Influence Maximization in a Real-Time Bidding Environment

David Dupuis
Kwanko & Léonard de Vinci Pôle Universitaire, Research Center, CNAM
Paris, France
david.dupuis@devinci.fr

Cédric du Mouza
CNAM
Paris, France
cedric.dumouza@cnam.fr

Nicolas Travers
Léonard de Vinci Pôle Universitaire, Research Center
Paris, France
nicolas.travers@devinci.fr

Gaël Chareyron
Léonard de Vinci Pôle Universitaire, Research Center
Paris, France
gael.chareyron@devinci.fr

ABSTRACT

Influence Maximization (IM) is a well studied maximum coverage problem, which consists in finding in a network the top-k influencers who will maximize the diffusion of a piece of information. It is commonly associated with basic online advertising strategies. However, today, the exponential growth of online advertising is due to Real-Time Bidding (RTB) which allows advertising agencies to target specific users with specific ads on specific webpages. It requires complex ad placement decisions in real-time to face a high-speed stream of online users. In order to stay relevant, the IM problem should be updated to answer RTB needs. While most traditional IM methods generate a static set of top-k influencers, they do not deal with the topic of influence maximization in a real-time bidding environment which requires dynamic influence targeting.

This paper studies this topic and proposes RTIM, the first IM algorithm capable of taking influence maximization decisions within a real-time bidding environment. We also analyze influence scores of users in several social networks in order to show their impact in our approach. Finally, we offer a thorough experimental process to compare static versus dynamic IM solutions *wrt.* influence scores.

KEYWORDS

Real-Time Bidding, Influence Maximization, Social Network

1 INTRODUCTION

Since Kempe et al. [1], Influence Maximization (IM) is a well studied maximum coverage problem which consists in finding the smallest subset of individuals in a social network whom when targeted with a piece of information will maximize its diffusion through social influence. Simply put, IM attempts to maximize the influential impact of a set of users in a social network. The term "*influence*" is defined as "*the power of causing an effect in indirect or intangible ways*" (*Merriam-Webster*). In this paper, we consider that a user can be influenced because he saw the ad, interacted with it, purchased the product or was encouraged to do so in the future.

This article has been published in the WISE'19 conference

However, today, online advertising revenue outpaced all other advertising strategies thanks to the advent of real-time bidding (RTB) [2, 3] and social network services (SNS). SNSs allow individuals to stay connected with more people, share more, and on a world scale. RTB is an online auction system which allows advertisers to bid in real-time for ad locations on a webpage loaded by users; this facilitates specific user targeting. In RTB, advertisers see a stream of users, one at a time and have less than 100 ms [2] to decide to bid or not. If the bid is won, the ad is displayed immediately. The bid winner cannot cancel or resale the bid offer. The bidders do not know what the auction landscape looks like and therefore cannot predict which user will appear in the stream.

Our observation is that in order to keep influence maximization relevant it is important that it should, now, take into consideration the time and targeting requirements of real-time bidding.

In addition, RTB ad targeting is initially based on the content of the web page and users' consumer profile, their intrinsic value, but fails to take into consideration the social value of each customer as suggested by Domingos and Richardson [4]. As far as we know no RTB algorithms attempt to find an IM solution to improve bidding decisions. However, in this article, we focus on developing an IM algorithm capable of running with RTB constraints. It is important to note that there is potential here for IM to integrate bidding in order to improve RTB, but this approach is left for future work.

Existing IM algorithms propose, based on propagation models, various optimization technics to statistically choose a seed set of users that maximizes influence. However, as far as we know, no existing IM algorithm can work within a real-time bidding environment and satisfy it's requirements. Indeed, whereas existing algorithms take hours or days to find seed sets up to 200 seeds in a large social network [5], they do not cope with ad campaign requirements with thousands of users or take into account the fact that chosen users can be available online or not. The maximization problem needs to rely on both propagation and real-time decision.

This articles targets the issue with the following constraints:

- *Real-Time Bidding*: Only an online user can be targeted,
- *Processing Time*: Deciding whether to target a user must be done in under 100ms,
- *Social Networks*: The propagation influence score relies on a social network containing millions of users and relationships,
- *Influence Maximization*: Thousands of users must be targeted in real-time while maximizing scores of large seed sets.

Therefore, to target influential users in a RTB environment it is necessary to develop an IM algorithm capable of deciding in real-time which users are worth targeting.

To achieve this we propose the RTIM approach which stands for `Real-Time Influence Maximization`. RTIM is an IM algorithm which decides in real-time the effectiveness of an online user u, while static IM models only verify if this user u has been chosen in the pre-computed seed set.

Our main contributions are as follow:

- We propose an elegant approach for real-time influence maximization focusing on the stream of online users,
- We provide a deep analysis of users' influence scores for various social network datasets in order to showcase users' behavior in IM,
- We set up a thorough experimental setting for RTIM and IMM models on different social networks.

In this article, we first explain the state of the art on influence maximization. We then explain the two stages of our algorithm: pre-processing and live, how they relate to each other and allow us to solve the influence maximization problem under RTB constraints. Furthermore, we present the implementation of the different stages of the algorithm. Finally, we go through the experimental process allowing us to compare our algorithm which makes use of a dynamic stream of available users compared to the other state of the art algorithms which are evaluated using a static social network, supposing that all users are necessarily available.

2 IM STATE OF THE ART

Influence Maximization takes place in a social network graph $\mathcal{G} = (V, E)$ where V is the set of vertices (users), E the set of directed edges (influence relationships). In this graph $\mathcal{G}$, a user is **activated** if he has successfully been influenced by a neighbor and therefore influences his own outgoing neighbors. A **targeted user** is a user who is not yet activated but for whom a piece of information is shown to be propagated.

The goal of IM is to produce a **seed set** $\mathcal{S}$ of targeted users which maximizes its influence on $\mathcal{G}$. The optimal seed set, or the final result is defined as $\mathcal{S}^*$.

2.1 Propagation models

In their seminal work Kempe et al. [1] propose two common propagation models: *Independent Cascade* (IC) and *Linear Threshold* (LT). The *IC* model considers that each user can be influenced by a neighbor independently of any of his other neighbors. The *LT* model considers that a user is activated if the sum of successful influence probabilities from his neighbors is greater than his activation threshold.

Under the IC model, time unfolds in discrete steps. At any timestep, each newly activated node $u_i \in V_a, \forall i \in V$ gets one independent attempt to activate each of its outgoing neighbors $v_j \in Out(u_i), \forall j \in V\{i\}$ with a probability $p(u, v) = e_{ij}$. In other words, e_{ij} denotes the probability of u_i influencing v_i.

As explained in [6] there is a real challenge in acquiring real-world data to build datasets containing accurate influence probabilities. Therefore theoretical edge weight models may be assumed, like in the following edge weight models for the *IC* model:

- **Constant:** Each weight e_{ij} has a constant probability. In most solutions, p is set at 0.01 or 0.1, see [1, 7–11]. Some define $p \in [0.01, 0.1]$ [12, 13].
- ***Weighted Cascade (WC):*** In this model, $e_{ij} = \frac{1}{|In(v_j)|}$ where $In(v_j)$ is the number of neighbors that influence u. Thus, all neighbors that influence u_i do so with the same probability. Therefore, it is easier to influence a user with a low in-degree [1, 7–10, 12, 14–17].
- ***Tri-valency Model:*** Here, the weight of edges is randomly chosen from a list of probabilities such as {0.001, 0.01, 0.1} [7, 14, 18]. In very large or dense networks, such as `Twitter`, the tri-valency model may actually have edge weights far greater than the weighted cascade model due to the number of neighbors each user has.

For the *LT* model the general edge weight rule is that the sum of the weights must equal one. Therefore, the WC model applies to LT. Additional alternative models can be found in [19] where an extensive IM state-of-the art is done.

The *IC* model is very useful to model information diffusion when a single exposition to a piece of information from one source is enough to influence an individual. It is also a simpler model to study than *LT*. The LT model doesn't change the fundamental approach of our algorithm and we believe that it should be simple to extend it to LT. For these reasons, we limit our approach to *IC*. In addition, we define the edge weights using the *WC* model, because it corresponds better to the simulation of the diversity of influence between individuals in a real-world social network.

2.2 Properties

Kempe et al. [1] prove that the influence maximization problem is a monotone and sub-modular function. It is also an NP-Hard problem under both the *IC* and *LT* models. Chen et al. [7] prove that computing the influence score of a seed set is #P-Hard under the *IC* model.

In fact, *IC* and *LT* models add users to the seed set and the weighting propagation automatically increase its global influence score which corresponds to the positive monotone property.

Moreover, the propagation function f is sub-modular if it satisfies a natural diminishing returns property i.e., the marginal gain from adding an element v to a set $\mathcal{S}$ is at least as high as the marginal gain from adding the same element to a superset of $\mathcal{S}$. Formally, a sub-modular function satisfies: $\forall S \subseteq T \subseteq \Omega$ and $x \in \Omega \backslash T, f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$. This sub-modular property is essential as it guarantees that a *greedy* algorithm will have a $(1 - 1/e - \epsilon)$ approximation to the optimal value [20]. As it is presented in this state of the art, many IM algorithms rely on this theoretical guarantee to validate their strategy.

2.3 Computing score

Influence Score: Authors in [21] elaborate the exact influence spread function for the IC model as an inclusion-exclusion based equation. We generalize their inclusion-exclusion based equation into equation 1.

In Equation 1, the influence score of a seed set $\mathcal{S}$, of size k, is defined as the sum of activation probabilities $a_{\mathcal{S}}(v)$ of any node $v \in V$ when users in $\mathcal{S}$ are targeted. The activation probability of a user is the probability that there exists a path between that user and any targeted user. As we write in Equation 1, the probability that a path exists is the union of the existence of any path between a user and any targeted user. It's clear here that computing this formula is exponential in complexity.

$$\sigma(\mathcal{S}) = \sum_{v_i \in V} a_S(v_i) = \sum_{v_i \in V} \mathrm{P}(\bigcup_{p_j \in P_{uv_i}} p_j), \qquad (1)$$
$$P_{uv_i} = \{all\ paths\ event\ existence\ between\ u\ and\ v_i\}$$

Determining the seed set $\mathcal{S}$ of k users among N which provides the maximum global influence score $\sigma(\mathcal{S})$ is a NP-hard problem since it requires computing the global influence score of any combination of k users, so there are $\binom{N}{k}$ combinations to test for any given k. Even when assuming the seed set is known, computing its global influence score has been proven to be #P-Hard [18]. Due to the exponential nature of computing the influence score, Kempe et al. [1] offer an alternative which consists in running $n = 10,000$ influence propagation simulations and averaging the scores into a final influence score result. This is termed the *Monte Carlo* approach and we write an influence score computed with this method as $\sigma_{MC}()$. This estimation method allows us to produce a good approximation for the influence score and we can thus efficiently compute:

$$\sigma_{MC}(\mathcal{S}) \sim \sigma(\mathcal{S})$$

In IM there are two major challenges:

- The first challenge is finding the best seed set in the exponential universe of possibilities, among $2^{|V|}$ subsets of users. As mentioned previously, even defining k leaves $\binom{N}{k}$ subsets to test.
- The second challenge is computing the influence score (#P-Hard) of any given seed set.

No matter their strategy, overcoming these two challenges is the objective of any influence maximization algorithm, which we will now discuss.

2.4 Algorithms

Clearly presented by Arora et al. [5] there are three main categories of IM algorithms: greedy, sampling and approximation.

GREEDY [1], CELF [22] and CELF++ [11] are all three lazy-forward algorithms which take advantage of the sub-modularity property of the IM problem proven by [1] and thus guarantee an approximation of $(1 - 1/e - \epsilon)$. To find $\mathcal{S}^*$ they start with $\mathcal{S} = \emptyset$ and incrementally add to $\mathcal{S}$ the node v which brings the largest marginal gain: $\sigma_{MC}(\mathcal{S} \cup v) > \sigma_{MC}(\mathcal{S})$, until $|\mathcal{S}| = k$. However, continuously computing $\sigma_{MC}(\mathcal{S})$ is costly. CELF and CELF++ attempt to remedy this by storing certain scores to take advantage of the sub-modular property and avoid recomputing other scores but this doesn't provide any significant gain in runtime. Thus, those greedy algorithms do not scale for large seed sets or large graphs.

Borg et al's method [23] (referred to as RIS: *Reverse Influence Sampling*), TIM, TIM+ [24], or IMM [16] use topological sampling. In the transpose graph, they generate a set $\mathcal{R}$ of size θ of random paths of greatest influence by picking users uniformly at random (Reverse Reachable (RR) sets). Using a greedy method, they build $\mathcal{S}^*$ by continuously adding to $\mathcal{S}^*$ the user who covers the greatest number of RR sets and removing them from $\mathcal{R}$. As shown by [5] sampling algorithms are significantly faster because $\sigma_{MC}(\mathcal{S}^*)$ is only computed once the final solution is found. However, their theoretical guarantee depends on θ which is computed by ϵ in $(1 - 1/e - \epsilon)$ and l which determines there runtime factor $l^2 * log(n)$. Experimentally they use $\epsilon = 0.5$ and $l = 1$ which means that precision is sacrificed for scalability.

Approximation algorithms such as EaSyIM [9], IRIE [18], SIMPATH [25], LDAG [26] or IMRANK [14], offer heuristics to compute $\sigma(\mathcal{S})$. Instead of computing the union of all paths as indicated in Equation 1 they consider the most probable path, or the independence of all paths. They can run greedy or select the top k influential users. While scalable, as proven by [5], they do not provide theoretical guarantees.

Conclusions: All of these algorithms provide proper scientific solutions to solve the IM problem. The common rule is that an algorithm which has high accuracy will take weeks to find its seed set and vice-versa an algorithm which runs in a couple of hours will have less accuracy. However, these algorithms compute the seed set in a static graph environment and assume that every user must be available at any time to be targeted. This approach is not appropriate to graphs with more than tens of millions of users with proportionally many edges [5]. There exists a great number of specific IM contributions which have been listed in [19]. It shows clearly, that very few contributions have been made regarding the analysis of the IM challenge in a stream of online users. We must notice [27] which proposes an interesting solution with sliding windows that computes local Influence Maximization. However it does not scale up for large seed set lists; more than 100 while thousands are required.

Therefore, none of the existing IM solutions can perform well under real-time bidding constraints. Indeed it requires that all users in $\mathcal{S}$ appear during the campaign to guarantee the maximization of the static IM strategies. In addition to lacking theoretical guarantees, these algorithms compute the seed set in a static environment rather than a dynamic one, as we hope to achieve, with RTB constraints.

To this end, we offer the algorithm RTIM: *Real-Time Influence Maximization.* RTIM targets influential users in real-time, henceforth generating a seed set of influencers under real-time bidding constraints. To ensure this, RTIM takes places in two stages: a pre-processing stage and a live stage which we now present.

3 RTIM APPROACH

RTIM is meant to perform in a RTB environment. The latter, consists of users who, through their devices connected to the Internet, are navigating on websites. As soon as a user arrives on a webpage which sells its ad slots through a real-time bidding environment, the IM algorithm has to determine wether it is useful for targeting. In minutes, millions of users are quickly navigating through many dozens of websites each. This enormous online activity, for any RTB advertising agency appears as a continuous stream of online users [2]. Advertisers can only target, with advertisements, users who appear in the stream and target them under 100ms which corresponds to the bidding platform delay. As we know, these users, all belong to a very large social network through which they may be sharing information and influencing one another. It is therefore in the advertisers interest to take advantage of the social network value of each user that appears in the RTB stream. In addition, these same advertisers, with large budgets seek to acquire or convert (i.e. activate) many users, most through targeting.

The originality of our approach lies in its ability to target users who appear in this dynamic stream by estimating whether they will have a significant gain based on previously targeted users in the same stream and belonging to the same social network. Thus providing a solution to the influence maximization problem and producing a large set of users for a realistic ad campaign. While traditional approaches determine the best seed set of targeted users, at the cost of expensive computation, by processing a static graph in which any user is considered online and available for targeting at any given moment in time. Contrary to these solutions, RTIM allows us to adapt our influence maximization strategy to an advertisement campaign taking place in a RTB streaming environment and which requires targeting tens of thousands of users.

Furthermore, we choose to compare RTIM with the static algorithm IMM ("Influence Maximization with Martingales" [16]). Indeed, IMM has proven that it can compute an effective seed set in reasonable time regardless of the graph size. Even more so, it can compute large seed set sizes of tens of thousands of users as a RTB advertising campaign requires it.

Static algorithms, such as IMM [16], correspond to an optimistic approach where they assume that the users from their pre-computed seed set will necessarily be online in the stream. However, without integrating a probabilistic model based on real and precise figures about the connection rate of the different users, which is hardly possible on large social networks like `Twitter`, many users of the pre-defined seed set won't be available to target during the advertisement campaign. In contrast, our greedy approach, which can be considered as a pessimistic approach, allows us to dynamically fill our seed set with online users of interest for the advertisement campaign.

Our RTIM algorithm is composed of two steps. First a pre-processing step which computes the influence score of every user in the graph. Second, when reading the dynamic stream in real-time (called the "live stage"), for each user in the stream, we determine whether his influence score is high enough or the probability of him being activated by a previously targeted user is low enough. When a user is targeted during this live stage, we update the activation probability of the users in his neighborhood.

3.1 Step I: Pre-processing - Building the Influence Graph

First, we attribute a weight to each edge which estimates the influence that depends on the number of incoming edges of a vertex. This influence estimation between direct neighbors is commonly adopted in influence propagation [13]. We call this graph the *influence graph* $\mathcal{G}_I(V, E, w_I)$ defined formally as follows:

Definition 1 (Influence graph). *Consider $\mathcal{G}(V, E)$ the social graph where V is the set of vertices and $E \subseteq V^2$ is the set of oriented edges. The* influence graph *for $\mathcal{G}$ is the graph $\mathcal{G}_I(V, E, w_I)$ with the same sets of vertices and edges and a weighted function $w_I : E \longrightarrow \mathbb{R}$ such that for an edge e_{ij} from vertice v_i to v_j:*

$$w_I(e_{ij}) = \frac{1}{indegree(v_j)}$$

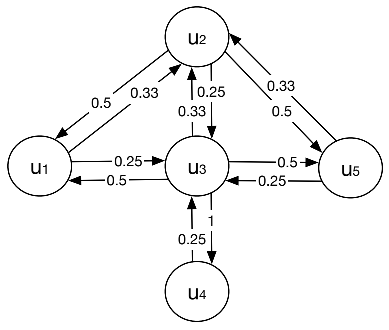

Figure 1: Influence graph $\mathcal{G}_I$ with weighted edges

Figure 1 depicts the influence graph for a social network between 5 users. For instance, user u_2 who follows or is influenced by users u_1, u_3 and u_5 has each of his incoming edge $e \in E$ weighted by:

$$w_I(e_{12}) = w_I(e_{32}) = w_I(e_{52}) = 1/3 = 0.33$$

To estimate the influence score, we use the *Monte Carlo* approach by running n simulations, where n is a large number ([1] set $n = 10,000$). The influence score of each user u is the average number of users activated for all simulations.

For each single simulation, we test the existence of each outgoing edge of a user (*i.e.* followers of u_1) in $\mathcal{G}$ by generating a random number $r \in [0, 1]$ and checking whether r reaches the activation probability equal to the edge weight, so that $r < w_I(e_{ij})$. If it is, the edge e_{ij} exists with probability $w_I(e_{ij})$. For example, for user u_1 we test his followers, the two edges e_{12} and e_{13} are tested with random values that give $0.3 < w_I(e_{12})$ and $0.6 > w_I(e_{13})$. Consequently, only u_2 is considered activated since we can consider that a path exists between u_1 and u_2.

When a neighbor is activated we can then recursively test each of the neighbor's outgoing edges with the same method. We stop when no more neighbors are activated (the influence propagation stops along the edges). That is u_2 cannot reactivate a node already activated by u_1. For instance, with $0.7 > w_I(e_{23})$ and $0.4 < w_I(e_{25})$, user u_5 is activated. Recursively, we test the edge e_{53}. In our example, the influence score of user u_1 (for the first simulation) is equal to 2 with activated users: u_2 and u_5.

Since the simulations are all independent and the graph data structure is only read during the process, we can run the n simulations in parallel. However running 10,000 *Monte Carlo* simulations for each user $u \in \mathcal{G}$ remains extremely costly when considering real large advertisement campaigns where the expected seed set reaches tens of thousands of users. Consequently, this computation must be performed offline.

We store all the values in a vector I of user influence scores:

$$\forall u_i \in \mathcal{G}, I_i = \sigma_{MC}(u_i)$$

3.2 Step II : User targeting at runtime

With the influence score computed in the pre-processing step, RTIM is able to select, during the RTB stream, users to target. Consider the temporal stream of users $\mathcal{T}$ in which appears every online connection event of the users $u \in \mathcal{G}$. Since a user can only be

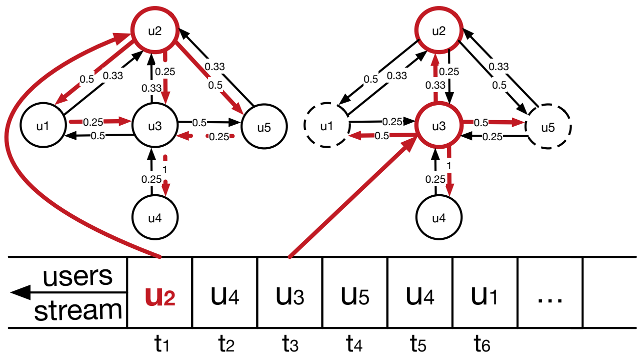

Figure 2: Ex. of the live stream ($\mathcal{T}$) of available users

targeted when he appears in the stream, we need to decide in real-time whether he is worth targeting or not. To make this decision, our RTIM algorithm takes into consideration two criteria:

i) Is the user influential enough?
ii) What is the probability that he was already activated by the ad through one of his influential and targeted neighbors?

To verify these two criteria, we set two thresholds, θ_I and θ_A, respectively the minimum influence score and the activation probability. Whenever a user is online, we check whether his influence score is important enough to be a potential target for the advertisement campaign or not. If his influence score is above the influence threshold θ_I (*i.e.,* considered to be an influencer) we check the probability for this user to be presented the advertisement by the users he follows (*i.e.,* already activated). If this probability is above the threshold θ_A, we estimate that it is not worth presenting the advertisement since it is very likely that it has already been presented to a user he follows who will have influenced him. Otherwise, the user is targeted and added to the seed set.

When we decide to target a user who satisfies these two thresholds, his activation probability is set to 1. This change, which impacts the activation probability of other users in the network, must be taken into account.Therefore, from the targeted user, we update the activation probability of neighboring users by propagating his influence. This will enable us to make better targeting decisions for future users who appear in the stream.

Figure 2 illustrates the stream of users that are online and how they are connected in the graph. $\mathcal{T}$ is a basic example of a RTB stream where users appear one at a time in discrete steps (in red/bold) and can only be targeted when available. As soon as RTIM makes a decision to target or not the user, he is no longer available. When the first user u_2 appears (time t_1), we verify his influence score I_2. His activation probability is necessarily 0 because he is the first user in $\mathcal{T}$. If $I_2 > \theta_I$ then we consider that u_2 tries to activate his followers u_1, u_3 and u_5, and propagate to their own neighbors. Then we update their activation probability. Assume that u_1 is activated ($A_1 > \theta_A$) while u_3 and u_5 are not.

When user u_4 is online (t_2), his influence score is insufficient to be targeted. We skip him and wait the for the following online user.

Then, when user u_3 appears in the stream (t_3), he is considered to be an influencer ($I_3 > \theta_I$) and not activated by u_2 ($A_3 < \theta_A$). As for u_2, he is targeted and propagates the activation probability to his neighbors u_1, u_2, u_5 and u_4.

When u_5 appears in $\mathcal{T}$ at t_4, even if his influence score is higher than θ_I, he is considered to be influenced by both u_2 and u_3 (assume that $A_5 > \theta_A$). Thus it is not worth targeting him.

By applying the whole stream of users $\mathcal{T}$, our approach generates the seed set $\mathcal{S}^*$ where every user $u \in \mathcal{G}$ verifies θ_I and θ_A. The key point resides in the fact that RTIM maximizes the influence of connected users while removing those who are too close to users already targeted.

We present the different algorithms, within RTIM, which are required for the runtime processing, in the following section.

4 RTIM MODEL

Traditional influence maximization algorithms, like IMM, have an optimistic approach since they determine statically the users to target based on the final global influence score of the set of targeted users. So they assume with a probability of 1 that these users will connect within the advertisement campaign period. If the advertisement campaign is not time-limited, *i.e.*, we consider an infinite stream of users online, these solutions potentially maximize the total score of the campaign. However, with a limited time-window for the advertisement campaign, not all these users will likely appear online.

RTIM's strategy is quite different since it considers that the probability that a user will appear in the stream is undefined. Therefore, the decision of targeting a user is done in real-time when he is available, considering whether this user is a good "influencer" while not already having been influenced by other users during the campaign. So RTIM can be considered as a pessimistic algorithm since we decide to add a user to the final seed set instantaneously, even if a "better" user to add to the seed set appears later in the stream.

Upon targeting a user, his activation probability is immediately set to 1 because he is considered to be activated on the spot. This change in the targeted user's status means that other users around him are likely to also be activated through his influence. In the following part, we discuss the activation probability graph which allows us to update the activation probability of neighboring users, with respect to time (i.e., the user position in the stream).

Activation probability graph

At time t_0, when $\mathcal{T}$ starts, we create the activation probability graph as the influence graph $\mathcal{G}_I$ described in Section 3.1.

We can adopt the matrix representation for the graph in the following equation:

$$M_{\mathcal{G}_I}(V, E) = A_{\mathcal{G}} \times InDeg_V$$

where $A_{\mathcal{G}}$ is the adjacency matrix, *i.e.*, $A_{\mathcal{G}}[i, j] = 1$ if there exists an edge from user u_i to user u_j, 0 otherwise, and $InDeg_V$ is the indegree vector, with $InDeg_V[i] = \frac{1}{indegree(u_i)}$.

The activation probability vector AV is initialized as the vector with only 0 values.

Activation probability updates

Consider we have at time $t_{k-1} > t_0$, an activation probability vector $AV(t_{k-1})$. Then assume that at time t_k, a user u_i connects and we decide to target him. So his activation probability $AV(t_{k-1})[i]$ is now set to 1. This probability update impacts other probabilities in the graph. Indeed, users who follow u_i are now more likely to see this advertisement and consequently we may avoid targeting them in the future. We must update other activation probabilities through influence propagation according to existing links in the graph to obtain the $AV(t_k)$ probability vector.

Definition 2 (Activation probability propagations). *Consider the social graph $\mathcal{G}(V,E)$ and its influence graph $\mathcal{G}_I(V,E,w_I)$ as defined in Section 3. The* activation probability *vector $AV(t_k)$ for $\mathcal{G}$ at instant t_k is recursively defined as:*

$$\begin{cases} AV^{(0)}(t_k) = M_{\mathcal{G}_I}(V,E) \times AV(t_{k-1}) \\ AV^{(i+1)}(t_k) = M_{\mathcal{G}_I}(V,E) \times AV^{(i)}(t_k) \end{cases}$$

So, after targeting a user u_i ($AV(t_{k-1})[i] = 1$) the vector is recursively combined with the activation probability graph $M_{\mathcal{G}_I}$ in order to propagate the activation while obtaining a convergence after i iterations:

$$AV(t_k) = AV^{(\infty)}(t_k) = M_{\mathcal{G}_I}(V,E) \times AV^{(\infty)}(t_k)$$

Since this model corresponds to a *Matrix population model* [28][1], we can guarantee its convergence due to the fact that the Eigenvalues of $M_{\mathcal{G}_I}(V,E)$ are real strictly positive (the matrix is real, asymmetric and non-diagonal). Moreover the propagation is an increasing and monotone function bounded to $\overrightarrow{1}$.

As the stream of users goes by RTIM will take less risk and target the best user when available because there is no certainty that a better user will appear later on. On the contrary, if the stream is infinite, then the probability that a user will appear is necessarily 1 and thus IMM, on an infinite stream, will always perform better than RTIM.

The aim of RTIM is to determine in real-time if a user is a good influencer while not already having been influenced by other users. To achieve this, our model relies on the probability a targeted and activated user has of influencing and activating other users. The issue is to determine the proper thresholds in order to fill the seed set both efficiently and effectively.

To target influencers, we need to determine users worth targeting but also when users are considered activated by influencers. For this we define the threshold θ_I as the minimum influence score for which we can consider the top-k influencers. Therefore we propose to set θ_I to the influence score of the kth influencer. We also define the activation probability threshold θ_A which is the likelihood of a user being activated. By default we set θ_A to 0.5. Any user whose activation probability is greater than θ_A is considered to have been activated and therefore will have attempted himself to propagate the information provided by an influencer and is therefore not worth targeting.

During the live stage we need to update the current online user's activation probability while checking if he is a worthwhile influencer. To achieve this, we need to compute all of the most reliable paths between this user and any activated neighbor of depth less than d. For that user, if his influence score is above θ_I and his activation probability is below θ_A, the user is targeted. Otherwise we ignore him.

In Equation 1 we explained that the probability of a user v being activated by any node $u \in \mathcal{S}$ is the probability of all paths between v and any user in $\mathcal{S}$, (i.e., $a_{\mathcal{S}}(u) = \mathrm{P}(\bigcup_{p_j \in P_{uv_i}} p_j)$). Here, we consider $\mathcal{S} = u$, where u is the online user who has just been targeted and considered activated.

Notice that if we consider the paths between u and v as being independent then we obtain equation 2:

$$A[u] = \mathrm{P}(\bigcup_{p_j \in P_{uv}} p_j) = 1 - \prod_{w_i \in \mathcal{P}^d_{uv}} (1 - w_i), \tag{2}$$
$$w_i \in \mathcal{P}^d_{uv} = \{all\ path\ weights\ of\ length\ d\ from\ u\ to\ v\}$$

This theoretical bias is validated if we consider the paths to be of length no more than 2. This is true because between two nodes all paths of length 2 are necessarily independent. Therefore, we use Equation 2 with a maximum depth of 2 to update the activation probability of a user.

Due to the directed property of the influence graph, for algorithmic and computational purposes it is easier to update the activation probabilities of a recently targeted and activated user. Hence, when a user is targeted, we update the activation probabilities of all his neighbors up to a maximum depth d.

Algorithm 1 Updating activation probabilities

```
Require: a graph $\mathcal{G}$, nodes $u$ and $v$, user $v$'s activation probability
    $A[v]$, the set of user $u$'s neighbors $\mathcal{N}_u$, current path weight $p$,
    depth $d$
1: procedure ActivationScores($\mathcal{G}, u, p, d$)
2:     for $v \in \mathcal{N}_u$ do
3:         $A[v] \leftarrow 1 - (1 - A[v]) * (1 - p * w_{uv})$
4:         if d > 1 then
5:             ActivationScores($\mathcal{G}, v, p * w_{uv}, d - 1$)
```

Algorithm 1 illustrates updates of activation probabilities. For each neighbor v of user u we propagate his activation probability (line 3). Then, while the depth of propagation is sufficient we follow the propagation recursively (line 4&5). In the worst case it runs in $O(|V|^d)$ when all users are interconnected. Since in most cases our networks are not dense (see Table 1), updating the activation probabilities is done very fast (see Table 4) and we set d to 2. However, it is not necessary to be extremely fast for very large and dense networks, such as *Twitter*, as updating the activation probabilities can take place in a separate thread during the live stage.

For the live stage of RTIM, we consider that if any neighbor (of depth d) of a user is targeted then we update his activation probability. First, Algorithm 2 initializes the activation probabilities to the 0 vector (line 1). Then, while the seed set is not filled (line 2) we check each new incoming user u if he validates both θ_I and θ_A (line 3&4). Deciding to target a user (line 4) is done in $O(1)$ and is thus instantaneous. If he does we add u to the seed set and propagate the activation by applying Algorithm 1 (line 5&6).

Consider Figure 2 as an example. Its influence scores vector I is:

$$I^\top = \begin{bmatrix} 2.3423 & 3.0232 & 3.7802 & 1.6932 & 2.3423 \end{bmatrix}$$

[1]Our model is not a Markov chain since the sum of a column can exceed 1: $M_{\mathcal{G}_I}(V,E) = A_{\mathcal{G}} \times InDeg_V$

	# of nodes	# of edges	Degree Mean	Degree Variance	Degree Standard Deviation
`Youtube`	1.13M	5.97M	10.53	10,304.01	101.50
`LiveJournal`	3.99M	69.3M	34.70	7,381.26	85.91
`Twitter`	41M	1.46B	70.50	6,426,184.47	2,534.99

Table 1: Datasets characteristics

Algorithm 2 RTIM Live

```
Require: a graph $\mathcal{G}$, a user $u$, the sorted list of influence scores $I$, influence threshold $\theta_I$, $u$'s activation probability $A[u]$ of size $|\mathcal{V}|$, a depth $d$, a temporal stream of users $\mathcal{T}$, the seed set $\mathcal{S}$, seed set max size $k$
1: Initialize $A \leftarrow \overrightarrow{0}$
2: while $|\mathcal{S}| < k$ do
3:     $u \leftarrow next(\mathcal{T})$
4:     if $I[u] \geq \theta_I$ and $A[u] \leq \theta_A$ then
5:         ActivationScores($\mathcal{G}$, $u$, 1, $d$)
6:         $\mathcal{S} \leftarrow \mathcal{S} \cup u$
```

During the live stage we read the RTB stream $\mathcal{T}$ with maximum seed size $k = 3$, $\theta_A = 0.5$ and $\theta_I = 3.0$, as example settings. At $t = 0$, we initialize the activation probability vector to $A^0 = \overrightarrow{0}$ (line 1). At $t = 1$, the first user in $\mathcal{T}$, u_2, has $I[u_2] = 3.0232 > \theta_I$ and $A[u_2] = 0$, so we target him and update the activation probability of his neighbors. Which gives us:

$$A^{1^\top} = \begin{bmatrix} 0.54125 & 1.0 & 0.4257 & 0.25 & 0.54125 \end{bmatrix}$$

At $t = 2$, the second user u_4, has $I[u_4] = 1.6932 < \theta_I$. We ignore him because his influence score is too low. At $t = 3$, for u_3, $I[u_3] = 3.7802 > \theta_I$ and $A[u_3] = 0.4257 < \theta_A$, so we target u_3, and update the activation probability vector:

$$A^{2^\top} = \begin{bmatrix} 0.7303 & 1.0 & 1.0 & 1.0 & 0.7303 \end{bmatrix}$$

In the remainder of the stream $\mathcal{T}$, $A[u_5] = 0.7303 > \theta_A$ and u_4 is already targeted ($A[u_4] = 1.0$), so $\mathcal{S}^* = \{u_2, u_3\}$. Notice how the size of the final seed set is less than the maximum seed set size $k = 3$. This is because, with the live parameters we have set, we weren't able to fill the final seed set up to its maximum potential size.

5 INFLUENCE ANALYSIS

Our model is experimented with empirical datasets of different sizes: `LiveJournal, Youtube` and `Twitter`. These graph datasets have interesting properties that can be exploited in order to understand the impact of the Real-Time Bidding environment on the influence maximization interactions of users.

To achieve this, we need to understand more closely the way that users are interconnected for each dataset and study their topologies. We can see in Table 1 the global statistics that highlight the different sizes. We see here the graphs composition of all three datasets (# nodes and # edges), and node degrees (mean, variance and standard deviation).

We can see that `Youtube` is the "smallest" graph with less connections (mean degree of 10) but with a high variation of degree compared to its size. `LiveJournal` is highly connected with a high

Dataset	B	r_0	α	X^2-Pearson value
`Youtube`	8×10^4	11	0.78	0.976
`LiveJournal`	1.55×10^3	0	0.395	0.971
`Twitter`	1.7×10^6	6	0.99	0.969

Table 2: Zipf-Mandelbrot parameters for graph datasets

number of edges and a mean degree of 34. However we can see that users are more homogeneously connected with a low variance and standard deviation. `Twitter`, on the other hand, is the biggest graph in which users can have varying numbers of connections with a mean degree of 70 but a variance of 6.4M and a standard deviation of 2.5k. This analysis helps understand the subtle performance of RTIM for each dataset.

As for the distribution of users' influence score, to estimate the influence score of each user we applied `Monte Carlo` simulations on each dataset for every node in the social networks. The result of each MC influence score is a correct approximation of the real score.

Figure 3 shows the distribution of influence scores for our graph datasets (reduced to 10^5 but analysis was done on all users). These distributions can be characterized by a standard *Zipf-Mandelbrot* distribution [29], traditionally used for distribution of ranked data. It is defined by:

$$\frac{B}{(r_0 + r)^\alpha}$$

where r is, here, the rank of the influencer. r_0 is a constant representing the number of top influencers. B corresponds to the starting score modifier and α is the decreasing speed of scores.

Table 2 gives the corresponding values for those Zipf-Mandelbrot distributions and the Pearson Xi-Square values (observation probabilities) found for each distribution.

`Youtube` and `Twitter` behave similarly with a huge r_0 (resp. 11 and 6) leading to 100 to 750 top-influencers. We can see that Twitter has more top influencers with a high score and then drops faster than Youtube.

`LiveJournal` behaves differently with very few top influencers compared to Youtube or Twitter. The low value r_0 shows that top-influencers' score decreases faster at the beginning of the curve.

However, the decreasing speed of the influence score α witnesses really high values (resp. 0.78 and 0.99) which means that it is harder to become a top influencer on `Twitter` than `Youtube`. Likewise, the absolute number of influencers is really high with a B value between 10^4 and 10^6 leading to a long tail which only starts after more than 10^5 for `Youtube` and 2×10^5 users for `Twitter`.

On the other hand, `LiveJournal`'s scores curve decreases slower than the others with an α of only 0.395 giving the idea that the number of connections between users are closer to the average

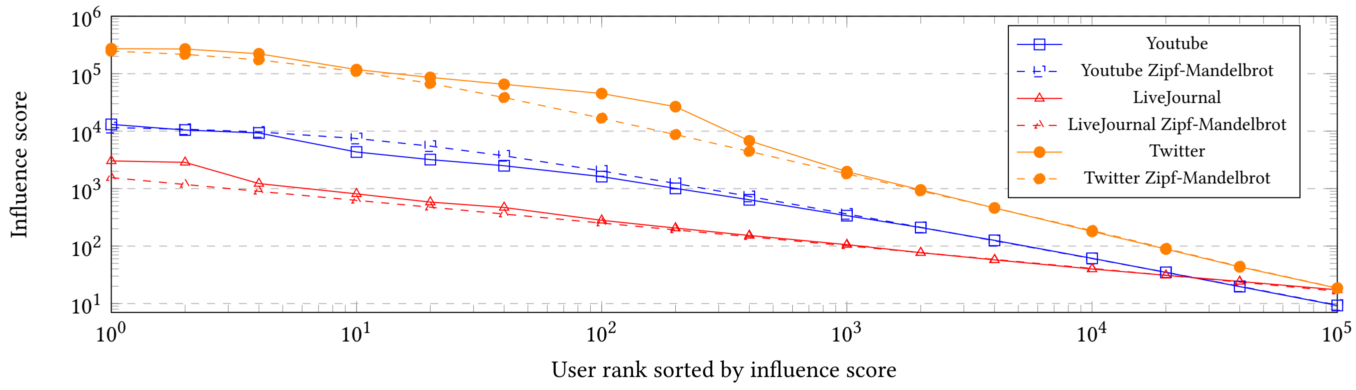

Figure 3: Datasets' influence score distributions

than `Twitter` or `Youtube`. Consequently, the long tail is reached more slowly than the others (4×10^5 users).

This conclusion is interesting in order to understand the impact of these social networks on influence maximization. Indeed, targeting top influencers in real-time requires choosing influencers according to there estimated score. For instance, users from the long tail are pretty identical and cannot be differentiated from each other, thus the decision to target or not an influencer depends on α which tells us how much an influencers score evolves.

6 EXPERIMENTS

All of our experiments are run on a server of make Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz with 56 CPUs and 462Gb of RAM memory.

We wish to show in this section the impact of the strategy of choosing influencers in a real-time bidding stream of users. In order to do this, we need to compute the influence scores of users, generate multiple streams of users with varying distributions and compare the final solution of each algorithm for different graph datasets.

6.1 Experimental process

Since RTIM is an IM algorithm which runs under RTB constraints, we want to compare it to an existing IM algorithm. We choose IMM [16] because in [5] it is proven to have the best compromise between computation speed, scalability and accuracy. In particular, IMM can compute seed sets with thousands of users on our largest dataset. We run all algorithms in a specific experimental process involving three stages: pre-processing, live stream generation and live stream process. The code for IMM is provided by [5] in C++.

6.1.1 Stage I: Pre-processing. First, `IMM` is run in its entirety and adds k users to its seed set $\mathcal{S}_{IMM}$. Recall that IMM relies on the fact that every user in the graph has the same probability to appear, and more precisely assumes that they will necessarily appear in the stream. However, during the marketing campaign, it is more likely that not all the users will be available online. Since our objective is to target a large number of users in the stream, as would happen in a marketing campaign, we set $k = 10,000$. We compute IMM's optimal seed using $\epsilon = 0.1$. It can do this in seconds and very easily scale to large graphs (see [5] for a proper analysis of IMM performance).

`RTIM` uses the *Monte Carlo* approach to compute the influence score of each user in the graph. We run n simulations per node ($n = 10,000$). This can be parallelized. Even so, for large graphs, this operation can take several days, so to make it scalable we limit it to a depth of 3 on large graphs to reduce pre-processing time to a few hours at most. Thanks to the graph topology with a high connectivity (see Section 5), the Monte Carlo simulations converge faster. As a counterpart we lose in influence score precision but believe the error to be negligible. The influence scores of each user are stored in a vector I for future use.

Regardless of the algorithm or external parameters, we restrict pre-processing time to be overnight, i.e. 6 hours or less. Thus, static algorithms, must compute their optimal seed set under 6 hours regardless of its size and RTIM must compute the influence score of all users regardless of the size of the graph.

6.1.2 Stage II: Live stream generation. It's during this stage that we read our RTB stream and both algorithms have the opportunity to target influential users. Since no real streams of connected users are available online, we simulate users' behavior in the social networks with different distributions. To the best of our abilities we haven't found well-founded models for RTB social stream generation. So, we simulate the RTB stream of users in the three different social networks by randomly picking users based on two distributions. Notice that a user can appear several times in the stream. Here are the two distributions:

- *Uniform*: we suppose that all users have equal probability of appearing in the stream. This distribution can be considered to be the worst case where highly connected users can appear as frequently as poorly connected users or where top influencers can appear as frequently as low influencers.
- *Log*: we suppose that users who have more in/out edges in the graph are more likely to be connected. The probability of user u_i being in the stream is therefore:

$$P(u_i \in \mathcal{S}) = \frac{\log(deg_{u_i})}{\sum_{u_i \in V} \log(deg_{u_i})}$$

Influence score	Youtube		LiveJournal		Twitter	
	Uniform	*Log*	*Uniform*	*Log*	*Uniform*	*Log*
$1 \leq I \leq 2$	37.79%	19.92%	37.43%	17.78%	86.33%	80.55%
$2 < I \leq 5$	43.25%	40.69%	36.71%	39.42%	12.02%	16.13%
$5 < I \leq 10$	11.01%	18.88%	17.18%	26.27%	1.13%	2.12%
$10 < I \leq 100$	7.46%	18.71%	8.66%	16.45%	0.48%	1.08%
$100 < I \leq 1,000$	0.47%	1.71%	0.03%	0.08%	0.04%	0.10%
$1,000 < I$	0.02%	0.10%	0.00%	0.00%	0.00%	0.02%
Stream size	226,978		399,796		4,165,223	

Table 3: Streams' influence score distributions

where deg_{u_i} is the degree of u_i, the sum of respectively in-degree and out-degree. We apply a logarithm on the number of edges per user in order to give users with a low influence score a reasonable probability of showing up in the stream. In fact, due to the distribution of edges in the graph, some users are highly connected to other ones. Thus, they will represent the large majority of online users in the stream even though this does not reflect the reality. This *Log* stream can be considered to be the best case where highly linked users are more likely to be present in the stream, and potentially top-influencers.

We also consider that the size of each stream is a subset of the total number of users, we choose 10% of the total number of users. In thus setting, we insure that all the users are not necessarily available online during the marketing campaign.

$$|\mathcal{T}| = 10\% \times |V|$$

To ensure the stability of our model, we generated multiple random versions of each stream. Because two versions of each stream proved to have equivalent results, our results are averages of both versions.

Table 3 gives for each stream the proportions of influence scores and size in our experiments. As expected *Uniform* distributions contains more low influence scores while *Log* focuses more on higher scores (more than 10). As seen in Figure 5, `LiveJournal` produces lower scores so does the stream. According to `Twitter` it witnesses a huge amount of low scores with more than 80% in both *Uniform* and *Log* distributions. It is due to the large stream (above 4M connections) and stick to the distribution of `Twitter` scores. However, with respect to the proportions even with 1.20% of the stream, the number of high influence scores is higher than `LiveJournal` and `Youtube`.

6.1.3 Stage III: Live stream process. For `IMM`, during the live stage, if a user in the stream belongs to $\mathcal{S}_{IMM}$, he is targeted and immediately added to the final seed set $\mathcal{S}^*_{IMM}$. Regardless of the number of times he appears in the stream, the user is only targeted once. At the end of the stream:

$$\mathcal{S}^*_{IMM} = \mathcal{S}_{IMM} \cap \mathcal{T}$$

For each user u_i in the stream, `RTIM` will verify its targeting conditions. It will check that the user's activation probability $ap(u_i)$ is below the activation threshold θ_A, and if his influence score

	Average	Median	Max
`Youtube`	70.3 ms	6.30 ms	193.4 ms
`LiveJournal`	61.1 ms	6.03 ms	192.0 ms
`Twitter`	85.9 ms	44.5 ms	411.1 ms

Table 4: Update activation probabilities time

$\sigma_{MC}(u_i)$ is greater than the influence threshold θ_I. If both conditions are satisfied than u_i is targeted instantaneously (see Algorithm 2). At which point he is added to the final seed set $\mathcal{S}^*_{RTIM}$ and we update the activation probability of his neighbors up to depth 2 (see Algorithm 1). This update operation can be done by a separate thread.

Table 4 gives the time spent to update propagation probabilities in the network. It shows that the average update time is less than 100ms which satisfies our real-time requirement, and in most of the case far less (median). However, some updates, especially on large dense graphs can take up to 411ms. This is still negligible since a user cannot influence another in less than half a second.

6.1.4 Stage IV: Seed Sets Evaluation. To compare the performance of both IM algorithms we compute and compare $\sigma_{MC}(\mathcal{S}^*_{IMM})$ and $\sigma_{MC}(\mathcal{S}^*_{RTIM})$. This is also something we have done during the live stage to analyze how the seed set score evolves over the duration of the stream. Because of the cost of computation of $\sigma_{MC}(\mathcal{S})$, during the live stage, the score of the seed set is computed for every 100 users added to the seed set, or 1,000 for large seed sets on Twitter.

6.2 Experimental results

6.2.1 `Youtube`*.* In the following experiments, we see the evolution of the seed set during the stream both for influence score and size. Figure 4 and 5 give the results produced for the evolution of seed sets with a stream of 2.27×10^5 connected users over the `Youtube` dataset.

Figure 4 shows the evolution of the seed set size. We clearly see that IMM hardly finds pre-defined influencers, especially for the *Uniform* distribution. RTIM evolves almost linearly with twice as many seeds for the *Uniform* distribution and 3.3 times more for the *Log* one. According to the *Log* distribution, RTIM finds more influencers and reaches k faster. The sudden stop of the RTIM seed set at 1.89×10^5 users is due to the fact that the marketing campaign is over with a full seed set of $k = 10,000$ users.

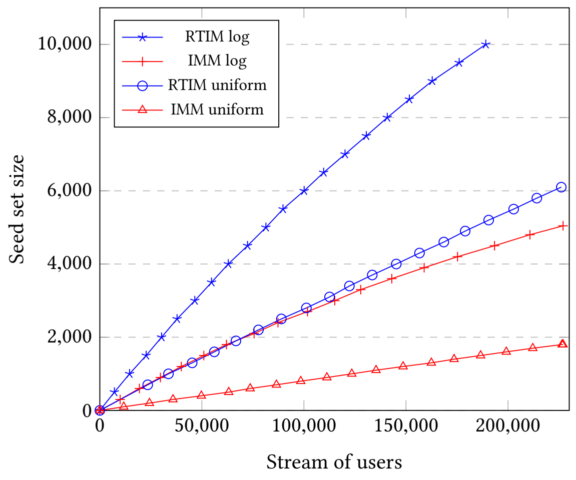

Figure 4: Seed set *size* evolution with `Youtube`

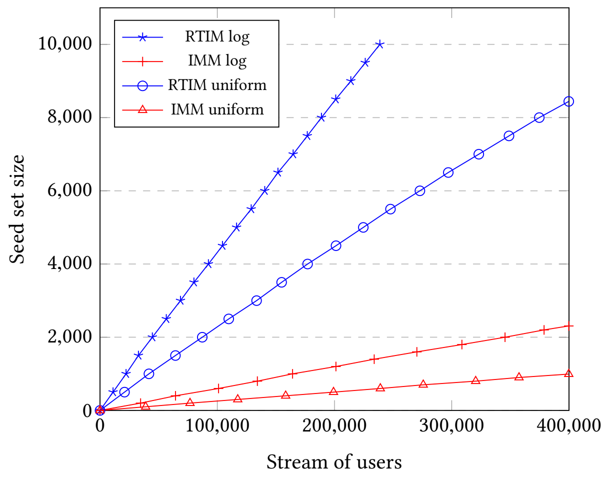

Figure 6: Seed set *size* evolution with `LiveJournal`

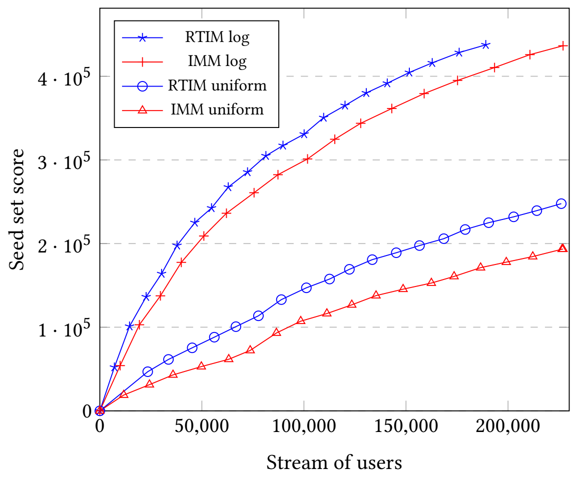

Figure 5: Seed set *score* evolution with `Youtube`

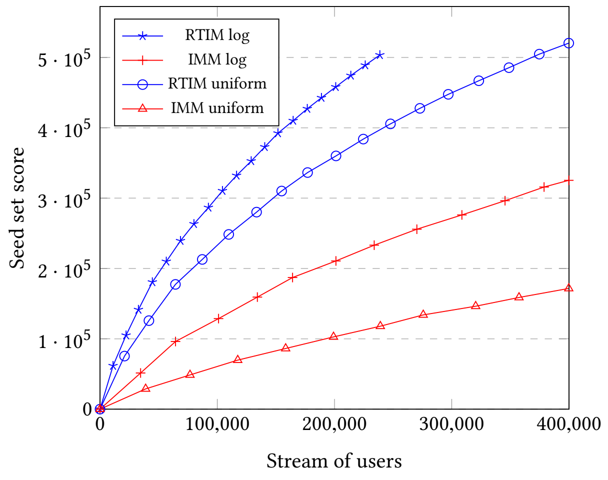

Figure 7: Seed set *score* evolution with `LiveJournal`

Figure 5 shows that RTIM produces seed sets with higher scores than IMM. We can see different evolutions from the *Uniform* and *Log* streams. In fact, IMM hardly finds influencers in the *Uniform* distribution, this is due to the fact that highly connected users are less likely to be available online. This explains why IMM's seed set evolves slowly. On the other hand, RTIM targets users according to their local influence on the graph and thus has more targeting opportunities.

According to the *Log* distribution, IMM is closer to RTIM since top-influencers are more present in the stream. Consequently, it takes time for IMM to reach this goal by the end of the stream with a similar score (1,105 less), while RTIM stopped earlier when the seed set size reached k. This confirms the fact that IMM is better at maximizing k than RTIM in an infinite stream, however in a finite campaign this is not the case.

Interestingly, the score growth is higher at the beginning of the stream and we observe a logarithmic evolution of the score. In fact, during the stream process finding new influencers is more unlikely since they have already been selected (IMM) or activated by targeted neighbors (RTIM). Since the evolution of size is almost linear (Figure 4), this logarithmic evolution of the seed set score confirms the fact that chosen users bring less influence in the seed set than previous chosen ones. This is due to the sub-modular property of the influence maximization problem.

6.2.2 `LiveJournal`. Figure 6 and 7 focus on the `LiveJournal` dataset. The stream contains almost 4×10^5 online users.

Figure 6 shows the evolution of seed set sizes. As we can see that IMM finds very few expected influencers and produces 8.5 times less seeds for the *Uniform* stream (respectively 7 for the *Log* stream) than RTIM. In fact, RTIM targets influencers more easily than IMM.

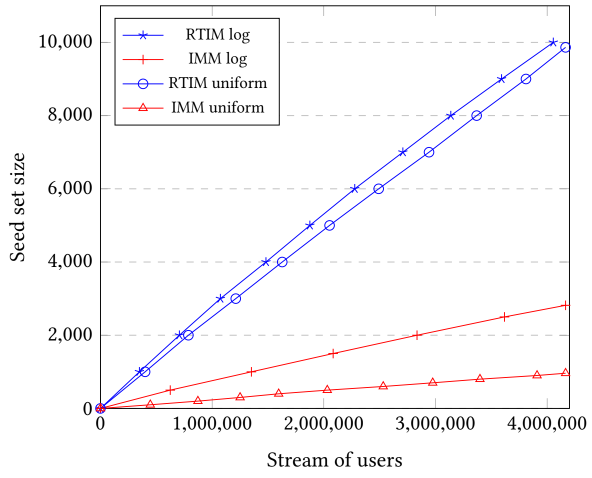

Figure 8: Seed set size evolution with `Twitter`

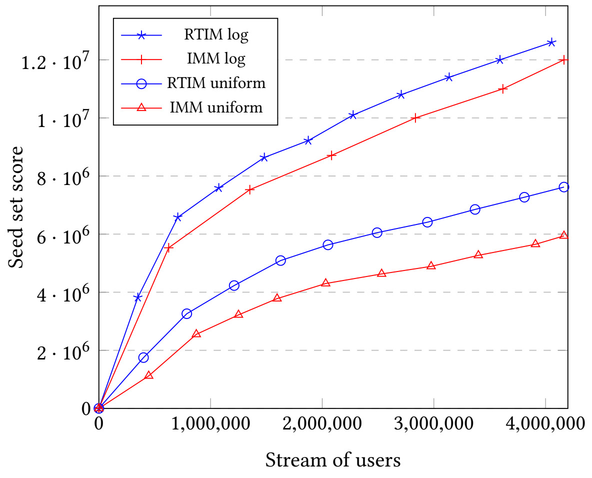

Figure 9: Seed set score evolution with `Twitter`

This can be explained by the specific distribution of scores we explained in Section 5 with a very slow decreasing of the scores ($\alpha = 0.395$). This can be confirmed by the fact that `LiveJournal` has a low degree standard deviation and variance. Thus RTIM adapts locally to the users connection with similar scores while IMM only focuses on pre-chosen seeds.

We can see in Figure 7 that the evolution of seed set scores are really different from the `Youtube` dataset. The impact of pre-determined seeds have a huge impact on the final seed set score since very few influencers appears in the stream while RTIM has the opportunity to choose a "similar" score in the neighborhood.

We can also see that RTIM obtains a lower seed set score for the *Uniform* distribution than the *Log* one. It is due to the fact that RTIM Log fills the seed set more quickly after only 2.38×10^5 users in the stream. The impact of the specific distribution of scores of `LiveJournal` and the fact that users have high mean degrees (with a low variance) give more chances for common users (lower scores) to ease information propagation.

6.2.3 `Twitter`. The seed sets produced for the `Twitter` dataset are presented in Figure 8 and 9. The stream is composed of 4.17×10^6 connected users over the `Twitter` dataset.

We observe in Figure 8 that RTIM seed sets evolves very quickly for both *Uniform* and *Log* streams. This is due to the huge amount of high score users of the `Twitter` distribution (see Section 5 with $B = 1.7 \times 10^6$), consequently RTIM targets any user in the stream that reaches the threshold θ_I. On the other hand, IMM evolves more slowly, 10 times less for *Uniform* and 3.5 times less for the *Log* stream. RTIM fulfills the marketing campaign $k = 10,000$ after 4×10^6 users in the stream.

In Figure 9 the seed set scores evolve similarly to the `Youtube` dataset (Figure 5) with close scores for the *Log* stream, even if the gap is higher due to huge seed set scores (600,000 less). The effect of the high decrease of the influence score ($\alpha = 0.99$ in Table 1) is observable here where IMM targets high influencers that have sufficient impact to grow rapidly while RTIM targets good influencers to guarentee a global impact in a minimum amount of time.

6.2.4 `Conclusions`. Our experiments showed that RTIM provides better seed sets score while maximizing the score in a minimum of time while IMM succeeds in maximizing on the whole dataset. The impact of the live stream distribution between *Uniform* and *Log* is such that both methods behave clearly better on users with very high degrees (more likely to be top influencers) however IMM is more sensitive to this setting.

The seed set score curve is logarithmic, this is due to the sub-modular property of the influence maximization problem. Indeed, the more users we add to the final seed the smaller the marginal gain to the overall seed set.

We saw that the distribution of users' influence score has an impact on the effictiveness of both IMM and RTIM methods. First, the decrease of those scores is in favor of RTIM when α is low (`LiveJournal`) where IMM makes a choice on similar influencers while RTIM targets only available ones. Second, graphs with very high influence scores (induced by B) give RTIM more choices of influencers (even with average scores) and so it fills up the seed set quickly.

Consequently, the seed set size evolves more quickly for RTIM and k is reached rapidly, especially for users with a high degree.

7 CONCLUSION

In this article we have shown, that it is possible to answer the influence maximization problem in a real-time bidding environment, that up to now have not been applied to IM algorithms. We have shown, that static IM algorithms, such as IMM, that pre-compute the best seed set of size k can solve this problem so long as they are capable of generating in reasonable time a large seed set with $k \geq 10,000$. We have shown, in addition, that it is possible, in this setting, to compete with these powerful static IM algorithms by using a dynamic IM algorithm, such as RTIM, based on the local influence of each user. In fact, we have proven, that dynamic IM algorithms such as RTIM can outperform static algorithms when the stream of users is finite in size (or a fixed period of time).

It is important to note, that the RTB environment is more complex so than the constraints which we used. It is for instance, not guaranteed that a user having been displayed an advertisement will see it, click on it, or even convert. Indeed, the influence probability for a targeted user is equal to 1. For future works, we propose to extend RTIM to answer the real-world RTB constraints. Contrary to IM algorithms, RTIM could choose to target another user if a previous user who was targeted was not considered as activated. We can therefore, make RTIM much more interactive with dynamic user behavior while static solutions like IMM cannot.

In addition, should the graph be updated, it is not difficult to recompute local influence scores, if necessary, or keep targeting users in the live stream. Whereas, static IM algorithms need to recompute the best possible seed set for each new graph.

We can also improve RTIM by adapting the θ_I threshold when processing the live stream. In fact, online user behavior, such as periodicity of connection during the day or week, has an impact on the final seed set score. We can therefore change the influence threshold dynamically to fill the final seed set more effectively.

ACKNOWLEDGMENTS

This research is supported and financed by the research partnership between Kwanko, the DeVinci Research Center and the CNAM. We thank them for their contributions.

REFERENCES

[1] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, (New York, NY, USA), pp. 137–146, ACM, 2003.

[2] S. Yuan, J. Wang, and X. Zhao, "Real-time bidding for online advertising: Measurement and analysis," *CoRR*, vol. abs/1306.6542, 2013.

[3] S. Spencer, J. O'Connell, and M. Greene, "The arrival of real-time bidding," tech. rep., Google, 2011.

[4] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'01, (New York, NY, USA), pp. 57–66, ACM, 2001.

[5] A. Arora, S. Galhotra, and S. Ranu, "Debunking the myths of influence maximization: An in-depth benchmarking study," in *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, (New York, NY, USA), pp. 651–666, ACM, 2017.

[6] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, (New York, NY, USA), pp. 241–250, ACM, 2010.

[7] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, (Washington, DC, USA), pp. 1029–1038, ACM, 2010.

[8] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck, "Sketch-based influence maximization and computation: Scaling up with guarantees," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM'14, (Shanghai, China), pp. 629–638, 2014.

[9] S. Galhotra, A. Arora, and S. Roy, "Holistic influence maximization: Combining scalability and efficiency with opinion-aware models," in *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, (New York, NY, USA), pp. 743–758, ACM, 2016.

[10] S. Galhotra, A. Arora, S. Virinchi, and S. Roy, "ASIM: A scalable algorithm for influence maximization under the independent cascade model," in *Proceedings of the 24th International Conference on World Wide Web Companion*, WWW'15, (Florence, Italy), pp. 35–36, 2015.

[11] A. Goyal, W. Lu, and L. V. Lakshmanan, "Celf++: Optimizing the greedy algorithm for influence maximization in social networks," in *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, (New York, NY, USA), pp. 47–48, ACM, 2011.

[12] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'09, (Paris, France), pp. 199–208, 2009.

[13] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi, "Fast and accurate influence maximization on large networks with pruned monte-carlo simulations," in *Proceedings of the Twenty-Eighth Conference on Artificial Intelligence*, AAAI'14, (Québec City, Québec, Canada), pp. 138–144, 2014.

[14] S. Cheng, H. Shen, J. Huang, W. Chen, and X. Cheng, "Imrank: influence maximization via finding self-consistent ranking," in *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, (Gold Coast , QLD, Australia), pp. 475–484, 2014.

[15] S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng, "Static greedy: solving the apparent scalability-accuracy dilemma in influence maximization," *CoRR*, vol. abs/1212.4779, 2012.

[16] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, (New York, NY, USA), pp. 1539–1554, ACM, 2015.

[17] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: near-optimal time complexity meets practical efficiency," in *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pp. 75–86, 2014.

[18] K. Jung, W. Heo, and W. Chen, "Irie: Scalable and robust influence maximization in social networks," in *2012 IEEE 12th International Conference on Data Mining*, ICDM'12, (Brussels, Belgium), pp. 918–923, 12 2012.

[19] S. N., A. B., and S. Bhattacharya, "Influence maximization in large social networks: Heuristics, models and parameters," *Future Generation Comp. Syst.*, vol. 89, pp. 777–790, 2018.

[20] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical Programming*, vol. 14, pp. 265–294, Dec 1978.

[21] M. Zhang, C. Dai, C. Ding, and E. Chen, "Probabilistic solutions of influence propagation on social networks," in *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, (New York, NY, USA), pp. 429–438, ACM, 2013.

[22] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, (New York, NY, USA), pp. 420–429, ACM, 2007.

[23] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, (Philadelphia, PA, USA), pp. 946–957, Society for Industrial and Applied Mathematics, 2014.

[24] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, (New York, NY, USA), pp. 75–86, ACM, 2014.

[25] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "SIMPATH: an efficient algorithm for influence maximization under the linear threshold model," in *11th IEEE International Conference on Data Mining*, ICDM '11, (Vancouver, BC, Canada), pp. 211–220, 2011.

[26] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," in *The 10th IEEE International Conference on Data Mining*, ICDM'10, (Sydney, Australia), pp. 88–97, 2010.

[27] Y. Wang, Q. Fan, Y. Li, and K.-L. Tan, "Real-time influence maximization on dynamic social streams," *Proceedings of the VLDB Endowment*, vol. 10, 02 2017.

[28] P. H. Leslie, "On the Use of Matrices in Certain Population Mathematics," *Biometrika*, vol. 33, pp. 183–212, 11 1945.

[29] B. B. Mandelbrot, *The fractal geometry of nature*, vol. 1. WH freeman New York, 1982.