Gerrymandering Under Uncertain Preferences

Ben Kelly

May 3, 2021

Abstract

Gerrymandering is the attempt to use redistricting processes to manipulate the results of a election, often for some type of political gain. While this problem has existed for hundreds of years, the advent of large-scale data collection and increased computational power allows for gerrymandering to be more accurate and precise than ever before. The ability to reason about the speed and accuracy of algorithms for solving this problem allows us to understand the magnitude of the effect of gerrymandering in real elections as well as inform attempts to create more balanced district assignments.

One aspect of both common discussion of gerrymandering as well as attempts to formalize it a decision problem is the assumption of *perfect information*, or the ability to know how each voter will vote before the election takes place. However, real applications of gerrymandering might not have this certainty and requires considerations of *imperfect information* to reason about the likelihood of election outcomes. In this paper we define a formalization of gerrymandering under probabilistic preference profile distributions as a decision problem, explore its complexity and present a greedy algorithm for solving the problem in polynomial time under constant candidate number and bounded voter weights.

1 Introduction

Gerrymandering refers to the intentional manipulation of district boundaries to favor a specific election result. Gerrymandering has been a reality in American redistricting processes and elections since nearly its inception as a nation, and its effects are felt every time an election is held for the United States House of Representatives. Around the world, gerrymandering remains an issue in many other nations where geographic districting is a part of the political process. Usually seen a net-negative in the democratic process, gerrymandering allows the group in control of drawing the district maps immense control over the results of future elections. A wealth of work has been done on gerrymandering from legal, political, mathematical, and more perspectives, but the advent of computation has created a new lens with which to analyze this centuries old problem. The existence of large amounts of detailed population data and predictive algorithms for voting allows those drawing the lines to be more detailed and precise than previously possible, and so gerrymandering has been a mainstay in national politics in the last few redistricting cycles. Recent legislation such as H.R. 1 aims to curtail the effects of gerrymandering, but even if it passes, gerrymandering is unlikely to disappear completely. Thus, ways of reasoning about the problem of gerrymandering and the magnitude of its effect in elections is important for understanding the impact it has on society and developing methods to detect and curtail it.

In many simple, visual examples of gerrymandering as well as existing literature (discussed in section 2), the assumption of *perfect information* of voter preference is made. While useful for helping people understand the problem, this assumption fails to match the real-world reality of the way people vote, which is not always consistent. Many voters are very consistent in their preferences, but often so called "swing voters," whose preferences change election to election, can decide close battles. By incorporating probabilistic models of voter preferences into the problem of gerrymandering, we can more accurately mimic real world scenarios and reason more effectively about election outcomes as we draw district maps. Even if we are not attempting to be manipulative and wish to draw "fair" districts, a way of determining such districts is incomplete if perfect information is assumed. In the following paper, we present a formalization of probabilistic gerrymandering as a decision problem. We discuss the increase in complexity that results in incorporating probabilistic preference profiles as well as assumptions that allow the problem to become NP. We also present a greedy algorithm to solve this problem in polynomial time under constant candidate number and bounded voter weights. Finally, we discuss simulations run testing this algorithm and results obtained as well as limitations and future work in this problem.

2 Related Works

Gerrymandering is a very well researched topic within many types of academic literature. More recently, gerrymandering has been worked on more and more as a computational problem for both the feasibility and approximation algorithms as well as fairness metrics. [6] represents the problem in a geographic setting where voters must vote at their closet polling location. They show that this problem is NP-Complete in the worst case as well as presenting a greedy algorithm to make the problem tractable. Not long after the same group published a formalization of gerrymandering as a graph problem [3], departing from a geographic representation. Ito et al. [5] followed up on this paper by showing that this problem is NP-Complete for even certain types of graphs but can be made polynomial under certain graph conditions. This paper intends to extend the problem definition presented in [3] to include metrics of imperfect information and look at results of the elections in a probabilistic manner.

Other recent work on gerrymandering includes work done on fairness metrics and attempts to make district assignments that maximize social welfare in some way. For example, Stoica et al. [7] redefine the problem as a minimization of maximum margin of victory across the groups. They prove that problem, as well as well as related problems that restricted groupings, are NP-Complete, and propose heuristic functions that decide the problem in polynomial time. These approaches focus on only the outcome of the elections, and do not consider other factors that determine the strength of a given district map. An important consideration missing in these papers that of shape, which has been studied and debated for a long time. Even the United States Supreme Court (Bush v. Vera) raises issue with strange district shape (namely low compactness) as a potential constitutional issue that could violate the Voting Rights Act if it affects minority racial groups. Alexeev & Mixon [1] prove an impossibility theorem that means that bounding district sizes, shapes, and "wasted" votes cannot be satisfied by a districting function in the general case. All of this research indicates that the problem of good redistricting is both hard and subjective, with many different considerations needing to be weighed. In fact, Barnes & Solomon [2] considering district shape and compactness can lead to manipulation, making it difficult to rely on compactness measurements as a legal or algorithmic tool.

Finally, we draw part of our proof and method from work done on calculating the probability of winning elections under imperfect information. Hazon et al. [4] prove that this problem is #P-Hard with parametric candidate number and present a polynomial time algorithm based on using dynamic programming with compiled voting results for certain voting rules such as methods presented by Xia and Conitzer in [8]. The structure of this algorithm is exceedingly useful in helping to develop a greedy algorithm for probabilistic gerrymandering since it essentially solves the verification problem for the problem.

3 Probabilistic Gerrymandering

3.1 The Voting Scenario

We take the general description of the voting setting from Cohen-Zemach et al. [3], which describes the setting of this election as a two stage process. Each voter $v \in V$ reports their preferences for the set of candidates C as a linear order over the candidates where the set of all such orders is $\pi(C)$. Further, a *voting rule* is defined as a function $f : \pi(C)^n \rightarrow \pi(C)$ that given the set of all voter preferences, selects a "winning" aggregation of the voter preferences. The set of voters V is partitioned into disjoint subsets (or districts) $V_1 \dots V_k$ where $\bigcup_{i=0}^{k} V_i = V$ and $\forall i\ |V_i| > 0$. Local elections are carried out within each district according to some voting rule f_1. Then, the aggregate profiles of these local elections are given to a secondary voting rule f_2, and a final preference is selected. The most common voting rule for many real world elections is plurality for both f_1 and f_2. For example, the local elections might decide on Congressional representatives, and the second round is those representatives voting on a bill in the United States House of Representatives. For the purposes of this problem, we do not assume anything about f_1, but we assume f_2 to be plurality, since it is by far the most prevalent rule in the second stage of the election. However, the way we define the problem does not rely on f_2, so this choice is not particularly relevant.

3.2 Definition of the Problem

Again, we build on the work done by [3] in representing gerrymandering as a problem related to a graph containing a network of voters which is divided into connected components which represent the new districts. Cohen-Zamach et al. indicate that this representation is a departure from traditional geographic representations of the problem, but we contend that a geographic version of gerrymandering can be transformed into this type of graph problem. For example, a procedure which tiles the geographic shape as nodes and connects adjacent tiles with edges would allow a geographic gerrymandering question to be represented in our graphical problem. Thus, by using a graph representation we do not lose the ability to solve the geometric questions, but extend our ability to solve non-geometric questions like dividing groups of people into committees with certain restrictions.

Definition 3.1: The input for gerrymandering under uncertainty, *PROB-GERRY*, is $\langle G, w, C, p, P_{map}, R, k, r, l, p_l, m, p_m \rangle$ where:

- G is an undirected graph (V, E) where V is the set of voters
- w is a function $V \to \mathbb{N}$ which gives the weight of each voter in the election.
- C is the set of candidates
- $p \in C$ is the target candidate
- P_{map} is a function $V \times \pi(C) \to [0, 1]$ which defines the probability of voter choosing a specific preference profile.
- R is a voting rule.
- $k \in \mathbb{N}$ is the number of districts
- $r \in \mathbb{R} \mid r \geq 1$ is the ratio-cap
- $l, m \in \mathbb{N} \mid l, m \leq k$
- $p_l, p_m \in [0, 1]$

We define *PROB-GERRY* as asking if there exists a way to remove edges from G to construct a new graph G' such that G' contains k connected components (or districts) where under voting rule R, p has at least a p_l chance to win at least l districts in the election and has no more than a p_m chance to win at most m districts in the election. Additionally, the largest connected component of G' must not be bigger than the smallest by a factor of r.

We choose to define both this minimum bound l and maximum bound m because there exist situations where we care both about the chance we're able to win a certain number of districts but also to minimize the risk of losing lots of district elections. The ratio cap helps us guarantee that the district sizes remain close, which is often a requirement for real life scenarios.

4 Complexity of Probabilistic Gerrymandering

In this section we explore the complexity of *PROB-GERRY*. We will discuss two main cases: when the number of candidates is bounded and when the number of candidates is unbounded.

4.1 Complexity with Constant Candidate Number and Bounded Weights

Let n be the number of voters and m be the number of candidates.

Theorem 4.1. *PROB-GERRY is NP-Complete for constant candidate number and weights bounded by poly(n).*

Proof. First, we show that *PROB-GERRY* is in NP by describing a polynomial time verifier. Given a district assignment G', we can easily gather the set of all connected components in polynomial time. We can then verify that the number of districts is correct and that the ratio cap is observed in polynomial time as well.

Then, we must be able to calculate the probability of a given candidate p winning a specific number of districts n, $P_{win}(p, n)$. If we are able to do that, we can calculate p_l as $\sum_{i=l}^{k} P_{win}(p, i)$ and p_m as $\sum_{i=0}^{m} P_{win}(p, i)$. We show that this $P_{win}(p, n)$ calculation takes polynomial time. First, we must calculate the probability that a candidate wins each district. The size of a single district is upper bounded by $O(n)$, and from the results shown in Hazon et al. [4], for constant candidate number and weights bounded by *poly*(n) the time complexity of calculating the probability of a single election takes polynomial time. We must simulate k elections to get all the probabilities, but this is still polynomial time. Once we have the probabilities for each district, we can simulate the second round of the election using the same procedure in [4] with the k districts as voters. This step is also polynomial time. Thus, we can make an algorithm to compute $P_{win}(p, n)$ in polynomial time, so therefore we can verify this assignment in polynomial time.

Next, we give a reduction from the deterministic version of this problem A_{GM} defined in [3]. Given an input for A_{GM} $\langle G, w, C, p, k, l, \succ\rangle$. The main difference between this input and the input for *PROB-GERRY* is $\succ$ which defines the preferences for each voter as opposed to P_{map} which we use in *PROB-GERRY*. The mapping reduction from $\langle G, w, C, p, k, l, \succ\rangle$ to $\langle G, w, C, p, P_{map}, R, k, r, l, p_l, m, p_m\rangle$ is the following:

1. Copy over G, w, C, p, k, and l.
2. Define P_{map} for all voter-preference pairs $(v, L(C)) \in V \times \pi(C)$ to be 1 if $\succ_v = L(C)$ and 0 otherwise.
3. R is plurality voting
4. Set $m = l$
5. Set $p_l = p_m = 1$

This transforms the deterministic version of the problem into the probabilistic version by assigning the probability of a voter's preference to be 1 and 0 for all other preferences. In A_{GM} the voting rule is assumed to be plurality, so we use that voting rule. And finally, we ensure that we choose the exact number of districts by asking if there a way to guarantee at least l districts are won and no more than l districts are more. That is, exactly l districts are won with probability 1. Since this reduction is done in polynomial time, $A_{GM} \leq_P$ *PROB-GERRY*. Since *PROB-GERRY* is in NP and there is a reduction from an NP-Complete problem, *PROB-GERRY* is NP-Complete when candidate number is constant and weights are bounded by *poly*(n). □

4.2 Complexity in the General Case

Theorem 4.2. *If P $\neq$ NP, PROB-GERRY is not in NP in the general case.*

Proof. Assume P $\neq$ NP. We will show that there does not exist a polynomial time verifier for *PROB-GERRY*. Let $k = 1$. Therefore, to find the probability of a candidate winning the single district, we must calculate the probability of them winning an election with n voters. Since the number of candidates is not constant (or the weights are not bounded by *poly*(n), this calculation is #P-Hard as shown in [4]. Since we assumed P $\neq$ NP, no polynomial time algorithm exists for this problem. Thus, there is no polynomial time verifier for *PROB-GERRY*, so *PROB-GERRY* is not in NP. □

5 A Greedy Algorithm

Next, we present a greedy algorithm for solving *PROB-GERRY* in polynomial time, assuming constant candidate number and weights bounded by $poly(n)$. The algorithm we developed was a combination of the greedy algorithm for deterministic gerrymandering developed in [3] and an algorithm for determining probability of winning under uncertainty developed by [4]. Drawing from the work done in [3], we build a new graph G' iteratively by greedily adding edges from G. In Cohen et al. this edge is chosen to maximize the ratio of the districts in which p wins and the number of districts won by the candidate who wins the most districts. They also restrict the edge so that an edge cannot be added if it it would violate the ratio cap, i.e. adding the edge would cause $\frac{\max_{c \in CC(G')}|c|}{\min_{c \in CC(G')}|c|} > r$. We deviate from this method in two important ways. The first is to choose edges that maximize the probability that p wins at least the same proportion of districts that we are aiming to win at the end. Namely, we maximum the probability that p wins at least $\frac{l}{k} \cdot N_{CC}(G')$ where $N_{CC}(G')$ is the number of connected components in G'.

Algorithm 1: Greedy Algorithm

```
init $G' \leftarrow (V(G), \emptyset)$ ;
while $N_{CC}(G') < k$ do
    forall $e \in E(G) - E(G')$ do
        $P_{districts}[\,] \leftarrow 0$ ;
        forall $cc \in CC(G' \cup e)$ do
            $T[,] \leftarrow 0$ ;
            $VotingResult(T, P_{map}(cc))$ [4] ;
            forall $\vec{row} \in T$ do
                if $R(\vec{row}) = p$ then
                    $P_{districts}[cc] \leftarrow P_{districts}[cc] + T[\vec{row}][n]$ ;
                end
            end
        end
        $T_{total}[,] \leftarrow 0$ ;
        $VotingResult(T_{total}, P_{districts})$ [4] ;
        $pr_e \leftarrow 0$ ;
        forall $\vec{row} \in T_{districts}$ do
            if $NumVictories(\vec{row}) \geq \frac{l}{k} \cdot N_{CC}(G' \cup e)$ then
                $pr_e \leftarrow pr_e + T_{districts}[\vec{row}][N_{CC}(G' \cup e)]$ ;
            end
        end
        $ratio_e \leftarrow ComputeRatio(G' \cup e)$ [3] ;
    end
    $new_e \leftarrow argmax_{e \in E(G)-E(G')}\{pr_e \mid ratio_e \leq r\}$ ;
    if $new_e = null$ then
        $new_e \leftarrow argmin_{e \in E(G)-E(G')}\{ratio_e\}$ ;
    end
    $G' \leftarrow G' \cup new_e$
end
```

One other addition is that for each edge, like [3] we restrict edges that would make the district size ratio to exceed the ratio cap. However, in their algorithm they are able to break ties (which are likely with their choice of greedy score for edges) by choosing the edge that minimizes the ratio cap in $G' \cup e$. Due to our way of evaluating edges, ties are exceedingly unlikely and thus cannot rely the same strategy. Instead, we add the edges greedily based on probability of winning the proportion of available districts until there is no edge which does not cause G' to exceed the ratio cap. If no edges are found, we instead greedily choose the edge which minimizes the ratio cap. This allows us to maintain the ratio cap and not run out of edges to add. Without this addition, graphs of size 100 voters would consistently fail due to running out of edges, especially for smaller ratio caps. We describe the algorithm in Algorithm 1

The running time of this algorithm is $O(|E(G)|^2 \cdot (n \cdot \mathit{rows\ in\ T} \cdot |\mathit{prof}|))$ where n is the number of voters, m is the number of candidates, and $|\mathit{prof}|$ is the number of preference profiles which must be considered for each voter. The number of rows in T depends on the way voting results are consolidated. For an arbitrary positional scoring rule, the row number can be upper bounded by $O(n^{m^2})$. However, for a specific voting rule, the runtime can be improved and the row complexity of many common voting rules is summarized in [4]. Assuming plurality voting allows us to reduce the number of rows to $O(n^m)$. In general, the number of profiles that must be considered for each candidate is $m!$ since there exist $m!$ possible preference profiles. However, this assumption of plurality voting allows us to reduce the number of profiles which must be considered to m, since we only care about the top candidate.

Finally, this algorithm can be improved by storing the previous tables for each district and then merging them together instead of recalculating the whole table for each new edge that gets added. Together with storing the probabilities for districts, this caching allows us to avoid recomputing the whole probability for every edge and allows us to reuse work. However, due to the nature of the $VotingResult$ algorithm in [4], this change does not provide as much speedup as we might like so we've omitted this optimization for this description and simply used the $VotingResult$ function every time instead for simplicity of description. For our experiments later on, we implemented this optimization as it does reduce runtime. Specifically, under our assumption of plurality voting, it reduces the runtime from $O(|E(G)|^2 \cdot (n \cdot n^m \cdot m))$ to $O(|E(G)|^2 \cdot (n^m \cdot m))$, removing a factor of n since it is not necessary to do the calculation over all voters again.

6 Results

6.1 Voter and Candidate Generation

To run simulations on this greedy algorithm, a way of generating a network of voters with probability distributions over all preference profiles is required. To accomplish this, we used an extension of the "line method" often used by instead of assigning each voter a candidate a location in $[0, 1]$, we assign each voter and candidate a location in the *trait space* $[0, 1]^t$. This increase in dimension allows candidates to differ in more ways, allowing for a more diverse political spectrum. For our simulations, we used $t = 2$ to be able to visualize voter and candidate locations on the plane. Each voter is assigned a number for each dimension with a truncated normal distribution centered around 0.5 with the idea that "extremist" voters are less likely than "centrist" voters.

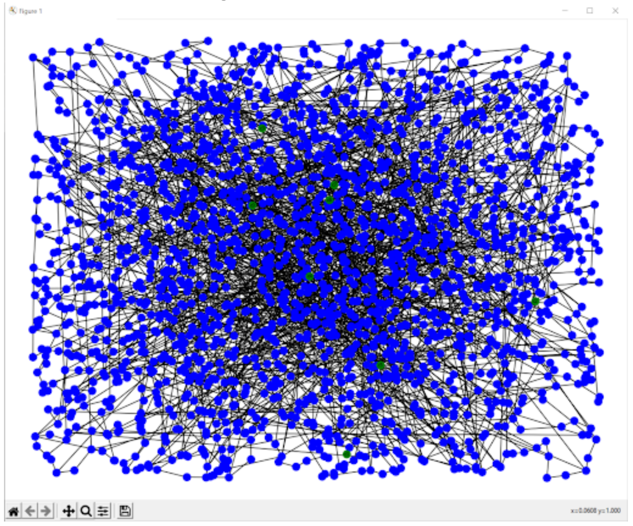
In the line model, a voter's preference profile is the candidates ordered by increasing distance from that voter's location. To extend this definition to probabilistic models, we use the common Plackett-Luce model for assigning probabilities to voter profiles. Plackett-Luce requires each voter to have a value for each candidate. We accomplish this by means of a *distance weighting function* that maps the distance between a voter and candidate to that voter's value of that candidate. Any function which is strictly positive over the interval $[0, \sqrt{t}]$ will work, but the only functions which make intuitive sense are strictly decreasing over that interval. For our simulations, we selected $\mathit{DWF}(x) = \frac{1}{x+\epsilon}$ for some small ϵ to prevent division by 0 for identical agents. Thus, the value of a candidate c for a voter v for the Plackett-Luce model is $\mathit{DWF}(\mathit{dist}(L_v, L_c))$ where L_i is the location of agent i in the trait space.

6.2 Voter Graphs

Next, we must find a way to connect voters in a graph to be able to reason about creating districts. We accomplish this task with the method set out in [3] using an Erdős–Rényi graph with an added "homophily factor" that increases the probability that ideologically similar voters are connected in the graph. The effect of the homophily factor is controlled by a paramater $\alpha \in [0, 1]$. Low α lowers the effect of the homophily factor and high alpha increase its effect. Figure 1 shows an example of a voter graph consisting of 2000 voters, d (expected degree of each voter) = 4, and $\alpha = 4$. For each of the results shown in the following subsections, we use the following simulation specifications:

- $\alpha = 0.25$
- $d = 8$
- $k = 5$
- $DWF(x) = \frac{1}{x+0.001}$
- $l = 3$
- $p_l = 0.7$
- $m = 2$
- $p_m = 0.25$

Figure 1: Example of a voter graph with 2000 voters, d = 4, $\alpha = 0.25$, t = 2



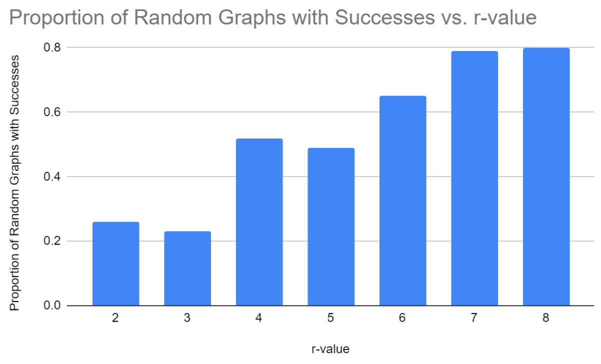
6.3 Effect of Ratio Cap

We explore the effect of the ratio cap on district size on the success rate of the greedy algorithm on finding a solution. We generated 100 graphs with 100 voters and ran the algorithm with 2 candidates. For all the graphs generated, we fixed $L_{c0} = (0.25, 0.25), L_{c1} = (0.75, 0.75)$ so both candidates are of equivalent strength in the average random graph. The result of this experiment is summarized in Figure 2. As expected, larger ratio caps increase the success rate of the algorithm against random graphs since larger values mean less restricted available district configurations. Results with low ratio cap such as 2 are not ideal, since many real world scenarios operate under low ratio cap. This result pinpoints ratio cap as a significant barrier both in the problem and in the greedy algorithm and a potential area for improvement.

6.4 Effect of Voter Number

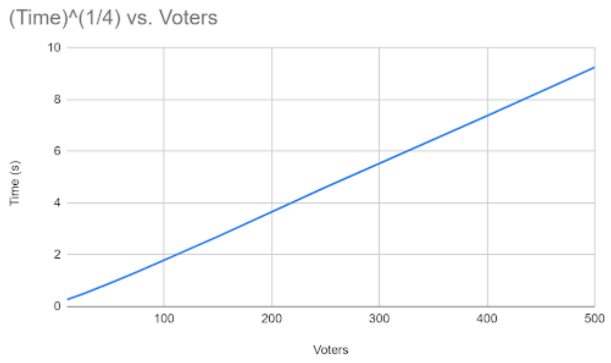
Next, we look at measures of runtime plotted against the number of voters. For this experiment, we fixed the ratio cap to be 5 and the number of candidates to be 2. We used plurality voting for this experiment, which allows us to use a representation of T that will generate $O(n^m)$ rows, where $|V| = n$ and $|C| = m$. Additionally, we implement the optimization described in section 5 that incrementally builds and combines the voting result tables for each district instead of recalculating every time. This reduces running time by a factor of n. We expect $O(n)$ edges in the graph, so the total runtime is expected to be $O(n \cdot n \cdot n^m) = O(n^{m+2}) = O(n^4)$ for fixed candidate number $m = 2$. The results are summarized in Figure 3. As expected,

Figure 2: Plot of ratio cap vs. proportion of successful graphs (n = 100)



the running time tracks extremely well with the number of voters to the fourth power. Despite being polynomial, the time required to compute a 500 voter graph was 7320 seconds, or just over 2 hours. This simulation was the best case scenario for the algorithm with $m = 2$ and plurality voting, so hopes that even this greedy algorithm can run in reasonable time on graphs the size of state populations are extremely small.

Figure 3: Plot of number of voters vs. the fourth root of time in seconds



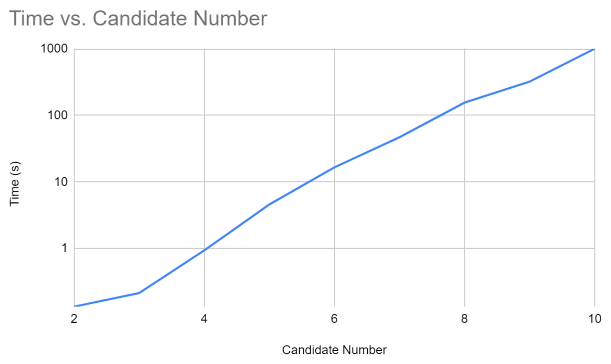
6.5 Effect of Candidate Number

Finally, we look at the effect of candidate number on the greedy algorithm. For this experiment, we fixed the ratio cap to be 5 and the number of voters to be 25. This number of voters is intentionally small, attempts at running the algorithm on 100 voters resulted in runtimes that exceeded what was reasonable for us to wait for even with $m = 6$. We also assume plurality voting and use the optimization used in the previous section. Thus, we can expect runtime of $O(25^m)$, which is exponential in m. The results are summarized in Figure 4 (log scale). As expected, the runtime scales exponentially with the number of candidates. This result is clear evidence that this algorithm and indeed problem is likely intractable on even a moderate number of candidates for reasonable voter number.

7 Limitations and Future Work

While the results obtained by this greedy algorithm are promising, there are significant limitations and places for improvement. The main area the algorithm struggles is with the ratio cap, which must be relatively high even on smaller voter graphs to have the algorithm have a reasonable chance to return the right answer. Especially for graphs that are sparser, the ratio cap is a challenging barrier to achieving a correct answer

Figure 4: Plot of number of candidates vs. the time in seconds



with this algorithm. Additionally, in real world scenarios like the United States Congressional districts, the ratio across the entire country is approximately 1.88. The ratio is only this high because the most populous district is the entirety of Montana. Within state borders where the district lines are actually drawn, it's likely that the acceptable range of district sizes is much lower. Applying this algorithm to real world data and parameters would therefore yield poor results, and is where I believe the most room for improvement lies.

Another place for improvement is the datasets used for the simulations run in section 6. While the ER Graph with the added homophily factor from [3] seems to be a reasonable way to generation this population data, drawing data from the real world would be interesting. As mentioned before, a tessellation of a geometric region that transforms a geometric shape of a region to redistrict into a graph of weighted voters would be a very interesting avenue for applying real world data to this project. On top of the connectivity of the graph, there is additional room for improvement on how to generating populations of voters with uncertain preferences. The Plackett-Luce model is a useful model to use, but the worth of each candidate to each voter can be adjusted by the distance weighting function, and exploring different distance weighting functions and which are appropriate is a good avenue for future work. More important, however, is exploring how to place people within the trait space. As mentioned, this project used a truncated normal distribution over [0,1] for each trait for each person. This may not accurately model the real world distribution of people, as elections in the United States tend to be consistently close in terms of popular vote, a phenomenon which would be difficult to explain with the distributions generated by this project, which creates many "centrist" voters that would tend to swing elections more drastically.

Finally, there might be chances for improvement in speed. This algorithm, while useful since if it finds an accepting district configuration, that configuration is guaranteed to be a true answer to the problem since the probabilities of winning the districts are calculated exactly. This feature is important but requires the completion of an NP-Complete problem if candidate number is not constant. To remedy this, the development of a general case P greedy algorithm would be useful. While unable to guarantee the same results since it is unable to verify it's own answer, this algorithm could be much more feasible to solve for larger populations with multiple candidates.

I hope to continue work on this project over the summer to more thoroughly explore what can be done in this topic and continue to develop novel work that improves upon the work done by the papers which this project used as a starting point.

References

[1] Boris Alexeev and Dustin G Mixon. An impossibility theorem for gerrymandering. *The American Mathematical Monthly*, 125(10):878–884, 2018.

[2] Richard Barnes and Justin Solomon. Gerrymandering and compactness: Implementation flexibility and abuse. *arXiv preprint arXiv:1803.02857*, 2018.

[3] Amittai Cohen-Zemach, Yoad Lewenberg, and Jeffrey S Rosenschein. Gerrymandering over graphs. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 274–282, 2018.

[4] Noam Hazon, Yonatan Aumann, Sarit Kraus, and Michael Wooldridge. On the evaluation of election outcomes under uncertainty. *Artificial Intelligence*, 189:1–18, 2012.

[5] Takehiro Ito, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. Algorithms for gerrymandering over graphs. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1413–1421, 2019.

[6] Yoad Lewenberg, Omer Lev, and Jeffrey S Rosenschein. Divide and conquer: Using geographic manipulation to win district-based elections. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 624–632, 2017.

[7] Ana-Andreea Stoica, Abhijnan Chakraborty, Palash Dey, and Krishna P Gummadi. Minimizing margin of victory for fair political and educational districting. *arXiv preprint arXiv:1909.05583*, 2019.

[8] Lirong Xia and Vincent Conitzer. Compilation complexity of common voting rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.