Rewiring What-to-Watch-Next Recommendations to Reduce Radicalization Pathways

Fabbri, Francesco

Association for Computing Machinery
2022-04

Fabbri , F , Wang , Y , Bonchi , F , Castillo , C & Mathioudakis , M 2022 , Rewiring What-to-Watch-Next Recommendations to Reduce Radicalization Pathways . in WWW'22: Proceedings of the ACM Web Conference 2022 . Association for Computing Machinery , pp. þÿ2719 2728 , The ACM Web Conference , Lyon , France , 25/04/2022 . h

http://hdl.handle.net/10138/343200
https://doi.org/10.1145/3485447.3512143

Rewiring *What-to-Watch-Next* Recommendations to Reduce Radicalization Pathways

Francesco Fabbri
Universitat Pompeu Fabra & Eurecat
Barcelona, Spain
francesco.fabbri@eurecat.org

Yanhao Wang
East China Normal University
Shanghai, China
yhwang@dase.ecnu.edu.cn

Francesco Bonchi
ISI Foundation, Turin, Italy
Eurecat, Barcelona, Spain
francesco.bonchi@isi.it

Carlos Castillo
ICREA & Universitat Pompeu Fabra
Barcelona, Spain
chato@icrea.cat

Michael Mathioudakis
University of Helsinki
Helsinki, Finland
michael.mathioudakis@helsinki.fi

ABSTRACT

Recommender systems typically suggest to users content similar to what they consumed in the past. If a user happens to be exposed to strongly polarized content, she might subsequently receive recommendations which may steer her towards more and more radicalized content, eventually being trapped in what we call a "*radicalization pathway*". In this paper, we study the problem of mitigating radicalization pathways using a graph-based approach. Specifically, we model the set of recommendations of a "*what-to-watch-next*" recommender as a d-regular directed graph where nodes correspond to content items, links to recommendations, and paths to possible user sessions.

We measure the "*segregation*" score of a node representing radicalized content as the expected length of a random walk from that node to any node representing non-radicalized content. High segregation scores are associated to larger chances to get users trapped in radicalization pathways. Hence, we define the problem of reducing the prevalence of radicalization pathways by selecting a small number of edges to "*rewire*", so to minimize the maximum of segregation scores among all radicalized nodes, while maintaining the relevance of the recommendations.

We prove that the problem of finding the optimal set of recommendations to rewire is NP-hard and NP-hard to approximate within any factor. Therefore, we turn our attention to heuristics, and propose an efficient yet effective greedy algorithm based on the absorbing random walk theory. Our experiments on real-world datasets in the context of video and news recommendations confirm the effectiveness of our proposal.

CCS CONCEPTS

• **Information systems → Web applications**; • **Theory of computation → Random walks and Markov chains**.

KEYWORDS

recommender systems, random walks, radicalization, polarization, extremist content, filter bubbles

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France.

ACM Reference Format:
Francesco Fabbri, Yanhao Wang, Francesco Bonchi, Carlos Castillo, and Michael Mathioudakis. 2022. Rewiring *What-to-Watch-Next* Recommendations to Reduce Radicalization Pathways. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3485447.3512143

1 INTRODUCTION

"*What-to-watch-next*" (W2W) recommenders are key features of video sharing platforms [55], as they sustain user engagement, thus increasing content views and driving advertisement and monetization. However, recent studies have raised serious concerns about the potential role played by W2W recommenders, specifically in driving users towards undesired or polarizing content [29]. Specifically, radicalized communities[1] on social networks and content sharing platforms have been recognized as keys in the consumption of news and in building opinions around politics and related subjects [30, 48, 53]. Recent work highlights the role of recommender systems, which may steer users towards radicalized content, eventually building "*radicalization pathways*" [30, 47] (i.e., a user might be further driven towards radicalized content even when this was not her initial intent). In this paper, we study the problem of reducing the prevalence of radicalization pathways in W2W recommenders while maintaining the relevance of recommendations.

Formally, we model a W2W recommender system as a directed labeled graph where nodes correspond to videos (or other types of content) and directed edges represent recommendation links from one node to another[2]. In this scenario, each video is accompanied by the same number d of recommendation links, and thus every node in the graph has the same out-degree d. Moreover, each node has a binary label such as "harmful" (e.g., radicalized) or "neutral" (e.g., non-radicalized). The browsing activity of a user through the W2W recommendations is modeled as a *random walk* on the graph: after visiting a node (e.g., watching a video), the user moves to one of the d recommended videos with a probability that depends on its visibility or ranking in the recommendation list. In this setting, for

[1] From McCauley and Moskalenko [35]: "*Functionally, political radicalization is increased preparation for and commitment to intergroup conflict. Descriptively, radicalization means change in beliefs, feelings, and behaviors in directions that increasingly justify intergroup violence and demand sacrifice in defense of the ingroup.*"
[2] For ease of presentation, we focus on video sharing platforms. We note that the same type of recommendations occurs in many other contexts such as, for instance, news feeding platforms as shown in our experiments (see Section 5).

each harmful node v, we measure the expected number of consecutive harmful nodes visited in a random walk before reaching any neutral node. We call this measure the "*segregation*" score of node v: intuitively, it quantifies how easy it is to get "stuck" in radicalization pathways starting from a given node. Our goal is to reduce the segregation of the graph while guaranteeing that the quality of recommendations is maintained, where the quality is measured by the *normalized discount cumulative gain* [4, 24] (nDCG) of each node. An important challenge is that the underlying recommendation graph has intrinsically some level of homophily because, given that the W2W seeks to recommend relevant videos, it is likely to link harmful nodes to other harmful nodes.

We formulate the problem of reducing the segregation of the graph as selecting k rewiring operations on edges (corresponding to modifications in the lists of recommended videos for some nodes) so as to minimize the maximum of segregation scores among all harmful nodes, while maintaining recommendation quality measured by nDCG above a given threshold for all nodes. We prove that our k-Rewiring problem is NP-hard and NP-hard to approximate within any factor. We therefore turn our attention to design efficient and effective heuristics. Our proposed algorithm is based on the *absorbing random walk theory* [34], thanks to which we can efficiently compute the segregation score of each node and update it after every rewiring operation. Specifically, our method finds a set of k rewiring operations by greedily choosing the optimal rewiring for the special case of $k = 1$ – i.e., the 1-Rewiring problem, then updates the segregation score of each node. We further design a sorting and pruning strategy to avoid unnecessary attempts and thus improve the efficiency for searching the optimal rewiring. Though the worst-case time complexity of our algorithm is quadratic with respect to the number of nodes n, it exhibits much better performance (nearly linear w.r.t. n) in practice.

Finally, we present experiments on two real-world datasets: one in the context of video sharing and the other in the context of news feeds. We compare our proposed algorithm against several baselines, including an algorithm for suggesting new edges to reduce radicalization in Web graphs. The results show that our algorithm outperforms existing solutions in mitigating radicalization pathways in recommendation graphs.

In the rest of this paper, we first review the literature relevant to our work in Section 2. Then, we introduce the background and formally define our problem in Section 3. Our proposed algorithms are presented in Section 4. The experimental setup and results are shown in Section 5. Finally, we conclude this paper and discuss possible future directions in Section 6.

2 RELATED WORK

A great deal of research has been recently published about the potential created by unprecedented opportunities to access information on the Web and social media. These risks include the spread of misinformation [1, 50], the presence of bots [16], the abundance of offensive hate speech [33, 37], the availability of inappropriate videos targeting children [40], the increase in controversy [18] and polarization [20], and the creation of radicalization pathways [47]. Consequently, a substantial research effort has been devoted to model, detect, quantify, reduce, and/or block such negative phenomena. Due to space limitations, we only discuss the existing studies that are the most relevant to our work here – in particular, algorithmic approaches to optimizing graph structures for achieving the aforementioned goals [7, 8, 19, 21, 23, 25–28, 38, 49, 51, 54].

A line of research deals with limiting the spread of undesirable content in a social network via edge manipulation [25–28, 49, 51, 54]. In these studies, the graph being manipulated is a network of users where the edges represent connections such as friendship or interactions among users. In contrast, we consider a graph of content items (e.g., videos or news), where the edges represent recommendation links. Moreover, these algorithmic methods are primarily based on information propagation models, while our work is based on random walks.

Another line of work aims at reducing controversy, disagreement, and polarization by edge manipulation in a social network, exposing users to others with different views [7, 8, 19, 21, 23, 38]. Garimella et al. [19] introduce the *controversy score* of a graph based on random walks and propose an efficient algorithm to minimize it by edge addition. Musco et al. [38] introduce the *Polarization-Disagreement index* of a graph based on Friedkin-Johnsen dynamics and propose a network-design approach to find a set of "best" edges that minimize this index. Chen et al. [7] define the *worst-case conflict risk* and *average-case conflict risk* of a graph, also based on Friedkin-Johnsen dynamics, and propose algorithms to locally edit the graphs for reducing both measures. Chitra and Musco [8] analyze the impact of "filter bubbles" in social network polarization and how to mitigate them by graph modification. Interian et al. [23] define a polarization reduction problem by adding edges between users from different groups and propose integer programming-based methods to solve it. Another related line of work proposes to model and mitigate the disparate exposure generated by people recommenders (e.g. who-to-follow link predictions) in presence of effects like homophily and polarization [9, 14, 15, 45]. These studies also deal with networks of users, while in our case we consider a network of items.

The work probably most related to ours is the one by Haddadan et al. [21], which considers a graph of items (e.g., Web pages with hyperlinks) and defines the structural bias of a node as the difficulty/effort needed to reach nodes of a different opinion. They, then propose an efficient approximation algorithm to reduce the structural bias by edge insertions. There are three main differences between this and our work. First, two-directional edge manipulations (from class A to B and also from B to A) are considered by Haddadan et al. [21], but one-directional edge manipulations (from harmful to neutral nodes only) are considered in our work. Second, they consider inserting new links on a node, which better fits the case of Web pages, but we consider rewiring existing edges, which better fits the case of W2W recommenders. Third, they define the structural bias of the graph as the sum of the bubble radii of all nodes, while we define the segregation of the graph as the worst-case segregation score among all harmful nodes. We compare our proposed algorithm with theirs in our experiments.

A recent line of work introduces the notion of *reachability* in recommender systems [11, 13]. Instead of rewiring the links, they focus on making allowable modifications in the user's rating history

to avoid unintended consequences such as filter bubbles and radicalization. However, as the problem formulation is different from ours, their proposed methods are not applicable to our problem.

Finally, there are many studies on modifying various graph characteristics, such as shortest paths [42, 43], centrality [3, 10, 12, 36, 44, 52], opinion dynamics [2, 5], and so on [6, 31, 41, 56], by edge manipulation. We can draw insights from these methods but cannot directly apply them to our problem.

3 PRELIMINARIES

Let us consider a set V of n items and a matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, where each entry $s_{uv} \in [0, 1]$ at position (u, v) denotes the relevance score of an item v given that a user has browsed an item u. This expresses the likelihood that a user who has just watched u would be interested in watching v. Typically, a recommender system selects the d most relevant items to compose the recommendation list $\Gamma^+(u)$ of u, where the number of recommendations d is a design constraint (e.g., given by the size of the app window). We assume that the system selects the top-d items v w.r.t. s_{uv} and that their relevance score uniquely determines the ranking of the d items in $\Gamma^+(u)$. For each $v \in \Gamma^+(u)$, we use $i_u(v)$ to denote its ranking in $\Gamma^+(u)$. After a user has seen u, she/he will find the next item to see from $\Gamma^+(u)$, and the probability p_{uv} of selecting $v \in \Gamma^+(u)$ depends on the ranking $i_u(v)$ of v in $\Gamma^+(u)$. More formally, $p_{uv} = f(i_u(v))$, where f is a non-increasing function that maps from $i_u(v)$ to p_{uv} with $\sum_{v \in \Gamma^+(u)} p_{uv} = 1$.

This setting can be modeled as a directed probabilistic d-regular graph $G = (V, E, \mathbf{M})$, where the node set V corresponds to the set of all n items, the edge set E comprises $n \cdot d$ edges where each node $u \in V$ has d out-edges connected to the nodes in $\Gamma^+(u)$, and $\mathbf{M}$ is an $n \times n$ transition matrix with a value of p_{uv} for each $(u, v) \in E$ and 0 otherwise. A user's browsing session is thus modeled as a random walk on G starting from an arbitrary node in V with transition probability p_{uv} for each $(u, v) \in E$.

We further consider that the items in V are divided into two disjoint subsets V_n and V_h (i.e., $V_n \cap V_h = \emptyset$ and $V_n \cup V_h = V$) corresponding to "neutral" (e.g., not-radicalized) and "harmful" (e.g., radicalized) nodes, respectively.

The risk we want to mitigate is having users stuck in a long sequence of harmful nodes while performing a random walk. In order to quantify this phenomenon we define the measure of *segregation*. Given a set $S \subset V$ of nodes and a node $u \in V \setminus S$, we use a random variable $T_u(S)$ to indicate the first instant when a random walk starting from u reaches (or "hits") any node in S. We define $\mathbb{E}_G[T_u(S)]$ as the *hitting length* of u w.r.t. S, where the expectation is over the space of all possible random walks on G starting from u. In our case, we define the segregation score z_u of node $u \in V_h$ by its expected hitting length $\mathbb{E}_G[T_u(V_n)]$ w.r.t. V_n. The segregation $Z(G)$ of graph G is defined by the maximum of segregation scores among all nodes in V_h – i.e., $Z(G) = \max_{u \in V_h} z_u$. In the following, we omit the argument G from $Z(G)$ when it is clear from the context.

Our main problem in this paper is to mitigate the effect of segregation by modifying the structure of G. Specifically, we aim to find a set O of rewiring operations on G, each of which removes an existing edge $(u, v) \in E$ and inserts a new one $(u, w) \notin E$ instead, such that $Z(G^O)$ is minimized, where G^O is the new graph after performing O on G. For simplicity, we require that $u, v \in V_h$, $w \in V_n$, and $p_{uv} = p_{uw}$. In other words, each rewiring operation changes the recommendation list $\Gamma^+(u)$ of u by replacing one (harmful) item $v \in \Gamma^+(u)$ with another (neutral) item $w \notin \Gamma^+(u)$ and keeping the ranking $i_u(w)$ of w the same as the ranking $i_u(v)$ of v in $\Gamma^+(u)$.

Another goal, which is often conflicting, is to preserve the relevance of recommendations after performing the rewiring operations. Besides requiring only a predefined number k of rewirings, we also consider an additional constraint on the loss in the quality of the recommendations. For this purpose we adopt the well-known *normalized discounted cumulative gain* (nDCG) [4, 24] to evaluate the loss in the quality. Formally, the *discounted cumulative gain* (DCG) of a recommendation list $\Gamma^+(u)$ is defined as:

$$\texttt{DCG}(\Gamma^+(u)) = \sum_{v \in \Gamma^+(u)} \frac{s_{uv}}{1 + \log_2(1 + i_u(v))}$$

Then, we define the quality loss of $\Gamma^+(u)$ after rewiring operations by nDCG as follows:

$$L(\Gamma^+(u)) = \texttt{nDCG}(\Gamma^+(u)) = \frac{\texttt{DCG}(\Gamma^+(u))}{\texttt{DCG}(\Gamma_0^+(u))} \tag{1}$$

where $\Gamma_0^+(u)$ is the original (ideal) recommendation list where all the top-d items that are the most relevant to u are included.

Let $o = (u, v, w)$ be a rewiring operation that deletes (u, v) while adding (u, w) and O be a set of rewiring operations. For ease of presentation, we define a function $\Delta(O) \triangleq Z(G) - Z(G^O)$ to denote the decrease in the segregation after performing the rewiring operations in O and updating G to G^O. We are now ready to formally define the main problem studied in this paper.

Problem 1 (k-Rewiring). *Given a directed probabilistic graph $G = (V, E, \mathbf{M})$, a positive integer $k \in \mathbb{Z}^+$, and a threshold $\tau \in (0, 1)$, find a set O of k rewiring operations that maximizes $\Delta(O)$, under the constraint that $L(\Gamma^+(u)) \geq \tau$ for each node $u \in V$.*

The hardness of the k-Rewiring problem is analyzed in the following theorem.

Theorem 3.1. *The k-Rewiring problem is NP-hard and NP-hard to approximate within any factor.*

We show the NP-hardness of the k-Rewiring problem by reducing from the VertexCover problem. Furthermore, we show that finding an α-approximate solution of the k-Rewiring problem for any factor $\alpha > 0$ is at least as hard as finding the minimum vertex cover of a graph. Therefore, the k-Rewiring problem is NP-hard to approximate within any factor. The proof of Theorem 3.1 can be found in Appendix A.

3.1 Absorbing Random Walk

We now provide notions from the absorbing random walk theory [34] on which our algorithms are built.

The k-Rewiring problem asks to minimize *segregation*, which is defined as the maximum hitting length from any harmful node to neutral nodes. Specifically, in the context of k-Rewiring for the given probabilistic directed graph $G = (V, E, \mathbf{M})$, we equivalently consider a modified transition matrix $\mathbf{M}$ as follows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{hh} & \mathbf{M}_{hn} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

In the matrix $\mathbf{M}$ above, each *neutral* node has been set to be *absorbing*, i.e., its transition probability to itself is set to $p_{ii} = 1$ and 0 to other nodes (see the bottom row of $\mathbf{M}$). Intuitively, no random walk passing through an absorbing node can move away from it [34]. For each *harmful* node, its transition probabilities remain unmodified (see the top row of $\mathbf{M}$) and thus the node remains *transient* (i.e., *non-absorbing*).

The fundamental matrix $\mathbf{F}$ can be computed from the sub-matrix $\mathbf{M}_{hh}$ as follows [34]:

$$\mathbf{F} = (\mathbf{I} - \mathbf{M}_{hh})^{-1}$$

where the entry f_{uv} represents the expected total number of times that the random walk visits node v having started from node u. Then, the expected length of a random walk that starts from any node and stops when it gets absorbed is given by vector $\mathbf{z}$:

$$\mathbf{z} = \begin{bmatrix} (\mathbf{I} - \mathbf{M}_{hh})^{-1} \\ \mathbf{0} \end{bmatrix} \mathbf{1} \tag{2}$$

where $\mathbf{1}$ is an n-dimensional vector of all 1's. Here, the i-th entry z_i of vector $\mathbf{z}$ represents the expected number of random walk steps before being absorbed by any absorbing node, assuming that the random walk starts from the i-th node.

Given that the absorbing and transient nodes are set to correspond exactly to the neutral and harmful nodes, respectively, the values of $\mathbf{z}$ correspond exactly to the expected hitting length as used to define segregation. Hence, the k-Rewiring problem asks to choose a set of k rewiring operations to minimize the maximum entry $Z = \max_{1 \le i \le n} z_i$ of vector $\mathbf{z}$.

4 ALGORITHMS

Since k-Rewiring is NP-hard to approximate within any factor, we propose an efficient heuristic. The heuristic is motivated by the following observation: despite the NP-hardness of k-Rewiring, its special case when $k = 1$, which we call 1-Rewiring, is solvable in polynomial time. Given an optimal 1-Rewiring algorithm, k-Rewiring can be addressed by running it k times.

We begin our presentation of algorithms by showing a brute-force algorithm for finding the optimal solution of 1-Rewiring (Section 4.1), as well as a way to speed it up via incremental updates (Section 4.2). Subsequently, we propose our optimal 1-Rewiring algorithm that improves the efficiency of the brute-force algorithm by faster rewiring search (Section 4.3). Finally, we present how our 1-Rewiring algorithm is used for k-Rewiring (Section 4.4).

4.1 Brute-Force Algorithm for 1-Rewiring

Given a graph G and a rewiring operation o, we use $\Delta(o)$ to denote the decrease in Z after performing o on G. We present a brute-force algorithm to find the rewiring operation o^* that maximizes $\Delta(o)$. The algorithm has three steps: (1) enumerate the set Ω of all feasible rewiring operations for G and a given threshold τ; (2) get $\Delta(o)$ for each $o \in \Omega$ by computing Z using Eq. 2 on G before/after performing o; (3) find the operation o that has the largest $\Delta(o)$ as the optimal solution o^*. In the brute-force algorithm, since the number of existing edges is $O(dn)$ and the number of possible new edges to rewire is $O(n)$ for each existing edge, the size of Ω is $O(dn^2)$. In addition, the old and new values of Z can be computed by matrix inversion using Eq. 2 in $O(n^3)$ time. Therefore, the brute-force algorithm runs in $O(dn^5)$ time. As all feasible operations are examined, this solution is guaranteed to be optimal.

The brute-force algorithm is impractical if the graph is large, due to the huge number of feasible operations and the high cost of computing Z. We introduce two strategies to improve its efficiency. First, we update the vector $\mathbf{z}$ incrementally for a rewiring operation. Second, we devise efficient strategies to avoid unnecessary computation when searching for the optimal rewiring operation, leading to our optimal 1-Rewiring algorithm.

4.2 Incremental Update of Vector z

We analyze how the fundamental matrix $\mathbf{F}$ and vector $\mathbf{z}$ change after performing a rewiring operation $o = (u, v, w)$. Two edits will be performed on G for o: (1) the removal of an existing edge $(u, v) \in E$ and (2) the insertion of a new edge $(u, w) \notin E$ to E.

The two operations update the transition matrix $\mathbf{M}$ to $\mathbf{M}'$ as follows:

$$\mathbf{M}' = \mathbf{M} + \mathbf{e}\mathbf{g}^\top$$

where $\mathbf{e}$ is an n-dimensional column vector that indicates the position of the source node u:

$$e_j = \begin{cases} 1 & \text{if } j = u \\ 0 & \text{otherwise} \end{cases}$$

and $\mathbf{g}^\top$ is an n-dimensional row vector that denotes the changes in the transition probabilities. Specifically, for the removal of (u, v) and insertion of (u, w), the probability p_{uv} of (u, v) is reassigned to (u, w). We denote the probability as $p_o = p_{uv}$. Formally,

$$g_j = \begin{cases} -p_o & \text{if } j = v \\ +p_o & \text{if } j = w \\ 0 & \text{otherwise} \end{cases}$$

Thus, operation $o = (u, v, w)$ on the fundamental matrix $\mathbf{F}$ yields an updated fundamental matrix $\mathbf{F}'$:

$$\mathbf{F}' = ((\mathbf{I} - \mathbf{M}_{hh}) - \mathbf{e}\mathbf{g}^\top)^{-1} = (\mathbf{F}^{-1} + (-1)\mathbf{e}\mathbf{g}^\top)^{-1}$$

By applying the Sherman-Morrison formula [46], we can avoid the computation of the new inverse and express $\mathbf{F}'$ as:

$$\mathbf{F}' = \mathbf{F} - \frac{\mathbf{F}\mathbf{e}\mathbf{g}^\top\mathbf{F}}{1 + \mathbf{g}^\top\mathbf{F}\mathbf{e}} \tag{3}$$

Accordingly, the new vector $\mathbf{z}'$ is expressed as:

$$\mathbf{z}' = \mathbf{z} - \frac{\mathbf{F}\mathbf{e}\mathbf{g}^\top\mathbf{F}}{1 + \mathbf{g}^\top\mathbf{F}\mathbf{e}}\mathbf{1} \tag{4}$$

The denominator of the second term in Eq. 4 can be written as:

$$1 + \mathbf{g}^\top\mathbf{F}\mathbf{e} = 1 - p_o(f_{wu} \cdot \mathbf{1}_{w \in V_h} - f_{vu})$$

where $\mathbf{1}_{w \in V_h}$ is an indicator that is equal to 1 if $w \in V_h$ and 0 otherwise. Because, as mentioned in Section 3, we restrict ourselves to rewiring with $w \notin V_h$, the above expression is simplified as:

$$1 + \mathbf{g}^\top\mathbf{F}\mathbf{e} = 1 + p_o f_{vu}.$$

Meanwhile, the numerator of the second term in Eq. 4 is written as:

$$\mathbf{F}\mathbf{e}\mathbf{g}^\top\mathbf{F}\mathbf{1} = -\mathbf{f}_u(z_w \cdot \mathbf{1}_{w \in V_h} - z_v)p_o$$

Algorithm 1: Optimal 1-Rewiring

Input : Graph $G = (V, E, \mathbf{M})$, fundamental matrix $\mathbf{F}$, segregation vector $\mathbf{z}$, threshold τ
Output : Optimal rewiring operation o^*

1 Initialize $\Omega \leftarrow \emptyset$, $o^* \leftarrow NULL$, $\Delta^* \leftarrow 0$;
2 **foreach** *node* $u \in V_h$ **do**
3 Find node $w \in V_n$ s.t. $(u, w) \notin E$ and s_{uw} is the maximum;
4 **foreach** *node* $v \in V_h$ *with* $(u, v) \in E$ **do**
5 Add $o = (u, v, w)$ to Ω if $L(\Gamma^+(u)) \geq \tau$ after replacing (u, v) with (u, w);
6 Sort nodes in V_h as $\langle h_1, \ldots, h_{n_h} \rangle$ in descending order of z_h;
7 **foreach** $o \in \Omega$ **do**
8 Compute $\Delta(h_1, o)$ using Eq. 5;
9 **if** $z'_{h_1} > z_{h_2}$ **then**
10 $\Delta(o) \leftarrow \Delta(h_1, o)$;
11 **else**
12 Find the largest $j > 1$ such that $z'_{h_1} < z_{h_j}$;
13 Compute $\Delta(h_i, o)$ for each $i = 2, \ldots, j$;
14 $\Delta(o) \leftarrow z_{h_1} - \max_{i \in [1,j]} z'_{h_i}$;
15 **if** $\Delta(o) > \Delta^*$ **then**
16 $o^* \leftarrow o$ and $\Delta^* \leftarrow \Delta(o)$;
17 **return** o^*;

where $\mathbf{f}_u$ is the column vector corresponding to u in $\mathbf{F}$, z_w and z_v are the entries of $\mathbf{z}$ for u and v, respectively. As previously, because $w \notin V_h$, we have that Eq. 4 is simplified as:

$$\mathbf{z}' = \mathbf{z} - \frac{\mathbf{f}_u z_v}{1/p_o + f_{vu}}.$$

For any harmful node h, we calculate its decrease $\Delta(h, o)$ in segregation score after performing $o = (u, v, w)$ as:

$$\Delta(h, o) = z_h - z'_h = \frac{f_{hu} z_v}{1/p_o + f_{vu}} \tag{5}$$

The optimal 1-Rewiring we present next is based on Eq. 5.

4.3 Optimal 1-Rewiring Algorithm

We now introduce our method to find the optimal solution o^* of 1-Rewiring, i.e., the rewiring operation that maximizes $\Delta(o)$ among all $o \in \Omega$. The detailed procedure is presented in Algorithm 1, to which the fundamental matrix $\mathbf{F}$ and segregation vector $\mathbf{z}$ are given as input. The algorithm proceeds in two steps: (1) candidate generation, as described in Lines 2–5, which returns a set Ω of possible rewiring operations that definitely include the optimal 1-Rewiring, and (2) optimal rewiring search, as described in Lines 6–16, which computes the objective value for each candidate rewiring to identify the optimal one. Compared with the brute-force algorithm, this method reduces the cost of computing $\Delta(o)$ since it only probes a few nodes with the largest segregation scores. In addition, it can still be guaranteed to find the optimal solution, as all rewiring operations that might be the optimal one have been considered.

Candidate generation. The purpose of this step is to exclude from enumeration all rewiring operations that violate the quality constraint. Towards this end, we do not consider any rewiring operation that for any node u will lead to the *discount cumulative gain* (DCG) of u below the threshold τ. According to Eq. 5, we find that $\Delta(h, o)$ of node h w.r.t. $o = (u, v, w)$ is independent of (u, w). Therefore, for a specific node u, we can fix w to the neutral (absorbing) node with the highest relevance score s_{uw} and $(u, w) \notin E$ so that as many rewiring operations as possible are feasible. Then, we should select the node v where $(u, v) \in E$ will be replaced. We need to guarantee that $L(\Gamma^+(u)) \geq \tau$ after (u, v) is replaced by (u, w). For each node $v \in \Gamma^+(u)$, we can take s_{uv} and s_{uw} into Eq. 1. If $L(\Gamma^+(u)) \geq \tau$, we will list $o = (u, v, w)$ as a candidate. After considering each node $u \in V_h$, we generate the set Ω of all candidate rewiring operations.

Optimal rewiring search. The second step is to search for the optimal rewiring operation o^* from Ω. We first sort all harmful nodes in descending order of their segregation scores as $\langle h_1, h_2, \ldots, h_{n_h} \rangle$, where h_i is the node with the i-th largest segregation score. Since we are interested in minimizing the maximum segregation, we can focus on the first few nodes with the largest segregation scores and ignore the remaining ones. We need to compute $\Delta(o)$ for each $o \in \Omega$ and always keep the maximum of $\Delta(o)$. After evaluating every $o \in \Omega$, it is obvious that the one maximizing $\Delta(o)$ is o^*. Furthermore, to compute $\Delta(o)$ for some operation o, we perform the following steps: (1) compute $\Delta(h_1, o)$ using Eq. 5; (2) if $z'_{h_1} > z_{h_2}$, then $\Delta(o) = \Delta(h_1, o)$; (3) otherwise, find the largest j such that $z'_{h_1} < z_{h_j}$, compute $\Delta(h_i, o)$ for each $i = 2, \ldots, j$; in this case, we have $\Delta(o) = z_{h_1} - \max_{i \in [1,j]} z'_{h_i}$.

Time complexity. Compared with the brute-force algorithm, the size of Ω is reduced from $O(dn^2)$ to $O(dn)$. Then, sorting the nodes in V_h takes $O(n \log n)$ time. Moreover, it takes $O(1)$ time to compute $\Delta(h, o)$ for each h and o. For each $o \in \Omega$, $\Delta(h, o)$ is computed $O(n)$ times in the worst case. Therefore, the time complexity is $O(dn^2)$ in the worst case. However, in our experimental evaluation, we find that $\Delta(h, o)$ is computed only a small number of times. Therefore, if computing $\Delta(o)$ takes $O(1)$ time in practice, then the anticipated running time is $O\big(n(d + \log n)\big)$, as confirmed empirically.

Algorithm 2: Heuristic k-Rewiring

Input : Graph $G = (V, E, \mathbf{M})$, threshold τ, size constraint k
Output : A set O of k rewiring operations

1 Compute the initial $\mathbf{F}$ and $\mathbf{z}$ based on $\mathbf{M}$;
2 Acquire Ω using Lines 2–5 of Algorithm 1;
3 Initialize $O \leftarrow \emptyset$;
4 **for** $i \leftarrow 1, 2, \ldots, k$ **do**
5 Run Lines 6–16 of Algorithm 1 to get $o^* = (u^*, v^*, w^*)$;
6 $O \leftarrow O \cup \{o^*\}$;
7 Update G, $\mathbf{M}$, $\mathbf{F}$, and $\mathbf{z}$ for o^*;
8 Delete the existing rewiring operations of u^* from Ω and add new possible operations of u^* to Ω;
9 **if** $\Omega = \emptyset$ **then**
10 **break**;
11 **return** O;

4.4 Heuristic k-Rewiring Algorithm

Our k-Rewiring algorithm based on the 1-Rewiring algorithm is presented in Algorithm 2. Its basic idea is to find the k rewiring

operations by running the 1-Rewiring algorithm k times. The first step is to initialize the fundamental matrix $\mathbf{F}$ and segregation vector $\mathbf{z}$. In our implementation, instead of performing the expensive matrix inversion (in Eq. 2), $\mathbf{F}$ and $\mathbf{z}$ are approximated through the power iteration method in [34]. Then, the procedure of candidate generation is the same as that in Algorithm 1. Next, it runs k iterations for getting k rewiring operations. At each iteration, it also searches for the the optimal rewiring operation $o^* = (u^*, v^*, w^*)$ among Ω as Algorithm 1. After that, G, $\mathbf{M}$, $\mathbf{F}$, and $\mathbf{z}$ are updated according to o^* (see Eq. 3 and 4 for the update of $\mathbf{F}$ and $\mathbf{z}$). Since the existing rewiring operations of u^* are not feasible anymore, it will regenerate new possible operations of u^* based on the updated $\Gamma^+(u^*)$ and the threshold τ to replace the old ones. Finally, the algorithm terminates when k rewiring operations have been found or there is no feasible operation anymore.

Time complexity. The time complexity of computing $\mathbf{F}$ and $\mathbf{z}$ is $O(\text{iter} \cdot dn)$ where iter is the number of iterations in the power method. The time to update $\mathbf{F}$ and $\mathbf{z}$ for each rewiring operation is $O(n)$. Overall, its time complexity is $O(kdn^2)$ since it is safe to consider that iter $\ll n$. In practice, it takes $O(1)$ time to compute $\Delta(o)$ and iter $= O(k)$, and thus the running time of the k-Rewiring algorithm can be regarded as $O\left(kn\left(d + \log n\right)\right)$.

5 EXPERIMENTS

Our experiments aim to: (1) show the effectiveness of our algorithm on mitigating radicalization pathways compared to existing algorithms; (2) test the robustness of our algorithm with respect to different thresholds τ; and (3) illustrate how much our algorithm can reduce the total exposure to harmful content.

5.1 Experimental Setup

Datasets. We perform experiments within two application domains: video sharing and news feeding.

For the first application, we use the **YouTube** dataset [47], which contains 330,925 videos and 2,474,044 recommendations. The dataset includes node labels such as "*alt-right*", "*alt-lite*", "*intellectual dark web*" and "*neutral*". We categorize the first three classes as "radicalized" or harmful and the last class as "neutral," following the analysis done by this dataset's curators [47], in which these three classes are shown to be overlapping in terms of audience and content. When generating the recommendation graphs, we consider only videos having a minimum of 10k views. In this way, we filter out all the ones with too few interactions. We consider the video-to-video recommendations collected via simulations as implicit feedback interactions, where the video-to-video interactions can be formatted as a square matrix, with position (u, v) containing the number of times the user jumped from video u to video v. Using alternating least squares (ALS) [22], we can first derive the latent dimensions of the matrix, generate the scores (normalized to $[0, 1]$) and then build the recommendation lists for each video. We eventually create different d-regular graphs with $d \in \{5, 10, 20\}$. To evaluate the effect of graph size on performance, we also use a smaller subset of videos with only 100k or more views for graph construction. Finally, we have 3 smaller (**YT-D5-S**, **YT-D10-S**, and **YT-D20-S**) and 3 larger (**YT-D5-B**, **YT-D10-B**, and **YT-D20-B**) recommendation graphs.

Table 1: Characteristics of the recommendation graphs used in the experiments, including out-degree d, number of nodes n, number of edges m, fraction of nodes from V_h (i.e., n_h/n), and initial segregation Z^0 of each graph.

YouTube					
Name	d	n	m	n_h/n	Z^0
YT-D5-S	5	31524	157620	0.48	588.86
YT-D5-B		105143	525715	0.43	598.32
YT-D10-S	10	31524	315240	0.48	718.92
YT-D10-B		105143	1051430	0.43	718.37
YT-D20-S	20	31524	630480	0.48	328.03
YT-D20-B		105143	2102860	0.43	331.09
NELA-GT					
Name	d	n	m	n_h/n	Z^0
NEWS-1	10	27286	272860	0.61	88.53
NEWS-2		22296	222960	0.62	29.90
NEWS-3		28861	288610	0.61	335.23
NEWS-4		26114	261140	0.65	75.15

For the second application, we use the **NELA-GT** dataset [39], which is a collection of 713k news in English. Each news article includes title, text, and timestamp, as well as credibility labels (reliable or unreliable). Our task is to reduce the risk of users getting stuck in unreliable content via *"what-to-read-next"* recommendations. To build the recommendation graphs, we compute the pairwise semantic similarities between news through the pre-generated weights with RoBERTa [32]. After normalizing the scores in the range $[0, 1]$, in order to reproduce different instances of news feeding websites, we generate different subsets of news by month. We perform our experiments on the 4 months with the largest number of news: August (**NEWS-1**), September (**NEWS-2**), October (**NEWS-3**) and November (**NEWS-4**).

The characteristics of the ten recommendation graphs used in our experiments are reported in Table 1.

Algorithms. We compare our proposed heuristic (**HEU**) algorithm for k-Rewiring with three baselines and one existing algorithm. The first baseline (**BSL-1**) selects the set of k rewiring operations by running Algorithm 1. Instead of picking only one rewiring operation, it picks the k operations with the largest values of Δ all at once. The second baseline (**BSL-2**) considers the best possible k rewiring operations by looking at the initial values of the vector $\mathbf{z}$. It firsts select the k nodes with the largest z values, then among the possible rewiring operations from those nodes, it returns the k operations with the largest values of Δ. The third baseline (**RND**) just picks k random rewiring operations from all the candidates. Finally, the existing method we compare with is the *RePBubLik* algorithm [21] (**RBL**). It reduces the structural bias of the graph by looking at the bubble radius of the two partitions of nodes, returning a list of k new edges to add. The original algorithm is designed for the insertion of new links, and not for the rewiring (deletion + insertion). Consequently, we adapt the *RePBubLik* algorithm to our

objective as follows: (1) we run it to return a list of potential edges to be added for reducing the structural bias of the harmful nodes; (2) for each potential insertion, in order to generate a rewiring operation, we check among the existing edges to find the one edge that meets the quality constraint τ after being replaced by the new edge; (3) we finally select a set of k rewiring operations from the previous step.

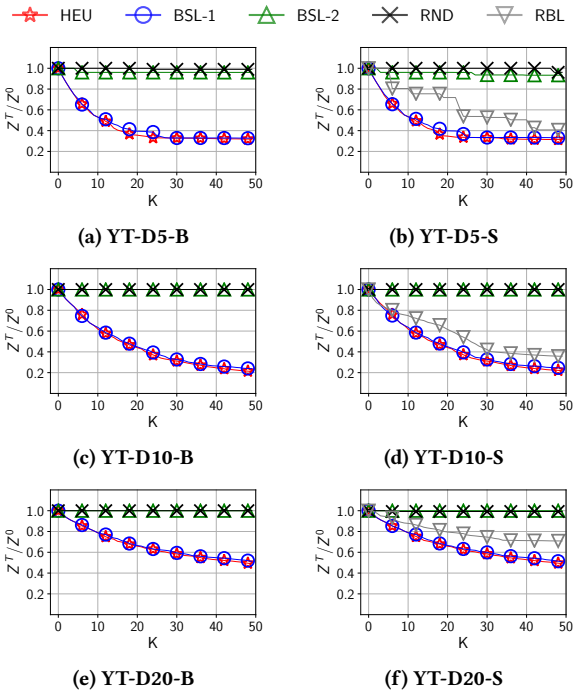
The experiments were conducted on a server running Ubuntu 16.04 with an Intel Broadwell 2.40GHz CPU and 29GB of memory. Our algorithm and baselines were implemented in Python 3. Our code and datasets are publicly available at https://github.com/FraFabbri/rewiring-what-to-watch. The implementation of *RePBubLik* is available at https://github.com/CriMenghini/RePBubLik.

5.2 Experimental Results

Effectiveness of our method. In Figure 1, we present the results on the YouTube recommendation graphs. On each graph, we evaluate the performance of each algorithm along 50 rewiring operations with the threshold of quality constraint is fixed to $\tau = 0.9$. We keep track of the relative decrease in the segregation Z^T/Z^0 after each rewiring operation, where Z^0 is the initial segregation and Z^T is the segregation after T rewiring operations. On all the graphs, it is clear that our heuristic algorithm (**HEU**) outperforms all the competitors. On the graphs with the smallest out-degree ($d = 5$), it decreases Z by over 40% within only 10 rewiring operations (i.e., $Z^{10}/Z^0 \leq 0.6$). In this case, it stops decreasing Z after 30 rewiring operations, which implies that only after a few rewiring operations our heuristic algorithm has found the best possible operations constrained by the threshold τ. On the graphs with $d = 10$, our heuristic algorithm is able to decrease Z by nearly 80%, which is even larger than the case of $d = 5$. This result is consistent in both smaller (YT-D10-S) and bigger (YT-D10-B) graphs. On the graphs with the largest out-degree ($d = 20$), the algorithm is still effective but, as expected, achieves a comparable reduction in Z after 50 operations.

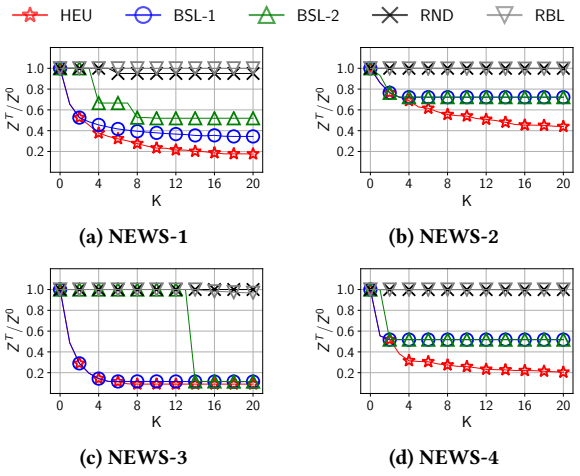
The first baseline (**BSL-1**) shows almost the same solution quality as **HEU**, since most of the operations found by both algorithms are the same. Although the rewiring operations provided by *RePBubLik* (**RBL**) also decrease the original Z_0 significantly, they are less effective than the ones given by our algorithm. Also, with a smaller size of recommendation list ($d = 5$), it reaches some steady states along the iterations, where the new rewiring operations do not decrease the Z value at all. For the YouTube dataset, we present only the results of **RBL** on the smaller graphs (the second column of Figure 1), since it cannot finish in reasonable time (24 hours) on larger graphs. The other baseline (**BSL-2**) and the random solution (**RND**) do not produce substantial decreases over the initial Z_0.

In Figure 2, we present the results on the NELA-GT recommendation graphs. We also fix $\tau = 0.9$ in these experiments. Given that the values of Z^0 are smaller in the news recommendation graph, we evaluate the performance of different algorithms with smaller k (i.e., $k = 20$). As for the previous case, our heuristic algorithm is the one achieving the best performance on every graph, which reduces Z by at least 60% after 20 rewiring operations. Furthermore, on the graph with the biggest Z value (NEWS-3), it decreases the initial segregation by more than 80% only after 4 rewiring operations. The two baselines (**BSL-1** and **BSL-2**) show comparable performance,



(a) YT-D5-B (b) YT-D5-S (c) YT-D10-B (d) YT-D10-S (e) YT-D20-B (f) YT-D20-S

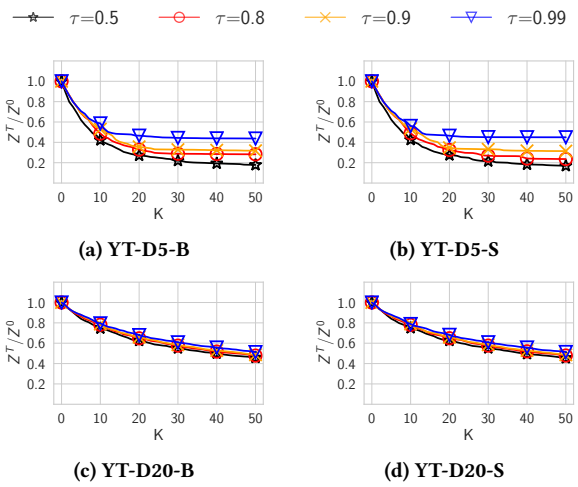
Figure 1: Performance comparison in the YouTube dataset.



(a) NEWS-1 (b) NEWS-2 (c) NEWS-3 (d) NEWS-4

Figure 2: Performance comparison in the NELA-GT dataset.

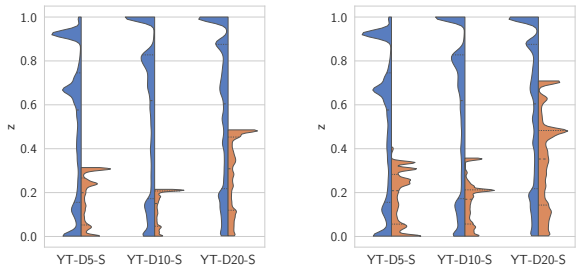
but only on NEWS-3 they obtain close drops in Z_0 to **HEU** after 20 iterations. In the other cases, they are stuck in steady states far from **HEU**. The rewiring provided by *RePBubLik* (**RBL**) shows no significant decrease over the initial Z_0, which is comparable only to **RND**. The difference in performance between YouTube and NELA-GT can be to some extent attributed to differences in their degree distributions. We compute the Gini coefficient of the in-degree distribution of the graphs: for the YouTube graphs the Gini coefficient of in-degree for the harmful nodes is never below 90%; while for the NELA-GT graphs this index is never above 50%.



(a) YT-D5-B (b) YT-D5-S

(c) YT-D20-B (d) YT-D20-S

Figure 3: Performance of our algorithm (HEU) with varying quality constraints τ.



(a) Heuristic (HEU) (b) RePBubLik (RBL)

Figure 4: Distribution of the segregation scores (z values) of harmful nodes before (blue) and after (red) performing 50 rewiring operations provided by HEU and RBL.

These differences imply that *RePBubLik* might not perform well when the in-degree distribution of the graph is not highly skewed.

Robustness w.r.t. threshold of recommendation quality. To investigate the role of the threshold τ of recommendation quality on the output of our algorithm, we test on the YouTube recommendation graphs with the same number of rewiring operations ($k = 50$) but different values of τ in $\{0.5, 0.8, 0.9, 0.99\}$. We present the results in Figure 3. As expected, under a more lenient quality constraint ($\tau = 0.5$), the algorithm achieves a larger decease in the value of Z. It is also clear that the differences are less evident on graphs with a larger out-degree ($d = 20$). Specifically, for a smaller out-degree ($d = 5$) all the τ configurations except $\tau = 0.5$ tend to stabilize after $k = 20$ rewiring operations. This is because the number of possible rewiring operations constrained by τ is small. It is also evident that the graph size, given different values of τ, does not impact the overall performance of our algorithm.

Total exposure to harmful content. Having tested the effectiveness of our algorithm in reducing the maximum segregation score, we study its effect on the distribution of the segregation scores over all harmful nodes. Figure 4 depicts the distribution of the z values before and after the rewiring operations (with $k = 50$ and $\tau = 0.9$) provided by **HEU** and **RBL** on the YouTube recommendation graphs. For each graph, the violin plot in blue (left) denotes the distribution of segregation scores before the rewiring operations and the one in red (right) the distribution after the rewiring operations. The range of segregation scores is normalized to $[0, 1]$, where the maximum corresponds to the initial segregation. We observe that reducing the maximum segregation also helps reduce the segregation scores of other harmful nodes. Compared to **RBL**, **HEU** generates a distribution more highly concentrated around smaller values; this discrepancy between the distributions is most significant when $d = 20$.

6 CONCLUSIONS AND FUTURE WORK

In this paper we studied the problem of reducing the risk of radicalization pathways in *what-to-watch-next* recommenders via edge rewiring on the recommendation graph. We formally defined the segregation score of a radicalized node to measure its potential to trap users into radicalization pathways, and formulated the k-Rewiring problem to minimize the maximum segregation score among all radicalized nodes, while maintaining the quality of the recommendations. We proposed an efficient yet effective greedy algorithm based on the absorbing random walk theory. Our experiments, in the context of video and news recommendations, confirmed the effectiveness of our proposed algorithm.

This work is just a first step and it has several limitations that we plan to tackle in future work. One main limitation is assuming a binary labelling of nodes, which limits each content to one of the two groups (harmful or neutral), which is not always realistic. A natural extension is to assume numerical labels in $[0, 1]$. This would require to re-define segregation accordingly.

Another limitation is that we are given the recommendation graph as input. This implies that the recommendations are precomputed and static. We plan to extend our setting to a scenario where recommendations are generated dynamically in an online fashion.

Finally, we showed through empirical evidence how our method, designed to reduce maximum segregation, may actually reduce the *total* segregation generated by all harmful nodes in the graph. We plan to design different algorithms which are able to directly tackle this objective.

ACKNOWLEDGMENTS

Francesco Fabbri is a fellow of Eurecat's *Vicente López* PhD grant program; his work was partially financially supported by the Catalan Government through the funding grant *ACCIÓ-Eurecat* (Project *PRIVany-nom*). Francesco Bonchi acknowledges support from Intesa Sanpaolo Innovation Center. Carlos Castillo has been partially supported by the *HUMAINT* programme (Human Behaviour and Machine Intelligence), European Commission, and by *"la Caixa"* Foundation (ID 100010434), under agreement LCF/PR/PR16/51110009. Michael Mathioudakis has been supported by the *MLDB* project of the Academy of Finland (decision number: 322046).

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

REFERENCES

[1] Hunt Allcott and Matthew Gentzkow. 2017. Social Media and Fake News in the 2016 Election. *J. Econ. Perspect.* 31, 2 (2017), 211–236.

[2] Victor Amelkin and Ambuj K. Singh. 2019. Fighting Opinion Control in Social Networks via Link Recommendation. In *KDD*. 677–685.

[3] Elisabetta Bergamini, Pierluigi Crescenzi, Gianlorenzo D'Angelo, Henning Meyerhenke, Lorenzo Severini, and Yllka Velaj. 2018. Improving the Betweenness Centrality of a Node by Adding Links. *ACM J. Exp. Algorithmics* 23 (2018).

[4] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. 2018. Equity of Attention: Amortizing Individual Fairness in Rankings. In *SIGIR*. 405–414.

[5] Matteo Castiglioni, Diodato Ferraioli, and Nicola Gatti. 2020. Election Control in Social Networks via Edge Addition or Removal. In *AAAI*. 1878–1885.

[6] Hau Chan, Leman Akoglu, and Hanghang Tong. 2014. Make It or Break It: Manipulating Robustness in Large Networks. In *SDM*. 325–333.

[7] Xi Chen, Jefrey Lijffijt, and Tijl De Bie. 2018. Quantifying and Minimizing Risk of Conflict in Social Networks. In *KDD*. 1197–1205.

[8] Uthsav Chitra and Christopher Musco. 2020. Analyzing the Impact of Filter Bubbles on Social Network Polarization. In *WSDM*. 115–123.

[9] Federico Cinus, Marco Minici, Corrado Monti, and Francesco Bonchi. 2021. The Effect of People Recommenders on Echo Chambers and Polarization. (2021). arXiv:2112.00626 [cs.SI]

[10] Pierluigi Crescenzi, Gianlorenzo D'Angelo, Lorenzo Severini, and Yllka Velaj. 2016. Greedily Improving Our Own Closeness Centrality in a Network. *ACM Trans. Knowl. Discov. Data* 11, 1 (2016), 9:1–9:32.

[11] Mihaela Curmei, Sarah Dean, and Benjamin Recht. 2021. Quantifying Availability and Discovery in Recommender Systems via Stochastic Reachability. In *ICML*. 2265–2275.

[12] Gianlorenzo D'Angelo, Martin Olsen, and Lorenzo Severini. 2019. Coverage Centrality Maximization in Undirected Networks. In *AAAI*. 501–508.

[13] Sarah Dean, Sarah Rich, and Benjamin Recht. 2020. Recommendations and user agency: the reachability of collaboratively-filtered information. In *FAT**. 436–445.

[14] Francesco Fabbri, Francesco Bonchi, Ludovico Boratto, and Carlos Castillo. 2020. The Effect of Homophily on Disparate Visibility of Minorities in People Recommender Systems. In *ICWSM*. 165–175.

[15] Francesco Fabbri, Maria Luisa Croci, Francesco Bonchi, and Carlos Castillo. 2021. Exposure Inequality in People Recommender Systems: The Long-Term Effects. arXiv:2112.08237 [cs.SI]

[16] Emilio Ferrara, Onur Varol, Clayton A. Davis, Filippo Menczer, and Alessandro Flammini. 2016. The rise of social bots. *Commun. ACM* 59, 7 (2016), 96–104.

[17] M. R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[18] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2016. Quantifying Controversy in Social Media. In *WSDM*. 33–42.

[19] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2017. Reducing Controversy by Connecting Opposing Views. In *WSDM*. 81–90.

[20] Pedro Henrique Calais Guerra, Wagner Meira Jr., Claire Cardie, and Robert Kleinberg. 2013. A Measure of Polarization on Social Media Networks Based on Community Boundaries. In *ICWSM*. 215–224.

[21] Shahrzad Haddadan, Cristina Menghini, Matteo Riondato, and Eli Upfal. 2021. RePBubLik: Reducing Polarized Bubble Radius with Link Insertions. In *WSDM*. 139–147.

[22] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*. 263–272.

[23] Ruben Interian, Jorge R. Moreno, and Celso C. Ribeiro. 2021. Polarization reduction by minimum-cardinality edge additions: Complexity and integer programming approaches. *Int. Trans. Oper. Res.* 28, 3 (2021), 1242–1264.

[24] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446.

[25] Elias Boutros Khalil, Bistra Dilkina, and Le Song. 2014. Scalable diffusion-aware optimization of network topology. In *KDD*. 1226–1235.

[26] Masahiro Kimura, Kazumi Saito, and Hiroshi Motoda. 2008. Minimizing the Spread of Contamination by Blocking Links in a Network. In *AAAI*. 1175–1180.

[27] Chris J. Kuhlman, Gaurav Tuli, Samarth Swarup, Madhav V. Marathe, and S. S. Ravi. 2013. Blocking Simple and Complex Contagion by Edge Removal. In *ICDM*. 399–408.

[28] Long T. Le, Tina Eliassi-Rad, and Hanghang Tong. 2015. MET: A Fast Algorithm for Minimizing Propagation in Large Graphs with Small Eigen-Gaps. In *SDM*. 694–702.

[29] Mark Ledwich and Anna Zaitsev. 2020. Algorithmic extremism: Examining YouTube's rabbit hole of radicalization. *First Monday* 25, 3 (2020).

[30] Rebecca Lewis. 2018. *Alternative Influence: Broadcasting the Reactionary Right on YouTube*. Technical Report. Data & Society Research Institute.

[31] Rong-Hua Li and Jeffrey Xu Yu. 2015. Triangle minimization in large networks. *Knowl. Inf. Syst.* 45, 3 (2015), 617–643.

[32] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692 [cs.CL]

[33] Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *RANLP*. 467–472.

[34] Charalampos Mavroforakis, Michael Mathioudakis, and Aristides Gionis. 2015. Absorbing Random-Walk Centrality: Theory and Algorithms. In *ICDM*. 901–906.

[35] Clark McCauley and Sophia Moskalenko. 2008. Mechanisms of Political Radicalization: Pathways Toward Terrorism. *Terror. Political Violence* 20, 3 (2008), 415–433.

[36] Sourav Medya, Arlei Silva, Ambuj K. Singh, Prithwish Basu, and Ananthram Swami. 2018. Group Centrality Maximization via Network Design. In *SDM*. 126–134.

[37] Mainack Mondal, Leandro Araújo Silva, and Fabrício Benevenuto. 2017. A Measurement Study of Hate Speech in Social Media. In *HT*. 85–94.

[38] Cameron Musco, Christopher Musco, and Charalampos E. Tsourakakis. 2018. Minimizing Polarization and Disagreement in Social Networks. In *WWW*. 369–378.

[39] Jeppe Nørregaard, Benjamin D. Horne, and Sibel Adali. 2019. NELA-GT-2018: A Large Multi-Labelled News Dataset for the Study of Misinformation in News Articles. In *ICWSM*. 630–638.

[40] Kostantinos Papadamou, Antonis Papasavva, Savvas Zannettou, Jeremy Blackburn, Nicolas Kourtellis, Ilias Leontiadis, Gianluca Stringhini, and Michael Sirivianos. 2020. Disturbed YouTube for Kids: Characterizing and Detecting Inappropriate Videos Targeting Young Children. In *ICWSM*. 522–533.

[41] Manos Papagelis. 2015. Refining Social Graph Connectivity via Shortcut Edge Addition. *ACM Trans. Knowl. Discov. Data* 10, 2 (2015), 12:1–12:35.

[42] Manos Papagelis, Francesco Bonchi, and Aristides Gionis. 2011. Suggesting ghost edges for a smaller world. In *CIKM*. 2305–2308.

[43] Nikos Parotsidis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2015. Selecting Shortcuts for a Smaller World. In *SDM*. 28–36.

[44] Nikos Parotsidis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2016. Centrality-Aware Link Recommendations. In *WSDM*. 503–512.

[45] Evaggelia Pitoura, Georgia Koutrika, and Kostas Stefanidis. 2020. Fairness in Rankings and Recommenders. In *EDBT*. 651–654.

[46] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge University Press.

[47] Manoel Horta Ribeiro, Raphael Ottoni, Robert West, Virgílio A. F. Almeida, and Wagner Meira Jr. 2020. Auditing radicalization pathways on YouTube. In *FAT**. 131–141.

[48] Kevin Roose. 2019. The Making of a YouTube Radical. *The New York Times* (2019). https://www.nytimes.com/interactive/2019/06/08/technology/youtube-radical.html

[49] Sudip Saha, Abhijin Adiga, B. Aditya Prakash, and Anil Kumar S. Vullikanti. 2015. Approximation Algorithms for Reducing the Spectral Radius to Control Epidemic Spread. In *SDM*. 568–576.

[50] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake News Detection on Social Media: A Data Mining Perspective. *SIGKDD Explor.* 19, 1 (2017), 22–36.

[51] Hanghang Tong, B. Aditya Prakash, Tina Eliassi-Rad, Michalis Faloutsos, and Christos Faloutsos. 2012. Gelling, and melting, large graphs by edge manipulation. In *CIKM*. 245–254.

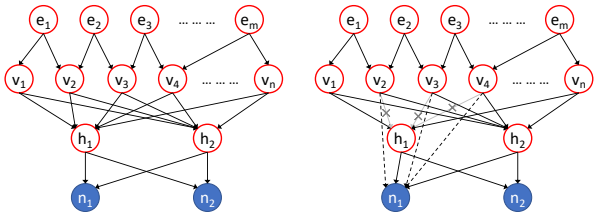
[52] Tomasz Was, Marcin Waniek, Talal Rahwan, and Tomasz P. Michalak. 2020. The Manipulability of Centrality Measures - An Axiomatic Approach. In *AAMAS*. 1467–1475.

[53] Bari Weiss and Damon Winter. 2018. Meet the Renegades of the Intellectual Dark Web. *The New York Times* (2018). https://www.nytimes.com/2018/05/08/opinion/intellectual-dark-web.html

[54] Ruidong Yan, Yi Li, Weili Wu, Deying Li, and Yongcai Wang. 2019. Rumor Blocking through Online Link Deletion on Social Networks. *ACM Trans. Knowl. Discov. Data* 13, 2 (2019), 16:1–16:26.

[55] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *RecSys*. 43–51.

[56] Weijie Zhu, Chen Chen, Xiaoyang Wang, and Xuemin Lin. 2018. K-core Minimization: An Edge Manipulation Approach. In *CIKM*. 1667–1670.



(a) Construct G^* from G **(b) k-Rewiring on G^***

Figure 5: Illustration of the reduction from the VertexCover problem to the k-Rewiring problem.

A PROOF OF THEOREM 3.1

Proof. We prove the NP-hardness of the k-Rewiring problem by a reduction from the VertexCover problem [17].

A VertexCover instance is specified by an undirected graph $G = (V, E)$, where $|V| = n$ and $|E| = m$, and an integer k. It asks whether G has a vertex cover of size at most k, i.e., whether there exists a subset $C \subseteq V$ with $|C| \leq k$ such that $\{v_i, v_j\} \cap C \neq \emptyset$ for every edge $e = (v_i, v_j) \in E$. We construct an instance of the k-Rewiring problem on G^* from a VertexCover instance on G as illustrated in Figure 5a. Given a graph $G = (V, E)$, the graph $G^* = (V^*, E^*)$ is constructed as follows: One vertex in G^* is created for each $e \in E$ and $v \in V$. Furthermore, four vertices h_1, h_2, n_1, n_2 are added to G^*. Let $V_h^* = E \cup V \cup \{h_1, h_2\}$ be the set of $(m + n + 2)$ "harmful" vertices (in red) and $V_n^* = \{n_1, n_2\}$ be the set of two "neutral" vertices (in blue). Then, for each $e = (v_i, v_j) \in E$, two directed edges (e, v_i) and (e, v_j) are added to G^*. For each $v \in V$, two directed edges (v, h_1) and (v, h_2) are added to G^*. Finally, four directed edges (h_1, n_1), (h_1, n_2), (h_2, n_1), and (h_2, n_2) are added to G^*. The out-degree d of each red node in G^* is 2. Accordingly, the transition probability of every edge in G^* is set to 0.5.

We first show that there will be a set O of at most k rewiring operations such that $\Delta(O) > 0$ after the rewiring operations in O are performed on G^* *if* G has a vertex cover of size at most k. For the original G^*, we have $z(h_1) = z(h_2) = 1$, $z(v) = 2$ for each vertex $v \in V$, and $z(e) = 3$ for each edge $e \in E$. Thus, we have $Z = z(e) = 3$. So, we will have $\Delta(O) > 0$ as long as $z'(e) < 3$ for each edge $e \in E$. Let $C = \{v_1, \ldots, v_k\}$ be a size-k vertex cover of G. We construct a set $O = \{o_1, \ldots, o_k\}$ of k rewiring operations on G^*, where $o_i = (v_i, h_1, n_1)$, corresponding to C, as illustrated in Figure 5b. After performing the set O of rewiring operations on G^*, we have two cases for $z'(e)$ of each $e = (v_i, v_j)$:

$$z'(e) = \begin{cases} 0.5 \times 3 + 0.5 \times 2 = 2.5, & \text{if } |\{v_i, v_j\} \cap C| = 2 \\ 0.5 \times 3 + 0.5 \times 2.5 = 2.75, & \text{if } |\{v_i, v_j\} \cap C| = 1 \end{cases}$$

Since C is a vertex cover, there is no edge $e = (v_i, v_j)$ such that $\{v_i, v_j\} \cap C = \emptyset$. Therefore, after performing the set O of rewiring operations on G^*, it must hold that $z'(e) < 3$ for every $e \in E$ and thus $\Delta(O) > 0$.

We then show that there will be a set O of at most k rewiring operations such that $\Delta(O) > 0$ after the rewiring operations in O are performed on G^* *only if* G has a vertex cover of size at most k. Or equivalently, if G does not have a vertex cover of size k, then any set O of k rewiring operations performed on G^* cannot make $\Delta(O) > 0$. Since G does not have a vertex cover of size k, there must exist some edge $\overline{e} = (v_i, v_j)$ with $\{v_i, v_j\} \cap \overline{C} = \emptyset$ for any size-k vertex set $\overline{C} \subseteq V$. Therefore, after performing the set $\overline{O}$ of k rewiring operations corresponding to $\overline{C}$, we have $z'(\overline{e}) = 3$ for an uncovered edge $\overline{e}$. So, we can say that any set of k rewiring operations from V cannot make $\Delta(O) > 0$. Furthermore, we consider the case of k rewiring operations from E, i.e., to find a set of k edges $\{e_1, \ldots, e_k\}$ and rewire one out-edge from each of them to n_1 or n_2. In this case, we can always find some unselected edge $\overline{e}$ with $z'(\overline{e}) = 3$ as long as $m > k$, which obviously holds as G does not have a vertex cover of size k. Finally, we consider the case of a "hybrid" set of k rewiring operations from both E and V. W.l.o.g., we assume that there are $(k - k')$ operations from V and k' operations from E for some $0 < k' < k$. Since G does not have a vertex cover of size k, we can say that any vertex set $\overline{C}$ of size $(k - k')$ can cover at most $(m - k' - 1)$ edges. Otherwise, we would find a vertex cover of size k by adding k' vertices to cover the remaining k' edges and thus lead to contradiction. Therefore, after performing only k' rewiring operations from E, there always exists at least one edge $\overline{e}$ that are covered by neither the vertex set nor the edge set, and thus $z'(\overline{e}) = 3$ and $\Delta(O) = 0$. Considering all the three cases, we prove that any set of k rewiring operations performed on G^* cannot make $\Delta(O) > 0$ if G does not have a vertex cover of size k.

Given that both the "*if*" and "*only-if*" directions are proven and G^* can be constructed from G in $O(m+n)$ time, we reduce from the VertexCover problem to the k-Rewiring problem in polynomial time and thus prove that the k-Rewiring problem is NP-hard.

To show the hardness of approximation, we suppose that there is a polynomial-time algorithm $\mathcal{A}$ that approximates the k-Rewiring problem within a factor of $\alpha > 0$. Or equivalently, for any k-Rewiring instance, if O^* is the set of k optimal rewiring operations, then the set O' of k rewiring operations returned by $\mathcal{A}$ will always satisfy that $\Delta(O') \geq \alpha \cdot \Delta(O^*)$. Let us consider a k-Rewiring instance on the above graph G^* constructed from G and k be the size of the minimum vertex cover of G. For this instance, the optimal solution O^* of the k-Rewiring problem exactly corresponds to the minimum vertex cover C^* of G with $\Delta(O^*) > 0$; any other solution O' will lead to $\Delta(O') = 0$, as we have shown in this proof. If $\mathcal{A}$ could find a solution for the k-Rewiring problem with any approximation factor $\alpha > 0$ in polynomial time, then $\mathcal{A}$ would have solved the VertexCover problem in polynomial time, which has been known to be impossible unless P=NP. Therefore, the k-Rewiring problem is NP-hard to approximate with any factor. □

B ETHICS STATEMENT

In this work, we aim at reducing the exposure to radicalized content generated by W2W recommender systems. Our approach does not include any form of censorship, and instead limits algorithmic-induced over-exposure, which is stimulated by biased organic interactions (e.g., the spread of radicalized content through user-user interactions). Our work contributes to raise awareness on the importance of devising policies aimed at reducing harmful algorithmic side-effects. Generally, we do not foresee any immediate and direct harmful impacts from this work.