_{2\times4}, K_s = 1)$ scores. The saddle is due to no boards receiving a P_B score of 0 when $K_s = 1$. Descriptive statistics are provided in Table 6.

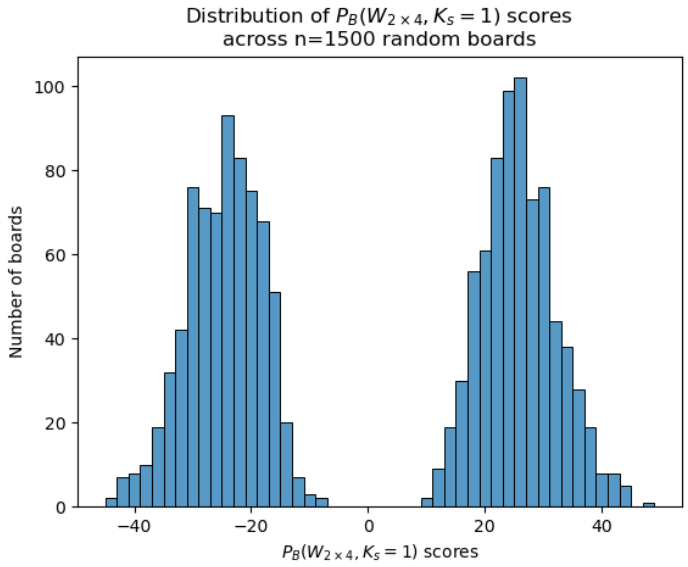


Figure 14: Distribution of the 1,500 $P_B(W_{2\times4}, K_s = 1)$ scores for the boards in the random sample.

	Party A	Party B
count	739	761
mean	-25.23	25.32
std	6.45	6.48
min	-45.00	10.00
25%	-30.00	21.00
50%	-25.00	25.00
75%	-20.00	29.00
max	-9.00	49.00

Table 6: Descriptive statistics of the $P_B(W_{2\times4}, K_s = 1)$ scores for both parties for the boards in the random sample.

3.2.2 Score deficiencies

Apart from the need for continuous improvements to the sliding kernel algorithm, we identify the following potential deficiencies in our scoring methodology that currently limit the $P_B(W_{2\times4}, K_s = 1)$ score's correlation with the benchmark EV score.

1. Sensitivity of the scoring method.

   There is a trade-off between using a simple kernel shape as a parameter, such as the $W_{2\times4}$ kernel, and more complex shapes such as the set of T_{19} kernels. Using the T_{19} kernels results in higher score sensitivity evident from the stronger correlation coefficient results for T_{19} in Table 4 when compared to the results from the $W_{2\times4}$ kernel.

2. Crackability and potential other properties of vote distributions.

   In addition to packability, the EV score used as a benchmark throughout this work inherently accounts for crackability. This means that the EV score has an additional component that the P_B score alone does not account for. In addition, other components of the EV score may exist that reflect unknown properties of a party's vote distribution, which may also not be reflected in the P_B score.

CHAPTER 4

Hypotheses Testing

This chapter presents an example of how the newly developed P_B score can be used to test hypotheses in order to draw insight on the real-world issue of mathematical redistricting.

4.1 Low P_B scores and tied district plans

In Distrix each party gets 50% of the votes on any given board (81 votes per party), i.e. the two parties are equally popular among the voters. We refer to this as a 50:50 board. The district plan that best represents the equal popularity of the two parties is the tied district plan. On a tied plan each party wins an equal number of districts, with the vote in the remaining districts ending up even. This is illustrated on Figure 15.

An argument can be made that for a given random vote distribution in Distrix, a relatively low P_B score, regardless of its party sign $(-/+)$, is associated with an increased likelihood of a tied end-game scenario. The reason to suggest this, is that boards that are less biased are expected to provide more fertile ground for tight electoral races. Thus, we hypothesize that in our model, as the absolute value of the $P_B(W_{2\times 4}, K_s = 1)$ score decreases, the number of tied district plans increases. Since a tied plan is considered a fair plan in Distrix, statistical evidence supporting our hypothesis would indicate that low P_B scores correlate with electoral results that more accurately reflect voter preferences.

There are 5 categories of tied plans that can occur in Distrix. The frequency distributions on Figures 16a, 16b, and 16c provide insights on each of these 5 categories.

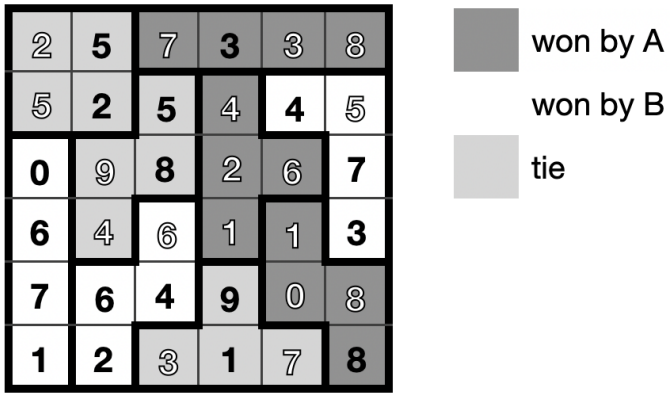


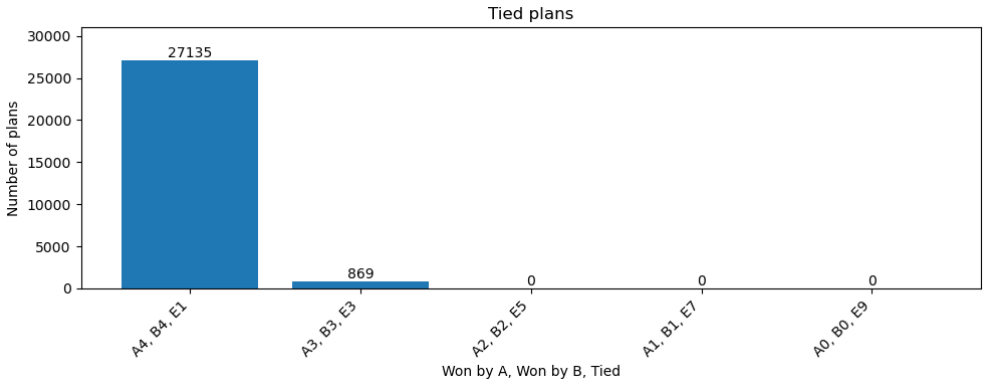
Figure 15: An example of a tied Distrix plan. There are three districts won by party A, three won by party B, and three districts where the vote is tied.

4.2 The TTP_B benchmark score

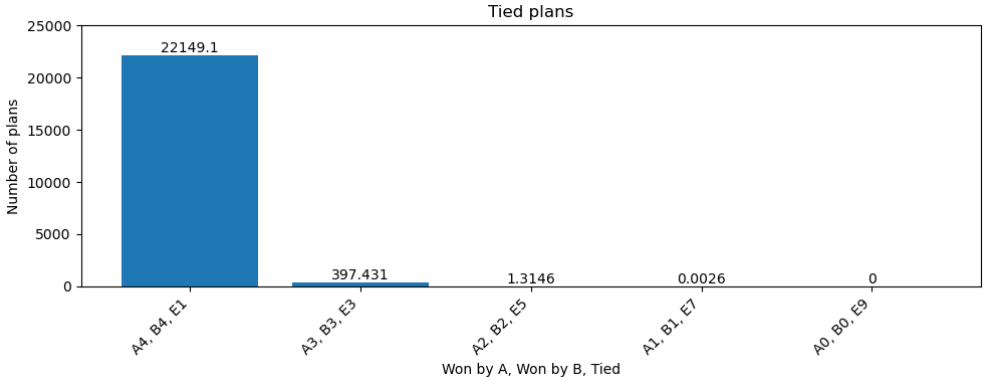
A new benchmark score, termed Total Tied Plans for board B (TTP_B) is created to help formally define our hypothesis. The score is calculated as the sum of all tied plans for a given random board as obtained from the 178,939 tilings. Table 7 shows descriptive statistics, for all TTP_B scores for the sample of 1,500 random boards.

TTP_B Score	
count	1500
mean	22547.82
std	6947.51
min	2944.00
25%	17729.25
50%	22406.00
75%	27196.25
max	43452.00

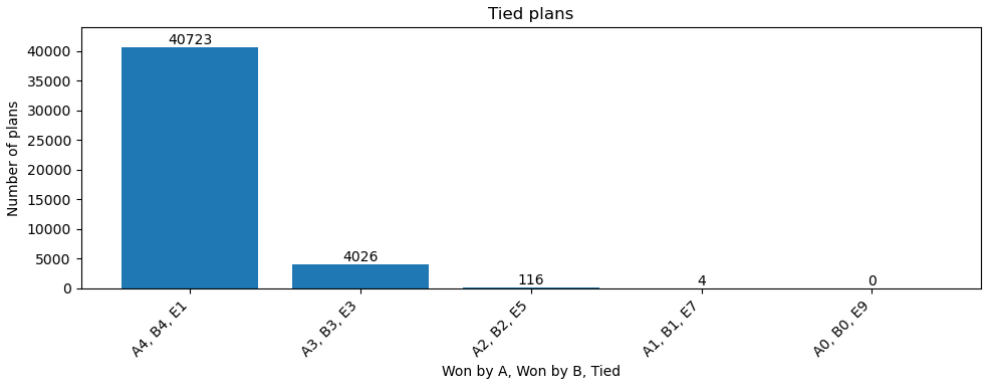
Table 7: Descriptive statistics of the TTP_B score across the sample of 1,500 random boards.



(a) Distribution of the 5 categories of tied plans for the board in Figure 15 over all 178,939 possible plans.



(b) Distribution of the averages of the 5 categories of tied plans for the 1,500 random boards over all 178,939 possible plans.



(c) The maximum number of times each of the 5 tied plans appears on a board in the sample of 1,500 random boards. One board in the sample has 40,723 $A4, B4, E1$ plans, a different board has 4,026 $A3, B3, E3$ plans, etc.

Figure 16: Insight on the likelihood of occurrence of each of the 5 categories of tied plans in Districx.

4.3 Correlation analysis

The Pearson correlation coefficient is used to measure the linear correlation between $P_B(W_{2\times4}, K_s = 1)$ and TTP_B for the 1,500 boards in our random sample. The correlation coefficient result is

$$r = -0.20$$

This indicates a weak but existing negative correlation between the approximation $P_B(W_{2\times4}, K_s = 1)$ score and the exhaustive search TTP_B score.

4.4 The hypothesis test

The following hypothesis test intends to assess the statistical significance of the findings from the correlation analysis presented in Section 4.3. This will provide evidence supporting the hypothesis that as the $P_B(W_{2\times4}, K_s = 1)$ score decreases, the number of tied end-game scenarios increases. The Pearson's correlation hypothesis test is formally defined as follows:

> $\boldsymbol{H_0}$: There is no negative linear correlation ($r \geq 0$) between $|P_B(W_{2\times4}, K_s = 1)|$ and TTP_B.
>
> $\boldsymbol{H_a}$: There is a negative linear correlation ($r < 0$) between $|P_B(W_{2\times4}, K_s = 1)|$ and TTP_B.

The test is conducted at significance level of $\alpha = 0.05$. The test statistic for the significance of the Pearson correlation coefficient r is calculated in Eq. 13.

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}} = \frac{-0.20\sqrt{1500-2}}{\sqrt{1-(-0.20)^2}} = -7.90 \tag{13}$$

where:

- $r = -0.20$ is the sample Pearson correlation coefficient,

- $n = 1500$ is the sample size,
- The test statistic follows a t-distribution with $n - 2$ degrees of freedom.

The resulting p-value calculated for a one-tailed test (since $H_0 \geq 0$) is:

$$p = 2.67 \times 10^{-15}$$

This p-value is less than the chosen significance level of $\alpha = 0.05$, therefore H_0 is rejected and it can be concluded that there is statistically significant evidence to support the claim of an existing negative linear correlation between $|P_B(W_{2\times4}, K_s = 1)|$ and TTP_B.

In relation to real-world redistricting, the hypothesis test results from our small-scale model indicate that a decrease in the packability score is associated with an increase in the number of electoral plans that fairly represent voter preferences.

4.5 Empirical observations

The hypothesis test results are supported by empirical observations, as demonstrated by the sequence in Table 8, where we observe that as the mean $|P_B(W_{2\times4}, K_s = 1)|$ decreases, the mean TTP_B score increases. This observations is across 3 sub-samples q_{30}, q_{20}, and q_{10}, which consist of boards in the 30th, 20th, and 10th $|P_B(W_{2\times4}, K_s = 1)|$ score percentile from the 1,500 board sample.

	$\lvert P_B(W_{2\times4}, K_s = 1)\rvert$	TTP_B
count	1500	1500
mean	25.28	22547.82
std	6.46	6947.51
min	9.00	2944.00
25%	21.00	17729.25
50%	25.00	22406.00
75%	30.00	27196.25
max	49.00	43452.00

(a) The sample of 1,500 boards.

	$\lvert P_B(W_{2\times4}, K_s = 1)\rvert$	TTP_B
count	522.00	522.00
mean	18.52	23617.27
std	2.80	6803.10
min	9.00	6418.00
25%	17.00	19000.25
50%	19.00	23763.00
75%	21.00	28283.50
max	22.00	43452.00

(b) Sub-sample q_{30}: All boards in the 30th percentile of the large sample, based on $\lvert P_B(W_{2\times4}, K_s = 1)\rvert$ score.

	$\lvert P_B(W_{2\times4}, K_s = 1)\rvert$	TTP_B
count	367.00	367.00
mean	17.26	23937.35
std	2.39	6666.54
min	9.00	7204.00
25%	16.00	19144.50
50%	18.00	24108.00
75%	19.00	28434.00
max	20.00	43452.00

(c) Sub-sample q_{20}: All boards in the 20th percentile of the large sample, based on $\lvert P_B(W_{2\times4}, K_s = 1)\rvert$ score.

	$\lvert P_B(W_{2\times4}, K_s = 1)\rvert$	TTP_B
count	169.00	169.00
mean	15.17	24718.69
std	1.88	6673.82
min	9.00	7204.00
25%	14.00	20226.00
50%	16.00	24863.00
75%	17.00	29271.00
max	17.00	43017.00

(d) Sub-sample q_{10}: All boards in the 10th percentile of the large sample, based on $\lvert P_B(W_{2\times4}, K_s = 1)\rvert$ score.

Table 8: Mean $\lvert P_B(W_{2\times4}, K_s = 1)\rvert$ (light gray) and TTP_B (gray) scores for the large sample and quantile-based sub-samples of boards in the 30th, 20th, and 10th $\lvert P_B(W_{2\times4}, K_s = 1)\rvert$ score percentile. As the average $\lvert P_B(W_{2\times4}, K_s = 1)\rvert$ score decreases, the average TTP_B score increases.

CHAPTER 5

Future work

5.1 Logistic regression for detecting biased boards

A binary logistic regression model can be developed to classify a board as being biased against one of the parties or not, using the packability score. To build this model, the set of 1,500 random boards can be split into a training set and a test set. For training, boards with an $|EV| > 0.5$ (1 standard deviation from the mean as shown in Table 2) will be labeled as "biased against one party" (1), and all others as "not biased against one party" (0). The test set can be used to fine-tune the probability threshold for classifying a board as biased, deviating from the typical threshold of 0.5 if needed. For example, if greater certainty is required, the probability threshold could be set at 0.8. Additionally, a logistic regression model can help mitigate outliers, such as when a party with a high P_B score performs unexpectedly well on EV score despite the P_B score indicating a bias.

5.2 A measure for the susceptibility to cracking

In Section 3.2.2, it was stated that the benchmark EV score inherently accounts for cracking in addition to packing, while our P_B score solely covers packing. This prompts an investigation of the "crackability" (C_B) score of a board to complement the P_B score. Such a C_B score should be focused on vote concentrations separate from the ones that have been "marked" as suitable for packing by the P_B scoring algorithm. As in real-world cracking, these will be areas on the board where the opponent can win districts by relatively narrow margins.

One component of the C_B score can take into account the compactness of the opponent's vote clusters. This can be measured by the distance between the votes

within the cluster. If the votes are 1 or 2 tiles apart, they are more likely to end up in different tetrominoes. In contrast, if they are sharing an edge, the cluster will be less "crackable" and the votes are more likely to end up in the same tetromino. For example, in a $W_{2\times4}$ kernel dominated by party A, if the 2 highest vote tiles for party A are in the top two corners of the kernel, the party is likely to win an I_{90}-shaped district in some district plans. However, in most of the 178,939 plans, those two tiles are likely to fall under different districts, which may or may not be won by party A. This approach can also be used to refine our existing P_B scoring methodology, since packing will be less likely to succeed in a $W_{2\times4}$ area where the opponent votes are further apart.

Another potential building block of the C_B score can be the location of the opponent's vote cluster on the board. From the tetromino saturation scores in Section A.2, we see that the center of the 6x6 board can be covered by more tetromino placements than its edges. If the opponent's vote tiles have significantly different weights, chances are those tiles will end up in different districts. This means that clusters closer to the center would be more "stable", while ones closer to the edges of the board would be more "crackable", when observed over all 178,939 plans.

The C_B and P_B scores can be combined into a G_B score reflecting the overall "gerrymanderability" of the board. This combined score is expected to have a stronger correlation with the EV score.

5.3 Sliding kernel algorithm parameters

In Section 2.2.2, we noted the need for an additional mechanism within the sliding kernel algorithm in order to produce reasonable scores when using K_s parameter values of 2 or 3. Such mechanism would ensure that only local scores covering distinctly different regions of the board are included in the list L_s. One

way to achieve this is to introduce an additional algorithm parameter, ST, for kernel "stride". This parameter, representing the step size with which the kernel slides across the board, would take an integer value. Instead of the kernel moving one position forward and taking another local score, as illustrated in Figure 9, it could skip 1, 2, 3, or more tiles depending on the ST value, before it stops to take another local score. This mechanism can be implemented as two parameters, ST_h and ST_v, which modify the stride horizontally and vertically.

CHAPTER 6

Summary

In this thesis we demonstrated how a board game can serve as a tool for conducting research in mathematical redistricting. We believe that using a board game to study the problem provides for an engaging research experience and facilitates the generation of interesting questions to investigate. The idea behind the approach is to conduct small-scale experiments that provide valuable insights applicable to real-world redistricting issues.

In Chapter 2, we developed a method to quantify the potential bias of the board's vote distribution against a player. The method calculates a "packability" (P_B) score which indicates the vulnerability of a player's votes to the gerrymandering tactic of "packing". We also parameterized the method so that its efficacy can be tested over various parameter values. A separate "Expected Value" (EV) score was created to use as a ground truth against which the P_B score is evaluated. The EV score is based on tiling the game's rectangular board with tetrominoes. A special recursive backtracking algorithm was designed to generate all possible tilings via exhaustive search. The time complexity of this algorithm was analyzed, and its upper bound was established at $\mathcal{O}(2^k)$, where k is the number of tetromino placements.

In Chapter 3 we used correlation analysis to evaluate the P_B scores produced at various parameters of our method against the benchmark EV score. We identified the sets of parameters that produced the most consistent and reliable scores, and selected one of these sets for use in hypotheses testing. Additionally, we discussed potential deficiencies of the existing P_B scoring methodology.

In Chapter 4, we demonstrated how the newly developed P_B score can be utilized to test hypotheses related to real-world redistricting. Specifically, we tested a hypothesis to investigate whether a negative correlation exists between the P_B scores and the number of plans that produce a tied election result for a given vote distribution. It was found that there is statistically significant evidence to support this hypothesis, which was further confirmed through empirical observations. Insight related to real-world redistricting was drawn from the hypothesis test results.

In Chapter 5, we proposed directions for extending this research. We suggested developing a logistic regression model based on the P_B score to classify boards as biased or not biased. Additionally, we recommended exploring a method to quantify a party's susceptibility to the gerrymandering practice of "cracking". The resulting "crackability" score can be combined with the P_B score to form an overall "gerrymanderability" score of a board. Finally, we recommended introducing a "stride" parameter in the sliding kernel algorithm. This parameter would aim to prevent duplicate counting of vote concentrations and ensure that only distinct clusters contribute to the P_B score.

LIST OF REFERENCES

[1] MIT Election Lab, "Redistricting," Jan. 2023. [Online]. Available: https://electionlab.mit.edu/research/redistricting

[2] M. Petering, *The Distrix Puzzle Book*. Milwaukee, WI: Distrix Games LLC, 2020.

[3] Distrix Games LLC, "The Distrix Puzzle Book." [Online]. Available: https://www.distrixgames.com/distrix-puzzle-book.html#/

[4] Wikimedia user: Anypodetos, "All 5 free tetrominoes.svg [midified]," 2023, online; Image [original modified by changing colors]; ©Anypodetos; GFDL version 1.2 or later; CC-BY-SA 3.0. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:All_5_free_tetrominoes.svg&oldid=789099523

[5] S. Butler, J. Ekstrand, and S. Osborne, "Tiling Tetris® Boards." [Online]. Available: https://www.stevebutler.org/research/publications

[6] S. W. Golomb, *Polyominoes: puzzles, patterns, problems, and packings*, 2nd ed. Princeton University Press.

[7] S. T. Coffin, *The puzzling world of polyhedral dissections*. Oxford Univ. Pr. [Online]. Available: https://johnrausch.com/PuzzlingWorld/chap02.htm#p7

[8] M. Garvie and J. Burkardt, "A new mathematical model for tiling finite regions of the plane with polyominoes." *Contributions to Discrete Mathematics*, vol. 14, pp. 95–131, 2020.

[9] R. A. Johnson and G. K. Bhattacharyya, *Chapter 5: Probability Distributions*, 6th ed. Hoboken, NJ: John Wiley Sons, 2010, p. 186.

[10] D. DeFord, M. Duchin, and J. Solomon, "Recombination: A family of markov chains for redistricting," *Harvard Data Science Review*, vol. 3, no. 1, Mar. 2021.

[11] K. H. Rosen, *Sets*, 7th ed. New York: McGraw-Hill, 2012, p. 122.

[12] S. W. Golomb, *Backtracking and Impossible Constructions*, 2nd ed. Princeton, N.J: Princeton University Press, 1994, p. 33.

[13] R. A. Horn and C. R. Johnson, *Section 5.2 Examples of norms and inner products*, 2nd ed. Cambridge; New York: Cambridge University Press, 2012, p. 321.

[14] S. Butler, J. Ekstrand, and S. Osborne, "Tetris® Tiling." [Online]. Available: https://oeis.org/A230031

[15] The On-Line Encyclopedia of Integer Sequences, "Number of fixed hexagonal polyominoes with n cells." online. [Online]. Available: https://oeis.org/A001207

[16] U.S. Election Assistance Commission, "Election Administration and Voting Survey 2020 Comprehensive Report, A Report From The U.S. Election Assistance Commission To The 117th Congress." [Online]. Available: https://www.eac.gov/sites/default/files/document_library/files/2020_EAVS_Report_Final_508c.pdf

[17] H. Akoglu, "User's guide to correlation coefficients." *Turkish journal of emergency medicine*, vol. 18, no. 3, p. 91–93, Sept. 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6107969

[18] Lafair Family Games, "The Gerrymandering Manual," 2018, Mapmaker: The Gerrymandering Game. [Online]. Available: https://www.kickstarter.com/projects/1639370584/mapmaker-the-gerrymandering-game

[19] S. W. Golomb, *Polyominoes and Checkerboards*, 2nd ed. Princeton, N.J: Princeton University Press, 1994, p. 4.

[20] The On-Line Encyclopedia of Integer Sequences, "Number of fixed polyominoes with n cells." online. [Online]. Available: https://oeis.org/A001168

[21] Wikimedia Commons, "Tricomb.svg," 2020, [Online; Image; Released into the public domain by the copyright holder.]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Tricomb.svg&oldid=517035876

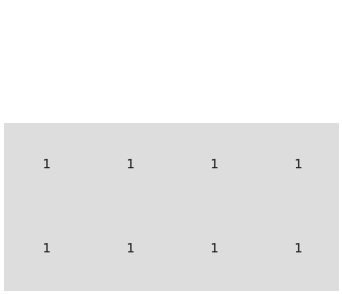
[22] Wikimedia Commons, "Tetracomb.svg," 2020, [Online; Image; Released into the public domain by the copyright holder.]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Tetracomb.svg&oldid=514764565

[23] Wikimedia Commons, "Pentacomb.svg," 2020, [Online; Image; Released into the public domain by the copyright holder.]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Pentacomb.svg&oldid=459686908

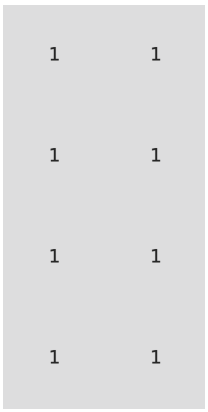
APPENDIX A

Kernels Used in the Experiment

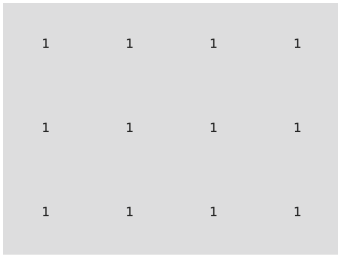
A.1 W kernels



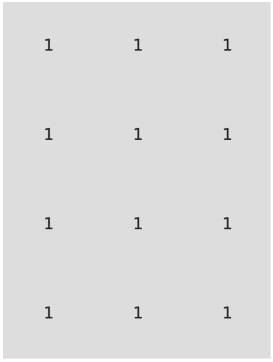
(a) $W_{2\times4}$



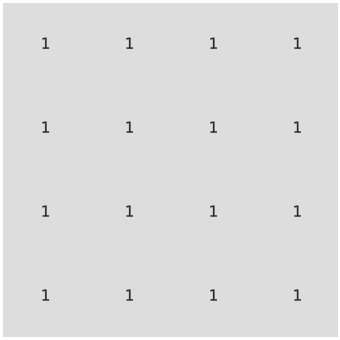
(b) $W_{4\times2}$



(c) $W_{3\times4}$



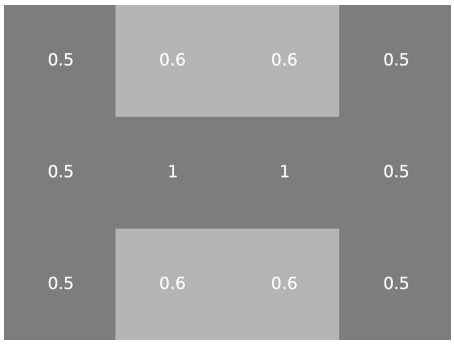
(d) $W_{4\times3}$



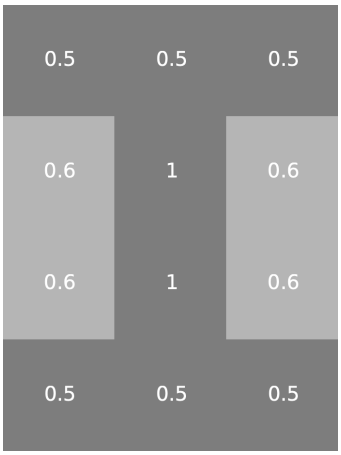
(e) $W_{4\times4}$

Figure A.1: All kernels W with weights set to 1.

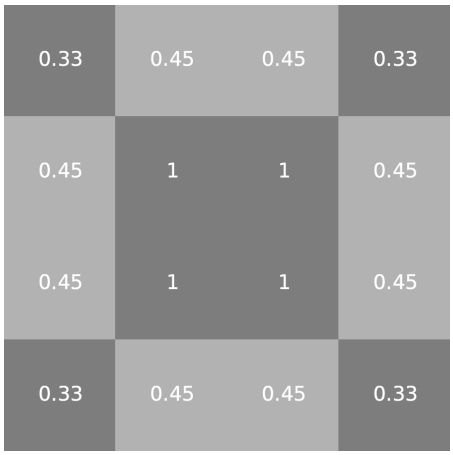
A.2 V kernels



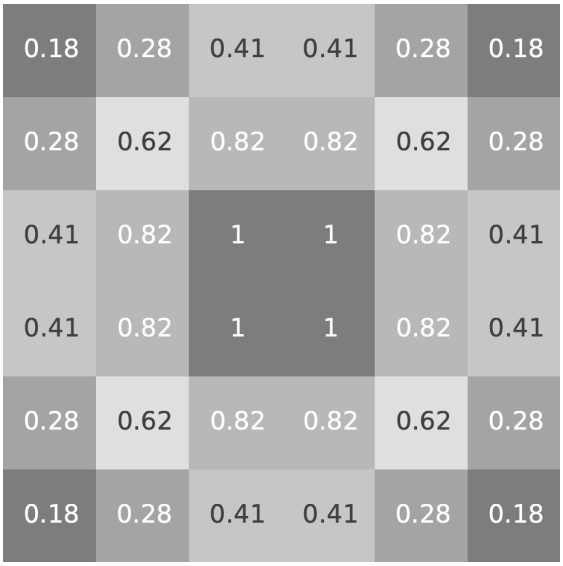
(a) $V_{3\times4}$



(b) $V_{4\times3}$



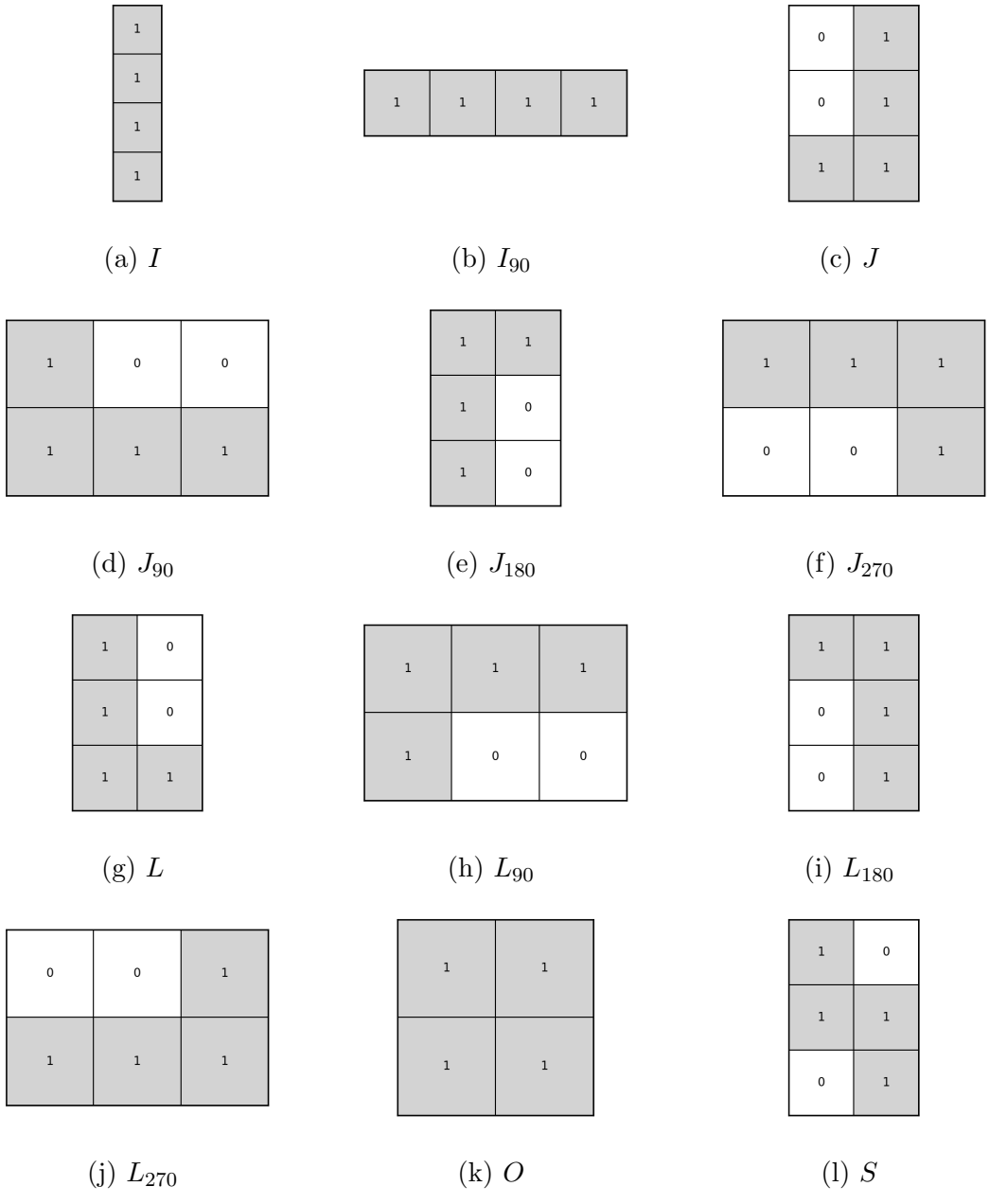
(c) $V_{4\times4}$



(d) $V_{6\times6}$

Figure A.2: All kernels V with variable weights.

A.3 T_{19} kernels



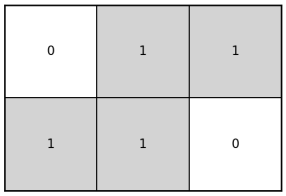
(a) I (b) I_{90} (c) J

(d) J_{90} (e) J_{180} (f) J_{270}

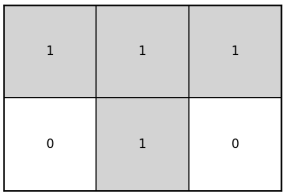
(g) L (h) L_{90} (i) L_{180}

(j) L_{270} (k) O (l) S

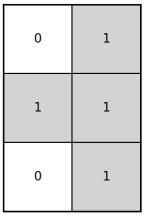
Figure A.3: The T_{19} fixed tetromino kernels.



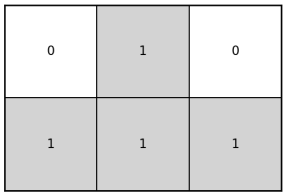
(m) S_{90}



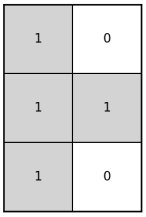
(n) T



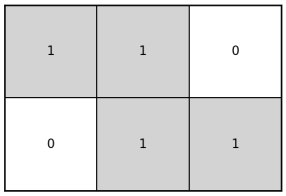
(o) T_{90}



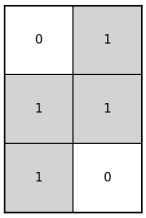
(p) T_{180}



(q) T_{270}



(r) Z



(s) Z_{90}

Figure A.3: The T_{19} fixed tetromino kernels (Continued).

APPENDIX B

Another Redistricting Game

Mapmaker is a game similar to Distrix. The board of the two-player version is a 37-tile hexagonal grid. Each hexagon in the grid represents a voting unit. Colored tokens placed on each of the hexagons represent the historic voting preferences in the units. The players place edges around the voting units to create districts. The player that controls the most districts at the end of the game wins. Figure B.1 shows an example Mapmaker board from the game's manual[18].

The 37-tile Mapmaker board is of a size close to the 36-tile Distrix board. For this reason, it may seem at first that the method outlined in Chapter 2 can use either Distrix or Mapmaker, interchangeably. The hexagonal shape of the tiles and board, however, make working with Mapmaker a lot more challenging.

Figure B.1: A Mapmaker game in progress, using the 2-player, 37-tile board[18].

To create an EV benchmark score for Mapmaker, we need the set of all possible tilings for the 37-tile hexagonal board. Instead of polyominoes, a combination of polyhexes needs to be used to tile the board (see Figure B.2) due to $37 \mod 4 \neq 0$. We have verified through "checkerboard coloring"[19], that the board can be tiled with at least the following combinations of polyhexes:

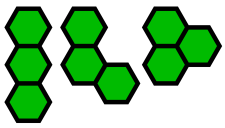
- 7 tetrahexes and 3 trihexes[1]
- 1 trihex, 6 tetrahexes and 1 pentahex
- 2 pentahexes, 1 trihex and 6 tetrahexes

Although other combinations may also exist, including any involving higher-degree polyhexes, we have not explored these. From [20] it is known that the total number of fixed polyhex forms of degree 3, 4 and 5 is 241 (11 + 44 + 186). This is significantly more than the 19 tetrominoes used to tile Distrix. We have not found the exact number of polyhex placements (PPs)[2] produced by sliding the 241 polyhexes on the 37-tile board. However, it is reasonable to assume that it will far surpass the 381 TPs used to tile Distrix. Given the substantial increase, a combinatorial explosion is likely, posing significant challenges for generating all possible tilings using the backtracking algorithm of exponential time complexity detailed in Algorithm 2. We are also not aware of any work similar to [14] that provides the total number of possible tilings for a 37-tile hexagonal board. Such a number would be useful for verifying any tiling results. Thus, fully enumerating all possible tilings for the 37-tile board with the recursive backtracking algorithm appears elusive.

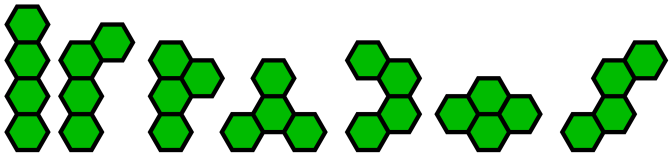
[1]This means that there will be 3-unit districts on the board which are not allowed in the formal rules of the Mapmaker game. This rule however, can be relaxed for the purposes of using the game as a research tool.

[2]Here PP denotes a polyhex placement, equivalent to the TP described in Chapter 2.

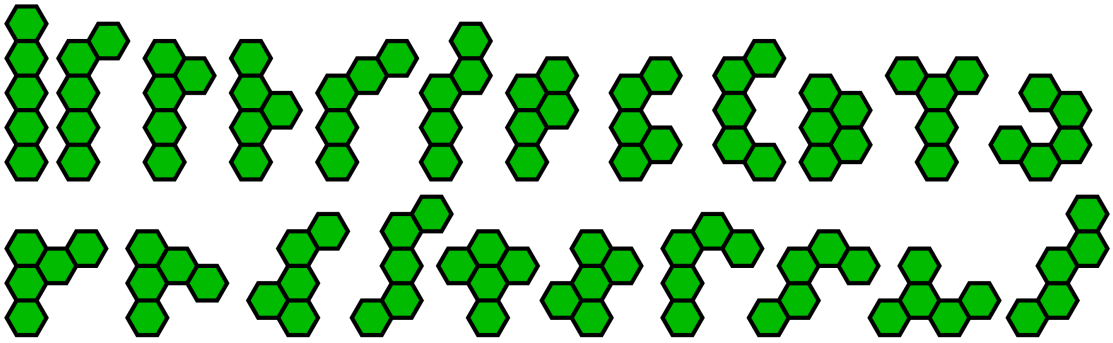
From [7] we understand that a limited solution to the Mapmaker tiling problem can be achieved by using each of the 7 free tetrahexes, together with each of the 3 free trihexes. This results in 12,290 unique tilings, which, however, is far less than the 178,939 tetromino tilings for the 6x6 Distrix board. In addition, this means that we have to use both 3 and 4 unit districts in Mapmaker versus the uniform 4-unit districts used in Distrix which, depending on the goal of the research, can be considered a deviation from the real world requirement of equal population size districts. For these reasons, the Mapmaker board appeared less suitable for the experiments that we wanted to run in this study. It could, however, suit other experiments where the shape of the voting unit is considered important. For example each hexagonal unit - apart from the ones on the edges of the board - has 6 neighbors, which could better represent certain real-world redistricting scenarios.



(a) The 3 free trihexes[21]. Reflections and rotations result in 11 fixed trihexes.



(b) The 7 free tetrahexes[22]. Reflections and rotations result in 44 fixed tetrahexes.



(c) The 22 free pentahexes[23]. Reflections and rotations result in 186 fixed pentahexes.

Figure B.2: The free polyhexes of degree 3,4 and 5.

BIBLIOGRAPHY

Akoglu, H., "User's guide to correlation coefficients." *Turkish journal of emergency medicine*, vol. 18, no. 3, p. 91–93, Sept. 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6107969

Becker, A. and Solomon, J., "Redistricting algorithms," in *Political Geometry: Rethinking Redistricting in the US with Math, Law, and Everything In Between*, Duchin, M. and Walch, O., Eds. Springer International Publishing, 2022, pp. 303–340. [Online]. Available: https://doi.org/10.1007/978-3-319-69161-9_16

Butler, S., Ekstrand, J., and Osborne, S., "Tetris® Tiling." [Online]. Available: https://oeis.org/A230031

Butler, S., Ekstrand, J., and Osborne, S., "Tiling Tetris® Boards." [Online]. Available: https://www.stevebutler.org/research/publications

Coffin, S. T., *The puzzling world of polyhedral dissections.* Oxford Univ. Pr. [Online]. Available: https://johnrausch.com/PuzzlingWorld/chap02.htm#p7

DeFord, D., Duchin, M., and Solomon, J., "Recombination: A family of markov chains for redistricting," *Harvard Data Science Review*, vol. 3, no. 1, Mar. 2021.

Distrix Games LLC, "The Distrix Puzzle Book." [Online]. Available: https://www.distrixgames.com/distrix-puzzle-book.html#/

Duchin, M., "Gerrymandering metrics: How to measure? What's the baseline?" [Online]. Available: http://arxiv.org/abs/1801.02064

Duchin, M. and Walch, O., Eds., *Political geometry: rethinking redistricting in the US with math, law, and everything in between*, corrected publication ed. Cham: Birkhäuser, 2022.

Gardner, M., "Chapter 11: Polyhexes and Polyaboloes," in *Mathematical Magic Show*, ser. Martin Gardner's Mathematical Games. Mathematical Association of America, vol. 7, pp. 148–150, https://bookstore.ams.org/view?ProductCode=GARDNER/7.

Garvie, M. and Burkardt, J., "A new mathematical model for tiling finite regions of the plane with polyominoes." *Contributions to Discrete Mathematics*, vol. 14, pp. 95–131, 2020.

Golomb, S. W., *Polyominoes: puzzles, patterns, problems, and packings*, 2nd ed. Princeton University Press.

Golomb, S. W., *Backtracking and Impossible Constructions*, 2nd ed. Princeton, N.J: Princeton University Press, 1994, p. 33.

Golomb, S. W., *Polyominoes and Checkerboards*, 2nd ed. Princeton, N.J: Princeton University Press, 1994, p. 4.

Horn, R. A. and Johnson, C. R., *Section 5.2 Examples of norms and inner products*, 2nd ed. Cambridge; New York: Cambridge University Press, 2012, p. 321.

Johnson, R. A. and Bhattacharyya, G. K., *Chapter 5: Probability Distributions*, 6th ed. Hoboken, NJ: John Wiley Sons, 2010, p. 186.

Lafair Family Games, "The Gerrymandering Manual," 2018, Mapmaker: The Gerrymandering Game. [Online]. Available: https://www.kickstarter.com/projects/1639370584/mapmaker-the-gerrymandering-game

MIT Election Lab, "Redistricting," Jan. 2023. [Online]. Available: https://electionlab.mit.edu/research/redistricting

Petering, M., *The Distrix Puzzle Book*. Milwaukee, WI: Distrix Games LLC, 2020.

Rosen, K. H., *Sets*, 7th ed. New York: McGraw-Hill, 2012, p. 122.

Sipser, M., *Chapter 8: Space Complexity*, 2nd ed. Boston: Thomson Course Technology, 2006, p. 308–309.

The On-Line Encyclopedia of Integer Sequences, "Number of fixed hexagonal polyominoes with n cells." online. [Online]. Available: https://oeis.org/A001207

The On-Line Encyclopedia of Integer Sequences, "Number of fixed polyominoes with n cells." online. [Online]. Available: https://oeis.org/A001168

U.S. Election Assistance Commission, "Election Administration and Voting Survey 2020 Comprehensive Report, A Report From The U.S. Election Assistance Commission To The 117th Congress." [Online]. Available: https://www.eac.gov/sites/default/files/document_library/files/2020_EAVS_Report_Final_508c.pdf

Wikimedia Commons, "Pentacomb.svg," 2020, [Online; Image; Released into the public domain by the copyright holder.]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Pentacomb.svg&oldid=459686908

Wikimedia Commons, "Tetracomb.svg," 2020, [Online; Image; Released into the public domain by the copyright holder.]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Tetracomb.svg&oldid=514764565

Wikimedia Commons, "Tricomb.svg," 2020, [Online; Image; Released into the public domain by the copyright holder.]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Tricomb.svg&oldid=517035876

Wikimedia user: Anypodetos, "All 5 free tetrominoes.svg [midified]," 2023, online; Image [original modified by changing colors]; ©Anypodetos; GFDL version 1.2 or later; CC-BY-SA 3.0. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:All_5_free_tetrominoes.svg&oldid=789099523