Content Recommendation for Viral Social Influence

Sergei Ivanov
Skoltech
Manolis Terrovitis*
IMIS, Athena R.I.C.

Konstantinos Theocharidis
University of Peloponnese & IMIS, Athena R.I.C.
Panagiotis Karras
Aalborg University

ABSTRACT

How do we create content that will become viral in a whole network after we share it with friends or followers? Significant research activity has been dedicated to the problem of strategically selecting a seed set of initial adopters so as to maximize a meme’s spread in a network. This line of work assumes that the success of such a campaign depends solely on the choice of a tunable seed set of adopters, while the way users perceive the propagated meme is fixed. Yet, in many real-world settings, the opposite holds: a meme’s propagation depends on users’ perceptions of its tunable characteristics, while the set of initiators is fixed.

In this paper, we address the natural problem that arises in such circumstances: Suggest content, expressed as a limited set of attributes, for a creative promotion campaign that starts out from a given seed set of initiators, so as to maximize its expected spread over a social network. To our knowledge, no previous work addresses this problem. We find that the problem is NP-hard and inapproximable. As a tight approximation guarantee is not feasible, we design an efficient heuristic, Explore-Update, as well as a conventional Greedy solution. Our experimental evaluation demonstrates that Explore-Update selects near-optimal attribute sets with real data, achieves 30% higher spread than baselines, and runs an order of magnitude faster than the Greedy solution.

ACM Reference format:
DOI: 10.1145/3077136.3080788

1 INTRODUCTION

Online networking offers opportunities for new types of marketing. A prime example of such a new marketing technique is viral marketing, whereby organizations run promotion campaigns through word-of-mouth effects within online social networks. The influence maximization (IM) problem [26], studied intensively during the last decade, aims to find well-chosen seed nodes from which to launch such campaigns so as to achieve good results.

Recent works [17, 18] have focused on the parameters that define the popularity of a post, campaign, idea, or meme within a network. Such works were the first to study the question of how commercial brand posts engage online social network users, drawing from the theory of Uses & Gratifications [25]; they examine post parameters such as content type (e.g., entertaining, informational), media type (e.g., vivid, interactive), posting time (e.g., workday, peak hours) and valence of comments (e.g., positive, negative). Interestingly, such studies have reached some ambivalent conclusions; for instance, [18] ascertains that entertaining content decreases the number of “likes”, while [17] claims the exact opposite.

Concurrent research has studied the problem of viral product design [3, 6], which calls for engineering products by incorporating viral attributes so as to generate peer-to-peer influence that encourages adoption within a network. Aral and Walker [3] study the question of viral attribute selection under randomized trials only; Barbieri and Bonchi [6] allude to the same problem as a complement to the standard IM problem of selecting a set of seed nodes that maximizes influence, but do not investigate it as a stand-alone problem in its own right. Conceptually, both these works pertain to attributes attached to products; they do not investigate the more general problem of choosing content, out of a set of eligible options, for any kind of meme spreading in a network, so as to make it viral.

In this paper, we introduce and study the problem of selecting content that characterizes any type of meme, so as to maximize its expected spread through a network, starting out from a fixed set of initial adopters. For instance, an advertisement post may feature aspects such as topics, people, locations and abstract themes. We are particularly interested in those content aspects that are associated with specific online social network pages; we denote such aspects as content attributes. Fittingly, online social network users themselves are associated with such non-personal network pages: they express their preferences for specific brands, topics of interest, public persons, hobbies, or locations by subscribing to or “liking” such pages. Thereby, an attribute’s popularity can be gauged via its number of subscribers or page “likes”. For our purposes, we denote the pages that a user subscribes to or “likes” as user attributes. We contend that, the more content and user attributes overlap, the more likely that user is to propagate that post.

We envisage an organization that aims to achieve high viral effect of a campaign initiated from its fixed set of subscribers. For example, assume FlyFast airways wants to launch a promotion campaign in social media. FlyFast already has a social network presence, and its page has a subscribers’ set S fixed at a given moment, while it faces constraints related to its budget and people’s attention span. In their design, FlyFast consultants are interested to identify a set of k content attributes, out of a universe of eligible, mutually compatible options, that will maximize the expected network spread
of a post starting out from its subscribers’ set \( S \). Assume that, for \( k = 4 \), the optimal attribute set is \{“Best travel Accessories”, “Airline food guide”, “Hipster Europe”, “Backpacker tips”\}. Guided by this knowledge, FlyFast can infuse its post with complementary content that appeals to users interested in those topics, e.g., promotions to backpackers, references to its hipster audience, and highlights on its food quality. Thereby, it can maximize its promotion’s reach.

To our knowledge, we are the first to study the influence maximization problem in which the seed is given and post content is sought. Our related contributions are as follows:

**Problem Setting** We motivate the influence maximization problem in settings where the set of initial adopters is fixed, or even a single point of origin, and the content of a propagated meme can be tuned. We formulate the concept of digital influence as a special case of social influence.

**Propagation Model** We devise a content-aware propagation model, whereby the probability of influence across edges depends on content. We show that, with this model: (i) the problem of choosing content attributes that maximize influence is NP-hard; (ii) the spread function is not submodular, hence no submodularity-based approximation algorithm applies; and (iii) it is NP-hard to approximate the optimal solution within a factor of \( n^{1−\varepsilon} \) for \( \varepsilon > 0 \).

**Algorithm** We design a fast algorithm, Explore-Update, which achieves higher influence spread than baselines; its effectiveness is based on the iterative estimation of the marginal spread achieved by each attribute, while its efficiency is gained by limiting such computations only to nodes within a probability-based distance threshold \( \theta \) and attributes potentially affecting such nodes.

**Experiments** We compare Explore-Update to two baselines and show that it always achieves better propagation results, while it is significantly faster than a naïve Greedy approach; we calculate the optimal solution on a reduced dataset with a small universe of attributes, showing Explore-Update can achieve optimality; last, we demonstrate the scalability of Explore-Update on seed set size.

### 2 MOTIVATION AND BACKGROUND

We start our discussion drawing from cognitive science and marketing, paving the way for our problem definition.

#### 2.1 Idea Habitats

An idea habitat \([8]\) is the set of environmental cues that make people think about an idea and pass it along. Regardless of how well an idea is encoded, it will persist and spread only if the environment cues people to retrieve it regularly. In other words, even if an idea can be easily recalled, if it is only rarely cued by the environment, it may remain rare and be forgotten. It follows that, for an idea to spread, it should be not only well encoded and easily recalled, but also regularly retrieved. Promotion campaigns aim to assist the spread of any meme in the same way as idea habitats assist the spread of ideas. For instance, assume that FlyFast food is good and therefore memorable. Still, without a viral promotion, FlyFast will miss the spotlight, letting other airlines gain public attention.

#### 2.2 Digital Influence

Social influence is defined based on peer behavior \([1]\), so as to distinguish it from confounding factors \([2, 19, 21, 29]\).
Aslay et al. [4] studied online TIM queries; the incentive for this online scenario is that many independent advertisers wish to instantly detect the $k$ most influential users for advertising purposes; each advertisement contains a different set of keywords and hence induces a new probabilistic graph creating a separate TIM instance; the authors proposed an offline-online solution, INFLEX, based on an index used to identify similarities among a new and log TIM queries; pre-computed solutions for log queries are aggregated online so as to provide an approximate solution for a TIM query.

The online TIM problem is also studied in [10, 12]. Chen et al. [12] studied topic-aware influence results on two real networks and utilized the derived properties to form three preprocessing-based algorithms, of which MIS is the best; its main difference from INFLEX is that, in MIS, pre-computed seed sets are based on each separate topic rather than on a mixture of topics from different log queries. Chen et al. [10] utilized the maximum influence arborescence (MIA) model [13] to achieve high influence spread with a theoretical guarantee. The core idea is to utilize upper- and lower-bounding techniques, so that an exact marginal influence is computed only for the most promising nodes. This work provides the state-of-the-art solution for the online TIM problem [4].

Recently, Li et al. [28] proposed a variation on the online TIM problem, namely the alternative problem of Keyword-Based Targeted Influence Maximization (KB-TIM). By KB-TIM, each user is associated with a weighted vector of preferences for distinct keywords, which stand for topics. This vector can be generated by applying topic modeling techniques [23] on aggregated user social activities, such as posts, likes, etc. An advertisement then achieves an impact determined by its own topic-oriented keywords. The KB-TIM problem aims to maximize an advertisement’s impact, expressed in terms of its spread to target users relevant to its keywords. The solution in [28] draws from previous work in [32], with the main difference being that, while in [32] $\theta$ users in a sampled Reverse Reachable (RR) set [9] are counted without prejudice, in [28] these sampled users are accounted in terms of exerted advertisement impact; [28] also employs two indexing methods to precompute RR sets for different keywords, so as to obtain RR sets associated with the query keywords on the fly. Nevertheless, results in [28] are not compared to those in [10].

Our work differs drastically from all aforementioned works: rather than aiming to detect an influential set of users given a post’s content attributes, we are interested to find viral content attributes themselves, given a set of initial adopting users, so as to form an influential post. The aforementioned works do not examine what kind of posts would be most promising or powerful given all topics in the network. Besides, the topic-aware approach is based on general topical terms, like music, soccer, cars, etc., ignoring the high variation among different specimens within such terms. In contrast, we search for specific attributes that can form posts successful in terms of influence spread.

Influence Maximization with VPD. Aral and Walker [3] investigated the problem of viral product design under randomized trials focusing on product features like personalized referrals and broadcast notifications. Thereafter, Barbieri and Bonchi [6] studied the problem of influence maximization in conjunction with that of viral product design, aiming to detect a combination of seed nodes and product attributes that maximize influence in a network. The proposed solutions are generic methods named Local Update and Genetic Update; the former is a greedy algorithm allowing for both addition and removal of attributes at each greedy iteration; the latter is a brute-force method that randomly selects a subset of all attributes. By contrast, we investigate the problem of content selection for a post (not a product) as a stand-alone problem in its own right and study its distinctive characteristics.

### 2.4 Distinctiveness

We emphasize three distinct elements of this work.

**Problem Formulation** While our model takes into consideration the influence with regard to a post exercised by users themselves, we seek to maximize the influence exercised by the appeal and quality of a post’s content within a network. To the best of our knowledge, we are the first to define this problem, which is distinct from, and cannot be treated by methods aiming at, user selection.

**Nature of Attributes** In related works, attributes are configured as general topics; for example, Barbieri and Bonchi [6] assign tags, conceived of as general topics, to the role of product attributes. By contrast, in our problem formulation and in our experimental study, an attribute corresponds to a topic of interest identifiable via a non-personal social network page. Thus, even a page on an abstract theme (e.g., Psychology of Relations) is a possible attribute in our setting; it has a specific commercial value (4,719,837 followers) that differentiates it from other pages on similar topics.

**Use of Tags** The type of content attributes we investigate can hardly be articulated via tagging. Tags express highly idiosyncratic user impressions, focusing on arbitrary aspects of content; they are volatile as descriptors of content, whereas we need a stable ground truth to represent content. For example, a video post may relate to several specific topics of interest, yet it would be hard to identify these via user tagging. In consequence, past tagging does not offer valuable information in our problem setting.

### 3 PROBLEM STATEMENT

From the preceding discussion, we conclude that any brand would gain by maximizing the expected effectiveness of its product promotion campaigns within an online social network. We assume that there exists a certain set of subscribers to the brand’s social network page, and a promotion campaign aims to influence the maximum number of non-subscribers; as we discussed, such users are associated with topics expressing their interests.

#### 3.1 Content-Aware Cascade Model

We model an online social network as a directed graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of nodes, each of which corresponds to an individual user, and $E \subseteq V \times V$ is a set of directed edges representing social relations among users. Each node $v$ has a set of associated attributes $F_v = \{f_{v1}, f_{v2}, \ldots\}$, from a universe $\Phi$, that define user preferences; we identify these attributes as the non-personal network pages a user expresses interest in. A meme propagated through the network is associated with a set of attributes $F = \{f_{p1}, f_{p2}, \ldots, f_{pn}\} \subseteq \Phi$; these content attributes, along with the user attributes $F_v$ associated with the targeted node $v$, affect the probability of its propagation across a network edge $e_{uv}$. 

Accordingly, we define the Content-Aware Cascade model (CAC) as a variant of the Independent Cascade model (IC), in which edge propagation probabilities depend on content and user attributes. A CAC diffusion process unfolds in steps, starting from an initial seed set of activated nodes. A node \( u \) activated at time step \( t \) has a single chance to activate its out-neighbors. The process is incremental, as nodes can alternate only from inactive to active states; the diffusion ends when there is no newly activated node at a given step. At any step, a newly activated node \( u \) activates its out-neighbor \( v \) with probability \( p(u, v) \) equal to:

\[
p_{uv} = b_{uv} + q_{uv} \cdot h_{uv}(F_v, F), \quad b_{uv}, q_{uv} \in [0, 1]
\]

\[
h_{uv}(F_v, F) = \min \left\{ \frac{1 - b_{uv}}{q_{uv}} \left| F_v \cap F \right| \right\}
\]

(1)

where \( b_{uv} \) is a base probability on an edge and \( q_{uv} \) a marginal probability that indicates how much the probability on an edge increases for each selected attribute in \( F \) matching a preference of node \( v \), as indicated by the transition function \( h_{uv}(F_v, F) \), with a sanity bound of \( \frac{1 - b_{uv}}{q_{uv}} \). We emphasize that the marginal probability \( q_{uv} \) distinguishes among different user links, albeit not among different attributes for a given link; a more complex model could distinguish among different attributes, or even define a probability distribution function over the set of all attributes [7], to be learned by historical logs. We choose to relegate the problem of defining and learning such probability distribution functions to future work, and now study the problem under the modeling assumption that each attribute has the same independent effect on the probability function. Nevertheless, our simplified model forms a special case of any more complex model in which each attribute would have a different effect on the probability function; i.e., in this special case, such effects are rendered equal. Therefore, our subsequent hardness and inapproximability results hold for any such more complex model as well. Furthermore, parameters \( q_{uv} \) and \( b_{uv} \) can be obtained from past data, as in [7]; in our setting, we assume that such parameters have been obtained in advance.

Given a seed set \( S \) of subscribers, for every set of attributes \( F \), we can obtain the total number of activated nodes after running several trials of the diffusion process from \( S \) [26]. The expected number of activated nodes for a given seed set \( S \) and a selected set of attributes \( F \) is called influence spread, denoted as \( \sigma(F) \), or, as \( S \) is fixed in our problem, just \( \sigma(F) \). Thus, \( \sigma(F) \) is the expected spread of the diffusion, which we can calculate using live-edge instances of the graph (i.e., instances of activated-only edges [26]) as:

\[
\sigma(F) = \sum_X \text{Prob}[X] \cdot \sigma_X(F)
\]

(2)

where \( \sigma_X(F) \) is the influence spread in live-edge instance \( X \).

### 3.2 Content-Aware Influence Maximization

We define the CAIM problem as follows:

**Problem 1.** *Given a directed graph \( G = (V, E) \), where each node \( v \) is associated with user attributes \( F_v \) and a universe of possible attribute sets \( \Phi \), and each possible attribute set \( F \subset \Phi \) corresponds to a different probabilistic graph, on which we can compute the influence spread of the seed set \( S \); the attribute set \( F \) that achieves maximum spread constitutes the CAIM solution. We emphasize that, due to the drastic difference between classical IM and CAIM in the way influence spread is achieved, the solutions to these two problems cannot be qualitatively compared against each other.*

### 4 HARDNESS AND INAPPROXIMABILITY

We now show the hardness of the CAIM problem and study the properties of influence spread function \( \sigma(F) \). To calculate \( \sigma(F) \), we first calculate edge probabilities with respect to the selected content attributes \( F \) and then estimate the expected spread on the graph starting from the given set of subscribed nodes \( S \).

![Figure 1: A graph instance demonstrating that the CAIM problem is NP-hard.](image)

**Theorem 4.1.** *The CAIM problem with the Content-Aware Cascade model is NP-hard.*

**Proof.** Consider an instance of the NP-complete Set Cover problem, defined by a collection of subsets \( S_1, S_2, \ldots, S_m \), a universe of elements \( U = \{u_1, u_2, \ldots, u_n\} \) and an integer \( k \). We are asked whether there are \( k \) sets that will cover all elements in \( U \). We show that Set Cover can be reduced to a trivial instance of CAIM as follows: We construct a bipartite graph with one activated node on the left side that connects to \( n \) nodes on the right side, as shown in Figure 1. We map each member \( u_i \) of universe \( U \) to a node on the right side and add an attribute \( f_j \) to set \( F_{u_i} \) if \( u_i \) belongs to subset \( S_j \). We set \( b_{uv} = 0 \) and \( q_{uv} = 1 \) for all edges \((u, v) \in E\), i.e., a node \( v \) is influenced if at least one of its user attributes is selected. In this trivialized version of CAIM, the spread can be computed deterministically; there is no need for expected spread computations. Then, an algorithm that could optimally solve this trivial instance of CAIM, among others, would decide any instance of Set Cover: if we can target all nodes in the CAIM instance of a conference paper.
using $k$ attributes, we can in effect cover all elements in $U$ using $k$ subsets in Set Cover. Otherwise, if the optimal spread in CAIM does not reach all nodes, it follows that there is no set of $k$ subsets that covers all elements in Set Cover. Thus, by reduction from Set Cover, CAIM is at least as hard as any problem in NP. □

By Theorem 4.1, there is no polynomial-time algorithm to find an optimal set of attributes $F$, unless $P=NP$. We now proceed to study the properties of the influence spread function $\sigma(F)$.

A function $\sigma(F)$ is submodular if it follows a diminishing returns rule: the marginal gain from adding an element to a set $F$ is at most as high as the marginal gain from adding the same element to a subset of $F$. That is, $\sigma(F_1 \cup \{f\}) - \sigma(F_1) \geq \sigma(F_2 \cup \{f\}) - \sigma(F_2)$, where $F_1 \subset F_2 \subset \Phi$, for any $f \in \Phi$.

We call a transition function $h_{uv}(F_v, F)$ monotonic on $F$ if, for subsets of attributes $F_1 \subset F_2 \subset \Phi$, it holds that $h_{uv}(F_v, F_1) \leq h_{uv}(F_v, F_2)$, for any node $v$. If the transition function is not monotonic, then the influence spread function is neither monotonic, nor submodular, because selecting more attributes may reduce probabilities $p(u, v)$ and thereby reduce the total influence spread.

We assume that attributes have nonnegative effects on users, rendering the transition function $h_{uv}(\cdot)$ monotonic: edge probabilities can only increase if we add attributes to $F$, i.e. $h_{uv}(F_v, F) \leq h_{uv}(F_v, F + \{f\})$ for any $f \in \Phi$ and $v \in G$; hence $\sigma(F)$ is monotonic.

We now examine whether $\sigma(F)$ is also submodular. This turns out to not be the case, even for a monotonic and submodular transition function, as the following counterexample demonstrates.

\textbf{Figure 2: Increasing & decreasing marginal returns}

\textbf{Example 4.2.} Consider the graph on the left-hand side in Figure 2, with a universe of attributes $\Phi = \{A, B, C\}$, sets of preferred attributes for each node be $F_{V_1} = \{A\}$ and $F_{V_2} = F_{V_3} = \{A, B, C\}$, $h_{uv} = 0.5$, $q_{uv} = \frac{1}{2}$ on all edges, and one active node $s$. Then, consider two subsets of attributes $F_1 = \emptyset$, $F_2 = \{B, C\}$, where $F_1 \subset F_2$, and a attribute $f = A \in \Phi \setminus F_2$. The achieved spreads for each attribute subset, and the respective marginal gains obtained after adding attribute $f$ to subsets $F_1$ and $F_2$, are calculated as follows. For subset attribute $F_1$ selected, we have:

\begin{align*}
\rho_{SV_1} &= \frac{1}{2}, \\
\rho_{SV_2} &= \rho_{SV_1} = \frac{1}{2}, \\
\sigma(F_1) &= \frac{1}{2} + 2 \cdot \frac{1}{2} = 1
\end{align*}

whereas when $f = A$ is added to $F_2$, we get:

\begin{align*}
\rho_{SV_1} &= 1, \\
\rho_{SV_2} &= \rho_{SV_1} = \frac{1}{2}, \\
\sigma(F_1 + \{A\}) &= 1 + 2 \cdot \frac{1}{2} = \frac{3}{2}
\end{align*}

Hence $\Delta_1 = \sigma(F_1 + \{A\}) - \sigma(F_1) = \frac{3}{2} - 1 = \frac{1}{2}$. Similarly, for $F_2$ selected, we have:

\begin{align*}
\rho_{SV_1} &= \frac{1}{2}, \\
\rho_{SV_2} &= \rho_{SV_1} = \frac{1}{2}, \\
\sigma(F_2) &= \frac{1}{2} + 2 \cdot \frac{5}{8} = \frac{9}{4}
\end{align*}

while when $f = A$ is added to $F_2$, we get:

\begin{align*}
\rho_{SV_1} &= 1, \\
\rho_{SV_2} &= \rho_{SV_1} = \frac{1}{2}, \\
\sigma(F_2 + \{A\}) &= 3
\end{align*}

Hence $\Delta_2 = \sigma(F_2 + \{A\}) - \sigma(F_2) = 3 - \frac{4}{3} = \frac{5}{3}$. Since $\Delta_2 > \Delta_1$, the submodularity of $\sigma(F)$ does not hold.

Given this negative result, the influence function $\sigma(F)$ might have an increasing returns property (supermodularity), whereby it would hold that $\sigma(F_1 \cup \{f\}) - \sigma(F_1) \leq \sigma(F_2 \cup \{f\}) - \sigma(F_2)$, for $F_1 \subset F_2 \subset \Phi$ and any attribute $f \in \Phi$. The following counterexample shows that this property does not hold either.

\textbf{Example 4.3.} Consider the graph on the right-hand side in Figure 2, with a universe of attributes $\Phi = \{A, B\}$, sets of preferred attributes per node $F_{V_1} = \{A, B\}$ and $F_{V_2} = \{A\}$, $h_{uv} = 0.5$ and $q_{uv} = \frac{1}{2}$ on all edges, and one active node $s$. Consider two subsets of attributes $F_1 = \emptyset$ and $F_2 = \{B\}$. Then, for subset attribute $F_1$ selected, we have:

\begin{align*}
\rho_{SV_1} &= \frac{1}{2}, \\
\rho_{SV_2} &= \rho_{SV_1} = \frac{1}{2}, \\
\sigma(F_1) &= \frac{1}{2} + 1 \cdot \left(1 - \left(1 - \frac{1}{2}\right) \frac{1}{2}\right) = \frac{3}{4}
\end{align*}

wheras when $f = A$ is added to $F_1$, we get:

\begin{align*}
\rho_{SV_1} &= \frac{1}{2}, \\
\rho_{SV_2} &= \rho_{SV_1} = \frac{1}{2}, \\
\sigma(F_1 + \{A\}) &= \frac{1}{2} + 1 = \frac{3}{2}
\end{align*}

Hence $\Delta_1 = \sigma(F_1 + \{A\}) - \sigma(F_1) = \frac{3}{2} - \frac{3}{4} = \frac{3}{4}$. Similarly, for $F_2$ selected, we have:

\begin{align*}
\rho_{SV_1} &= \frac{1}{2}, \\
\rho_{SV_2} &= \rho_{SV_1} = \frac{1}{2}, \\
\sigma(F_2) &= \frac{3}{4} + 1 \cdot \left(1 - \left(1 - \frac{1}{2}\right) \frac{1}{2}\right) = \frac{23}{16}
\end{align*}

while when $f = A$ is added to $F_2$ we get:

\begin{align*}
\rho_{SV_1} &= 1, \\
\rho_{SV_2} &= \rho_{SV_1} = 1, \\
\sigma(F_2 + \{A\}) &= 2
\end{align*}

Hence $\Delta_2 = \sigma(F_2 + \{A\}) - \sigma(F_2) = 2 - \frac{23}{16} = \frac{9}{16}$. Since $\Delta_1 > \Delta_2$, the influence function $\sigma(F)$ is not supermodular either.

Eventually, we have established the following:

\textbf{Theorem 4.4.} The spread function $\sigma(F)$ with a probability transition function $h_{uv}(F_v, F)$ is neither submodular nor supermodular.

By Theorem 4.4, it follows that we cannot use a greedy algorithm with an approximation guarantee based on submodularity, as in [26]. Moreover, in the following we show that it is NP-hard to approximate the optimal solution to CAIM.

\textbf{Theorem 4.5.} It is NP-hard to approximate the optimal solution to the CAIM problem with the Content-Aware Cascade model within a factor $n^{1-\epsilon}$ for any $\epsilon > 0$.

\textbf{Proof.} Consider an instance of the Set Cover problem, in which we need to decide whether we can cover all elements of a universe $U = \{u_1, u_2, \ldots, u_k\}$ by selecting at most $k$ subsets out of a collection of $S_1, S_2, \ldots, S_m \subset U$. 


We then construct a graph $G$ for the CAIM problem with a single subscriber node $s$ and nodes $u_1, u_2, \ldots, u_n$ corresponding to elements in $U$, connected so that $u_{i-1}$ points towards $u_i$ for all $i = 2 \ldots n$, and $s$ is connected to $u_1$, and, for every subset $S_j$ an element $u_i$ belongs to, we add an attribute $f_j$ to the preferred attributes of $u_i$. Next, for some integer $c$ we add $\eta = n^c - n - 1$ more nodes $x_1, x_2, \ldots, x_\eta$ such that $u_n$ has outgoing edges to them and each $x_i$ has the same preferred attributes as $u_n$. Graph $G$, shown in Figure 3, has $N = n^c$ nodes. We set $b_{u,v} = 0$ and $q_{u,v} = 1$ for all edges, so that an edge becomes active if at least one of the attributes associated with its target node is selected. Then, if it is possible to select $k$ subsets that cover all elements of universe $U$, we can also have $N = n^c$ activated nodes. Conversely, if there is no selection of $k$ subsets that covers all $U$, then there is at least one node $u_i$ that does not get activated, precluding influence spread to nodes $x_1, x_2, \ldots, x_\eta$. We can then only target at most $n$ out of $n^c$ nodes, a fraction of $n^{-c} = N^{1/\eta}$. Thus, if we had a polynomial-time algorithm that approximated the optimal solution to CAIM within a factor of $N^c$ for any $\epsilon > 0$, then it would suffice to set $c = \lceil \frac{1}{\epsilon} \rceil$ and use that algorithm so as to decisively distinguish between a case that accepts a solution activating all $N$ nodes and one that does not, and thereby also decide Set Cover. Thus, by reduction from Set Cover, we have shown that it is NP-hard to approximate the optimal solution to CAIM within a reasonable factor. □

Figure 3: A graph instance demonstrating that it is NP-hard to approximate the optimal solution to the CAIM problem.

5 THE EXPLORE-UPDATE ALGORITHM

As it is NP-hard to approximate the CAIM solution within a factor of $n^{-c}$ with the Content-Aware Cascade model, we proceed to design heuristic solutions therefor. We structure our exposition as follows: we first present a simple, yet time-consuming greedy heuristic; then, through a sequence of simplifying assumptions, we will generate a much more efficient algorithm called Explore-Update.

Algorithm 1: Greedy($G, S, \theta, k$)

1. $F = \emptyset$
2. while $|F| < k$ do
3. for every $f \in \Phi \setminus F$ do
4. calculate $\sigma(F + \{f\})$ using Monte Carlo simulations
5. $F = F \cup \argmax_f \{\sigma(F + \{f\})\}$
6. return $F$

Our first proposal is a baseline greedy algorithm that selects the attribute of highest marginal gain to add at each iteration, shown in Algorithm 1. This is an adaptation of the Local Update algorithm in [6] to our problem. Intuitively, it is reasonable to greedily select the locally best attribute in each iteration, especially for small values of $k$. This kind of algorithm has been shown to achieve better quality than others in classical Influence Maximization [11, 13, 27].

Though simple and effective, Algorithm 1 is inefficient due to its calculation of influence spread by MC simulations. In a manner reminiscent of [13], we can improve efficiency by considering maximum influence paths between nodes and the seed set. We call a path $P_{\text{max}} = \langle u = u_0, u_1, u_2, \ldots, u_n \rangle$ between vertices $u \in S$ and $v \in G$ maximum influence path (MIP) if this path is the most probable among all paths between $u$ and $v$: $P_{\text{max}} = \arg\max_{P} \prod_{i=0}^{n-1} \text{prob}(u_i, u_{i+1})$. Under the simplifying assumption that influence is propagated only through MIPs, we can estimate influence spread in polynomial time as follows: For a threshold $\theta$ and a node $v$, we build a tree structure called in-arborescence $A_{\text{in}}(v)$, which includes all MIPs of probability higher than $\theta$ from any node to $v$: $A_{\text{in}}(v) = \{\text{MIP}(u, v) \mid \text{prob}(\text{MIP}(u, v)) > \theta, u \in G\}$. Then, given a node $u$, the seed set $S$, and an arborescence $A_{\text{in}}(v)$, Algorithm 2 recursively estimates the probability that $u$ is activated in $A_{\text{in}}(v)$, i.e., its activation probability $ap(u, A_{\text{in}}(v))$.

Algorithm 2: calculateAP($u, A_{\text{in}}(v), S$)

1. if $u \in S$ then
2. $ap(u, A_{\text{in}}(v)) = 1$
3. else if $u$ has no in-neighbors in $A_{\text{in}}(v)$ then
4. $ap(u, A_{\text{in}}(v)) = 0$
5. else
6. $ap(u, A_{\text{in}}(v)) = 1 - \prod_{u \in \text{in}(u)} \left(1 - ap(\text{in}(u), A_{\text{in}}(v)) \cdot \text{prob}(\text{in}(u), u)\right)$
7. return $ap(u, A_{\text{in}}(v))$

Based on these calculations, for all nodes $u \in G$, we can calculate the influence spread $\sigma(F)$ as follows:

$$\sigma(F) = \sum_{u \in G} ap(u, A_{\text{in}}(u)) \tag{3}$$

We can then employ Equation (3) so as to estimate influence spread in Algorithm 1, in lieu of MC simulations, deriving Algorithm 3; at each iteration, we compute the in-arborescence of node $u$ for a given threshold $\theta$ by converting each probability $p_e$ on an edge $e$ to $-\log p_e$ and employing an efficient implementation of Dijkstra’s algorithm. If computing an arborescence takes time $t$, then Lines 4-6 take $nt$ and the total time is $O(k|\Phi|nt)$.

Algorithm 3: Arb($G, S, \theta, k$)

1. $F = \emptyset$
2. while $|F| < k$ do
3. for every $f \in \Phi \setminus F$ do
4. compute $A_{\text{in}}(u)$ with threshold $\theta$
5. $ap(u, A_{\text{in}}(u)) = \text{calculateAP}(u, A_{\text{in}}(u), S)$
6. $\eta = 0$
7. $\sigma(F + \{f\}) = \sum_{u \in G} ap(u, A_{\text{in}}(u))$
8. $F = F \cup \argmax_f \{\sigma(F + \{f\})\}$
9. return $F$

We further reduce the runtime of Algorithm 3 by eschewing redundant iterations of the loops over nodes $u$ and attributes $f$. First, we limit the calculation of in-arborescences and activation probabilities only to nodes whose in-arborescence under threshold $\theta$ reaches at least one node in $S$; only such nodes can yield non-zero estimated activation probability. To find out these nodes, we compute the out-arborescence of all nodes in $S$, $A_{\text{out}}(S)$, consisting of all MIPs of probability higher than $\theta$ from a node $v \in S$ to other nodes in $G$. Nodes in $A_{\text{out}}(S)$ yield non-zero activation probability...
estimates. Yet the set of paths in $A_{out}(S)$ may contain directed loops, hence we cannot apply a recursive algorithm like Algorithm 2 directly on $A_{out}(S)$; we still need to obtain the in-arborescence $A_{in}(u)$ of each $u \in A_{out}(S)$; we do so while building $A_{out}(S)$, by adding MIP($u$, $v$) to $A_{in}(u)$ for each $u \in A_{out}(v)$. Algorithm 4 illustrates this Explore process.

Algorithm 4: Explore($G, F, S, \emptyset$)

1. $A_{out}(S) = \emptyset$
2. $A_{in}(u) = \emptyset$ for every $u$ in $G$
3. For every $v$ in $S$ do
   4. compute $A_{out}(v)$ for given $0$ and $F$
   5. update $A_{in}(u)$ for each $u \in A_{out}(v)$
6. return $(A_{in}(u) \neq \emptyset \text{ for } u \in G)$

Then we can calculate influence spread $\sigma(F)$ using the union of such in-arborescences, $A_{in}$, by Algorithm 5.

Algorithm 5: Update($A_{in}$, $S$)

1. For every $u$ in $A_{in} do$
   2. $ap(u) = \text{CalculateMIP}(u, A_{in}(u), S)$
   3. $\sigma(F) = \sum_{u \in A_{in}} ap(u)$
4. return $\sigma(F)$

Second, we limit the calculation of marginal gain in Algorithm 3 only to those attributes that can affect the influence spread. We call an edge in $G$ participating, if at least one of its endpoints are in $A_{out}(S)$. Figure 4 presents a graph for a seed set $S$ (and selected attributes set $F$) in the green area; the yellow area includes nodes in $A_{out}(S)$; the set of participating edges $\Pi$ is shown in solid and dotted lines; dotted edges have only one endpoint in $A_{out}(S)$; non-participating edges are shown in dashed lines, in the gray area.

Figure 4: Participating and non-participating edges

Non-participating edges cannot increase influence spread, regardless whether their probability is increased; only participating edges have such potential. We limit the attributes Algorithm 3 considers based on this observation. Let $E(f)$ be the set of edges that include attribute $f$ among their preferred attributes, hence their probability is affected when adding $f$ to $F$. Then, at any iteration, if none of the edges in $E(f)$ is a participating edge, i.e., $E(f) \cap \Pi = \emptyset$, then attribute $f$ need not be examined as a candidate to be added to $F$; it bears no effect to influence function $\sigma(F + \{f\})$.

Putting together our enhancements to Algorithm 3, we design the polynomial-time Explore-Update algorithm (Algorithm 6). In a nutshell, at each iteration, this Explore-Update algorithm selects the hitherto unselected attribute $f$ affecting participating edges that brings about the largest increase of influence spread, using the Explore procedure for calculating in-arborescences and the Update procedure for calculating influence spread, while updating the set of participating edges $\Pi$ at each iteration and using it to determine which attributes need to be examined at the next iteration.

Algorithm 6: Explore-Update($G, S, k, \emptyset$)

1. $F = \emptyset$
2. $A_{in} = \text{Explore}(G, F, S, \emptyset)$
3. $\Pi = \{(u, v) \in G | u \in A_{in} \text{ or } v \in A_{in}\}$
4. While $|F| < k$ do
   5. for $f \in \Phi \setminus F$ do
     6. if $E(f) \cap \Pi = \emptyset$ then
     7. $A_{in} = \text{Explore}(G, F + \{f\}, S, \emptyset)$
   8. $\Pi_f = \{(u, v) \in G | u \in A_{in} \text{ or } v \in A_{in}\}$
   9. $\sigma(F + \{f\}) = \text{Update}(A_{in}, S)$
10. $f_{max} = \text{argmax}(\sigma(F + \{f\}))$
11. $F \leftarrow F \cup f_{max}$
12. $\Pi \leftarrow \Pi \cup \Pi_f$
13. return $F$

Let the time complexity to calculate an out-arborescence for node in $S$ be $t_{out}$, then the Explore procedure takes $|S|t_{out}$ and the Update procedure takes $O(n_{ing}n_{out})$ time, where $n_{ing}$ is the number of nodes in in-arborescences, and $n_{out}$ is the number of nodes in out-arborescence of $S$. Therefore, if we perform $\kappa$ calculations of $A_{in}$ per iteration, the total runtime is $O(k|S|t_{out} + n_{ing}n_{out})$. In effect, the Explore-Update algorithm is expected to perform well when the size of arborescences is small, and the number of updates $\kappa$ per iteration is smaller than $|\Phi|$. As propagation probabilities on edges are usually small in real networks, the size of arborescences is expected to be small. The number of updates depends on the structure of the network. In a large-diameter network where multiple hops are required to reach most nodes from $S$ via a MIP, there is a good chance to reduce the number of computations significantly. We investigate this matter experimentally in the following.

6 EXPERIMENTAL STUDY

In this section we present a comprehensive experimental study on the Greedy and Explore-Update algorithms we have introduced. All experiments were run on a 32GB Intel Core i5-2450M CPU machine @ 2.56GHz, while algorithms were implemented$^2$ in C++.

As there is no previous work on the CAIM problem, we compare to basic baselines. Still, as we discussed, the previous work that comes closest to our problem is that by Barbieri and Bonchi [6]; yet that work solves primarily the problem of selecting a set of seed nodes, and secondarily a set of product attributes, so as to maximize product influence in a network. The best-performing algorithm for updating an attribute set in [6], Local Update, performs one addition or removal of an attribute to/from the current attribute set at each iteration; in effect, our Greedy algorithm can be considered as an adaptation of Local Update to our problem, where only additions of attributes are needed. Therefore, to the extent that a comparison to [6] is possible, we conduct it via the comparison to the Greedy algorithm itself. Another method for updating an attribute set proposed in [6], Generic Update, is a hard-to-tune genetic algorithm, which may lead to an unpredictable number of output attributes. Besides, as the experimental study in [6] shows, Genetic Update offers no qualitative advantage while it is much slower than Local Update, which is already far the most time-consuming algorithm in our study. Therefore, we do not consider a genetic algorithm in our experimental study.

$^2$Documentation and code is available at https://github.com/mdf141/Explore-Update
Diffusion models. In the Content-Aware Cascade model the probability on edge \((u, v)\) is a linear function of product and base probabilities \(q_{uv}\) and \(b_{uv}\). To assign these probabilities we use two techniques prevalent in previous work [13].

- **Weighted Cascade model**: probability \(q_{uv}\) is assigned to edge \((u, v)\), where \(d_v\) is the in-degree of node \(v\). We use this model for the sake of compatibility with previous works, even while it may fit less to our problem setting.
- **Multivalency model**: the probability for edge \((u, v)\) is drawn uniformly at random from a set of probabilities. We choose that set to be \([0.02, 0.04, 0.08]\).

We calculate \(b_{uv}\) for every edge \((u, v)\), and set \(q_{uv} = \frac{b_{uv}}{F_{uv}}\).

Algorithms. We compare the Explore-Update algorithm under different threshold \(\theta\) values to three other algorithms:

- **Greedy** This is Algorithm 1 in this paper, which is effectively an adaptation of *Local Update*, the best algorithm in [6]. A similar algorithm has been used extensively in the context of the Influence Maximization problem, and always demonstrated top performance in terms of spread, while being slower than other heuristics [11]; it requires specifying the number of Monte-Carlo simulations to calculate influence spread, as we do in the following.
- **Top-Nodes** This algorithm measures each attribute’s frequency among node preferences and selects the \(k\) most frequent ones.
- **Top-Edges** This algorithm assigns to each edge \(e = (u, v)\) the attribute preferences of node \(v\), \(F_{uv}\), and select the \(k\) most frequent attributes across all edges.
- **Brute-Force** This algorithm finds all possible sets of attributes of size \(k\), computes each one’s influence spread using Monte-Carlo simulations, and opts for the best. Because the solution space is exponential, we use this method on reduced datasets.

Datasets. We run experiments in two real-world networks. The first network is a peer-to-peer file sharing directed network Gnutella\(^3\), where nodes represent hosts and edges represent connections between the Gnutella hosts. Our second network is extracted by crawling the social network VK\(^4\), nodes are users and edges are friendships among them. Statistics are presented in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Gnutella</th>
<th>VK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>10,876</td>
<td>7,420</td>
</tr>
<tr>
<td>Edges</td>
<td>15,959</td>
<td>17,688</td>
</tr>
<tr>
<td>Average Clustering Coefficient</td>
<td>0.0062</td>
<td>0.02</td>
</tr>
<tr>
<td>Number of Triangles</td>
<td>934</td>
<td>168,284</td>
</tr>
<tr>
<td>Diameter</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>Attributes/Seed sets</td>
<td>151</td>
<td>3,882</td>
</tr>
<tr>
<td>Default Seed Size</td>
<td>54</td>
<td>11</td>
</tr>
</tbody>
</table>

\(^3\)https://snap.stanford.edu/data/p2p-Gnutella04.html
\(^4\)https://vk.com/
6.1 Influence spread

Figures 5(a-b) present our results on competing algorithms’ influence spread\(^3\) on the Gnutella network, varying number of selected attributes \(k\) from 1 to 51. We used 10000 MC simulations for Greedy, and \(\theta = 1/320\) for Explore-Update. We observe that Explore-Update arrives just 1% and 5% below the performance of Greedy with the Multivalency and Weighted Cascade model, respectively. On the other hand, the Top-Edges and Top-Nodes algorithms reach only 88% and 85% of the spread of Explore-Update. Figures 5(d-e) present influence spread in VK network. Now Greedy used with just 500 Monte-Carlo simulations comfortably achieves 15%, 37%, and 48% higher spread than Explore-Update with \(\theta = 1/40\), Top-Nodes, and Top-Edges, respectively, in MV model. The picture is similar with the WC model, where Greedy achieves spread 9%, 38%, 40% higher than Explore-Update, Top-Nodes, and Top-Edges. Overall, our results confirm that Explore-Update achieves high influence spread for networks where the local neighborhood of the seed set has structure amenable to long distance arborescences.

6.2 Runtime

We now compare algorithms in terms of runtime. Figure 5(c) presents the results with Gnutella for \(k = 50\); Explore-Update (\(\theta = 1/320\)) runs an order of magnitude faster than Greedy (10000 simulations); Top-Edges and Top-Nodes output a selected set in less than a second, hence we do not include them. Next, we investigate how the algorithms scale with increasing network size. We extract subnetworks of VK consisting of 1855, 3710, and 7420 nodes of the original network (i.e., 1/4, 1/2, and full network) and proportional edge density to the full network. In all cases, we compute the runtime on a seed set \(S\) of size 15, with the Multivalency model for \(k = 20\), for Greedy (10000 simulations), Explore-Update (\(\theta = 1/40\)), and the Top-Edges and Top-Nodes heuristics. Figure 5(f) shows that runtime scales linearly in network size in all cases. Moreover, we ascertain that while Explore-Update fares no better than Greedy in terms of influence spread, it is much faster.

6.3 Effect of Seed Size

We now test the performance of Explore-Update for different sizes of the seed set \(S\). We select different seed sets from size 21 (minimal size for the current block partition) to 101 with step 10 on Gnutella. Figure 6(a-b) presents the influence spread for Explore-Update and Top-Edges for \(k = 50\), as well as the runtime of Explore-Update, whereas Greedy is orders of magnitude slower for this setup, and Top-Nodes performs worse than Top-Edges. We note that Explore-Update always achieves better influence spread than Top-Edges. Interestingly, influence spread and runtime do not always grow with \(|S|\). This is explicable by the fact that different seed sets induce different local structures.

6.4 Effect of \(\theta\)

Next, we study the effect of the \(\theta\) threshold, which controls the size of arborescences and thereby the influence spread achievable from seed set \(S\). Figure 6(c) presents the influence spread and runtime with the Gnutella network for \(\theta\) in \(\{1, 1/2, \ldots, 1/100\}\), with the WC model for \(k = 50\). The runtime of Explore-Update grows logarithmically in the inverse threshold \(\theta\), while influence spread grows logarithmically in it. A good tradeoff between quality and runtime is found at the knee point in the influence spread curve for \(\theta = 1/30\).

6.5 Comparison to the Optimal Solution

By Theorem 4.5, we proved it is NP-hard to approximate the optimal solution to CAIM. Now, we compare the results of heuristics to the optimal solution obtained by brute force; we reduce the total number of attributes to 16 and use a reduced Gnutella network by selecting 2k nodes, yielding similar degree distribution properties to the original. Figure 7 shows the influence spread results, with the Multivalency model, for a random seed set of size 10. Remarkably, Explore-Update finds the optimal set of attributes with varying \(k\).

---

\(^3\)We use 10000 Monte-Carlo simulations to compute the final spread of all solutions.
We formulated a content-aware cascade model and showed that the problem of content-aware influence maximization is NP-hard and inapproximable, while the influence spread from fixed points of departure is maximized, whereby different attributes of a propagated meme’s content, such that its spread across a network is maximized, is liked by seed nodes does not yield good results. Nevertheless, selected attributes exhibited a remote, yet unpredictable, resemblance to the attributes liked by seed set nodes. For example, with a group titled “La vie et l’amour” as seed, the selected attributes in our VK network sample included “Home Comfort | Design | Interior Design | Style”. With “Psychology of Relations” as seed, the selected attribute set included “Philosophy of Life”. Such analogies between seed set and selected attributes, while retrospectively intuitive, would not be derived otherwise; they depend on the way nodes of diverse interests interact within the overall network structure. Such results vindicate our problem motivation.

We also checked how result sets change when we vary k. For example, we select 100 out of 431,374 subscribers of “Esoterica YOGA MEDITATION” as seed set. With k = 3, the selected attributes are [“MODA”, “La vie et l’amour”, “Blog for Men”]. As “Esoterica YOGA MEDITATION” targets primarily women, results such as “MODA” and “La vie et l’amour” are unsurprising. Nevertheless, interestingly, both E-U and Greedy also return “Blog for Men” as a selected attribute, whereas the simple Top-Nodes and Top-Edges heuristics do not. This result shows that our algorithm can select nontrivial attributes.

### 7 CONCLUSION

This paper proposed the problem of content-aware influence maximization (CAIM). The goal is to select k attributes that characterize a propagated meme’s content, such that its spread across a network from fixed points of departure is maximized, whereby different attribute sets yield different propagation probabilities across network edges. To our knowledge, there is no previous work on this problem. We formulated a content-aware cascade model and showed that the problem is NP-hard and inapproximable, while the influence function is neither submodular, nor supermodular. We developed an efficient algorithm for CAIM using bounded local arborescences to calculate influence spread. Our experimental study demonstrates that this Explore-Update algorithm selects topics sets that achieve high spread and is orders of magnitude faster than a conventional Greedy solution resembling algorithms developed for related problems. We also provide evidence that our E-U algorithm can achieve the optimal solution when the number of selected topics is small.

In the future, we plan to study other propagation models and investigate the parallelization of Explore-Update.

### REFERENCES


