Gerrymandering Over Graphs

Amittai Cohen-Zemach
School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel
amittai.cohen-zemac@mail.huji.ac.il

Yoad Lewenberg
School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel
yoadlew@cs.huji.ac.il

Jeffrey S. Rosenschein
School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel
jeff@cs.huji.ac.il

ABSTRACT

In many real-life scenarios, voting problems consist of several phases: an overall set of voters is partitioned into subgroups, each subgroup chooses a preferred candidate, and the final winner is selected from among those candidates. The attempt to skew the outcome of such a voting system through strategic partitioning of the overall set of voters into subgroups is known as "gerrymandering".

We investigate the problem of gerrymandering over a network structure; the voters are embedded in a social network, and the task is to divide the network into connected components such that a target candidate will win in a plurality of the components. We first show that the problem is NP-complete in the worst case. We then perform a series of simulations, using random graph models incorporating a *homophily* factor. In these simulations, we show that a simple greedy algorithm can be quite successful in finding a partition in favor of a specific candidate.

KEYWORDS

Voting; Social Choice; Gerrymandering; Districts; Social Networks

ACM Reference Format:
Amittai Cohen-Zemach, Yoad Lewenberg, and Jeffrey S. Rosenschein. 2018. Gerrymandering Over Graphs. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

Collective decision-making often arises when agents interact with one another. In these situations, any individual in the group may have a preference over the possible choices that can be made, yet a single choice may need to be made for the entire group. It is thus necessary to aggregate the opinions or preferences of multiple agents to achieve a unified decision. The rules that define how to reach a group decision are often called social choice functions.

We can frame these group choices as elections, where each agent is a voter with a personal preference over available candidates. Voters report their preferences to a center that applies some social choice function (i.e., voting rule) to aggregate the preferences and output a collective decision. A key topic of interest is *manipulation*, that is, strategic exploitation of the protocol by an agent, aimed at improving the final outcome from that agent's perspective.

Most commonly, manipulation refers to strategic voting, i.e., misreporting true preferences. The most fundamental theorems in social choice theory, Arrow's theorem [1] and the Gibbard-Satterthwaite theorem [11, 17], are closely-related impossibility results regarding the existence of social choice functions that meet some basic criteria.

However, it is not only the voter that can influence outcomes via strategic behavior; the center, or "election chair", may change aspects of the election to affect the result. This class of manipulative actions is known as strategic *control*.

In this paper, we focus on hierarchical voting systems such as those that are district-based. These systems consist of two phases:

(1) Each voter reports their preferences, and these are aggregated locally with other voters in the same group or district (using some predefined voting rule);
(2) The results from phase one are aggregated globally, using a second voting rule (potentially different from the first).

Hierarchical voting systems are common; one example is the U.S. presidential election system. In hierarchical voting and district-based elections, manipulating the borders between districts in order to skew the final outcome in favor of a preferred candidate is known as "gerrymandering". This technique has been used by political parties in the U.S. for many years [9, 12].

In this paper we present a model where voters are embedded in a social network. Each voter is represented as a vertex, and a connection between voters is represented as an edge. Such scenarios arise, for example, in a company divided into divisions. In this case, a link between two workers (vertices) could represent a professional connection between the two. The company is split into divisions, where each division is a cluster of workers in the graph. In this example, every division might reach an internal decision, and then based on the outcome from the first stage, the final decision for the entire organization is made.

We study gerrymandering as a control problem. We first show that given a graph with voter preferences, finding a partition into connected components that helps a specific candidate win is a computationally hard problem. Turning to simulations, we present a greedy algorithm that tries to find the best partition for a given candidate, and test its performance on graphs with various properties. Specifically, we are interested in examining how the *homophily* level of the graph affects the algorithm's performance; in graphs with a high level of homophily, voters with similar preferences are more likely to have a link between them. Our simulations show that if we wish to divide the graph into a small number of components then it is easier as the homophily level decreases. However, for graphs with a high level of homophily, it is easier to find partitions that help a specific candidate as the number of components increases.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

2 RELATED WORK

The effect of districts on election outcomes has been widely studied; mostly, however, it has been examined within political science, where data analysis from past elections is used to determine whether gerrymandering has occurred (see, e.g., [9]). In the computational social choice community, the problem of voter manipulation has been extensively studied [5, 10]. For example, Xia et al. [20] and Zuckerman et al. [21] studied coalitional manipulation, where a coalition of manipulators collude in order to make their favorite candidate win. Control problems that emerge in voting scenarios have been studied in the context of non-hierarchical voting systems, where, for example, Procaccia et al. [16] studied the complexity of control problems both in the worst case (by proving a hardness result) and in the average case (using simulations rather than analyzing real world data). Other work dealt with control problems in the context of hierarchical voting systems, where the task is to divide the voters into groups [3, 8, 14]. Those models, however, do not assume a network structure of the voters.

A relatively small number of papers deal with district-based elections within the computational social choice community. Pegden et al. [15] designed a protocol for dividing a state into districts, and showed that the protocol has provable guarantees for both the number of districts in which each party has a majority of supporters and the extent to which either party has the power to pack a specific population into a single district.

Taking a somewhat similar point of view, Bachrach et al. [2] define a ratio that indicates how unrepresentative a district election is, under some voting rule f. That ratio is computed compared to the outcome when applying f to the entire population, without districts; they show some bounds on this ratio, for various known voting rules.

In a later study, Lewenberg et al. [13] show that gerrymandering in a geographical setting, in which voters have to vote at the closest polling place, is NP-complete. They demonstrate, using both simulations and real data from elections in Israel and the UK, that in practice gerrymandering may not be so hard, and that it is possible to gerrymander reasonably well in favor of multiple candidates using a simple greedy algorithm. This paper can be seen as a direct extension of their research, with one major difference, namely, that here we renounce the geographical setting in favor of a graphical setting, substituting physical distances between voters with more abstract links of a network, that could have various interpretations (such as organizational structure, friendship, or any other shared property).

Clough [4] studies strategic voting in social networks, using a 13×13 grid-based undirected graph. In their model, voters are grouped according to ideology, and therefore voters are connected primarily with those of similar political perspectives. They show how the degree of homogeneity of a political discussion network affects voter coordination on two parties in a single-member plurality system. Homogeneity is somewhat equivalent to the use of homophily in our work.

Coombs and Avrunin [6] study the emergence of single-peaked preferences in various domains and the psychological factors that shape it. They show that single-peakedness is psychologically inevitable if there is only one domain over which there exists a preference order, as is the case in our work.

Our work also has similarities to Tsang and Larson [19]. Tsang and Larson model voters as vertices in a network. They model the voting process as an iterative game, and study the effects of strategic behavior of the voters on the convergence of the iterative process. The graph modeling in this paper is similar to their modeling, featuring two simple random graph models, as well as a homophily factor.

Talmon [18] also models voters as vertices in a network. The paper considers multi-winner elections and studies the computational complexity of dividing the voters into districts that satisfy certain structural properties.

3 THE VOTING SETTING

We start with some general definitions needed for our theoretical analysis. An election is comprised of a set V of n voters (possibly weighted) and a set of candidates C. Let $\pi(C)$ be the set of orders over the elements of C. Each voter $v \in V$ has a preference order $\succ_v \in \pi(C)$. A voting rule is a function $f : \pi(C)^n \rightarrow \pi(C)$ that returns a ranking, given the profile of n voters.

In a hierarchical voting system, voters are divided into disjoint sets $V_1, \ldots, V_s$ such that $\cup_{i=1}^{s} V_i = V$. These sets define a set of s elections, where in each one of them a voting rule f_1 determines the winning candidate. Next, another voting rule f_2 is applied to the outcome of the first-stage elections, and selects the final winner of the election (note that f_2 is selected from the subclass of voting rules that use as input just the top-rated candidate in each preference order). It is common to set f_1 to be the plurality voting rule, in which the winner is simply the candidate with the most supporters (not to be confused with "majority", where a candidate is required to be chosen by a majority of the voters in order to win). Using plurality is both common in real-world scenarios and convenient for theoretical analysis. In particular, plurality can be thought of as the simplest voting rule possible, as it considers only the top-ranked candidate for every voter and not the full ranking. When showing theoretical hardness for plurality, it may be suggestive of the general hardness of the problem for other voting rules as well.

As for f_2, we slightly abuse the notation, and require that the target candidate would win in as many districts as possible (either directly, as done in the simulations, or indirectly, as done in the theoretical analysis, by casting the problem as a decision problem, asking whether it is possible to make the target candidate win in at least l out of k districts). We can therefore think of f_2 as also being the plurality rule; however, any other voting rule may be applied.

4 GERRYMANDERING OVER GRAPHS

Although it classically refers to manipulation of borders between districts over a geographic layout such as a plane, gerrymandering can also be interesting in non-geographic settings. Instead of associating a point on the plane with each voter, one could think of voters as vertices in a graph, such that edges represent some sort of connection between them (e.g., a social network, or a structural network of a company). We can ask whether it is possible (and how) to gerrymander such structures.

A graphical representation, for example, could be appropriate in a group selecting its president, assuming that the group members are divided into committees. Every committee first recommends a candidate, and based on those recommendations the president is selected. In this example, the members of the group might be embedded in a social network, where a link between two vertices (members) might indicate that they are friends. The control problem in this case is to find a partition into committees, based on their links, such that a target candidate will win.

The graphical representation can also be seen as an abstraction of an underlying geographical structure. In this case, a link between two vertices could indicate that the underlying distance between them is bounded by some constant. We thus obtain a gerrymandering problem in which the connected components of the graph translate into underlying territorial continuity.

5 THE COMPLEXITY OF GERRYMANDERING OVER GRAPHS

We begin with a formal description of the problem.

Definition 5.1. The input of the Gerrymandering Over Graphs problem, A_{GM}, is $\langle G, w, C, p, k, l, \succ \rangle$, where:

- $G = (V, E)$ is an undirected connected graph over $|V|$ voters (vertices),
- $w : V \to \mathbb{R}_+$ is a weight function on voters ($w(v) = r$ can be thought of as creating r duplicates of voter v),
- C is the set of candidates,
- $p \in C$ is the target candidate,
- $l, k \in \mathbb{N}$ are integers such that $1 \leq l \leq k \leq |V|$; and
- in addition, for each voter $v \in V$ we have a preference order $\succ_v \in \pi(C)$.

We are asked whether it is possible to remove edges from G such that we get a new graph G' with k connected components, such that the candidate p wins (according to the plurality rule) in l of the k "district elections" (defined by the connected components in G').

To ease specification of the proof, there is no need to set a specific voting rule f_2. However, the problem can be easily modified, without changing its complexity, such that the second voting rule will be plurality.

We add a further restriction to the problem's definition: we do not allow connected components of size 1. This has an intuitive interpretation of avoiding components in which a single voter has all the power. This also has a crucial role in the correctness part of the reduction.

Theorem 5.2. *A_{GM} is NP-complete.*

Proof. To show that A_{GM} is NP-complete, we reduce the Exact Cover by 3-Sets ($X3C$) problem, a known NP-complete problem, to our gerrymandering over graph problem. The input of $X3C$ is $X = (x_1, \ldots, x_{3n})$ a set of $3n$ elements and $\mathcal{S} = (S_1, \ldots, S_m)$ a collection of m subsets of X, where for every $S \in \mathcal{S} : |S| = 3$. We can represent an instance of the problem as a bipartite graph $G = (X \cup \mathcal{S}, E)$, where there is an undirected edge between $x \in X$ and $S \in \mathcal{S}$ if and only if $x \in S$. We are asked whether there is a subset $\overline{\mathcal{S}} \subset \mathcal{S}$ of n subsets such that for every element $x \in X$ there is exactly one subset $S \in \overline{\mathcal{S}}$ such that $x \in S$.

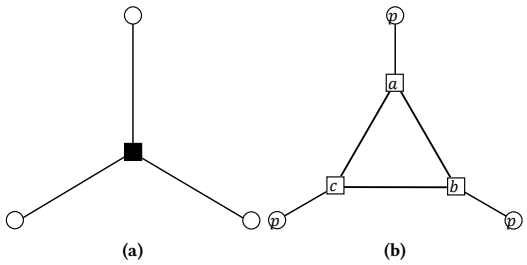

Figure 1: (a) An X3C gadget; the circles are the elements and the square is the set. (b) A Gerrymandering Over Graphs gadget; p's supporters are the circles, the squares are supporters of the other candidates. The weights of p's supporters are $1 + \varepsilon$, the weights of the other voters are 3.

We reduce an arbitrary $X3C$ instance to the following A_{GM} instance. First, for every $S \in \mathcal{S}$ we denote $S = \{x_a^S, x_b^S, x_c^S\}$, that is, we enumerate the elements in S arbitrarily. In the reduced A_{GM} instance, given as a graph $G' = (V', E')$, there are $3n + 3m$ voters and 4 candidates, a, b, c and p. For every element $x \in X$ there will be a voter $v^x \in V$ with weight $1 + \varepsilon$ that supports candidate p. For every collection $S = \{x_a^S, x_b^S, x_c^S\}$ there will be three voters $v_a^S, v_b^S, v_c^S \in V$ with weight 3, and edges among them. Voter v_a^S supports candidate a, voter v_b^S supports candidate b and voter v_c^S supports candidate c. For every element $x \in X$ and collection $S \in \mathcal{S}$ such that $x \in S$: if x is labeled as x_a^S then there will be an edge between v^x to v_a^S; if x is labeled as x_b^S then there will be an edge between v^x to v_b^S; and if x is labeled as x_c^S then there will be an edge between v^c to v_c^S. We are asked whether we can divide G' into m connected components such that p will win n of them. A graphical illustration of the reduction is given in Figure 1.

Now, assume there is a subset $\overline{\mathcal{S}} \subset \mathcal{S}$ such that for every element $x \in X$: $\left|\left\{S \in \overline{\mathcal{S}} : x \in S\right\}\right| = 1$. If $S \in \overline{\mathcal{S}}$ then the voters $\{v^x : x \in S\} \cup \left\{v_a^S, v_b^S, v_c^S\right\}$ will form a district; and if $S \notin \overline{\mathcal{S}}$ then the voters $\left\{v_a^S, v_b^S, v_c^S\right\}$ will form a district. Since $\overline{\mathcal{S}}$ is an exact cover, for every $x \in X$ voter v^x belongs to exactly one district. If $S \in \overline{\mathcal{S}}$ then p will win in the corresponding district with $3 + 3\varepsilon$ votes comparing to 3 for every other candidate. If $S \notin \overline{\mathcal{S}}$ then p will not win in the corresponding district. Therefore, we can divide G' into m connected components such that p will win n of them.

Now, assume that we can divide G' into m connected components such that p will win n of them. Note that in every district that p is winning we must have at least three voters that support p. This holds because the size of a district must be at least two, and every two p's supporters are connected via an opponent supported with weight 3. Since p is winning in n districts and there are exactly $3n$ p's supporters, it holds that in every district where p is winning there are exactly three voters that support p, and at most one voter that supports a, one voter that supports b, and one voter that supports c. Therefore, every district in which p is winning must consist of the voters $\{v^x : x \in S\} \cup \left\{v_a^S, v_b^S, v_c^S\right\}$ for some $S \in \mathcal{S}$. Let $\overline{\mathcal{S}}$ be the

collection such that $S \in \overline{\mathcal{S}}$ if and only if $\{v^x : x \in S\} \cup \left\{v_a^S, v_b^S, v_c^S\right\}$ form a district. Since v^x belongs to exactly one district it holds that x belongs to exactly one set in $\overline{\mathcal{S}}$, and thus there is a subset $\overline{\mathcal{S}} \subset \mathcal{S}$ such that for $\left|\left\{S \in \overline{\mathcal{S}} : x \in S\right\}\right| = 1$. □

We formulated the problem as trying to find a partition of the graph into k components such that p wins in *exactly* l components. However, given an A_{GM} instance, one can ask whether it is possible to make p win in *at least* l components. Denote the "at least" version of the problem by $A_{GM}^{\leq}$. Formally, $A_{GM}^{\leq}$ is the set of all instances $\langle G, w, C, p, k, l, > \rangle$ such that $\exists l \leq \theta \leq k$ such that $\langle G, w, C, p, k, \theta, > \rangle$ is a "yes" instance. Clearly, if we had an oracle for A_{GM}, we could use it to find the answer to $A_{GM}^{\leq}$ by simply querying the oracle for all $l \leq \theta \leq k$. In this sense, A_{GM} is at least as hard as $A_{GM}^{\leq}$. The next theorem states that $A_{GM}^{\leq}$ is still a hard problem.

Theorem 5.3. *$A_{GM}^{\leq}$ is NP-complete.*

The full proof is omitted as it is very similar to the proof of Theorem 5.2 (it builds on the same reduction and adds another $k - l$ vertices supporting p so that p will now win in "at least" k out of k components).

Graphs can model geographical settings where nodes are geographical voting units (e.g., voting precincts) and there is an edge between units that share a border. In this case, the induced graphs are planar. Therefore, it is natural to ask whether there is an efficient algorithm for the Gerrymandering Over Planar Graphs problem. The next theorem states that Planar-A_{GM} is still a hard problem.

Theorem 5.4. *Planar-A_{GM}, the problem defined as A_{GM}, only for planar graphs, is NP-complete.*

Proof. When applying the same reduction as in Theorem 5.2 to a Planar-Exact Cover by 3-Set (Planar-X3C) instance, the obtained graph is also planar. The Planar-X3C problem is known to be NP-complete [7]. □

Remark 5.5. *In Definition 5.1 we do not require that the size of the districts be equal. Our theoretical results show that without this constraint the problem is not tractable. Note that a wide family of partition problems and graph partition problems are known to be NP-complete. We thus chose to focus on the problem without any additional constraints so that we could establish that the hardness stemmed from the voting constraints, and not from other constraints.*

Nevertheless, there are real-world situations where gerrymandering occurs without the equal population constraint. For example, in the UK the number of electors in a UK constituency can vary considerably, with the smallest constituency currently having fewer than a fifth of the electors of the largest.

6 EMPIRICAL FEASIBILITY

In this section we discuss the simulations we performed. These focus on whether certain instances of the problem may be susceptible to gerrymandering, despite the worst-case hardness results shown above.

6.1 Graphs with "Natural" Properties

For our simulations, we restrict ourselves to a particular family of graphs, which we assume to have some "natural" properties (corresponding to those of real-world structures, such as a social network or an organization). The most basic notion we use is that vertices in the graph that are connected represent voters that are close in some social or structural sense, and therefore tend to have similar ideologies. This is a known phenomenon in the social science literature, often referred to as "homophily". We use the common and simple *line model* to generate candidates and voters. In the line model, both candidates and voters are represented as points on a line (we use the real interval $[0, 1]$), the voters rank the candidates by their distance from them. This results in preference relations that are called "single-peaked preferences".

Single-peaked preferences are widely used in social choice to simulate real-life preferences. Note that we used the plurality voting rule and only the top ranked candidate is considered. In this case using single-peaked preferences we may generate candidates with different support levels, and we may ensure that voters with the same top preference will be more likely to be connected (see Subsection 6.2 below).

Single-peaked preferences have two properties:

(1) each voter has an ideal choice, and
(2) each voter ranks choices lower as they get farther from his ideal choice.

For example, if we have three candidates defined by their locations $c_1 = 0.2$, $c_2 = 0.4$, and $c_3 = 0.7$, then a voter with location $v = 0.5$ will have the preferences $c_2 \succ c_3 \succ c_1$. Following the model proposed by Tsang and Larson, we use two random graph models to generate the underlying network structure:

(1) the Erdös-Rényi (ER) random graph model (in which each edge is realized independently with the same probability q), and
(2) the Barabasi-Albert (BA) preferential attachment model (in which we start with a small clique and then add vertices to it, such that each new vertex we add is connected to d existing vertices, which are chosen independently with probability proportional to their current degree).

Additionally, we incorporate the homophily property by multiplying the probabilities to generate edges between pairs of voters (vertices) by a factor h that depends on the distance between the two voters. This can be done in various ways, creating different kinds and strengths of homophily, and is discussed in the following section.

The degree distribution in ER graphs is typically very peaked: the probability of having a vertex with a certain degree *deg* decays exponentially fast as *deg* grows. Most real-world social networks, however, have a polynomial decay in these probabilities, meaning that a few vertices with very high degree exist (such networks are often called "scale-free networks"). Unlike the ER model, the BA model is scale-free, and we wish to examine if this property affects the ability to gerrymander.

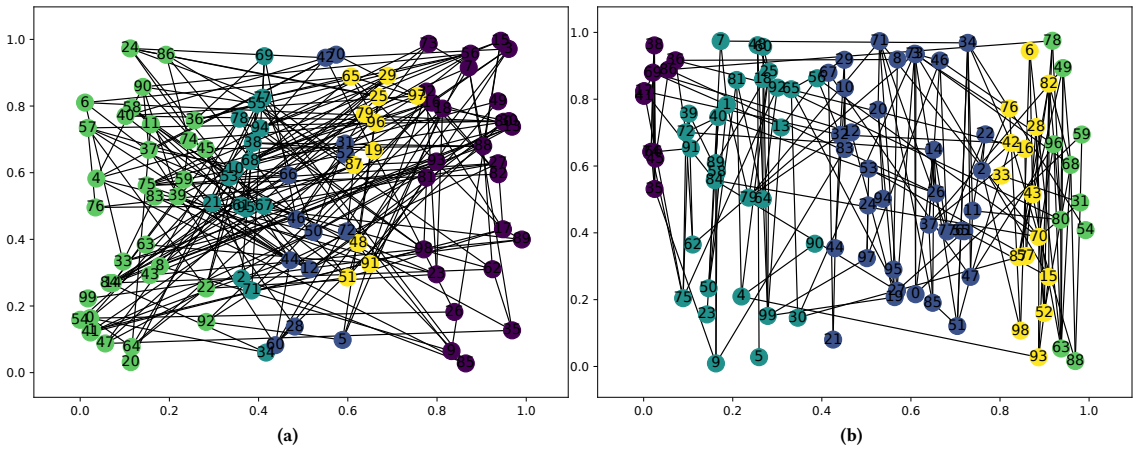

Figure 2: A sample of ER graph generated with $\alpha = 0$ (a) and $\alpha = 1$ (b). When α is high, the probability of an edge (u, v) to be realized is higher if u and v have similar preferences.

6.2 The Homophily Function

We want voters with similar preferences to be more likely to be connected; we thus modify the graph models to incorporate homophily. An edge between voters v, u may be realized with some initial probability (depending on the particular graph model). This probability of realization is then modified, multiplying by a homophily factor $h = h(u, v)$ which is a function of the two voters. The homophily function $h(u, v)$ is a function of the distance between the two voters $|u - v|$. Intuitively, we would like the homophily function to output a high value given two voters that are closer to each other (on the line) and who thus have relatively similar opinions. Therefore, the homophily function should be inversely correlated with the distance between its input voters.

Moreover, we would like to examine the effect of the homophily property, in addition to the graph model (ER or BA), on the ability to gerrymander successfully. To do so, we introduce another parameter, $\alpha \in [0, 1]$ that governs the relative weight of homophily in determining which links will be realized in the graph. As α grows, the probability of a voter being connected to other voters with similar preferences increases. Specifically, we use the following formula to compute the probability that an edge (v, u) will be realized:

$$P[(u, v)] = \alpha \cdot \beta \cdot q \cdot h(u, v) + (1 - \alpha) q,$$

where q is the initial probability of connecting v and u; $h(u, v)$ is the homophily coefficient for the voters v and u; and β is a constant that is added so that for any value of α, the expected edge density of the generated graph will be the same.

Figure 2 shows two ER graphs, one with $\alpha = 0$ and the other with $\alpha = 1$. These illustrate the effect of homophily on the edges that are realized in the generated graph. Specifically, one can see that in the graph that was generated with $\alpha = 1$, there are relatively more links between vertices with the same color (the color indicates their preferred candidate), compared to the graph generated with $\alpha = 0$, in which the probability is not affected by the preferences, so many edges link vertices with different colors. Note that the total number of edges in the two graphs is roughly the same.

In order to quantify the effects of different α values on the generated graph, we define the following score: for every vertex v, let $H(v)$ be the percentage of v's neighbors that prefer the same candidate as v. Let $H(G) = \frac{\sum_{v \in G} H(v)}{|G|}$. We expect that the $H(\cdot)$ score of a graph generated with high α values will be higher than a graph generated with low α values.

For our simulations, we used the following homophily function: $h(u, v) = \min\{1, \frac{\lambda}{|u-v|+\epsilon}\}$. The reason for using that function is because it allows a sufficient degree of modification via the parameter α; the H value increases (almost linearly) as a function of α, from ~0.25 to ~0.75 for both graph types.[1] Taking the minimum between $\frac{\lambda}{|u-v|+\epsilon}$ and 1 prevents us from getting too-large values. The purpose of ϵ is to avoid division by zero. λ serves to adjust the effective range of $h(u, v)$ to be "reasonably between 0 and 1".

In ER graphs the parameter q is often scaled as a function of the number of vertices n, for example $q = \frac{d}{n}$ or $q = d\frac{\ln n}{n}$ for some constant $d > 0$. This is done to maintain a constant expected degree. We here use $q = \frac{d}{n}$ (for ER graphs) and set $d = 4$ (for BA graphs, d is directly the number of vertices to which each new vertex is connected). This means that the expected edge density in both models is the same.

6.3 Gerrymandering in favor of different candidates

Finally, we wish to examine the ability to gerrymander in favor of candidates with different power levels (that is, candidates with different fractions of supporting voters). In our simulations we used $|C| = 5$, and tried to gerrymander in favor of all five candidates. An empirical average of the power of the five candidates was computed (over 100,000 instances). It shows that the strongest candidate has an average power of ~32%, that is, about 32% of the voters ranked her first; the middle candidate has an average power of ~19%; and the weakest candidate has an average power of ~10.5%. The two

[1] We also experimented with other homophily functions; however, as they allow smaller ranges of H values, and due to space constraints, we do not elaborate on this issue.

remaining candidates (called the "mid-strong" and the "mid-weak" candidates) had roughly the same power distribution as the middle candidate. This means that when sampling candidates and voters on a line, the results tend to naturally create candidates with different power levels, which is a good basis for investigating the effect of candidate power on the ability to gerrymander.

6.4 A Greedy Algorithm

Until now we described the underlying models used for the simulations. We now present a greedy algorithm for gerrymandering over a graph G (called the "greedy graph GM" algorithm). Given an input graph $G = (V, E)$, a candidate p, and a parameter k, the algorithm maintains a graph $G' = (V, E')$, starting with $E' = \emptyset$ and iteratively adding edges from E to E', and thus reducing the number of connected components in G' (denoted $N_{cc}(G')$), from its initial value $|V|$, until it is k. It then returns G' as a suggestion of a way to partition G into k connected components, while trying to make candidate p win in as many components as possible.

The choice of which edge to add to E' at each step of the algorithm is made greedily by picking an edge that maximizes the ratio between the number of connected components in which p wins and the maximum, over all candidates, of the number of connected components in which this particular candidate wins. At each step the greedy choice of which edge to add to E' is made while requiring that a given cap on the ratio $\frac{\max_{c\in CC(G')}|c|}{\min_{c\in CC(G')}|c|}$ is maintained (where $CC(G')$ denotes the set of connected components of G', and the size of a component c is simply the number of vertices in it). The idea behind this constraint is that we do not want the algorithm to find very lopsided graphs G', such as, in the extreme case, a graph in which there is a single giant component, and all $k-1$ other components are very small. Lopsided graphs like these are not desired as they typically cause misrepresentation of voters' opinions: few voters in small components have much more power than a large group of voters that resides in a single giant component.

Algorithm 1 Greedy Algorithm

```
procedure GreedyGraphGM (⟨G, w, C, p, k, l, ≺⟩, RatioCap)
    initialize G' = (V(G), E' = ∅)
    while N_CC(G') > k do
        for all e ∈ E\E' do
            r_e ← ComputeRation(G' ∪ e)
        end for
        best ← arg max_{e∈E\E'} {r_e | max_{c∈CC(G'∪{e})}|c| / min_{c∈CC(G'∪{e})}|c| ≤ RatioCap}
        G' ← G' ∪ {best}
    end while
    return G'
end procedure
procedure ComputeRatio(G')
    n_p ← |{c ∈ CC(G') : p wins in c}|
    n_o ← max_{a∈C} |{c ∈ CC(G') : a wins in c}|
    return n_p / n_o
end procedure
```

In addition, when multiple edges maximize the aforementioned ratio, the algorithm chooses the edge that, when added to the graph, maximizes the harmonic mean of the components' sizes. This is a way to make the algorithm prefer evenly-distributed component sizes over lopsided ones. This criterion is applied whenever more than a single edge maximizes the ratio (which happens often), thus providing another "soft" way to select edges, on top of the "hard" cap on the ratio $\frac{\max_{c\in CC(G')}|c|}{\min_{c\in CC(G')}|c|}$, given by the parameter *RatioCap* (in the simulations we set $RatioCap = 5$). The pseudo-code for the greedy algorithm is given in Algorithm 1.

6.5 Simulation Specifications

In this section we present and discuss the results of the simulations. We start with a brief overview of the different parameters used.

6.5.1 Parameters. We used two random graph models: ER and BA (as explained above). For each model, and each α value (determining the homophily strength in the generated graph) from the set $\{0, 0.25, 0.5, 0.75, 1\}$, 2000 graph instances were generated. The network size parameter was chosen such that the instances will be small enough for practical purposes, and other parameters were chosen accordingly. The hyperparameters in the homophily function were chosen according to the explanation in the homophily function section. The ratio cap was set to the lowest value that did not significantly impair overall performance. Specifically, we used the following parameters:

- Number of voters: 100
- Number of candidates: 5
- Average degree: 4
- *RatioCap*: 5
- h function: $h(u,v) = \min\left\{1, \frac{1}{200\cdot|u-v|+10^{-4}}\right\}$

For each such graph instance, the greedy algorithm was run with the parameter k (target number of components) ranging from 1 to 30, for each one of the candidates as p (the target candidate).

It is worth noting that although using the exact parameter values mentioned above for the number of voters and average degree throughout the simulations, we also carried out a short set of simulations with these parameters being set to other values (such as 200 voters, and average degree of 8). Notably, however, the results seemed to be very similar to the results we will present in the Results section below; it is suggestive that these parameters have no great impact on the greedy algorithm's ability to gerrymander.

6.5.2 Performance Evaluation. To rate the algorithm's performance for different candidates, the following measure was used: given the graph G' returned by the greedy algorithm, the success rate of candidate p, for some fixed α and k values, is the fraction of times (out of 2000 graphs) that p won in more components in G' than any other candidate. We refer to the case of requiring that p wins in *strictly* more components than any other candidate, as "strong success rate" (as opposed to "weak success rate"). Since the number of candidates and voters is fairly small in the simulations, we evaluate the performance using the strong success rate.

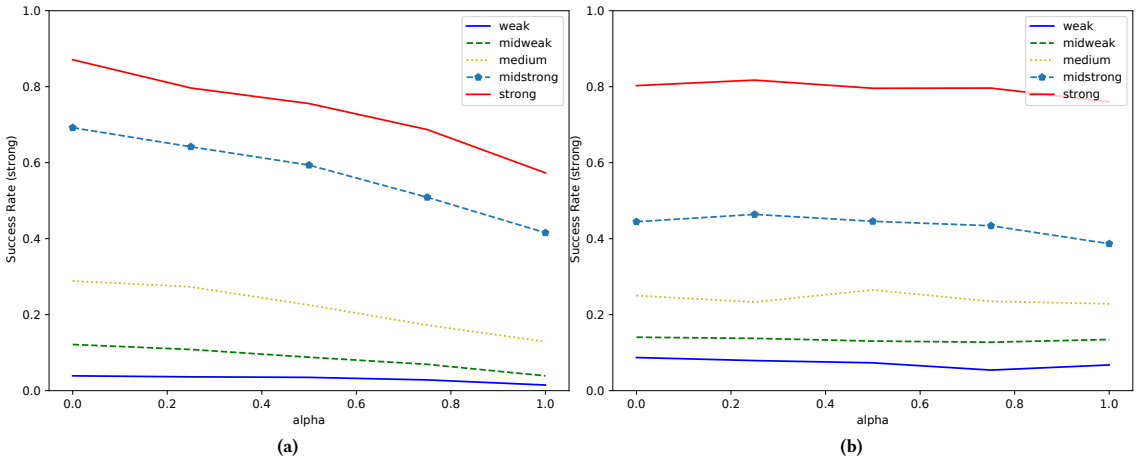

Figure 3: *The correlation of the homophily with the success rate of the algorithm.* Figure (a) shows that α negatively correlates with performance for low values of k ($k = 3$), and Figure (b) shows that α barely correlates with performance for high values of k ($k = 25$)

6.6 Results

6.6.1 General success rate and non-linear gaps. Most generally, the output graphs of the greedy algorithm indicated a reasonable success rate, depending on the initial power of the candidates. For example, when gerrymandering in favor of the strong candidate, the algorithm succeeded in making her win in a strict plurality of components ~70% – 80% of the time (depending on the other parameters, α and k). In addition, as expected, the results were correlated with the candidate's power level very consistently. Moreover, the gap in performance between candidates seems to be non-linear: the strong candidate had typically a far greater success rate than the other candidates, in comparison to the gaps between the four other candidates.

Note that the success rates of the algorithm may seem low; however, when summing the success rate of all five candidates, the result is significantly higher than 1. This means that our algorithm was able to improve the outcome for all candidates. This is supportive of the conclusion that in the average case the problem is tractable.

6.6.2 Graph model effects. Generally, the graph model appeared to have no impact on the performance of the algorithm. In other words, the success rate of our greedy algorithm was the same for ER and BA graphs. A possible explanation for this may be that our algorithm does not utilize the difference in structure between ER and BA graphs: in BA graphs there are few highly connected hubs, and a polynomial degree distribution, while in ER graphs there are no such hubs, and the degree distribution is exponential. It may be the case that the existence of hubs only makes it more likely that these hubs will be added to existing connected components in relatively early iterations of the algorithm (since they have a lot of incident edges), but does not affect the algorithm's performance any further, since it does not affect the maximization criterion. Similarly, the degree distribution seems to have no major impact on the algorithm. In what follows we discuss some other effects that emerged, and use mainly the ER results to demonstrate them.

6.6.3 Effects of α and k. The parameter α governs the homophily strength when generating a graph instance, and ranges from 0 (no homophily at all) to 1 (strong homophily). As α grows, the generated graphs tend to have a higher H score, indicating that more of a vertex's neighbors agree with his preferred candidate on average. The parameter k is the target number of components in the graph that the algorithm outputs (in range from 1 to 30). For $k = 1$ obviously the strongest candidate always wins, so we are interested in $k \geq 2$. The two main effects found were interactions between α and k:

For low k values, increasing α hurts performance. This effect is most significant for $k = 3$. As can be seen in Figure 3a, as α increases the performance drops from around 85% for the strong candidate at $\alpha = 0$ to around 60% at $\alpha = 1$. The performance for the other candidates drops as well (except for the two weakest candidates, which have very low success rate for all values of α). In contrast, for high k values, such as $k = 25$ for example (see Figure 3b), α no longer negatively correlates with performance, and in fact seems not to have any significant effect on performance. A possible explanation for this may be the fact that high α values tend to create clusters of vertices supporting the same candidate. Therefore, when trying to disconnect the graph into a small number of components, such as 3, it is hard for the algorithm to find a good solution since it has a smaller operating space. This results from the inability to disconnect these clusters while preserving a small total number of clusters. However, when k is relatively large, the algorithm has a larger operating space, that allows it to find solutions regardless of how strongly clustered the graph initially was.

For high α, increasing k boosts performance (mainly of the strong candidate). As can be seen in Figure 4a, as k increases, the performance of the strong candidate rises from around 60% for the strong

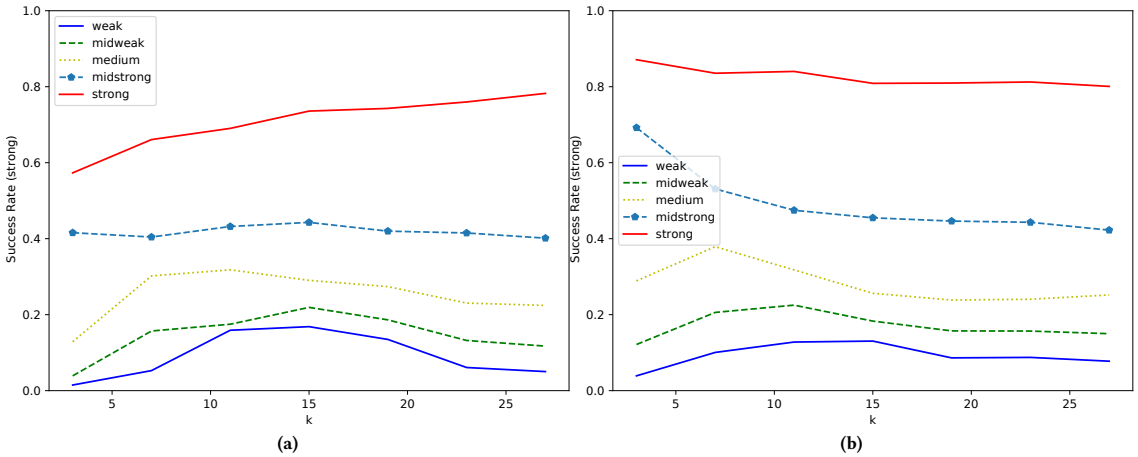

Figure 4: *The correlation of the target number of components with the success rate of the algorithm.* Figure (a) shows that k correlates with performance for high values of α ($\alpha = 1$), and Figure (b) shows that k barely correlates with performance for low values of α ($\alpha = 0$)

candidate at $k = 3$ to around 80% at $k = 25$. The performance of the mid-strong candidate stays roughly the same, whereas the three other candidates also exhibit some improvement while $k \leq 15$, after which the performance drops again. In contrast, for low α values, such as $\alpha = 0$ (see Figure 4b), k now negatively correlates with performance for the medium and mid-strong candidates, and has a slight positive effect for the two weakest candidates (both effects occur for $k \leq 15$). A possible explanation for this may again result from the fact that high α values tend to create clusters of vertices supporting the same candidate. In this case, the more components, the easier it is for the algorithm to find a partition into components that utilizes the relatively large number of supporters the strong candidate has. The rest of the candidates do not exhibit the same improvement since they have fewer supporters to begin with. In particular, the weakest candidates suffer both from low and from high k values, since both low and high values require more flexibility (i.e., a large space of possibilities) in order to be successful. Regarding the effect in Figure 4b, one could postulate that when there is no strong cluster structure, increasing the required number of components makes it harder to find a partition that makes a medium powered candidate win, since as there are more components, p has to win in more components as well, using its initially not so large number of supporters. As for the weakest candidates, they hardly manage to find good partitions under any value of k.

7 CONCLUSION

In this work we studied the gerrymandering problem over a graph structure representing voters with some type of links between them. While the problem is theoretically hard in the worst case, simulations suggest that there exist many instances in which it is not that hard to gerrymander. We examined the effects of various models and parameters on the hardness of the problem, such as the graph model, homophily strength, target candidate power, and target number of districts. We saw that success rates were reasonably high, and in particular, homophily makes it harder to gerrymander, especially when requiring a small number of districts.

There are several possible directions for future research. It would be interesting to understand what instances of the problem are harder and why, in a more general sense. Such a characterization should not be restricted to the performance of a single algorithm (like we did here with a simple greedy algorithm), or a specific graph model (like the ER and BA models used here). In addition, it might be interesting to check if and how certain parameters that we fixed in this work would affect the ability to gerrymander. For example, one could define another objective for the greedy algorithm to maximize in each iteration, and see how the performance changes and if we get similar effects.

ACKNOWLEDGMENT

This research has been partially supported by the HUJI Cyber Security Research Center in conjunction with the Israel National Cyber Directorate (INCD) in the Prime Minister's Office.

REFERENCES

[1] Kenneth J Arrow. 1950. A difficulty in the concept of social welfare. *Journal of political economy* 58, 4 (1950), 328–346.
[2] Yoram Bachrach, Omer Lev, Yoad Lewenberg, and Yair Zick. 2016. Misrepresentation in District Voting. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*. New York, 81–87.
[3] John J Bartholdi, Craig A Tovey, and Michael A Trick. 1992. How hard is it to control an election? *Mathematical and Computer Modelling* 16, 8-9 (1992), 27–40.
[4] Emily Clough. 2007. Talking Locally and Voting Globally: Duverger's Law and Homogeneous Discussion Networks. *Political Research Quarterly* 60, 3 (2007), 531–540.
[5] Vincent Conitzer and Toby Walsh. 2016. Barriers to Manipulation in Voting. In *Handbook of computational social choice*, Felix Brandt, Ulle Endriss Vincent Conitzer and, Jérôme Lang, and Ariel D. Procaccia (Eds.). Cambridge University Press, Cambridge, Chapter 6, 127–145.
[6] Clyde H Coombs and George S Avrunin. 1977. Single-peaked functions and the theory of preference. *Psychological review* 84, 2 (1977), 216.
[7] Martin E. Dyer and Alan M. Frieze. 1986. Planar 3DM is NP-complete. *Journal of Algorithms* 7, 2 (1986), 174–184.
[8] Gábor Erdélyi, Edith Hemaspaandra, and Lane A Hemaspaandra. 2015. More natural models of electoral control by partition. In *International Conference on*

Algorithmic DecisionTheory. Springer, 396–413.

[9] Robert S Erikson. 1972. Malapportionment, gerrymandering, and party fortunes in congressional elections. *American Political Science Review* 66, 4 (1972), 1234–1245.

[10] Piotr Faliszewski and Jörg Rothe. 2016. Control and Bribery in Voting. In *Handbook of computational social choice*, Felix Brandt, Ulle Endriss Vincent Conitzer and, Jérôme Lang, and Ariel D. Procaccia (Eds.). Cambridge University Press, Cambridge, Chapter 7, 146–168.

[11] Allan Gibbard. 1973. Manipulation of voting schemes: a general result. *Econometrica: journal of the Econometric Society* (1973), 587–601.

[12] Samuel Issacharoff. 2002. Gerrymandering and political cartels. *Harvard Law Review* (2002), 593–648.

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

[14] Cynthia Maushagen and Jörg Rothe. 2017. Complexity of Control by Partition of Voters and of Voter Groups in Veto and Other Scoring Protocols. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 615–623.

[15] Wesley Pegden, Ariel D. Procaccia, and Dingli Yu. 2017. A partisan districting protocol with provably nonpartisan outcomes. (2017). arXiv:arXiv:1710.08781

[16] Ariel D. Procaccia, Jeffrey S. Rosenschein, and Aviv Zohar. 2007. Multi-Winner Elections: Complexity of Manipulation, Control and Winner-Determination. In *The Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*. Hyderabad, India, 1476–1481.

[17] Mark Allen Satterthwaite. 1975. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of economic theory* 10, 2 (1975), 187–217.

[18] Nimrod Talmon. 2018. Structured proportional representation. *Theoretical Computer Science* 708 (2018), 58–74.

[19] Alan Tsang and Kate Larson. 2016. The echo chamber: Strategic voting and homophily in social networks. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 368–375.

[20] Lirong Xia, Michael Zuckerman, Ariel D. Procaccia, Vincent Conitzer, and Jeffrey S. Rosenschein. 2009. Complexity of Unweighted Coalitional Manipulation Under Some Common Voting Rules. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. Pasadena, California, USA, 348–353.

[21] Michael Zuckerman, Ariel D. Procaccia, and Jeffrey S. Rosenschein. 2008. Algorithms for the Coalitional Manipulation Problem. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '08)*. San Francisco, California, 277–286.