RTIM: a Real-Time Influence Maximization Strategy

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

RTIM: a Real-Time Influence Maximization Strategy

David Dupuis[1,2], Cédric du Mouza[3], Nicolas Travers[1,3], and Gaël Chareyron[1]

[1] Léonard de Vinci Pôle Universitaire, Research Center, Paris La Défense, France
firstname.lastname@devinci.fr
[2] Kwanko, Paris, France
[3] CEDRIC Lab, CNAM Paris, France
prenom.nom@cnam.fr

Keywords: Real-Time Bidding · Influence Maximization · Social Network

Abstract. Influence Maximization (IM) consists in finding in a network the top-k influencers who will maximize the diffusion of information. However, the exponential growth of online advertisement is due to Real-Time Bidding (RTB) which targets users on webpages. It requires complex ad placement decisions in real-time to face a high-speed stream of users. In order to stay relevant, the IM problem should be updated to answer RTB needs. While traditional IM generates a static set of influencers, they do not fit with an RTB environment which requires dynamic influence targeting. This paper proposes RTIM, the first IM algorithm capable of targeting users in a RTB environment. We also analyze influence scores of users in several social networks and provide a thorough experimental process to compare static versus dynamic IM solutions.

1 Introduction

Since Kempe et al. [16], Influence Maximization (IM) is a well studied maximum coverage problem which consists in finding the smallest subset of individuals in a social network who will maximize information diffusion through social influence. In this paper, we are interested in enhancing IM methods with Real-Time Bidding (RTB) constraints. We consider that a user can be influenced because he saw an ad, interacted with it, or purchased the product. To be more relevant, IM algorithms need to take into consideration time and targeting requirements of RTB.

However, online advertising revenue outpaced all other advertising strategies thanks to the advent of RTB [24, 27] and Social Network Services (SNS). RTB is an online auction system which allows online advertisers to bid in real-time for ad locations on a webpage in less than 100ms [27]. RTB ad targeting is initially based on the content of the web page and users' consumer profile, but fails to rely on the social value of each customer as suggested in [9]. As far as we know no RTB algorithms attempt to find an IM solution to improve bidding decisions. They only consider influence as a parameter and not as a propagation value on

a network. It is important to note that there is potential here for IM to integrate full bidding, however this approach is left for future work.

Existing IM algorithms propose 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 to find seed sets up to 200 seeds in a large social network [1], they do not scale up or take into account real time streams of users.

This article targets the issue with the following constraints: `a)` only an online user can be targeted, `b)` deciding whether to target a user must be done in under 100ms, `c)` the propagation influence score relies on a social network containing millions of users and relationships, `d)` 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`.

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 in Section 2 the state of the art on IM. Section 3 explains our two stages RTIM approach and Section 4 gives the RTIM model and algorithms. In order to understand the impact of our model, we propose an analysis of influence in different datasets in Section 5. This leads to the experimental process in Section 6 with a dynamic stream evaluation. Finally, we conclude and give some perspectives in Section 7.

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 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

Kempe et al. [16] propose two common propagation models: *Independent Cascade* (IC) and *Linear Threshold* (LT). *IC* considers that each user can be influenced by a neighbor independently of any of his other neighbors. *LT* 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 [12] there is a real challenge in acquiring real-world data to build datasets containing accurate influence probabilities. Common practice is to use theoretical edge weight models. For *IC*, the `Constant` model is where each weight e_{ij} is given a constant probability [3, 8, 10, 11, 13, 16]. Some define $p \in [0.01, 0.1]$ [4, 23]. The `Tri-Valency` model is where the weights are randomly chosen from a list of probabilities such as {0.001, 0.01, 0.1} [3, 6, 15]. Finally,the `Weighted Cascade (WC)` model is where $e_{ij} = \frac{1}{|In(v_j)|}$ where $In(v_j)$ is the number of neighbors that influence u [3, 4, 8, 10, 11, 6, 7, 16, 25, 26, 19]. Under `WC`, 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.

For *LT*, the general edge weight rule is that the sum of the weights which must equal 1. Therefore, `WC` applies to *LT*. Additional alternative models can be found in [21] where an extensive IM state-of-the art is done.

IC 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. *LT* doesn't change the fundamental approach of our algorithm, it should not be difficult to extend to. For these reasons, we limit our approach to *IC*. In addition, we define the edge weights using the *WC* model, because we believe it better corresponds to the diversity of influence between individuals in a real-world social network.

2.2 Properties

Kempe et al. [16] prove that influence maximization is NP-Hard under both the *IC* and *LT* models and computing the influence score is monotone and sub-modular which Chen et al. [3] prove to be #P-Hard under the *IC* model.

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 [22]. As it is presented previously, many IM algorithms rely on this theoretical guarantee to validate their strategy.

2.3 Computing score

Influence Score: Computing the influence score requires solving Eq. (1), which is a generalization of the inclusion-exclusion principle from [28].

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

In Equation 1, the influence score of a seed set $\sigma(\mathcal{S})$ is the sum of activation probabilities $a_S(v)$ 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.

Computing $\sigma(\mathcal{S})$ is proven by [15] to be #P-Hard but [16] approximate the exact result by running $n = 10,000$ *Monte Carlo* simulations. In this article, the result of these simulations is written $\sigma_{MC}(\mathcal{S})$.

[16] prove that IM is an NP-Hard problem. In fact, it simply consists of two challenges. The first is finding the optimal seed set out of $2^{|V|}$ subsets of users or $\binom{N}{k}$ if we know k, and computing the influence score according to Equation 1.

2.4 Algorithms

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

GREEDY [16], CELF [17] and CELF++ [13] are all three lazy-forward algorithms which 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 $|\mathcal{S}| = k$. CELF, CELF++ improve computation time with the submodular property by storing temporary results.

Reverse Influence Sampling method [2] from Borg et al. like in IMM [25], TIM or TIM+ [26], 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 sets). Using a greedy method, they build $\mathcal{S}^*$ by continuously adding to $\mathcal{S}^*$ the user who covers the greatest number of Reverse Reachable sets and removing them from $\mathcal{R}$.

Approximation algorithms such as EaSyIM [10], IRIE [15], SIMPATH [14], LDAG [5] or IMRANK [6], offer heuristics to compute $\sigma(\mathcal{S})$ Eq. (1), such as using the most probable path or the independence of paths.

Conclusions: Greedy solutions require hours or days of processing due to the repeated computation of $\sigma_{MC}(\mathcal{S})$. They perform poorly for seed sets larger than 50 and do not scale to large datasets [1]. Heuristics don't offer theoretical guarantee and often lack in precision. Sampling algorithms are significantly faster than greedy, are more precise than heuristics and offer a theoretical guarantee. In addition, none of these solutions are meant to dynamically generate a seed set with RTB constraints.There exists a great number of specific IM contributions

which have been listed in [21]. It shows clearly, that no contributions have been made regarding the analysis of the IM challenge in a stream of online users.

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

3 RTIM Approach

RTIM is meant to perform in a RTB environment. The latter, consists of users who connect to a website which sells ad slots. The IM algorithm has to determine wether it is useful for targeting. As we know, these users, all belong to a very large social network through which they may influence neighbors.

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. While traditional approaches determine the best seed set of targeted users by processing a graph in which any user is considered online. Contrary to these solutions, RTIM allows us to adapt our IM strategy to take place in a RTB streaming environment.

Static algorithms, such as IMM [25], correspond to an optimistic approach which assumes that pre-computed users from their seed set will necessarily be online in the stream. However, many users of the pre-defined seed set won't be available to target during the advertisement campaign. In contrast, our 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.

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

First, an *influence graph* $\mathcal{G}_I(V, E, w_I)$ is built with weights on each edge to estimate the influence based on the number of incoming edges of a vertex. This influence estimation is commonly adopted in influence propagation [23]. It is 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 $w_I : E \to \mathbb{R}$ 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: $w_I(e_{12}) = w_I(e_{32}) = w_I(e_{52}) = 1/3 = 0.33$

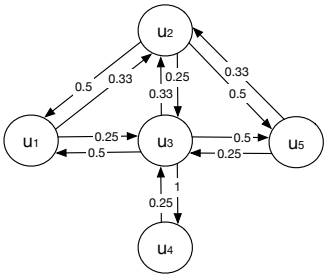

Fig. 1. Influence graph $\mathcal{G}_I$

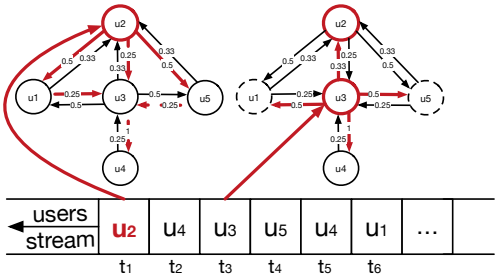

Fig. 2. Ex. of the live stream ($\mathcal{T}$) of available users

To estimate the influence score, we use the *Monte Carlo* approach by running n simulations, where n is a large number ([16] set $n = 10,000$). The influence score of each user u is the average number of users activated for all simulations.

Each simulation randomly test each outgoing edge of a user against the edge weight $w_I(e_{ij})$. When a neighbor is activated we can then recursively test neighbors until no more nodes are activated.

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. Consequently, this computation must be performed offline and all influence scores are stored in a vector I: $\forall u_i \in \mathcal{G}, I_i = \sigma_{MC}(u_i)$

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 stream, users to target. Consider the temporal stream of users $\mathcal{T}$ in which appears every online connection events of users from $\mathcal{G}$. 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 the influence score of users and the probability that they have already been targeted by 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 (above θ_I) to be a potential target for the ad campaign or not. We also check the probability for him to be activated by users he follows (above θ_A). If θ_I is validated and not θ_A, the user is targeted and added to the seed set. His activation probability is set to 1. This activation is propagated in the neighbourhood. 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 their correlation with 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. When the first user u_2 appears (time t_1), we verify his influence score I_2. 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 probabilities to their own neighbors. 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 and is avoided. 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 ($A_3 < \theta_A$). Thus the activation probability is propagated to u_1, u_2, u_5 and u_4. When u_5 appears in $\mathcal{T}$, even if his influence score is higher than θ_I, $A_5 > \theta_A$ since he has been influenced by u_2 and u_3. 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.

4 RTIM Model

Traditional influence maximization algorithms 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. If the advertisement campaign is not time-limited (infinite stream), these solutions potentially maximize the total score.

RTIM's strategy is quite different since it decides to target a user in real-time when he is available. So RTIM can be considered as a pessimistic algorithm since we decide to add him to the final seed set instantaneously, even if a "better" user to add to the seed set appears later 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 ad and we avoid targeting them in the future. We must update other activation probabilities through influence propagation to obtain the $AV(t_k)$ probability vector.

Definition 2 (Activation probability propagations) *Consider $\mathcal{G}(V, E)$ the social graph 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:*

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

Algorithm 1 Updating activation probabilities

Require: graph $\mathcal{G}$, nodes u and v, v's activation probability $A[v]$, u's neighbors $\mathcal{N}_u$, current path weight p, depth d

```
1: procedure ActivationScores(G, u, p, d)
2:     for v ∈ N_u do
3:         A[v] ← 1 − (1 − A[v]) ∗ (1 − p ∗ w_uv)
4:         if d > 1 then
5:             ActivationScores(G, v, p ∗ w_uv, d − 1)
```

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* [18][4], 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 $\overrightarrow{1}$.

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 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 to reach, set to the influence score of the k^{th} influencer. We also define the activation probability threshold θ_A, set by default to 0.5. Any user whose activation probability is greater than θ_A is considered to be 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 users' activation probability. To achieve this, we propagate probabilities at depth less than d. For a 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.

Equation 1 gave the activation probability of a user v. Since we consider paths of length 2 all paths between u and v are independent:

$$A[u] = \mathsf{P}\Big(\bigcup_{p_j \in P_{uv}} p_j\Big) = 1 - \prod_{w_i \in \mathcal{P}^d_{uv}} (1 - w_i), \tag{2}$$

Algorithm 1 illustrates activation probabilities updates. For each neighbor v of user u we propagate his activation probability (line 3). 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. However, updates 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

[4] Our model is not a Markov chain since the sum of a column can exceed 1.

Algorithm 2 RTIM Live

Require: graph $\mathcal{G}$, user u, vector of influence scores I, influence threshold θ_I, u's activation probability $A[u]$, depth d, stream of users $\mathcal{T}$, seed set $\mathcal{S}$ and 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: $\mathcal{S} \leftarrow \mathcal{S} \cup u$

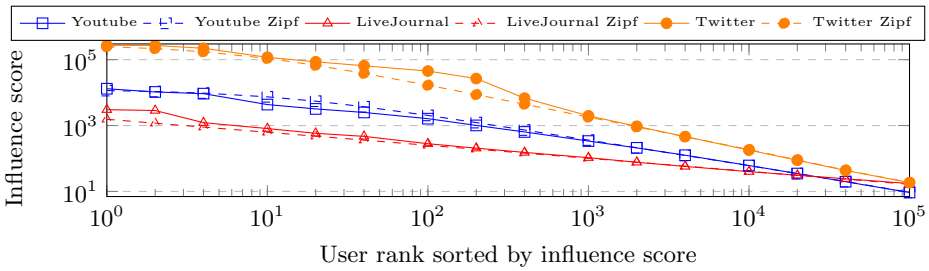

Fig. 3. Datasets' influence score distributions

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).

5 Influence Analysis

Our model is experimented with empirical datasets of different sizes: `Youtube`, `LiveJournal` and `Twitter`. We need their characteristics to understand the impact of our approach. We can see in Table 1 the global statistics: # nodes, # 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 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 (low variance and STD). `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.

Figure 3 shows the distribution of influence scores for our graph datasets. These distributions can be characterized by a standard *Zipf-Mandelbrot* distribution [20], traditionally used for distribution of ranked data. It is defined by: $ZM(r) = \frac{B}{(r_0+r)^\alpha}$ 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.

	# of nodes	**# of edges**	**Deg. Mean**	**Degree Var.**	**STD**
`Youtube`	1.13M	5.97M	10.53	10,304	101
`LiveJournal`	3.99M	69.3M	34.70	7,381	85
`Twitter`	41M	1.46B	70.50	6,426,184	2,534

Table 1. Datasets characteristics

B	r_0	α	$\mathcal{X}^2$Pearson Value
8×10^4	11	0.78	0.976
1.55×10^3	0	0.395	0.971
1.7×10^6	6	0.99	0.969

Table 2. Zipf-Mandelbrot parameters for graph datasets

Table 2 gives the corresponding values for those Zipf-Mandelbrot distributions and the $\mathcal{X}^2$-Pearson values (observation probabilities) found for each distribution. `Youtube` and `Twitter` behave similarly with a high r_0 (resp. 11 and 6) leading to 100 to 750 top-influencers. We can see that `Twitter` has more top influencers 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 more slowly 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 an influencers score evolves.

6 Experiments

We wish to show in this section the impact of choosing influencers in a real-time 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 [25] since it is the best compromise between computation speed, scalability and accuracy, especially on large datasets. The code for IMM is provided by [1] in C++.

	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 3. Update activation probabilities time

Stage I: Pre-processing. First, `IMM` is run in its entirety and adds k users to its seed set $\mathcal{S}_{IMM}$ ($k = 10,000$ and optimal seed $\epsilon = 0.1$). Recall that IMM relies on the fact that every user in the graph has the same probability to appear.

`RTIM` uses the *Monte Carlo* approach to compute the influence score of each user in the graph. We run n parallel simulations per node ($n = 10,000$) with a limit depth of 3 for scalability purposes. Thanks to the graph topology with a high connectivity (see Section 5), those *Monte Carlo* simulations converge faster. 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 stream and both algorithms have the opportunity to target influencers. Since no real streams of connected users are available online, we simulate users' behavior in the social networks with different distributions. Streams contain 10% of the total number of users and can appear several times with two different distributions: *Uniform* and *Log*.

The *Uniform* distribution supposes that all users have an equal probability of appearing in the stream. This distribution can be considered to be the worst case where top influencers can appear as frequently as low influencers.

The *Log* distribution supposes 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 $P(u_i \in \mathcal{S}) = \log(deg_{u_i}) / \sum_{u_i \in V} \log(deg_{u_i})$, where deg_{u_i} is the degree of u_i. We apply a logarithm on deg_{u_i} in order to give users with a low influence score a reasonable probability of showing up in the stream. 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.

To ensure the stability of our model, we generated multiple random versions of each stream. Our results are averages of all versions.

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, if $ap(u_i) < \theta_A$ and $\sigma_{MC}(u_i) > \theta_I$, RTIM targets u_i instantaneously. u_i is added to the final seed set $\mathcal{S}^*_{RTIM}$ and we update the activation probability of his neighbors in a separate thread.

Table 3 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.

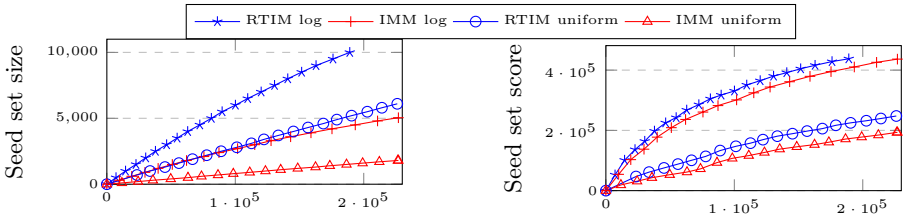

Fig. 4. Seed set score & size evolution with `Youtube`

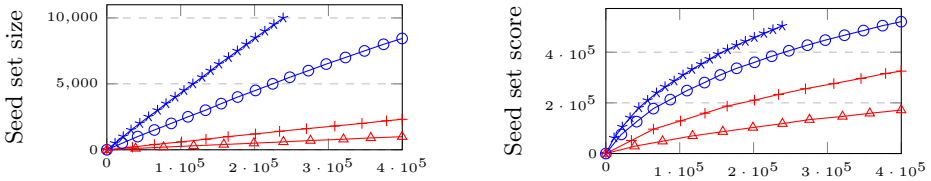

Fig. 5. Seed set score & size evolution with `LiveJournal`

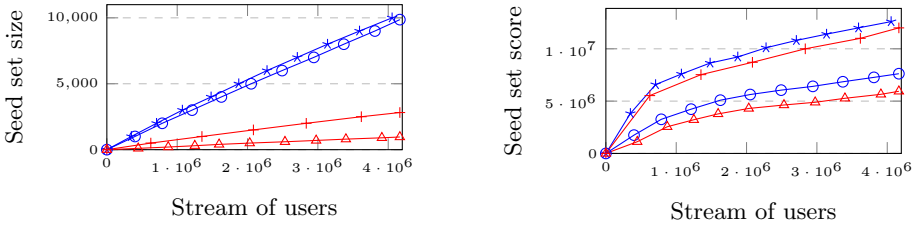

Fig. 6. Seed set score & size evolution with `Twitter`

6.2 Experimental results

In the following experiments, we see the real-time evolution of the seed set influence score and size on the `Youtube`, `LiveJournal` and `Twitter` datasets.

Youtube. Figure 4 gives the results produced for a stream of 2.27×10^5 connected users over the `Youtube` dataset. The left-hand side shows the evolution of the seed set size where 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 more quickly. 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.

The other side 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 targets few users in the *Uniform* distribution, since highly connected users are less often available online. On the other hand, RTIM targets more users according to their local influence on the graph. 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 ad campaign this is not the case.

LiveJournal. Figure 5 focuses on `LiveJournal`. The stream contains almost 4×10^5 online users. On the left part, we can see the evolution of seed set sizes for which 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 which can be explained by the distribution of scores (see 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 the evolution of seed set scores on the right part where 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 can choose a "similar" score in the neighbourhood.

Moreover, 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).

Twitter. The seed sets produced for `Twitter` are presented in Figure 6. The stream is composed of 4.17×10^6 connected users. We observe first 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.

For seed set scores, they evolve similarly to the `Youtube` dataset 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.

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 however IMM is more sensitive to this setting.

The seed set score curve is logarithmic since IM is sub-modular. Indeed, the more users we target the smaller the marginal gain to the overall seed set. 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 avail-

able ones. Second, graphs with very high influence scores (induced by B) give RTIM more choices of influencers and so it fills up the seed set quickly.

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 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. For future works, we propose to extend RTIM to fully answer RTB constraints. Contrary to IM algorithms, RTIM could choose to target another user if a previous targeted one 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 we can 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 dynamically 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.

Acknowledgments. This work was supported by Kwanko

References

1. Arora, A., Galhotra, S., Ranu, S.: Debunking the myths of influence maximization: An in-depth benchmarking study. In: SIGMOD. pp. 651–666. ACM (2017)
2. Borgs, C., Brautbar, M., Chayes, J., Lucier, B.: Maximizing social influence in nearly optimal time. In: SODA. pp. 946–957. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2014)
3. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: SGKDD. pp. 1029–1038. ACM, Washington, DC, USA (2010)
4. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: SIGKDD. pp. 199–208. Paris, France (2009)
5. Chen, W., Yuan, Y., Zhang, L.: Scalable Influence Maximization in Social Networks under the Linear Threshold Model. In: ICDM. pp. 88–97. Sydney, Australia (2010)
6. Cheng, S., Shen, H., Huang, J., Chen, W., Cheng, X.: IMRank: influence maximization via finding self-consistent ranking. In: SIGIR. pp. 475–484 (2014)

7. 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)
8. Cohen, E., Delling, D., Pajor, T., Werneck, R.F.: Sketch-based Influence Maximization and Computation: Scaling up with Guarantees. In: CIKM. pp. 629–638. Shanghai, China (2014)
9. Domingos, P., Richardson, M.: Mining the network value of customers. In: SIGKDD. pp. 57–66. ACM, New York, NY, USA (2001)
10. Galhotra, S., Arora, A., Roy, S.: Holistic influence maximization: Combining scalability and efficiency with opinion-aware models. In: SIGMOD. pp. 743–758. ACM, New York, NY, USA (2016)
11. Galhotra, S., Arora, A., Virinchi, S., Roy, S.: ASIM: A Scalable Algorithm for Influence Maximization under the Independent Cascade Model. In: WWW. pp. 35–36. Florence, Italy (2015)
12. Goyal, A., Bonchi, F., Lakshmanan, L.V.: Learning influence probabilities in social networks. In: WSDM. pp. 241–250. ACM, New York, NY, USA (2010)
13. Goyal, A., Lu, W., Lakshmanan, L.V.: CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. In: WWW. pp. 47–48 (2011)
14. Goyal, A., Lu, W., Lakshmanan, L.V.: SIMPATH: an efficient algorithm for influence maximization under the linear threshold model. In: ICDM. pp. 211–220. Vancouver, BC, Canada (2011)
15. Jung, K., Heo, W., Chen, W.: Irie: Scalable and robust influence maximization in social networks. In: ICDM. pp. 918–923. Brussels, Belgium (2012)
16. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: SIGKDD. pp. 137–146. ACM, New York, NY, USA (2003)
17. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: SIGKDD. pp. 420–429. ACM, NY, USA (2007)
18. Leslie, P.H.: On the Use of Matrices in Certain Population Mathematics. Biometrika **33**(3), 183–212 (1945)
19. Li, Y., Zhang, D., Tan, K.L.: Real-time targeted influence maximization for online advertisements. PVLDB **8**(10), 1070–1081 (2015)
20. Mandelbrot, B.B.: The fractal geometry of nature, vol. 1. Freeman New York (1982)
21. N., S., B., A., Bhattacharya, S.: Influence maximization in large social networks: Heuristics, models and parameters. Future Gene. Comp. Syst. **89**, 777–790 (2018)
22. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions. Mathematical Prog. **14**(1), 265–294 (1978)
23. Ohsaka, N., Akiba, T., Yoshida, Y., Kawarabayashi, K.: Fast and Accurate Influence Maximization on Large Networks with Pruned Monte-Carlo Simulations. In: AAAI. pp. 138–144. Québec City, Québec, Canada (2014)
24. Spencer, S., O'Connell, J., Greene, M.: The arrival of real-time bidding. Tech. rep., Google (2011)
25. Tang, Y., Shi, Y., Xiao, X.: Influence maximization in near-linear time: A martingale approach. In: SIGMOD. pp. 1539–1554. ACM, NY, USA (2015)
26. Tang, Y., Xiao, X., Shi, Y.: Influence maximization: Near-optimal time complexity meets practical efficiency. In: SIGMOD. pp. 75–86. ACM, NY, USA (2014)
27. Yuan, S., Wang, J., Zhao, X.: Real-time Bidding for Online Advertising: Measurement and Analysis. CoRR **abs/1306.6542** (2013)
28. Zhang, M., Dai, C., Ding, C., Chen, E.: Probabilistic solutions of influence propagation on social networks. In: CIKM. pp. 429–438. ACM, NY, USA (2013)