A General Stabilization Bound for Influence Propagation in Graphs

Pál András Papp
ETH Zürich, Switzerland
apapp@ethz.ch

Roger Wattenhofer
ETH Zürich, Switzerland
wattenhofer@ethz.ch

Abstract

We study the stabilization time of a wide class of processes on graphs, in which each node can only switch its state if it is motivated to do so by at least a $\frac{1+\lambda}{2}$ fraction of its neighbors, for some $0 < \lambda < 1$. Two examples of such processes are well-studied dynamically changing colorings in graphs: in majority processes, nodes switch to the most frequent color in their neighborhood, while in minority processes, nodes switch to the least frequent color in their neighborhood. We describe a non-elementary function $f(\lambda)$, and we show that in the sequential model, the worst-case stabilization time of these processes can completely be characterized by $f(\lambda)$. More precisely, we prove that for any $\epsilon > 0$, $O(n^{1+f(\lambda)+\epsilon})$ is an upper bound on the stabilization time of any proportional majority/minority process, and we also show that there are graph constructions where stabilization indeed takes $\Omega(n^{1+f(\lambda)-\epsilon})$ steps.

2012 ACM Subject Classification Mathematics of computing → Graph coloring; Theory of computation → Distributed computing models; Theory of computation → Self-organization

Keywords and phrases Minority process, Majority process

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.90

Category Track A: Algorithms, Complexity and Games

Related Version The full version of the paper is available at `https://arxiv.org/abs/2004.09185`.

1 Introduction

Many natural phenomena can be modeled by graph processes, where each node of the graph is in a state (represented by a color), and each node can change its state based on the states of its neighbors. Such processes have been studied since the dawn of computer science, by, e.g., von Neumann, Ulam, and Conway. Among the numerous applications of these graph processes, the most eminent ones today are possibly neural networks, both biological and artificial.

Two fundamental graph processes are majority and minority processes. In a *majority process*, each node wants to switch to the most frequent color in its neighborhood. Such a process is a straightforward model of influence spreading in networks, and as such, it has various applications in social science, political science, economics, and many more [29, 9, 12, 18, 23].

In contrast, in a *minority process*, each node wants to switch to the least frequent color in its neighborhood. Minority processes are used to model scenarios where the nodes are motivated to anti-coordinate with each other, like frequency selection in wireless communication, or differentiating from rival companies in economics [24, 6, 7, 11, 8].

© Pál András Papp and Roger Wattenhofer;
licensed under Creative Commons License CC-BY
47th International Colloquium on Automata, Languages, and Programming (ICALP 2020).
Editors: Artur Czumaj, Anuj Dawar, and Emanuela Merelli; Article No. 90; pp. 90:1–90:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Majority and minority processes have been studied in several different models, the most popular being the synchronous model (where in each step, all nodes can switch simultaneously) and the sequential model (where in each step, exactly one node switches). Since in many application areas, it is unrealistic to assume that nodes switch at the exact same time, we focus on the sequential model in this paper. We are interested in the worst-case stabilization time of such processes, i.e. the maximal number of steps until no node wants to change its color anymore.

Our main parameter describes how easily nodes will switch their color. Previously, the processes have mostly been studied under the basic switching rule, when nodes are willing switch their color for any small improvement. However, it is often more reasonable to assume a *proportional switching rule*, i.e. that nodes only switch their color if they are motivated by at least, say, 70% of their neighbors to do so. In general, we describe such proportional processes by a parameter $\lambda \in (0,1)$, and say that a node is switchable if it is in conflict with a $\frac{1+\lambda}{2}$ portion of its neighborhood. The stabilization time in such proportional processes (possibly as a function of λ) has so far remained unresolved.

The reason we can analyze proportional majority and minority processes together is that both can be viewed as a special case of a more general process of propagating conflicts through a network, where the cost of relaying conflicts through a node is proportional to the degree of the node. This more general process could also be used to model the propagation of information, energy, or some other entity through a network. This suggests that our results might also be useful for gaining insights into different processes in a wide range of other application areas, e.g. the behavior of neural networks.

In the paper, we provide a tight characterization of the maximal possible stabilization time of proportional majority and minority processes. We show that for maximal stabilization, a critical parameter is the portion φ of the neighborhood that nodes use as "outputs", i.e. neighbors they propagate conflicts to. Based on this, we prove that the stabilization time of proportional processes follows a transition between quadratic and linear time, described by the non-elementary function

$$f(\lambda) := \max_{\varphi \in (0, \frac{1-\lambda}{2}]} \frac{\log\left(\frac{1-\varphi}{\lambda+\varphi}\right)}{\log\left(\frac{1-\varphi}{\varphi}\right)}. \tag{1}$$

More specifically, for any $\epsilon > 0$, we show that on the one hand, $O(n^{1+f(\lambda)+\epsilon})$ is an upper bound on the number of steps of any majority/minority process, and on the other hand, there indeed exists a graph construction where the processes last for $\Omega(n^{1+f(\lambda)-\epsilon})$ steps.

2 Related Work

Various aspects of both majority and minority processes on two colors have been studied extensively. This includes basic properties of the processes [17, 36], sets of critical nodes that dominate the process [12, 15, 20], complexity and approximability results [21, 3, 10], threshold behavior in random graphs [14, 26], and the analysis of stable states in the process [16, 33, 4, 5, 34, 24]. Modified process variants have also been studied [35, 25], with numerous generalizations aiming to provide a more realistic model for social networks [2, 1].

However, the question of stabilization time in the processes has almost exclusively been studied for the basic switching rule (defined in Section 3.2). Even for the basic rule, apart from a straightforward $O(n^2)$ upper bound, the question has remained open for a long time in case of both processes. It has recently been shown in [13] and [27] that both processes can

exhibit almost-quadratic stabilization time in case of basic switching, both in the sequential adversarial and in the synchronous model. On the other hand, the maximal stabilization time under proportional switching has remained open so far.

It has also been shown that if the order of nodes is chosen by a benevolent player, then the behavior of the two processes differs significantly, with the worst-case stabilization time being $O(n)$ for majority processes [13] and almost-quadratic for minority processes [27]. In weighted graphs, where the only available upper bound on stabilization time is exponential, it has been shown that both majority and minority can indeed last for an exponential number of steps in various models [22, 28]. The result of [28] is the only one to also study the proportional switching rule, showing that the exponential lower bound also holds in this case; however, since the paper studies weighted graphs with arbitrarily high weights, this model differs significantly from our unweighted setting.

Stabilization time has also been examined in several special cases, mostly assuming the synchronous model. The stabilization of a slightly different minority process variant (based on closed neighborhoods) has been studied in special classes of graphs including grids, trees and cycles [30, 31, 32]. The work of [19] describes slightly modified versions of minority processes which may take $O(n^5)$ or $O(n^6)$ steps to stabilize, but provide better local minima (stable states) upon termination. For majority processes, stabilization has mostly been studied from a random initial coloring, on special classes of graphs such as grids, tori and expanders [14, 26].

Various aspects of majority processes have also been studied under the proportional switching rule, including sets of critical nodes that dominate the process, and sets of nodes that always preserve a specific color [38, 37]. However, to our knowledge, the stabilization time of the processes with proportional switching has not been studied before.

3 Model and Notation

3.1 Preliminaries

We define our processes on simple, unweighted, undirected graphs $G(V, E)$, with V denoting the set of nodes and E the set of edges. We denote the number of nodes by $n = |V|$. The neighborhood of v is denoted by $N(v)$, the degree of v by $\deg(v) = |N(v)|$.

We also use simple directed graphs in our proofs. A directed graph is called a DAG if it contains no directed cycles. A *dipartitioning* of a DAG is a disjoint partitioning (V_1, V_2) of V such that each source node is in V_1, and all edges between V_1 and V_2 all go from V_1 to V_2. We refer to the set of edges from V_1 to V_2 as a *dicut*.

Given an undirected graph G with edge set E, we also define the *directed edge set* of G as $\widehat{E} = \{(u, v), (v, u) \mid (u, v) \in E\}$, i.e. the set of directed edges obtained by taking each edge with both possible orientations.

A *coloring* is a function $\gamma : V \to \{\text{black, white}\}$. A *state* is a current coloring of G. Under a given coloring, we define $N_s(v) = \{u \in N(v) | \gamma(v) = \gamma(u)\}$ and $N_o(v) = \{u \in N(v) | \gamma(v) \neq \gamma(u)\}$ as the same-color and opposite-color neighborhood of v, respectively.

We say that there is a *conflict* on edge (u, v), or that (u, v) is a *conflicting edge*, if $u \in N_o(v)$ in case of a majority process, and if $u \in N_s(v)$ in case of a minority process. In general, we denote the conflict neighborhood by $N_c(v)$, meaning $N_c(v) = N_o(v)$ and $N_c(v) = N_s(v)$ in case of majority and minority processes, respectively. We occasionally also use $N_{\neg c}(v) = N(v) \setminus N_c(v)$.

If a node v has more conflicts than a predefined threshold (depending on the so-called *switching rule* in the model, discussed later) in the current state, then v is *switchable*. Switching v changes its color to the opposite color. If edge (u, v) becomes (ceases to be) a conflicting edge when node v switches, then we say that v has *created* this conflict (*removed* this conflict, respectively).

A *majority/minority process* is a sequence of steps (states), where each state is obtained from the previous state by a set of switchable nodes switching. In this paper, we examine sequential processes, when in each step, exactly one node switches. Such a process is *stable* when there are no more switchable nodes in the graph. By *stabilization time*, we mean the number of steps until a stable state is reached.

3.2 Model and switching rule

We study the worst-case stabilization time of majority/minority processes, that is, the maximal number of steps achievable on any graph, from any initial coloring. In other words, we assume the *sequential adversarial model*, when the order of nodes (i.e., the next switchable node to switch in each time step) is chosen by an adversary who maximizes stabilization time.

It only remains to specify the condition that allows a node to switch its color. The most straightforward switching rule is the following:

▶ **Rule I** (Basic Switching). *Node v is switchable if $|N_c(v)| - |N_{\neg c}(v)| > 0$.*

An equivalent form of this rule is $|N_c(v)| > \frac{1}{2} \cdot \deg(v)$. This rule is shown to allow up to $\widetilde{\Theta}(n^2)$ stabilization time for both majority [13] and minority [27] processes. However, it is often more realistic to assume a proportional switching rule, based on a real parameter $\lambda \in (0, 1)$:

▶ **Rule II** (Proportional Switching). *Node v is switchable if $|N_c(v)| - |N_{\neg c}(v)| \geq \lambda \cdot deg(v)$.*

Since we have $|N_c(v)| + |N_{\neg c}(v)| = \deg(v)$, this is equivalent to saying that v is switchable exactly if $|N_c(v)| \geq \frac{1+\lambda}{2} \cdot \deg(v)$. In the limit when λ is infinitely small (or, equivalently, as $\frac{1+\lambda}{2}$ approaches $\frac{1}{2}$ from above), we obtain Rule I as a special case of Rule II.

In case of Rule I, whenever a node v switches, it is possible that the total number of conflicts in the graph decreases by 1 only. On the other hand, Rule II implies that the switching of v decreases the total number of conflicts at least by $\lambda \cdot \deg(v)$ (we say that v *wastes* these conflicts), so in case of Rule II, the total number of conflicts can decrease more rapidly, allowing only a smaller stabilization time. Our findings show that the maximal number of steps is different for every distinct λ.

3.3 On the $f(\lambda)$ function

While the processes have a symmetric definition on each edge by default, it turns out that in order to maximize stabilization time, each edge has to be used in an asymmetric way. The most important parameter at each node v is the ratio of neighbors v uses as "inputs" and as "outputs". That is, the optimal behavior for each node v is to select $\varphi \cdot \deg(v)$ of its neighbors as outputs (for some $\varphi \in (0, 1)$), and create all new conflicts on the edges leading to these output nodes, and similarly, mark the remaining $(1 - \varphi) \cdot \deg(v)$ neighbors as inputs, and only remove conflicts from the edges coming from these input nodes. Note that with Rule II, whenever a node switches, it can create at most $\left(1 - \frac{1+\lambda}{2}\right) \cdot \deg(v) = \frac{1-\lambda}{2} \cdot \deg(v)$ new conflicts, so it is reasonable to assume $\varphi \in \left(0, \frac{1-\lambda}{2}\right]$.

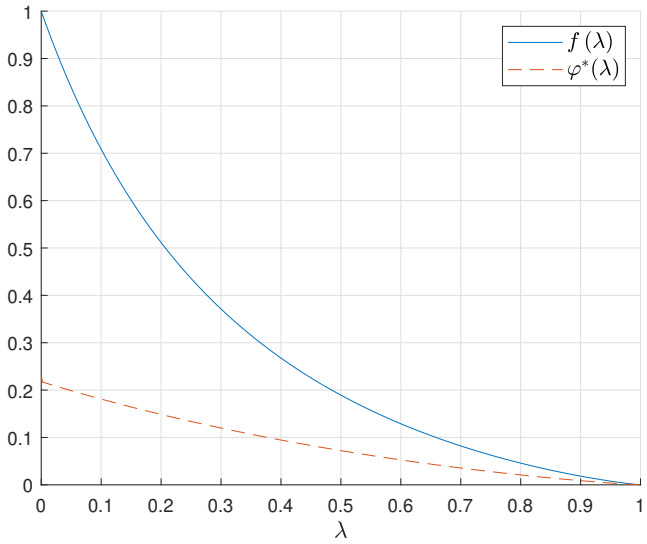

Figure 1 Plot of $f(\lambda)$ and $\varphi^*(\lambda)$ for $\lambda \in (0, 1)$.

Our results show that if all nodes select φ as their output rate, then the maximal achievable stabilization time is a function of

$$\frac{\log\left(\frac{1-\varphi}{\lambda+\varphi}\right)}{\log\left(\frac{1-\varphi}{\varphi}\right)}. \tag{2}$$

As such, the largest stabilization time can be achieved by maximizing this expression by selecting the optimal φ value, as shown in the definition of f in Equation 1. We denote the optimal value of φ (i.e., the argmax of Equation 2) by φ^*. The function f has no straightforward closed form, as such a form would require solving

$$(\lambda+1)\cdot\varphi\cdot\log\left(\frac{1-\varphi}{\varphi}\right) = (\lambda+\varphi)\log\left(\frac{1-\varphi}{\lambda+\varphi}\right),$$

for φ, with λ as a parameter. We discuss f in more detail in the full version of the paper.

Figure 1 shows the values of f and φ^* as a function of λ. The figure shows that both $f(\lambda)$ and $\varphi^*(\lambda)$ are continuous, monotonically decreasing and convex.

It is visible that $\lim_{\lambda\to 0} f(\lambda) = 1$ and $\lim_{\lambda\to 1} f(\lambda) = 0$. This is in line with what we would expect: the simple switching rule allows a stabilization time up to $\widetilde{\Theta}(n^2)$ [13, 27], while even for any large $\lambda < 1$, it is still straightforward to present a graph with $\Omega(n)$ stabilization time. Our main result is showing that $f(\lambda)$ describes the continuous transition between these two extremes.

4 General intuition behind the proofs

Note that initially, each node v can have at most $\deg(v)$ conflicts on its incident edges, and each time when v switches, it wastes $\lambda\cdot\deg(v)$ conflicts. Therefore, if each node were to "use" its own initial conflicts only, then each node could switch at most $\frac{1}{\lambda}$ times, and stabilization time could never go above $O(n)$.

Instead, the idea is to take the high number of conflicts initially available at high-degree nodes, and use these conflicts to switch the less wasteful low-degree nodes many times. Specifically, we could have a set of $\Theta(n)$-degree nodes that initially have $\Omega(n^2)$ conflicts

altogether on their incident edges, and somehow relay these conflicts to another set of $O(1)$-degree nodes, which only waste $O(1)$ conflicts at each switching. However, due to the large difference both in degree and in the number of switches, it is not possible to connect these two sets directly; instead, we need to do this through a range of intermediate levels, which exhibit decreasing degree and increasingly more switches. In order to maximize stabilization time, our main task is to move conflicts through these levels as efficiently (i.e., wasting as few conflicts in the process) as possible.

The formula of $f(\lambda)$ describes the efficiency of this process. The rate of inputs to outputs $\frac{1-\varphi}{\varphi}$ determines the factor by which the degree decreases at every new level. If φ is chosen small, then $\frac{1-\varphi}{\varphi}$ is high, so we only have a few levels until we reach constant degree, and hence the number of switches is increased only a few times. On the other hand, the increase in the number of switches per level is expressed by $\frac{1-\varphi}{\lambda+\varphi}$, which is a decreasing function of φ. If φ is too large, then although we execute this increase more times, each of these increases is significantly smaller.

With a degree decrease rate of $\frac{1-\varphi}{\varphi}$, we can altogether have about $\log_{\frac{1-\varphi}{\varphi}}(n)$ levels until the degree decreases from $\Theta(n)$ to $\Theta(1)$. If we increase the number of switches by a factor of $\frac{1-\varphi}{\lambda+\varphi}$ each time, then the $O(1)$-degree nodes will exhibit

$$\left(\frac{1-\varphi}{\lambda+\varphi}\right)^{\log_{\frac{1-\varphi}{\varphi}}(n)} = n^{\frac{\log\left(\frac{1-\varphi}{\lambda+\varphi}\right)}{\log\left(\frac{1-\varphi}{\varphi}\right)}} \leq n^{f(\lambda)} \tag{3}$$

switches, with an equation only if $\varphi = \varphi^*(\lambda)$. Having $\widetilde{\Theta}(n)$ nodes in the last level, this sums up to about $n^{1+f(\lambda)}$ switches altogether.

4.1 Conflict propagation systems

The upper bound on stabilization time is easiest to present in a general form that only focuses on this flow of conflicts in the graph. We define a simpler representation of the processes which only keeps a few necessary concepts to describe the flow of conflicts, and ignores e.g. the color of nodes or the timing of the switches at each node. In fact, we only require the number of times $s(v)$ each $v \in V$ switches, and the number $c(u,v)$ of conflicts that were created by node u and then removed by node v, for each $(u,v) \in \widehat{E}$.

For simplicity, given a function $c : \widehat{E} \to \mathbb{N}$, let us introduce the notation $c_{in}(v) := \sum_{u \in N(v)} c(u,v)$ and $c_{out}(v) := \sum_{u \in N(v)} c(v,u)$.

▶ **Definition 1** (Conflict Propagation System, CPS). *Given an undirected graph G, a conflict propagation system is an assignment $s : V \to \mathbb{N}$ and $c : \widehat{E} \to \mathbb{N}$ such that*

1. *for each $v \in V$, we have $c_{in}(v) + deg(v) \geq \lambda \cdot deg(v) \cdot s(v) + c_{out}(v)$,*
2. *for each $v \in V$, we have $c_{out}(v) \leq \frac{1-\lambda}{2} \cdot deg(v) \cdot s(v)$, and*
3. *for each $(u,v) \in \widehat{E}$, we have $c(u,v) \leq s(u)$.*

With the choice of $s(v)$ and $c(u,v)$ described above, any proportional majority or minority process indeed satisfies these properties, and thus provides a CPS. Hence if we upper bound the stabilization time (i.e. the total number of switches $\sum_{v \in V} s(v)$) of any CPS, this establishes the same bound on the stabilization time of any majority/minority process.

Condition 1 is the most complex of the three; it expresses the amount of "input conflicts" $c_{in}(v)$ required to switch v an $s(v)$ times altogether. Every time after v switches, it has at most $\frac{1-\lambda}{2} \cdot \deg(v)$ conflicts on the incident edges, so it needs to acquire $\lambda \cdot \deg(v)$ new conflicts to reach the threshold of $\frac{1+\lambda}{2} \cdot \deg(v)$ and be switchable again; this results in the

$\lambda \cdot \deg(v) \cdot s(v)$ term. Moreover, if in the meantime, the neighboring nodes remove some of the conflicts from the incident edges (expressed by $c_{out}(v)$), then this also has to be compensated for by extra input conflicts. Finally, the extra $\deg(v)$ term comes from the (at most) $\deg(v)$ conflicts that are already on the incident edges in the initial coloring. For a detailed discussion of this condition, see the full version of the paper.

Condition 2 also holds, since each time when v switches, it creates at most $\frac{1-\lambda}{2} \cdot \deg(v)$ conflicts on the incident edges. Each time u switches, it can only create one conflict on a specific edge, so condition 3 also follows. Hence any majority/minority process indeed provides a CPS.

Finally, we need a technical step to get rid of the extra $\deg(v)$ term in condition 1. Note that this term becomes asymptotically irrelevant as $s(v)$ grows; hence, our approach is to handle fewer-switching nodes separately, and require condition 1 only for nodes with large $s(v)$. More formally, we select a constant s_0, and we refer to nodes v with $s(v) < s_0$ as *base nodes*. We then consider *Relaxed CPSs*, where, given this extra parameter s_0, condition 1 is replaced by:

1R. *for each $v \in V$ with $s(v) \geq s_0$, we have $c_{in}(v) \geq \lambda \cdot deg(v) \cdot s(v) + c_{out}(v)$,*

This relaxation comes at the cost of an extra ϵ additive term in the exponent of our upper bound.

5 Upper bound proof

We now outline the proof of the upper bound on the number of switches. A more detailed discussion of this proof is available in the full version of the paper.

5.1 Properties of an optimal construction

We start by noting that since moving a conflict through a node is wasteful, it is suboptimal to have two neighboring nodes that both transfer a conflict to each other, or more generally, to move a conflict along any directed cycle. Therefore, in a CPS with maximal stabilization time, the conflicts are essentially moved along the edges of a DAG. To formalize this, given a CPS, let us say that a directed edge $(u, v) \in \widehat{E}$ is a *real edge* if $c(u, v) > 0$.

▶ **Lemma 2.** *There exists a CPS with maximal stabilization time where the real edges form a DAG.*

Proof. Among the CPSs on n nodes with maximal stabilization time, let us take the CPS P where the sum $\sum_{e \in \widehat{E}} c(e)$ is minimal. Assume that there is a directed cycle along the real edges of this CPS, and let $c(e_0)$ denote the minimal value of function c along this cycle.

Now consider the CPS P' where the value of c on each edge of this directed cycle is decreased by $c(e_0)$. Since in each affected node, the inputs and outputs have been decreased by the same value, P' still satisfies all three conditions, and thus it is also a valid CPS. Moreover, P' has the same amount of total switches as P. However, since $c(e_0) > 0$, the sum of $c(e)$ values in P' is less than in P, which contradicts the minimality of P. ◀

Hence for the upper bound proof, we can assume that the real edges of the CPS form a DAG. In the rest of the section, we focus on this DAG composed of the real edges of the CPS. We first show that for convenience, we can also assume that each base node is a source in this DAG.

▶ **Lemma 3.** *There exists a CPS with maximal stabilization time where each base node is a source node of the DAG.*

Proof. Note that by removing an input edge (u, v) of a base node v (that is, setting $c(u, v)$ to 0), the remaining CPS is still valid, since node v does not have to satisfy condition 1R, and in node u, only the sum of outputs was decreased. Therefore, we can remove all the input edges of each base node, and hence base nodes will all become source nodes of the DAG. ◀

▶ **Lemma 4.** *For each directed edge (u, v) in the DAG where u is a source node, $c(u, v) = O(1)$. More specifically, $c(u, v) \leq s_0$.*

Proof. If u is a base node, then $s(u) \leq s_0$, so $c(u, v) \leq s_0$ due to condition 3. Otherwise, condition 1R must hold, and since u has no input nodes, we get $0 \geq c_{out}(u) + \lambda \cdot \deg(u) \cdot s(u)$, hence $c_{out}(u) = 0$, so $c(u, v) = 0$ for every v. Thus $c(u, v) \leq s_0$. ◀

5.2 Edge potential

As a main ingredient of the proof, we define a way to measure how close we are to propagating conflicts optimally.

▶ **Definition 5** (Potential)**.** *Given a real edge $e \in \widehat{E}$, the potential of e is defined as $P(e) = c(e)^{1/f(\lambda)}$.*

For simplicity of notation, we also use P to denote the function $x \to x^{1/f(\lambda)}$ on real numbers instead of edges.

Intuitively speaking, the potential function describes the cost of sending a specific number of conflicts through a single edge, in terms of the number of initial conflicts used up for this. Note that since $f(\lambda) < 1$, the function P is always convex. This shows that sending a high number of conflicts through a single edge is more costly than sending the same amount of conflicts through multiple edges.

As the following lemma shows, the potential is defined in such a way that the total potential can never increase when passing through a node in the DAG; the best that a node can do is to preserve the input potential if it relays conflicts optimally.

▶ **Lemma 6.** *For any non-source node v of the DAG, with input edges from $N_{in}(v)$ and output edges to $N_{out}(v)$, we have*

$$\sum_{u \in N_{in}(v)} P(u, v) \geq \sum_{u \in N_{out}(v)} P(v, u).$$

Proof. If v is not a source, then by Lemma 3 it is not a base node, and thus has to satisfy condition 1R. In our DAG, c_{in} and c_{out} correspond to $\sum_{u \in N_{in}(v)} c(u, v)$ and $\sum_{u \in N_{out}(v)} c(v, u)$, respectively. Assume that we fix the value of c_{in} and c_{out}. Since the potential function P is convex, the incoming potential (left side) is minimized if c_{in} is split as equally among the input neighbors as possible. On the other hand, the outgoing potential (right side) is maximized if c_{out} is split as unequally among outputs as possible, so all output edges present in the DAG have the maximal possible number of switches, meaning $c(v, u) = s(v)$ for every $u \in N_{out}(v)$.

Assume that a fraction φ of v's incident edges are outgoing, i.e. $|N_{out}(v)| = \varphi \cdot \deg(v)$ and $|N_{in}(v)| = (1 - \varphi) \cdot \deg(v)$. By condition 1R, we have $c_{in} \geq \lambda \cdot \deg(v) \cdot s(v) + c_{out}$; with $c_{out} = \varphi \cdot \deg(v) \cdot s(v)$, this gives $c_{in} \geq (\lambda + \varphi) \cdot \deg(v) \cdot s(v)$. If split evenly among the $(1 - \varphi) \cdot \deg(v)$ inputs, this means

$$\frac{c_{in}}{|N_{in}(v)|} \geq \frac{(\lambda + \varphi) \cdot \deg(v) \cdot s(v)}{(1 - \varphi) \cdot \deg(v)} = \left(\frac{\lambda + \varphi}{1 - \varphi}\right) \cdot s(v)$$

switches for each input node. The inequality on the potential then comes down to

$$\sum_{u \in N_{in}(v)} P(u,v) \geq (1-\varphi) \cdot \deg(v) \cdot \left(\frac{\lambda+\varphi}{1-\varphi} \cdot s(v) \right)^{1/f(\lambda)} \geq$$
$$\geq \varphi \cdot \deg(v) \cdot s(v)^{1/f(\lambda)} \geq \sum_{u \in N_{out}(v)} P(v,u).$$

To show that the inequality in the middle holds, we only require

$$\left(\frac{\lambda+\varphi}{1-\varphi} \right)^{1/f(\lambda)} \geq \frac{\varphi}{1-\varphi},$$

or, put otherwise,

$$\frac{1}{f(\lambda)} \log \left(\frac{\lambda+\varphi}{1-\varphi} \right) \geq \log \left(\frac{\varphi}{1-\varphi} \right).$$

Since $\frac{\varphi}{1-\varphi} < 1$ (thus its logarithm is negative), we get

$$\frac{\log\left(\frac{\lambda+\varphi}{1-\varphi}\right)}{\log\left(\frac{\varphi}{1-\varphi}\right)} = \frac{\log\left(\frac{1-\varphi}{\lambda+\varphi}\right)}{\log\left(\frac{1-\varphi}{\varphi}\right)} \leq f(\lambda).$$

This holds by the definition of $f(\lambda)$. Note that this also shows that equality can only be achieved if the output rate φ is indeed chosen as the argmax value $\varphi^*(\lambda)$. ◀

Lemma 6 provides the key insight to the main idea of our proof: if we process the nodes of a DAG according to a topological ordering, always maintaining a dicut of outgoing edges from the already processed part of the DAG, then this potential cannot ever increase when adding a new node.

▶ **Lemma 7.** *Given a dicut S of a dipartitioning in the DAG, we have*

$$\sum_{e \in S} P(e) = O(n^2).$$

Proof (Sketch). Each dipartitioning can be obtained by starting from the trivial dipartitioning where V_1 only contains the source nodes of the DAG, and then iteratively adding nodes one by one to this initial V_1. The number of outgoing edges from this initial V_1 (the set of source nodes) is upper bounded by $|E| = O(n^2)$. According to Lemma 4, the number of switches (and hence the potential) on each edge of the dicut is at most constant, so the sum of potential in this initial dicut is also $O(n^2)$.

Now consider the process of iteratively adding nodes to this initial V_1 to obtain a specific dipartitioning. Whenever we add a new node v to V_1, the incoming edges of v are removed from the dicut, and the outgoing edges of v are added to the dicut. According to Lemma 6, the potential on the outgoing edges of v is at most as much as the potential on the incoming edges, so the sum of potential can not increase in any of these steps. Therefore, when arriving at the final V_1, the sum of potential on the cut edges is still at most $O(n^2)$. ◀

5.3 Upper bounding switches

Finally, we present our main lemma that uses the previous upper bound on potential in order to upper bound the number of switches in the CPS.

▶ **Lemma 8.** *Given a CPS and an integer $a \in \{1, ..., n\}$, let $A = \{v \in V \mid a \leq deg(v) < 2a\}$. For the total number of switches $s(A) = \sum_{v \in A} s(v)$, we have*

$$s(A) = O\left(n^{1+f(\lambda)} \cdot a^{-f(\lambda)}\right).$$

Proof (Sketch). If the input edges of the nodes in A would form the dicut of a dipartitioning, then we could directly use Lemma 7 to upper bound the number of switches in A through the potential of the input edges. However, the nodes of A might be scattered arbitrarily in the DAG, and if there is a directed path from one node in A to another, then the "same" potential might be used to switch more than one node in A. Thus we cannot apply Lemma 7 directly. Instead, our proof consists of two parts.

1. First, we define so-called responsibilities for the nodes in A. Given a node $v_0 \in A$, the idea is to devise two different functions: (i) a function $\Delta c(e)$, defined on each edge e which is contained in any directed path starting from v_0, and (ii) a function $\Delta s(v)$, which is defined on any node v that is reachable from v_0 on a directed path. Intuitively, we will consider the conflicts $\Delta c(e)$ and the switches $\Delta s(v)$ to be those that are indirectly "the effects of the switches of v_0". More specifically, Δc and Δs are chosen such that if they are removed (subtracted from the CPS), then v_0 has no output edges in the DAG anymore, and the resulting assignment $s'(v) = s(v) - \Delta s(v)$ and $c'(e) = c(e) - \Delta c(e)$ still remains a valid CPS. Hence the subtraction results in a CPS where v_0 has no directed path to other nodes in A anymore. This shows that we can keep on executing this step for each $v_0 \in A$ until no two nodes in A are connected by a directed path, at which point we can apply Lemma 7 to the resulting graph.

Whenever we process such a node $v_0 \in A$, we define the *responsibility* of v_0 as $R(v_0) := s(v_0) + \sum \Delta s(v)$, where the sum is understood over all the nodes $v \in A$ that are reachable from v_0. The main idea is that we "reassign" these switches to v_0 from other nodes in A. This method is essentially a redistribution of switches in the CPS, so we have $\sum_{v \in A} s(v) = \sum_{v \in A} R(v)$ altogether.

Furthermore, our definition of $\Delta s(v)$ will ensure that $R(v_0) = O(1) \cdot s(v_0)$. Intuitively, this can be explained as follows. Recall that with Rule II, the ratio of output to input conflicts is always upper bounded by a constant factor (below 1) at every node, since switching always wastes a specific proportion of conflicts. Hence, over any path starting from v_0, the number of outputs that can be attributed to v_0 forms a geometric series. As the ratio of the geometric series is below 1, the total amount of conflicts caused by v_0 this way is still within the magnitude of the input conflicts of v_0. Since each node in A has similar degree (and thus requires similar number of input conflicts for one switching), these conflicts can only switch nodes in A approximately the same number of times as v_0 can be switched by its own inputs. A detailed discussion of this responsibility technique is available in the full version of the paper.

2. For the second part of the proof, we show the claim in this modified CPS with no directed path between nodes in A. This implies that there exists a dipartitioning where the nodes of A are in V_2, but all their input nodes are in V_1. This means that all the input edges of each node in A are included in the dicut S of the partitioning.

Consider a node $v \in A$. Due to condition 1R, v has at least $\lambda \cdot \deg(v) \cdot s(v)$ input conflicts. Even if these are distributed equally on all incident edges of v (this is the case that amounts to the lowest total potential, since P is convex), this requires a total input potential of

$$\deg(v) \cdot P(\lambda \cdot s(v)) = \deg(v) \cdot s(v)^{1/f(\lambda)} \cdot \lambda^{1/f(\lambda)}$$

at least. Recall that Lemma 7 shows that the total potential on all edges in S is $O(n^2)$. Our task is hence to find an upper bound on $\sum_{v\in A} s(v)$, subject to

$$\sum_{v\in A} \deg(v) \cdot s(v)^{1/f(\lambda)} \cdot \lambda^{1/f(\lambda)} = O(n^2).$$

Since the last factor on the left side is a constant, we can simply remove it and include it in the $O(n^2)$ term. Furthermore, the degree of each node in A is at least a, so by lower bounding each degree by a, we get

$$\sum_{v\in A} s(v)^{1/f(\lambda)} = O(n^2) \cdot \frac{1}{a}.$$

Given this upper bound on $\sum_{v\in A} P(s(v))$, since the function P is convex, the sum of switches $\sum_{v\in A} s(v)$ is maximal when each node in A switches the same amount of times (i.e. there is an s such that $s(v) = s$ for every $v \in A$), giving

$$|A| \cdot s^{1/f(\lambda)} = O(n^2) \cdot \frac{1}{a}.$$

With this upper bound, $|A| \cdot s$ is maximal if $|A|$ is as large as possible and s as small as possible (again because P grows faster than linearly). Clearly $|A| \leq n$, so assuming $|A| = n$, we get

$$s^{1/f(\lambda)} = O(n) \cdot \frac{1}{a},$$

which means that

$$s = O(n^{f(\lambda)}) \cdot a^{-f(\lambda)},$$

and thus for the total number of switches in A, we get

$$|A| \cdot s = O(n^{1+f(\lambda)}) \cdot a^{-f(\lambda)}.$$ ◀

It only remains to sum up this bound for the appropriate intervals to obtain our final bound. Let us consider the intervals $[1, 2)$, $[2, 4)$, $[4, 8)$, ..., i.e. $a = 2^k$ for each factor of 2 up to n, which is a disjoint partitioning of the possible degrees. Note that for these specific values of a, the sum $\sum_{k=0}^{\infty}(2^k)^{-f(\lambda)}$ converges to a constant according to the ratio test. In other words, the sum is dominated by the number of switches of the lowest (constant) degree nodes, and hence, the total number of switches in the graph can be upper bounded by $O(1) \cdot n^{1+f(\lambda)}$.

Recall that since we work with Relaxed CPSs, we lose an ϵ in the exponent of this upper bound when we carry the result over to an original CPS.

▶ **Theorem 9.** *In any CPS with parameter λ, we have $\sum_{v\in V} s(v) = O(n^{1+f(\lambda)+\epsilon})$ for any $\epsilon > 0$.*

Since we have established that every majority/minority process provides a CPS, the upper bound on their stabilization time also follows.

▶ **Corollary 10.** *Under Rule II with any $\lambda \in (0, 1)$, every majority/minority process stabilizes in time $O(n^{1+f(\lambda)+\epsilon})$ for any $\epsilon > 0$.*

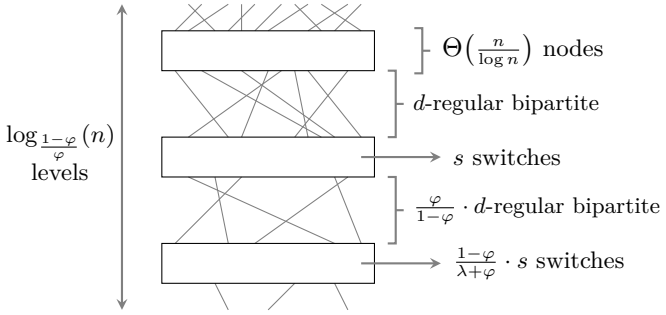

Figure 2 Consecutive levels of the lower bound construction.

6 Lower bound construction

Having established the most efficient way to relay conflicts, the high-level design of the matching lower bound construction is rather straightforward, following the level-based idea described in Section 4.

Given λ, we first determine the optimal output rate $\varphi = \varphi^*(\lambda)$. We then create a construction consisting of distinct levels, where each level has the same size, and each consists of a set of nodes that have the same degree. Since the degree should decrease by a factor of $\frac{\varphi}{1-\varphi}$ in each new level from top to bottom, we can add $L = \log_{\frac{1-\varphi}{\varphi}}(n)$ such levels to the graph. If each of these level has $\Theta(\frac{n}{\log n})$ nodes, then with the appropriate choice of constants, the total number of nodes is below n.

Each node in the construction is only connected to other nodes on the levels immediately above or below its own. All conflicts are propagated down in the graph, from upper to lower levels, so the upper neighbors of a node are always used as inputs, while the lower neighbors are always used as outputs. For the optimal propagation of conflicts, each node v must have the optimal input-output rate, i.e. an up-degree of $(1-\varphi) \cdot \deg(v)$ and a down-degree of $\varphi \cdot \deg(v)$. Thus each consecutive level pair forms a regular bipartite graph, with $\frac{\varphi}{1-\varphi}$ of the degree of the level pair above. The construction is illustrated in Figure 2.

Our parameters λ and φ also determine that the number of switches should increase by a factor $\frac{1-\varphi}{\lambda+\varphi}$ on each new level. If we can always increase the switches at this rate, then each node on the lowermost level will switch

$$\left(\frac{1-\varphi}{\lambda+\varphi}\right)^{\log_{\frac{1-\varphi}{\varphi}}(n)} = n^{\frac{\log\left(\frac{1-\varphi}{\lambda+\varphi}\right)}{\log\left(\frac{1-\varphi}{\varphi}\right)}} = n^{f(\lambda)},$$

times, where the last equation holds because we are using $\varphi = \varphi^*(\lambda)$. Since there are $\widetilde{\Theta}(n)$ nodes on the lowermost level, the switches in this level already amount to a total of $\widetilde{\Theta}(n^{1+f(\lambda)})$, matching the upper bound.

However, note that when $\varphi^*(\lambda)$ or $\frac{1-\varphi}{\lambda+\varphi}$ is irrational, we can only use close enough rational approximations of these values. This comes at the cost of losing a small ϵ in the exponent.

▶ **Theorem 11.** *Under Rule II with a wide range of λ values, there is a graph construction and initial coloring where majority/minority processes stabilize in time $\Omega(n^{1+f(\lambda)-\epsilon})$ for any $\epsilon > 0$.*

This level-based structure describes the general idea behind our lower bound construction. However, the main challenge of the construction is in fact designing the connection between subsequent levels. In particular, this connection has to make sure that conflicts are indeed always relayed optimally, i.e. no potential is wasted between any two levels.

Recall from the proof of Lemma 6 that this is only possible if between any two consecutive switches of a node v, it is exactly a $\frac{\lambda+\varphi}{1-\varphi}$ fraction of v's upper neighbors that switch. Moreover, these switching $\frac{\lambda+\varphi}{1-\varphi} \cdot \deg(v)$ upper neighbors always have to be of the right color, i.e. they need to switch to the opposite of v's current color in case of majority processes, and to the same color in case of minority processes. Since the upper neighbors of v are in the same level, we also have to ensure that throughout the entire process, each upper neighbor switches the same number of times altogether.

These conditions impose heavy restrictions on the possible ways to connect two subsequent levels. If the conditions hold for a node v (i.e. the sequence of switches of v's upper neighbors can be split into $\frac{\lambda+\varphi}{1-\varphi} \cdot \deg(v)$-size consecutive appropriate-colored subsets, in an altogether balanced way), then we say that v's upper neighbors follow a valid *control sequence*.

On the other hand, in order to argue about levels in general, we want each level to behave in a similar way. The easiest way to achieve this is to have a one-to-one correspondence between the nodes of different levels, and ensure that each level repeats the same sequence of steps periodically, but in a different pace. That is, we want to connect the levels in such a way that when a level exhibits a specific pattern of switches, then this allows the nodes of the next level to replicate the exact same pattern of switches, but more times.

Thus the key task in our lower bound constructions is to develop a so-called *control gadget*, which is essentially a bipartite graph that fulfills these two requirements: it admits a scheduling of switches such that (i) the upper neighborhood of each lower node follows a valid control sequence, and (ii) while the upper level executes a sequence s times, the lower level executes the same sequence $\frac{1-\varphi}{\lambda+\varphi} \cdot s$ times. Given such a control gadget, we can connect the subsequent level pairs of our construction using this gadgets. This allows us to indeed increase the number of switches by a $\frac{1-\varphi}{\lambda+\varphi}$ factor in each new level, resulting in a total of $\widetilde{\Theta}(n^{1+f(\lambda)})$ switches as described above.

However, developing a control gadget is a difficult combinatorial task in general: it depends on many factors including divisibility questions, and whether our parameters can be expressed as a fraction of small integers. A detailed discussion of control gadget design and the λ values covered by Theorem 11 is available in the full version of the paper. In particular, we present a method which allows us to develop a control gadget for every small λ value below a threshold of approximately 0.476 (more specifically, as long as $\frac{\lambda+\varphi}{1-\varphi} \leq \frac{3}{5}$). The same technique also provides a control gadget for some larger λ values above the threshold, but only when the corresponding switch increase ratio $\frac{1-\varphi}{\lambda+\varphi}$ can be expressed as a fraction of relatively small integers. Furthermore, the full version also describes a simpler solution technique to the control gadget problem; this leaves a slightly larger gap to the upper bound, but it works for any λ without much difficulty.

References

1. Victor Amelkin, Francesco Bullo, and Ambuj K Singh. Polar opinion dynamics in social networks. *IEEE Transactions on Automatic Control*, 62(11):5650–5665, 2017.
2. Vincenzo Auletta, Ioannis Caragiannis, Diodato Ferraioli, Clemente Galdi, and Giuseppe Persiano. Generalized discrete preference games. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 53–59. AAAI Press, 2016.
3. Cristina Bazgan, Zsolt Tuza, and Daniel Vanderpooten. Complexity and approximation of satisfactory partition problems. In *International Computing and Combinatorics Conference*, pages 829–838. Springer, 2005.
4. Cristina Bazgan, Zsolt Tuza, and Daniel Vanderpooten. The satisfactory partition problem. *Discrete applied mathematics*, 154(8):1236–1245, 2006.

5 Cristina Bazgan, Zsolt Tuza, and Daniel Vanderpooten. Satisfactory graph partition, variants, and generalizations. *European Journal of Operational Research*, 206(2):271–280, 2010.

6 Olivier Bodini, Thomas Fernique, and Damien Regnault. Crystallization by stochastic flips. In *Journal of Physics: Conference Series*, volume 226, page 012022. IOP Publishing, 2010.

7 Olivier Bodini, Thomas Fernique, and Damien Regnault. Stochastic flips on two-letter words. In *2010 Proceedings of the Seventh Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 48–55. SIAM, 2010.

8 Zhigang Cao and Xiaoguang Yang. The fashion game: Network extension of matching pennies. *Theoretical Computer Science*, 540:169–181, 2014.

9 Luca Cardelli and Attila Csikász-Nagy. The cell cycle switch computes approximate majority. *Scientific reports*, 2:656, 2012.

10 Ning Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.

11 Jacques Demongeot, Julio Aracena, Florence Thuderoz, Thierry-Pascal Baum, and Olivier Cohen. Genetic regulation networks: circuits, regulons and attractors. *Comptes Rendus Biologies*, 326(2):171–188, 2003.

12 MohammadAmin Fazli, Mohammad Ghodsi, Jafar Habibi, Pooya Jalaly, Vahab Mirrokni, and Sina Sadeghian. On non-progressive spread of influence through social networks. *Theoretical Computer Science*, 550:36–50, 2014.

13 Silvio Frischknecht, Barbara Keller, and Roger Wattenhofer. Convergence in (social) influence networks. In *International Symposium on Distributed Computing*, pages 433–446. Springer, 2013.

14 Bernd Gärtner and Ahad N Zehmakan. Color war: Cellular automata with majority-rule. In *International Conference on Language and Automata Theory and Applications*, pages 393–404. Springer, 2017.

15 Bernd Gärtner and Ahad N Zehmakan. Majority model on random regular graphs. In *Latin American Symposium on Theoretical Informatics*, pages 572–583. Springer, 2018.

16 Michael U Gerber and Daniel Kobler. Algorithmic approach to the satisfactory graph partitioning problem. *European Journal of Operational Research*, 125(2):283–291, 2000.

17 Eric Goles and Jorge Olivos. Periodic behaviour of generalized threshold functions. *Discrete Mathematics*, 30(2):187–189, 1980.

18 Mark Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.

19 Sandra M Hedetniemi, Stephen T Hedetniemi, KE Kennedy, and Alice A Mcrae. Self-stabilizing algorithms for unfriendly partitions into two disjoint dominating sets. *Parallel Processing Letters*, 23(01):1350001, 2013.

20 Clemens Jeger and Ahad N Zehmakan. Dynamic monopolies in reversible bootstrap percolation. *arXiv preprint arXiv:1805.07392*, 2018.

21 Dominik Kaaser, Frederik Mallmann-Trenn, and Emanuele Natale. On the voting time of the deterministic majority process. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, 2016.

22 Barbara Keller, David Peleg, and Roger Wattenhofer. How even tiny influence can have a big impact! In *International Conference on Fun with Algorithms*, pages 252–263. Springer, 2014.

23 David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

24 Jeremy Kun, Brian Powers, and Lev Reyzin. Anti-coordination games and stable graph colorings. In *International Symposium on Algorithmic Game Theory*, pages 122–133. Springer, 2013.

25 Yuezhou Lv and Thomas Moscibroda. Local information in influence networks. In *International Symposium on Distributed Computing*, pages 292–308. Springer, 2015.

26 Ahad N Zehmakan. Opinion forming in erdös-rényi random graph and expanders. In *29th International Symposium on Algorithms and Computations*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken . . . , 2018.

27 Pál András Papp and Roger Wattenhofer. Stabilization Time in Minority Processes. In *30th International Symposium on Algorithms and Computation (ISAAC 2019)*, volume 149 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:19, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

28 Pál András Papp and Roger Wattenhofer. Stabilization Time in Weighted Minority Processes. In *36th International Symposium on Theoretical Aspects of Computer Science (STACS 2019)*, volume 126 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

29 David Peleg. Local majorities, coalitions and monopolies in graphs: a review. *Theoretical Computer Science*, 282(2):231–257, 2002.

30 Damien Regnault, Nicolas Schabanel, and Éric Thierry. Progresses in the analysis of stochastic 2d cellular automata: A study of asynchronous 2d minority. In Luděk Kučera and Antonín Kučera, editors, *Mathematical Foundations of Computer Science 2007*, pages 320–332. Springer Berlin Heidelberg, 2007.

31 Damien Regnault, Nicolas Schabanel, and Éric Thierry. On the analysis of "simple" 2d stochastic cellular automata. In *International Conference on Language and Automata Theory and Applications*, pages 452–463. Springer, 2008.

32 Jean-Baptiste Rouquier, Damien Regnault, and Éric Thierry. Stochastic minority on graphs. *Theoretical Computer Science*, 412(30):3947–3963, 2011.

33 Khurram H Shafique and Ronald D Dutton. On satisfactory partitioning of graphs. *Congressus Numerantium*, pages 183–194, 2002.

34 Saharon Shelah and Eric C Milner. Graphs with no unfriendly partitions. *A tribute to Paul Erdös*, pages 373–384, 1990.

35 Ariel Webster, Bruce Kapron, and Valerie King. Stability of certainty and opinion in influence networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 1309–1320. IEEE, 2016.

36 Peter Winkler. Puzzled: Delightful graph theory. *Commun. ACM*, 51(8):104–104, August 2008.

37 Ahad N Zehmakan. Target set in threshold models. *Acta Mathematica Universitatis Comenianae*, 88(3), 2019.

38 Ahad N Zehmakan. Tight bounds on the minimum size of a dynamic monopoly. In *International Conference on Language and Automata Theory and Applications*, pages 381–393. Springer, 2019.