Real-Time Influence Maximization in a RTB Setting

David Dupuis, Cédric Du Mouza, Nicolas Travers, Gaël Chareyron

Noname manuscript No.
(will be inserted by the editor)

Real-Time Influence Maximization in a RTB Setting

David Dupuis · Cédric du Mouza · Nicolas Travers · Gaël Chareyron

Received: 30/02/2020 / Accepted: 08/06/2020

Abstract To maximize the impact of an advertisement campaign on social networks, the Real-Time Bidding (RTB) systems aim at targeting the most influential users of this network. Influence Maximization (IM) is a solution that addresses this issue by maximizing the coverage of the network with top-k influencers who maximize the diffusion of information. Associated with online advertising strategies at web scale, RTB is faced with complex ad placement decisions in real-time to deal with a high-speed stream of online users. To tackle this issue, IM strategies should be modified in order to integrate RTB constraints. While most traditional IM methods deal with static sets of top influencers, they hardly address the dynamic influence targeting issue by integrating short time decision, no interchange and stream's incompleteness. This paper proposes a real-time influence maximization (RTIM) approach which takes influence maximization decisions within a real-time bidding environment. A deep analysis of influence scores of users over several social networks is presented as well a strategy to guarantee the impact of an IM strategy in order to define the budget of an ad campaign. 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

D. Dupuis
Léonard de Vinci Pôle Universitaire, Research Center, Paris, France. E-mail: david.c.dupuis@gmail.com

C. du Mouza & N. Travers
CEDRIC laboratory, Conservatoire National des Arts et Métiers (CNAM), Paris, France. E-mail: firstname.lastname@cnam.fr

N. Travers & G. Chareyron
Léonard de Vinci Pôle Universitaire, Research Center, Paris, France. E-mail: firstname.lastname@devinci.fr

1 Introduction

Influence Maximization (IM) is a trend topic since Kempe et al. [18], known as a maximum coverage problem of social networks. The goal is to find the smallest subset of individuals in a social network, whom when targeted with a piece of information will maximize its diffusion through social influence. Thus, IM aims at maximizing the influence impact of a set of users. "*Influence*" is "*the power of causing an effect in indirect or intangible ways*" (*Merriam-Webster*). In other words, a user can be influenced if he saw the ad, interacted with it, purchased the product or was encouraged to do so in the future.

Today, real-time bidding (RTB) outpaced other advertising strategies in terms of online advertising [28,38] and social network services (SNS). RTB is an online auction system which allows advertisers to bid in real time for ad locations on a webpage loaded by users and thus to target them efficiently. In RTB, advertisers see a stream of users, one at a time and have less than 100 ms [38,37] to decide to bid or not. The bidders do not know what the auction landscape looks like and consequently cannot oversee future connections.

In addition, RTB ad targeting relies essentially on web page content and users' profile. However, it lacks the social value of each customer as suggested by Domingos and Richardson [10]. As far as we know no RTB algorithm attempts to find an IM solution to improve bidding decisions. Thus, the aim of our approach is to develop an IM algorithm capable of running with RTB constraints. It is worth noting that IM could integrate the bidding aspect in order to improve RTB, but this approach is left for future work.

Traditional IM algorithms, based on propagation models, propose various optimization technic to statistically choose a seed set of users that maximizes influence. Even if some IM algorithms compute it in real time, none of them can work within a real-time bidding environment and satisfy its requirements.

Interestingly, the IM problematic in an RTB context is a twofold issue. First, influence maximization needs to take into account time with real-time bidding constraints: a short time, no exchange with the past, no time windows, incompleteness of the stream (not all users are connected). Second, to maximize the impact of an ad campaign, it is necessary to know in advance how much user it requires to target in the social network to guarantee a sufficient coverage. In fact, it is necessary to define the more accurate budget to define the best seed set size which maximizes the influence on the social network.

Indeed, whereas existing algorithms take hours or days to find a seed set up to 200 seeds in a large social network [1], 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 article targets the issue with the following constraints:

- *Real-Time Bidding*: only an online user can be targeted,
- *Processing Time*: 100 ms to choose to target a user or not,
- *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 just by appending them in real time while maximizing scores of large seed sets.
- *Ad campaign guarantee*: the size of the ad campaign is set in advance in correlation to the social network to give a guarantee of its influence score.

Therefore, an IM algorithm is necessary to target influential users in an RTB environment, capable of deciding in real time which users are worth targeting. To achieve this we propose the `Real-Time Influence Maximization` (RTIM). It 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 follows:

- 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 give a model to give the estimation of the seed-set size in order to guarantee the influence efficiency of an ad campaign on the network,
- We set up a thorough experimental setting for RTIM and IMM models on different social networks.

In this article, we first review the literature 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. We follow on the RTIM implementation and we go through the experimental process which compares our dynamic algorithm with a static approach. We then present a methodology to estimate the impact of an ad campaign which gives the estimation of the influence of a seed-set size.

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.

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

2.1 Propagation models

Kempe et al. [18] 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 time step, 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 [13] 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's models for the *IC* model:

- **Constant:** Each weight e_{ij} has a constant probability. In most solutions [4, 9,11,12,14,18], p is set at 0.01 or 0.1. Some define $p \in [0.01, 0.1]$ [5,26].
- ***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 [4,5,9,11,12,7,8,18,30,31].
- ***Tri-valency Model*:** Here, the weight of edges is randomly chosen from a list such as {0.001, 0.01, 0.1} [4,7,17]. In very large or dense networks, 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 [24] with an extensive IM state-of-the art.

The *IC* model is very useful to model information diffusion when a single exhibition 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. [18] 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. [4] prove that computing the influence score of a seed set is #P-Hard under the *IC* model.

For both *IC* and *LT* models, adding users to the seed set always increases 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 [25]. Many IM algorithms of the state-of-the art rely on this theoretical guarantee to validate their strategy.

2.3 Computing score

Influence Score: Authors in [39] 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.

$$\begin{aligned} \sigma(\mathcal{S}) &= \sum_{v_i \in V} a_S(v_i) = \sum_{v_i \in V} \mathsf{P}(\bigcup_{p_j \in P_{uv_i}} p_j), \\ P_{uv_i} &= \{\textit{all paths event existence between } u \textit{ and } v_i\} \end{aligned} \tag{1}$$

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 [17]. Due to the exponential nature of computing the influence score, Kempe et al. [18] 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 called 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: $\sigma_{MC}(\mathcal{S}) \sim \sigma(\mathcal{S})$

2.4 Algorithms

Clearly presented by in [1] and [2] there are three main categories of IM algorithms: greedy, sampling and approximation.

GREEDY [18], CELF [20] and CELF++ [14] are all three lazy-forward algorithms which take advantage of the sub-modularity property of the IM problem and thus guarantee an approximation of $(1 - 1/e - \epsilon)$. To find $\mathcal{S}^*$ they start with $\mathcal{S} = \emptyset$ and incrementally add 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 [20,14] 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 [3] (referred to as RIS: *Reverse Influence Sampling*), TIM, TIM+ [31], or IMM [30] 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 [1] 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 their 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 [11], IRIE [17], SIMPATH [15], LDAG [6] or IMRANK [7], SSA-Fix [16], 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. RLP [22] uses *live edges* and *propagation paths* to optimize computation time. They can run select the top k influential users. While scalable they do not provide theoretical guarantees [1].

Mining and learning strategies try to enhance the extraction of seed sets. DIEM [32] proposes to learn propagation models on the influence graph to produce a prediction model. C2IM [27] focuses on influence communities' extraction in order to ease the connection to the influencers. [36] proposes the factorization of bandits' methods in order to predict influencers through iterations of reward strategies. However it hardly scales in seed set sizes (generally 50) and, moreover, models are not designed to be time dependent and flexible.

OIM [19], AIM [34] and TAIM [33] are recent works on real-time influence maximization which need to be noticed. They propose adaptative strategies that compute the seed set dynamically by incrementally watching the influence impact of users on time windows and choose the optimal one on this setting. OPIM [29] also proposed an extension of Borg's solution [3] by deriving in real time the approximated seed set and then choose the node with the largest marginal coverage. Even if those strategies properly approximate the seed set influence iteratively, it cannot fit with RTB constraints for which a decision must be made on a single user (bid) and not on a set of users at once. Thus it cannot be an approximation on the whole graph but a local decision at once.

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 [1]. There exists a large number of specific IM contributions which have

been listed in [24]. 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 [35] 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. 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 RTIM, *Real-Time Influence Maximization* which targets influential users in real time, henceforth generating a seed set of influencers under real-time bidding constraints. To ensure this, RTIM takes place in two stages: a pre-processing stage and a live stage which we now present.

3 RTIM Approach

RTIM is meant to perform in an 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 whether it is useful for targeting. In minutes, millions of users are quickly navigating through many dozens of websites each leading for any RTB advertising agency to a continuous stream of online users [38]. Advertisers can only target, with advertisements, users who appear in the stream and target them under 100 ms 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 advertiser's 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. Contrary to these solutions, RTIM allows us to adapt our influence maximization strategy to an advertisement campaign taking place in an 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" [30]). Indeed, IMM has proven that it can compute an effective seed set in a reasonable time regardless of

the graph size. Even more so, it can compute large seed set sizes of tens of thousands of users as an RTB advertising campaign requires it.

Static algorithms, such as IMM [30], 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 [26]. 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 $E \subseteq V^2$ 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 $w_I : E \to \mathbb{R}$ 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 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 ($10,000$ in [18]). 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, 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

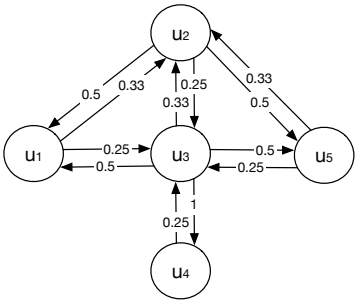

Fig. 1 Influence graph $\mathcal{G}_I$ with weighted edges

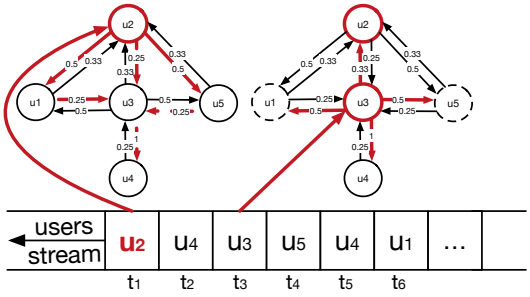

Fig. 2 Ex. of the live stream ($\mathcal{T}$) of available users

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 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 value 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 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 online users and their interconnected graph. $\mathcal{T}$ is a basic example of an 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 I_5 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:

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

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 $\overrightarrow{0}$ vector.

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 propagation) *Consider the social graph $\mathcal{G}(V, E)$ and its influence graph $\mathcal{G}_I(V, E, w_I)$ as stated in Section 3. The* activation probability *vector $AV(t_k)$ for $\mathcal{G}$ at 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:

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

This model corresponds to a *Matrix population model* [21][1]. Thus, we can guarantee its convergence since 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 k^{th} 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., $a_{\mathcal{S}}(u) = \mathsf{P}(\bigcup_{p_j \in P_{uv_i}} p_j)$). 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:

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

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

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($\mathcal{G}, v, p * w_{uv}, d - 1$)

	Youtube	LiveJournal	Twitter
# of nodes	1.13M	3.99M	41M
# of edges	5.97M	69.3M	1.46B
Degree Mean	10.53	34.70	70.50
Degree Variance	10,304.01	7,381.26	6,426,184.47
Degree Standard Deviation	101.50	85.91	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 θ_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 $A \leftarrow \vec{0}$
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: $\mathcal{S} \leftarrow \mathcal{S} \cup u$

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 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

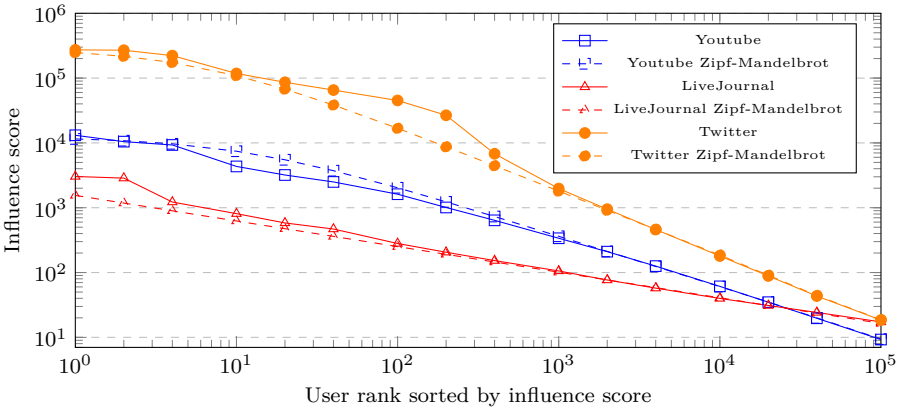

Fig. 3 Datasets' influence score distributions

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 score vector I is:

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

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 $t = 2$, 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 final seed set size is less than the maximum seed set size $k = 3$. It is due to the time-dependent live stream which prevents filling the seed set.

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

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 the way more closely 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 graph 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 fewer connections (mean degree of 10) but with a high variation of degree compared to its size. `LiveJournal` is highly connected with a high number of edges and a mean degree of 34. However, 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 [23], 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).

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

	Youtube		LiveJournal		Twitter	
Influence score	*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

`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 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 score 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 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 their 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 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 choosing influencers in a real-time bidding stream of users. In order to do this, we need to compute users' influence scores, 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 [30] because in [1] 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 [1] in C++.

Stage I: Pre-processing. First, we run `IMM` in its entirety and add k users to its seed set $\mathcal{S}_{IMM}$. Recall that IMM states that every user in the graph appears equiprobable, and more precisely assumes that they will appear in the stream. However, during the ad 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 [1] 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 parallelized simulations per node ($n = 10,000$). 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.

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. User probability of being in the stream is:

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

  where deg_{u_i} is the degree of u_i, the sum of 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 and represent the large majority of online users in the stream even though this does not reflect the reality. This *Log*

	Targeting time	Activation time		
	Average	Average	Median	Max
`Youtube`	0.5 μs	70.3 ms	6.30 ms	193.4 ms
`LiveJournal`	1.0 μs	61.1 ms	6.03 ms	192.0 ms
`Twitter`	3.1 μs	85.9 ms	44.5 ms	411.1 ms

Table 4 Target and update activation probabilities time

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.

Moreover each stream is a subset of the total number of users, we choose 10% of the total number of users. In this setting, we ensure that all the users are not necessarily available online during the ad campaign: $|\mathcal{T}| = 10\% \times |V|$

Table 3 gives for each stream the proportions of influence scores and size in our experiments. As expected *Uniform* distributions contains lower 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`.

Stage III: Live stream process. For `IMM`, if a user in the stream belongs to $\mathcal{S}_{IMM}$, he is targeted and added to the seed set $\mathcal{S}^*_{IMM}$. A user is targeted only once, even if he appears twice. 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 $\sigma_{MC}(u_i)$ is greater than the influence threshold θ_I. If both conditions are satisfied then 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). The update operation can be done by a separate thread.

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

6.1.1 Stage IV: Seed Sets Evaluation

To compare the performance of both IM algorithms we compute and compare $\sigma_{MC}(\mathcal{S}^*_{IMM})$ and $\sigma_{MC}(\mathcal{S}^*_{RTIM})$ during the live stage in order to analyze how the seed set score evolves. Because of the computation cost of $\sigma_{MC}(\mathcal{S})$, during

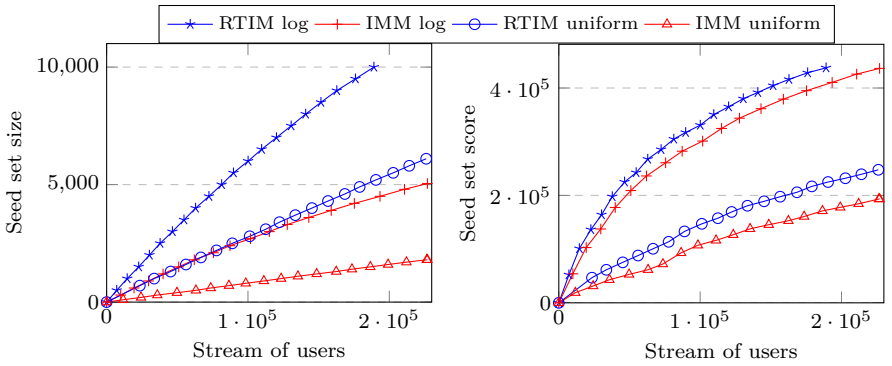

Fig. 4 Seed set *size* evolution with `Youtube` **Fig. 5** Seed set *score* evolution with `Youtube`

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

`Youtube`. In the following, 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 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 5 shows that RTIM produces seed sets with higher scores than IMM. *Uniform* and *Log* streams witness different evolution. In fact, IMM hardly finds influencers in the *Uniform* distribution where 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

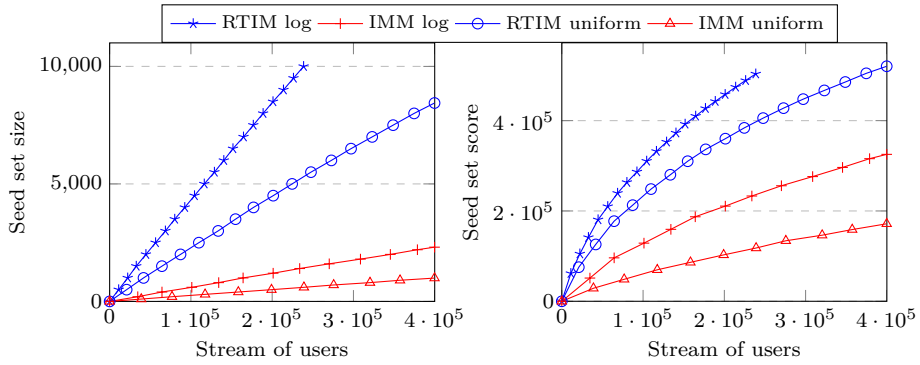

Fig. 6 `LiveJournal`'s seed set *size* evolution **Fig. 7** `LiveJournal`'s seed set *score*

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 than previous chosen ones, explained by the sub-modular property of the IM problem.

`LiveJournal`. This dataset is evaluated with a stream of 4×10^5 online users.

Figure 6 shows the evolution of seed set sizes. We can see that IMM finds very few expected influencers and produces 8.5 times fewer seeds for the *Uniform* stream (resp. 7 for the *Log* stream) than RTIM. In fact, RTIM targets influencers more easily than IMM. This is explained by the specific distribution of scores we explained in Section 5 with a very slow decreasing of the scores ($\alpha = 0.395$). It is 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. 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.

`Twitter`. 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.

Figure 8 shows that RTIM seed sets evolve very quickly for both *Uniform* and *Log* streams. This is due to the huge amount of high score users of the `Twitter` distribution ($B = 1.7 \times 10^6$ - Section 5), consequently RTIM targets

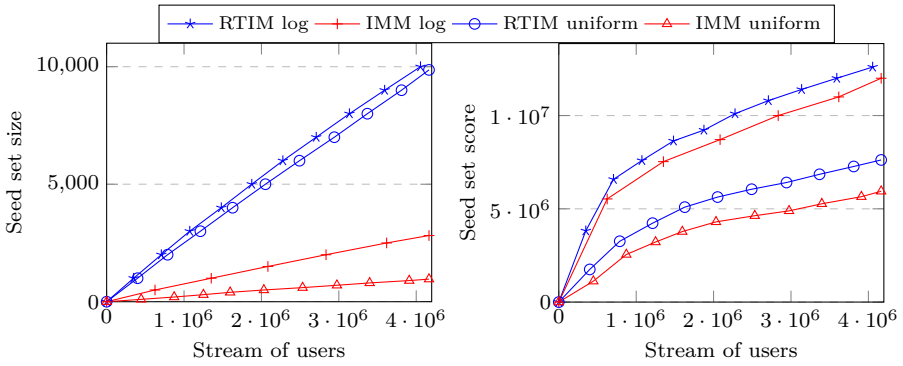

Fig. 8 Seed set size evolution with `Twitter` **Fig. 9** Seed set score evolution with `Twitter`

any user in the stream that reaches the threshold θ_I. On the other hand, IMM evolves more slowly, 10 times less for *Uniform* (resp. 3.5 times less for *Log*). RTIM fulfills the ad 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 guarantee a global impact in a minimum amount of time.

6.3 `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 (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 effectiveness 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.

7 Seed-Set Size Guarantee to Maximize Ad Campaigns

Although finding the optimal seed set of size k to maximize the influence of a network is the focus of Influence Maximization, one other issue is to determine in advance the correct value for k. In fact, making the choice of top-k influencers for an ad campaign is crucial in order to give a guarantee of the budget to reach a sufficient number of users on the network.

The choice of the seed-set size will help advertisers to decide how much they wish to influence of social networks. With a guarantee of efficiency, they will determine the impact of the ad campaign in advance and maybe more if the IM algorithm performs better than expected.

The issue is for a given seed set size to guarantee or estimate its influence score; the number of activated users. For instance, on the Twitter network with tens of millions of users, rather than targeting millions of users to reach everyone (100%); we may only want to target tens of thousands of users to reach 25% of the social network.

However, as discussed in the state-of-the-art, finding seed set in a network of N users, there are 2^N total seed sets to test in a brute force approach to search for the optimal seed set $\mathcal{S}^*$ and by setting k there are $\binom{N}{k}$ combinations. As fat as we know, no existing IM paper has established a method of determined the proper seed set size, nor do existing IM algorithms offer guarantees of finding a seed set capable of covering a given portion of the network.

7.1 Expected Seed Set Size

We denote k_p^* the expected seed set size which influences a sufficient number of users in the graph given by the rate p. Thus k_p^* is the seed set size that guarantees to reach $r \times |V|$ users, where p is a percentage between 0 and 1.

Although the expected k_p^*, is bounded between 1 and $p \times |V|$, we can infer that k_p^* is at least equal to the number of connected components in $\mathcal{G}$ in order to reach disconnected communities of users.

In order to give k_p^*, we propose an approach that estimates them. It consists in randomly sampling as many users necessary to influence $p \times |V|$ users. We apply propagation models on each sampled user in order to see if he activates his neighborhood or not. The final sample size is a possible seed-set size k'_p. This random sampling is performed until convergence of the sample size.

Of course, to give a better approximation of k_p^*, activated users during a sampling cannot be targeted. The number of uniformly chosen users that activate $p \times |V|$ users is an estimation k_p^*. The expected seed-set size is then computed using the average of the results for each iteration, where I is the number of iterations required to reach convergence: $k_p^* \approx \frac{\sum_{i \in I} k'_p(i)}{I}$.

Reach		Seed Set Size by edge weight models							
%	# users	**0.01**	**0.1**	**0.3**	**0.5**	**0.7**	**0.8**	**0.9**	**WC**
100	15,233.00	14,919	12,183	7,768	5,037	3,292	2,669	2,173	7,589
95	14,471.35	14,159	11,428	7,024	4,320	2,657	2,114	1,724	6,912
90	13,709.70	13,399	10,680	6,312	3,680	2,162	1,703	1,384	6,331
80	12,186.40	11,886	9,220	4,982	2,586	1,375	1,044	824	5,318
70	10,663.10	10,377	7,806	3,777	1,687	758	525	373	4,428
60	9,139.80	8,874	6,434	2,693	948	275	124	48	3,622
50	7,616.50	7,378	5,114	1,730	360	26	13	9	2,883

Table 5 `Nethept` expected Seed Set Sizes k_p^* according to the reach

Reach		Seed Set Size by edge weight models							
%	# users	**0.01**	**0.1**	**0.3**	**0.5**	**0.7**	**0.8**	**0.9**	**WC**
100	12,006.0	10,915	6,817	3,378	1,806	940	655	440	5,778
95	11,405.7	10,316	6,222	2,792	1,241	450	243	123	5,213
90	10,805.4	9,720	5,636	2,232	749	118	3	2	4,708
80	9,604.8	8,534	4,498	1,208	15	2	2	2	3,820
70	8,404.2	7,360	3,408	339	2	2	2	2	3,057
60	7,203.6	6,197	2,376	2	2	2	2	2	2,391
50	6,003.0	5,048	1,408	2	2	2	2	2	1,811

Table 6 `HepPh` expected Seed Set Sizes k_p^* according to the reach

7.2 Preliminary Experiments

To achieve those simulations, we used the `Nethept` and `HepPh` [14] graph datasets composed of 15,200 nodes (resp. 12,006) and 61,300 edges (resp. 80,578). We ran experiments using various edge weighting methods, like Weighted Cascade (like in RTIM) as well as edge weights in $[0.01, 0.1, 0.3, 0.5, 0.7, 0.8, 0.9]$ to test the effect of weights on the size of k_p^*. Reach is the number p and its correspondence in number of users who are activated by a sample.

Tables 5 and 6 give results of simulations of k_p^* for each edge weights strategy according to the given reach p. We can say that p corresponds to the goal of an ad campaign. We can see that the seed set size increases the smaller the edge weights are. For constant weights, when those values are higher than 0.3 reaching at least of half the network requires very low seed set sizes, 360 for Netept where the number of edges is lower than HepPh, and a size of 2 for HepPh which is the number of connected components.

It also shows that the WC weight model requires to target 7,589 users to reach 100% of users in Hep and 5,778 users must be targeted to reach 100% of users in Phy which corresponds to half the number of users in the graph.

It is interesting to notice that most of state-of-the-art algorithms deal with small seed set sizes (mostly 300) in order to make it scalable in terms of computation time. We show in these experiments that it requires far more users to influence at least half the number of users. However, remind that we give an expected seed set size and not the optimal one which means that any IM algorithm must give a higher influence score to be efficient.

7.3 Conclusion

Our approach offers an approximation of k_p^* which would provide a valuable indicator of the budget of an online advertising strategy given how much it would cost to target k^* number of users during the campaign. Moreover it gives a threshold of efficiency of IM algorithms.

8 Conclusion

In this article we have shown that it is possible to answer the influence maximization problem in a real-time bidding environment, which up to now has not been applied to IM algorithms. We have shown that static IM algorithms, such as IMM, can solve this problem in a reasonable time with a seed set of 10,000 influencers. However, RTIM is a solution that competes static IM by using a dynamic IM algorithm based on the local influence of each user. With streams or ad campaign of finite size can outperform static algorithms.

It is important to note that the RTB environment is more complex so than the constraints which we used. For instance, a user who was targeted should not have seen the ad, 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 more 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 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 needs to recompute the 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 connections, has an impact on the final seed set score.

Acknowledgements This research is supported and financed by Kwanko. We thank them for their contributions.

References

1. Arora, A., Galhotra, S., Ranu, S.: Debunking the myths of influence maximization: An in-depth benchmarking study. In: Proc. SIGMOD '17, pp. 651–666. ACM, New York, NY, USA (2017). DOI 10.1145/3035918.3035924
2. Aslay, C., Lakshmanan, L.V., Lu, W., Xiao, X.: Influence maximization in online social networks. In: Proc. WSDM'18, p. 775–776. ACM, New York, NY, USA (2018). DOI 10.1145/3159652.3162007
3. Borgs, C., Brautbar, M., Chayes, J., Lucier, B.: Maximizing social influence in nearly optimal time. In: Proc. SODA '14, pp. 946–957. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2014). DOI 10.5555/2634074.2634144

4. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proc. KDD'10, pp. 1029–1038. ACM, Washington, DC, USA (2010). DOI 10.1145/1835804.1835934
5. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proc. KDD'09, pp. 199–208. Paris, France (2009). DOI 10.1145/1557019.1557047
6. Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: Proc. ICDM'10, pp. 88–97. Sydney, Australia (2010). DOI 10.1109/ICDM.2010.118
7. Cheng, S., Shen, H., Huang, J., Chen, W., Cheng, X.: Imrank: influence maximization via finding self-consistent ranking. In: Proc. KDD'14, pp. 475–484. Gold Coast , QLD, Australia (2014). DOI 10.1145/2600428.2609592
8. Cheng, S., Shen, H., Huang, J., Zhang, G., Cheng, X.: Static greedy: solving the apparent scalability-accuracy dilemma in influence maximization. CoRR **abs/1212.4779** (2012)
9. Cohen, E., Delling, D., Pajor, T., Werneck, R.F.: Sketch-based influence maximization and computation: Scaling up with guarantees. In: Proc. CIKM'14, pp. 629–638. Shanghai, China (2014). DOI 10.1145/2661829.2662077
10. Domingos, P., Richardson, M.: Mining the network value of customers. In: Proc. KDD'01, pp. 57–66. ACM, New York, NY, USA (2001). DOI 10.1145/502512.502525
11. Galhotra, S., Arora, A., Roy, S.: Holistic influence maximization: Combining scalability and efficiency with opinion-aware models. In: Proc. SIGMOD '16, pp. 743–758. ACM, New York, NY, USA (2016). DOI 10.1145/2882903.2882929
12. Galhotra, S., Arora, A., Virinchi, S., Roy, S.: ASIM: A scalable algorithm for influence maximization under the independent cascade model. In: Proc. WWW'15, pp. 35–36. Florence, Italy (2015). DOI 10.1145/2740908.2742725
13. Goyal, A., Bonchi, F., Lakshmanan, L.V.: Learning influence probabilities in social networks. In: Proc. WSDM'10, pp. 241–250. ACM, New York, NY, USA (2010). DOI 10.1145/1718487.1718518
14. Goyal, A., Lu, W., Lakshmanan, L.V.: CELF++: Optimizing the greedy algorithm for influence maximization in social networks. In: Proc. WWW '11, pp. 47–48. ACM, New York, NY, USA (2011). DOI 10.1145/1963192.1963217
15. Goyal, A., Lu, W., Lakshmanan, L.V.S.: SIMPATH: an efficient algorithm for influence maximization under the linear threshold model. In: Proc. IEEE ICDM '11, pp. 211–220. Vancouver, BC, Canada (2011). DOI 10.1109/ICDM.2011.132
16. Huang, K., Wang, S., Bevilacqua, G., Xiao, X., Lakshmanan, L.V.S.: Revisiting the stop-and-stare algorithms for influence maximization. Proc. VLDB Endow. **10**(9), 913–924 (2017). DOI 10.14778/3099622.3099623
17. Jung, K., Heo, W., Chen, W.: Irie: Scalable and robust influence maximization in social networks. In: Proc. IEEE ICDM'12, pp. 918–923. Brussels, Belgium (2012). DOI 10.1109/ICDM.2012.79
18. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: Proc. KDD'03, pp. 137–146. ACM, New York, NY, USA (2003). DOI 10.1145/956750.956769
19. Lei, S., Maniu, S., Mo, L., Cheng, R., Senellart, P.: Online influence maximization. In: L. Cao, C. Zhang, T. Joachims, G.I. Webb, D.D. Margineantu, G. Williams (eds.) Proc. SIGKDD'15, pp. 645–654. ACM (2015). DOI 10.1145/2783258.2783271
20. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proc. KDD '07, pp. 420–429. ACM, New York, NY, USA (2007). DOI 10.1145/1281192.1281239
21. Leslie, P.H.: On the Use of Matrices in Certain Population Mathematics. Biometrika **33**(3), 183–212 (1945). DOI 10.1093/biomet/33.3.183
22. Liang, W., Shen, C., Li, X., Nishide, R., Piumarta, I., Takada, H.: Influence maximization in signed social networks with opinion formation. IEEE Access **7**, 68837–68852 (2019). DOI 10.1109/ACCESS.2019.2918810
23. Mandelbrot, B.B.: The fractal geometry of nature, vol. 1. WH freeman New York (1982)
24. N., S., B., A., Bhattacharya, S.: Influence maximization in large social networks: Heuristics, models and parameters. Future Generation Comp. Syst. **89**, 777–790 (2018). DOI 10.1016/j.future.2018.07.015

25. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions—i. Mathematical Programming **14**(1), 265–294 (1978). DOI 10.1007/BF01588971
26. Ohsaka, N., Akiba, T., Yoshida, Y., Kawarabayashi, K.: Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In: Proc. AAAI'14, pp. 138–144. Québec City, Québec, Canada (2014)
27. Singh, S.S., Kumar, A., Singh, K., Biswas, B.: C2im: Community based context-aware influence maximization in social networks. Physica A: Statistical Mechanics and its Applications **514**, 796 – 818 (2019). DOI https://doi.org/10.1016/j.physa.2018.09.142. URL http://www.sciencedirect.com/science/article/pii/S0378437118312822
28. Spencer, S., O'Connell, J., Greene, M.: The arrival of real-time bidding. Tech. rep., Google (2011)
29. Tang, J., Tang, X., Xiao, X., Yuan, J.: Online processing algorithms for influence maximization. In: Proc. SIGMOD'18, p. 991–1005. ACM, New York, NY, USA (2018). DOI 10.1145/3183713.3183749
30. Tang, Y., Shi, Y., Xiao, X.: Influence maximization in near-linear time: A martingale approach. In: Proc. SIGMOD'15, pp. 1539–1554. ACM, New York, NY, USA (2015). DOI 10.1145/2723372.2723734
31. Tang, Y., Xiao, X., Shi, Y.: Influence maximization: Near-optimal time complexity meets practical efficiency. In: Proc. SIGMOD'14, pp. 75–86. ACM, New York, NY, USA (2014). DOI 10.1145/2588555.2593670
32. Tian, S., Zhang, P., Mo, S., Wang, L., Peng, Z.: A learning approach for topic-aware influence maximization. In: J. Shao, M.L. Yiu, M. Toyoda, D. Zhang, W. Wang, B. Cui (eds.) Web and Big Data, pp. 125–140. Springer International Publishing, Cham (2019)
33. Tong, G., Wang, R., Ling, C., Dong, Z., Li, X.: Time-constrained adaptive influence maximization (2020). URL https://arxiv.org/abs/2001.01742
34. Tong, G.A., Wu, W., Tang, S., Du, D.Z.: Adaptive influence maximization in dynamic social networks. IEEE/ACM Transactions on Networking **25**, 112–125 (2017)
35. Wang, Y., Fan, Q., Li, Y., Tan, K.L.: Real-time influence maximization on dynamic social streams. Proc. VLDB Endow. **10**(7), 805–816 (2017). DOI 10.14778/3067421.3067429
36. Wu, Q., Li, Z., Wang, H., Chen, W., Wang, H.: Factorization bandits for online influence maximization. In: Proc. KDD'2019, pp. 636–646 (2019)
37. XXX, et al.: RTIM: A real-time influence maximization strategy. In: Proc. WISE'19, vol. 11881, pp. 277–292. Springer, Hong Kong, China (2019). DOI 10.1007/978-3-030-34223-4_18
38. Yuan, S., Wang, J., Zhao, X.: Real-time bidding for online advertising: Measurement and analysis. CoRR **abs/1306.6542** (2013)
39. Zhang, M., Dai, C., Ding, C., Chen, E.: Probabilistic solutions of influence propagation on social networks. In: Proc. CIKM '13, pp. 429–438. ACM, New York, USA (2013). DOI 10.1145/2505515.2505718