Influence Maximization for Big Data Through Entropy Ranking and Min-Cut

Agustin Sancen-Plaza
Dept. of Electrical Engineering
and Computer Science
CINVESTAV Guadalajara
Guadalajara, Mexico
Email: asancen@gdl.cinvestav.mx

Andres Mendez-Vazquez
Dept. of Electrical Engineering
and Computer Science
CINVESTAV Guadalajara
Guadalajara, Mexico
Email: amendez@gdl.cinvestav.mx

***Abstract*—As Big Data becomes prevalent, the traditional models from Data Mining or Data Analysis, although very efficient, lack the speed necessary to process problems with data sets in the range of million samples. Therefore, the need for designing more efficient and faster algorithms for these new types of problems. Specifically, from the field of social network analysis, we have the influence maximization problem. This is a problem with many possible applications in advertising, marketing, social studies, etc, where we have representations of influences by large scale graphs. Even though, the optimal solution of this problem, the minimum set of graph nodes which can influence a maximum set of nodes, is a NP-Hard problem, it is possible to devise an approximated solution to the problem. In this paper, we have proposed a novel algorithm for influence maximization analysis. This algorithm consist in two phases: the first one is an entropy based node ranking where entropy ranking is used to determine node importance in a directed weighted influence graph. The second phase computes the minimum cut using a novel metric. To test the propose algorithm, experiments were performed in several popular data sets to evaluate performance and the seed quality over the influences.**

Keywords*—*Influnce maximization, entropy, minimum cut

I. Introduction

Large scale data processing analysis has brought us new challenges with respect to algorithmic support. There are several problems, like finding statistics in data streams, clustering of information in data streams, classification and learning [1], [2], [3], where the need of light and efficient algorithm is necessary. For example, in pattern recognition there are algorithms [4] where the *top-k* most important patterns are found instead of all of them.

Specifically in the new world of social networks, being able to identify sets of users able to influence the rest of the social network has become highly important and necessary. For example, Influence Maximization (IM) is a classic problem in social analysis, and it has become of great interest in data mining, machine learning and optimization. The original problem was formalized by Domingos and Richardson in [5], and it has the following structure: Given a social network (a directed graph) G, we need to find m nodes (call seeds) who influence the maximum set of network nodes. Because IM is NP-Hard problem [6], most of related works [6], [7], [8], [9] are focused in trying to approximate the optimal solution.

Quality results in the IM problem depends of finding the best seed nodes, and there are two kind of seeds nodes, static and extra seed nodes [7].

Static seed nodes are those which are well know. For example, if we represent a series of twitter accounts, which talk about a famous pop music singer, as a graph. It is possible to identify fan club leaders as static nodes, and the rest of the followers as nodes being influenced by the leaders. The second type of seed nodes are related to nodes who potentially can influence others to change their opinion. These type of nodes are called extra seed nodes. The focus of this work is to try to find these extra seed nodes to define a solution of the IM problem.

Even though, several methods [10] [6] [11] [8] [12] allow to process large amounts of information, they lack of effective techniques for seed discovery. Therefore, it is necessary to have more effective methods to obtain faster and more complex answers. For example, Liu et al. [7] uses a greedy technique based on min-cuts that allows finding the most influential nodes by computing min-cuts for every node to be a potential extra seed node. A drawback of this technique, when applied to a large amount of data, is its combinatorial nature.

IM has been recently re-stated as a Big Data problem for marketing purposing in social networks. For example, traditional greedy techniques based in Kempe et al. [6] were adopted to get faster results using parallelizing algorithms [11]. In another example, Li et al. [12] were facing large datasets problems, for which they decide to introduce the "conformity" term in the network. This term means *"the inclination of one node to be influenced by others"*.

In this work, the proposed algorithm needs to calculate these two indexes, influence and conformity. Then, using these indexes, it gets a series of graph partitions to determine local seed nodes because after all influence and conformity are local properties.

Then, the initial problem is to find the extra seed nodes. Then, nodes need to be classified in order to determine the extra seed nodes.

A popular method is the use of ranking algorithms [13] for this task. For example, there are traditional ranking algorithms using the degree of the nodes, as the populars PageRank [14] and HITS [15], for web page classification. Another

COLLABORATECOM 2013, October 20-23, Austin, United States

DOI 10.4108/icst.collaboratecom.2013.254119

method [16] uses ranking for labeled directed graph for *hyper-linked document* ranking. There are other methods, as Dirichlet PageRank [17], designed to avoid cyclical structures to identify e.g. spammer pages in web map.

To process large datasets for IM, we propose a two-phase algorithm and a bounded greedy algorithm in order to reduce computational time and to obtain good approximations to the IM problem. In the first phase, we rank the graph nodes using an entropy ranking [18] to find *"a hidden organizational structure to select interesting prominent members"*. This entropy ranking does not take in account the node's degree which makes this ranking more effective for finding extra seed nodes. The influence entropy is calculated by removing the node from the graph, and calculating its effect. The second phase uses a well know min-cut in a local fashion to speed up the computational time.

This idea of using the min-cut is inspired by the Ising model [19]. This model was originally used for ferromagnetic material behavior analysis in mechanical statistics [20]. In Social Network Analysis (SNA), the Ising model is used to detect community structures [21]. In this type of problems, the computation of the min-cut for calculating the Ising model leads to a ground state, which is a configuration with a state of low energy. In an equivalent way, in social networks, the ground state is a configuration with low number of conflicts. Finally, the proposed method utilizes a greedy pruning method taken from IM traditional methods [15] [8] to obtain the extra seed nodes. These extra seed nodes are choose from the most ranked nodes, which are also attractor seeds.

Finally, in order to prove the proposed algorithm, we select a series of datasets to build weighted graphs. These datasets are from twitter trending topics, hashtags, user mentions and retweets to represent the users influence. In addition, we include random graphs for baseline comparison.

The paper is organized as follows. In Section II, we formally explain the problem, and the required background to understand the proposed algorithm. In Section III, we describe the proposed algorithm. In Section IV, we explain how the algorithm was implemented. In Section V, a series of experiments are described, and the results are reported including comparison with other algorithms. In this section, we show the complexity time improvement due to the two phase algorithm and the bounded greedy algorithm. Finally, in the conclusion, we describe some ideas for future improvements.

II. Problem Description and Background

A. Graph Construction

Graphs are a very used technique for data representation, specially in SNA [22] [23]. In our case, the graphs are used to represent influences between social network members. For this, we define a Social Network as a graph $G = \langle V, E \rangle$, in which every user is a graph node (V), every link (E) between nodes represents an influence. Links are defined as $E \subseteq \{(i,j) \in V \times V : i \neq j \}$, and we define influence in one way or direction. In other words, if there is a influence from i to j, there is not necessarily the same influence from j to i. Actually, this is the natural way to form user's link (followers) in twitter. Matrix weight W represents degrees of influence between nodes, $W : \{n\} \times \{n\} \rightarrow \mathbb{N}$ where $n = |V|$, W_{ij} means link weight from node i to node j and $i \neq j$.

For twitter data representation, graphs are used to represent twitter followers where users are nodes and links are follow relationships. For example, if a user a follows user b there are a link from a to b. Follow relationships do not necessary reflects influence because, for example, user a can follow user b but if b never writes a tweet, there is no influence. For influence representation, we can build it using retweets and mention graphs. This means that if user a retweets user's b tweets, there is an influence from b to a. The same happens for tweet mentions. In this way, the weight matrix W_{ij} is built by summing the number of times user j mentions or retweets user i those ideas are based in McKelvey and Menczer work [22].

Influence graphs are constructed to represent specific users opinion. Given an opinion set Ψ, every opinion $\psi \in \Psi$ belongs to a node ψ_i (the node's i opinion). To fit the notation to the min-cut problem model, we say, if node i has a positive opinion, then, it is represented by $\psi_i \in N^+$. When it has negative opinion, then $\psi_i \in N^-$ as is used in [7]. For example, we could use an opinion set to indicate if a user knows or does not know a new product.

B. Entropy Based Ranking

The node ranking problem is wildly studied in web search, document classification, citation analysis, anomaly detection, etc., and many of the proposed solutions use graph representation [9], [16], [18].

The most basic algorithm is the Degree Ranking (DG), it was mentioned in Kempe et. al [6] as a simple strategy to find extra seed nodes. In this algorithm, given the degree value of each node, the nodes are sorted by degree values with a expected complexity of $O(n \log n)$, if quicksort is used. For example, if we apply this for twitter followers, the node with most in-degree is the most popular node because in-degree represents the number of followers. In-degree is represented by a function $indegree(i)$ and the out-degree is represented by function $outdegree(i)$. In this algorithm, high out-degree values could represent spammers.

Although, most of the ranking methods are based in metrics related with high node degree, this metric does not necessarily implies "importance". This can be seen in (Fig. 1), where there are nodes linking two communities, both with low $indegree(v) = 3$ and low $outdegree(v) = 1$.

This is the reason why we decide to adopt graph entropy [18]. For this, we are using the entropy definition in [24], which has the following expression (Eq. (1)).

$$H(G,P) = \min_{x \in SatableSet(G)} \sum_{v \in V}^{n} p_v log(p_v) \qquad (1)$$

This equation involves finding the Stable Set (SS). This set is the node collection L such that there is no edge between L members. It is well know, that finding SS is NP-complete problem. To overcome this problem Shetty et al. [18] proposed to use the equation (Eq. (2)) to compute graph entropy. In

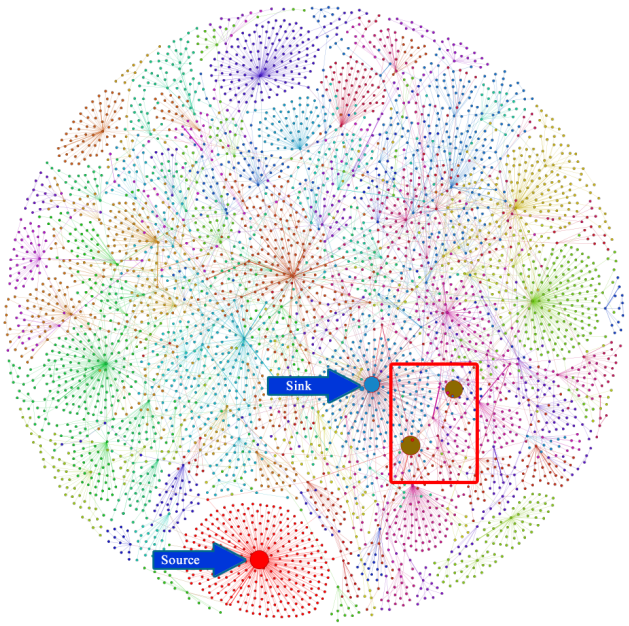

Figure 1. The graph represents twitter user interactions which are formed with hastag #israel. The brown nodes have low in-degrees and out-degrees, but are important because they link two big communities. Data was taken from Indiana University Truthy project [22].

addition, they also proposed an algorithm to determine node "effect". This metric evaluates the graph when a node is present and re-calculates the evaluation when the node has been removed. The node's effect is calculated as follows:

1) Compute node entropy $E(v)$.
2) Compute graph entropy without node to be evaluate $EN(v)$.
3) Calculate the effect with effect function (Eq. (3)).

$$H(G,P) = \sum_{v \in V}^{n} p_v log(1/p_v) \tag{2}$$

$$effect(v) = EN(v)/log(EN(v)/E(v)) \tag{3}$$

If these three steps are applied $\forall v \in V$, all possible effects are calculated. Then, the nodes can be sorted in descendant order using the ranking/effect generated by the entropy. Furthermore, entropy could be calculated at different levels (l). Here, level number is related to the direct and indirect influences between nodes. For example, $l = 1$ represents direct influence, i.e. node v_0 directly influences node v_1. For $l > 1$, we have indirect influences, i.e. for example, node v_0 influences node v_1, and this in turn influences v_2. This means that there is an indirect influence of level two from v_0 to v_2.

How the probability graph distribution P is calculated depends on analysis level. For $l = n$, we need to count every single path of size n for each node v. Probability of node v is equal to all paths of length l, which contain v, divided by the total number of graph paths of length l (ρ_l).

$$p_{v_0} = (\sum_{(v_0,v_1) \in E} \cdots \sum_{(v_{l-1},v_l) \in E} W_{l-1,l})/\rho_l \quad \forall v \in G, \quad l > 0 \tag{4}$$

A final note, it does not make sense to analyze l with large values because influence is lost as you increase level's value.

C. Min-cuts for Influence Maximization

Min-cuts are mainly used in social network analysis for graph reconstruction [21]. Suppose that we have a directed weighted graph G that represents social network's influences between users V by edges E, weight matrix W, and we know all members opinion about certain subject s. If all opinions are of two types, positive or negative, then we have two sets, N^+ for positive opinion nodes, and N^- for negative opinion nodes. In addition, N^+ and N^- are disjoint sets. Therefore, the min-cut can be defined as a function that:

$$c(N^+, N^-) = \min_{(i,j) \in N^+ \times N^-} \sum W_{ij} \tag{5}$$

Now, suppose we want to reconstruct the graph's node opinion set, and we know some specific opinions represented by subsets $\Psi^+ \subseteq N^+$ representing positive opinion, and $\Psi^- \subseteq N^-$ representing negative opinion. These subsets are called seed nodes, and they have the characteristic that they never change their opinion no matter what. We can use these seed sets and a min-cut algorithm [25] to obtain the original opinion sets, N'^+ and N'^-, which are known as source and sink sets. In addition, if we compare N^+ and N'^+ subsets obtained from the min-cut algorithm, from the original graph opinions, we can get an error e, defined as the number of nodes in N^+ which are misclassified. In other words, they are in N'^- after min-cut. This error e can be reduced, if we increase Ψ^+ size [21].

Then, we can use min-cuts for IM, with a dependable seed sets, to infer the other member opinions [7] in a very accurate manner. This final inference is determined depending on which side of the min-cut the node is. In order to use the min-cut algorithms in the IM problem, we need to specify, in addition to seed sets, the source s and sink t nodes.

The computation of the min-cut is accomplished by using a flow algorithm [25]. For example, if we use the Edmonds-Karp algorithm, the running time is in $O(VE^2)$. This is highly recommended for sparse graphs as the ones generated by social networks. Finally, we only need to determine the source and sink nodes. There are two strategies to determine these nodes:

- The first strategy takes the most representative node belonging to a community. This can be done if we have some knowledge about the graph. For example, in the karate club problem [21], if we do not choose as a source/sink the most representative nodes, and select them randomly, the results may give us a very different result. For this selection, we can use ranking algorithms to determine them, we can select one node with high values in the positive opinion side and one for the negative opinion side.

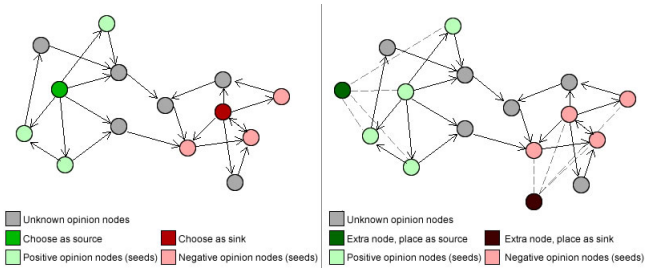

Figure 2. The first graph represents the source and sink election by high degree in their communities. The second graph represents the extra node placement for source and sink nodes.

- In the second strategy, we create extra source and sink nodes. These nodes do not represent any real graph member, it is only used to have a starting node and a ending node. This technique is used in flow problems (Fig. 2).

Once we have s and t, we need to link them with static seeds nodes, with $s \in N^+$ and $t \in N^-$. For this, we use the following directed weights:

- $W_{sv^+} = \infty, \forall v^+ \in \Psi^+$.
- $W_{v^-t} = \infty, \forall v^- \in \Psi^-$

Where set Ψ is defined as $\Psi = \Psi^+ \cup \Psi^-$. These rules are from multi-source/multi-sink flow problem. In addition, they are used to not allow N^+ nodes does not go to the N^- subset. References to all this notation can be seen in (Table I)

An example of an algorithm that uses these ideas of seed, source and sink nodes is the Greedy Placement (GP) algorithm [7]. This algorithm can find extra seed nodes by evaluation of node influence in running time of $O(mV^2E^2)$, where m is the number of seeds sought. This is the base algorithm for comparisons in the experiment (Section V).

III. Proposed Method

In order to analyze the IM problem for "Big Data," we need to have low computational complexity algorithms due to the size of the problems. Therefore, it is necessary to have fast algorithms that are suitable to obtain high quality results. However, most of the proposed approximation algorithms for the IM problem are based on greedy methods over combinatorial interpretations of the IM problem [7] [15] [12] [8]. This makes them vulnerable to runaway computation with exponential bounds. To address this, we propose a novel algorithm which has two phases. In the first phase, an analysis is carried out using node entropy ranking because is faster than use greedy approaches. These greedy approaches measure every single node influence, even when using min cut analysis. This implies to compute min-cut as many time as nodes have the graph. In the second phase, the min-cut algorithm, Edmonds-Karp, is used locally to obtain an approximation in the opinions of the graph.

TABLE I. Notation used in this paper

Notation	Description
G	Directed weight graph
V	Set of nodes (vertex) in G
E	Set of edges (links) in G
W	Node weights matrix
N^+	Set of nodes classified as positive
N^-	Set of nodes classified as negative
Ψ^+	Positive seed nodes
Ψ^-	Negative seed nodes
Ψ'^+	Positive extra seed nodes
Ψ'^-	Negative extra seed nodes
ψ_i	Opinion of the ith node
$outdegree(v)$	Function to count number of out edges
$indegree(v)$	Function to count number of in edges
$H(G, P)$	Entropy of G probability distribution P
s	Source node
t	Sink node
$c(N^+, N^-)$	Cut function
GP	Greedy placement algorithm
DG	Degree based algorithm
EMinG	Entropy based algorithm

A. Data Preparation

To apply the proposed algorithm, we need to decide in which part of the graph we are going to rank the nodes. It could be in the entire graph or only a set of nodes. This happens because local ranking of a node is not equal than the entire graph ranking. In our specific problem we need to know which m nodes in N^-, could attract more N^- nodes to N^+.

The local ranking will be done in the N^- side, in order to reduce the number of operations. To get N^-, the min-cut needs to be calculated. After the N^- subset is obtained, we rank the respective nodes.

B. Ranking Phase

In this phase, we want to identify "prominent" nodes. This characteristic must be related to an individual power of persuasion. Therefore, the ranking serves as data preparation. The top most m ranked nodes can be the extra seed nodes (Ψ'^+). Entropy ranking is only calculated in N^- subgraph to reduce the number of operation. Graph entropy is calculated by the formula 2, and node effect with 3. The algorithm for this phase is in (Algorithm 1).

Complexity of ranking is directly related to the number of nodes in the graph. The computation of probability is made in $O(N^-)^l$, and we only are going to compute influence level for $l = 1$ and $l = 2$. To compute effect is necessary to compute graph entropy, which is caluculated with complexity $O((N^-)^2)$. Finally, sorting of the nodes is done in $O(N^- logN^-)$ complexity, and addition of nodes is done in $O(N^-)$ complexity.

During experimentation, it was discovered that a good level of influence in this phase was $l = 1$ and $l = 2$. An increase of the level values may result in loss of time without improving the quality of the extra seeds selected.

C. Cut Phase

The min-cut splits the graph in two sets N^+ and N^-. Because Min-cut is computed locally, we need to establish source and sink nodes. Locally, this refers to obtain a local min-cut between these nodes and not the global min-cut. Then, every node $i \in N^+$ is classified as a positive, and every node $j \in N^-$ is classified as a negative. For our model, the m nodes

Algorithm 1 Ranking algorithm

Require: Directed Weight Graph G.

1: $\Psi'^{+} = \phi$
2: $N^- = mincut(G)$
3: *Combine all N^+ nodes as a supernode*
4: $entropy_{N^-} = H(N^-, P_{N^-})$
5: **for all** v_i^- in N^- **do**
6: $G^- = N^- \backslash v_i^-$
7: $entropy_{G^-} = H(G^-, P_{G^-})$
8: $effect[i] = entropy_{G^-}/log(entropy_{G^-}/E(v_i^-))$
9: **end for**
10: Sort($effect$)
11: **for** $j = 1$ to $\mid N^- \mid$ **do**
12: **if** $v_j^- \notin \Psi^+$ **then**
13: $\Psi'^{+} = \Psi'^{+} \cup \{v_j^-\}$
14: **end if**
15: **end for**
16: **return** Ψ'^{+}

Algorithm 2 Cut algorithm EMin

Require: Extra seed set Ψ'^{+}.

1: $j = 1$
2: **for** $i = 1$ to m **do**
3: $\psi^+ = \Psi'^{+}[j]$
4: **while** $\psi^+ \in N^+$ **do**
5: $j = j + 1$
6: $\psi^+ = \Psi'^{+}[j]$
7: **end while**
8: $W_{s\psi^-} = \infty$
9: $N+ = mincut(G)$
10: *Combine all N^+ nodes as a supernode*
11: **end for**
12: **return**

with high entropy ranking are chosen. Then, the following algorithm (algorithm 2) is applied.

As we see in the cut algorithm, in line 4 there is a validation that guarantees a special case, when a node with high entropy value was attracted previously to N^+ by other node. In (Fig. 3) we illustrates the situation. The algorithm deals with the situation in the following way: The attracted nodes are not added to Ψ'^{+} because all the possible nodes, which could be attracted by this node, are also includes when they are members of N^+ side.

In the ranking and cut algorithms in line 3 and 10 respectively, we note a special operation which combines all nodes in the N^+ side in a single node. Supernode is a technique in graph theory [25] to reduce the number of operations. In this technique, nodes in N^+ related by an edge to N^- nodes are called cut nodes. The technique takes all edges between positive cut nodes to negative cut nodes, and fused them in one edge which is the sum of all weights of cut nodes. This is also done for negative cut nodes linked to positive cut nodes. All weights between N^+ nodes are ignored because it does not make sense to analyze N^+ interactions, and they are also in the side that we need to increment. Finally, the supernode always will be the source node.

The algorithm time complexity consists of Min-cut complexity and the entropy effects calculations. It needs to compute Min-cut over using the seeds generated by the entropy ranking. Then, the cut algorithm complexity is $O(m(VE^2))$.

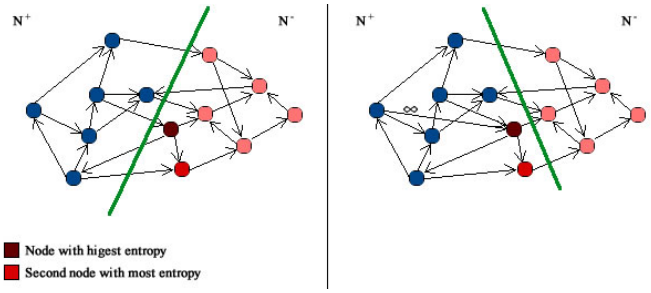

Figure 3. Special case representation. In the left graph, there are two top most ranked nodes in N^-. In the right, after adding most ranked node to the N^+ side. Also the second most ranked node is attracted because it is not necessary to take this node as extra seed.

D. Adding Greedy to Entropy

As we mention earlier, greedy techniques may lead us to a spend more time to obtain results, in addition to have low quality results. Still, we want to exploit the greedy methods main characteristic, the reduction of the combinatorial part of the IM as an optimization problem. Thus, we can prune the number of combinations to compute high quality seeds. We can take $m+m'$ top ranked nodes and test which attract more nodes, where m' is an extra search space to find best extra seeds. The value for m' could be determined by experimental tests. In addition, the traditional greedy algorithm is defined when $m+m' = |N^-|$. In (Eq. 6), we define EMinG algorithm as an optimization equation, where $\Psi'^{+}_{m+m'}$ are top $m+m'$ ranked nodes. The final algorithm with the new greedy part is defined in (algorithm 3).

$$\psi_j^+ = \underset{i \in N^- \backslash \Psi^-}{\arg\max}\ mincut(N^-, \Psi'^{+}_{m+m'}) \tag{6}$$

Algorithm 3 Cut algorithm with Greedy EMinG

Require: Extra seed set Ψ'^{+}.

1: $j = 1$
2: **for** $i = 1$ to m **do**
3: **for** $j = 1$ to $m + m'$ **do**
4: $a = mincut(N^+ \bigcup \Psi'^{+}[j])$
5: **if** $a > max$ **then**
6: $max = a$
7: $\psi^+ = \Psi'^{+}[j]$
8: **end if**
9: **end for**
10: $W_{s\psi^-} = \infty$
11: *Combine all N^+ nodes as a supernode*
12: **end for**
13: **return**

The complexity of EMinG is directly related with the number of seeds to be found. For example, Min-cut is computed $m(m+m')$ times, then we get a $O(m(m+m')VE^2)$ complexity for the algorithm. For IM, m is very small compared with the number of nodes in the graph. For example with $m = 5$,

as we will see in section V, we can attract 1033 nodes, in a 7500 nodes graph.

Finally, the proposed algorithm is a combination of min-cut and ranking algorithms with greedy technique over nodes which can potentially be influential, with this we can reduce time over greedy methods and get more quality extra seeds.

IV. Experiment Design

The proposed algorithm was applied over twitter data sets, which are constructed from certain subjects, and also it was applied over random build graphs. We run all the algorithms with the same seed nodes discovered in the entropy ranking phase Ψ^+ and Ψ^-. In addition to the same source s node, sink t node and those which were chosen from specific studied graphs.

A. Algorithms for Comparison

The proposed algorithm will be compared with two specific algorithms, the DG algorithm and the GP algorithm. DG was chosen to be compared with the algorithm in order to get an extra seed set Ψ'^+. As the experiments will show, the proposed algorithm is able to obtain more quality results (better extra seed nodes) when compared with the DG algorithm. Although, with the GP algorithm, it is possible to obtain higher quality results, the proposed algorithm is able to obtain faster results than GP when the size of the data sets increases. We also use random seed that were chosen heuristically together with a single discount algorithm. The single discount algorithm heuristic consist in a simple degree discount heuristic where each neighbor of a newly selected seed discounts its degree by one. This is used in cascade models.

B. Twitter Graphs

We use twitter data sets from *Truthy* project [22]. Those graphs, from politics, social movements and news, are used to try to understand how communication spreads on twitter. This project recollects tweets for the last nine months and packed them in data sets available to the public.

Using graph's software visualization such Gephi [26], it is possible to recognize some of the nodes' preferences at the graphs. Then, it is possible to use them as a seed nodes (positive and negative). We also recollected data from twitter for marketing datasets where the users are related with a certain subject. The subject could be hashtag or keyword, and the user relations are found by mentions or retweets.

C. Random Graphs

Although some random graphs generators could not represents any real problem, we are going to use a randomly generated datasets to compare the testing algorithms against the proposed algorithm in order to see how they behave as the graph increses in size, degree at the nodes, etc . For this, random graphs are generated with well know algorithm Barabasi-Albert [27] generator. The importance of using random graphs is because these can be configured to modify the average degree of the nodes, sparsity and of course we can handle the amount of nodes to be used. Modifying these parameters, we can create graphs to represent real possible behaviors.

V. Experiments and Results

The algorithms were implemented on Intel Core I7 3.4 GHz with 8Gb RAM, and this scenario was the same for all algorithm comparisons.

A. Attracted Nodes

As we mention earlier, most of the traditional IM algorithms are focused in maximizing the amount of nodes which can change their opinion. Under this assumptions, it is clear GP is going to have the best results. But, as we saw in the last section, it is possible to obtain most of the influential nodes by reducing the number of operations with EMinG.

In the #israel dataset (Fig. 4), GP and EMinG got the same results because high quality nodes are also included in top $m + m'$ ranked nodes. In this case $m' = \mid V \mid / 7$ was chosen for testing on all experiments. In #immigration, we got almost the same results with both algorithms. The variation was of at most 15 extra seeds. For example, node N4620 is not of high entropy, but it is influential. Single discount had acceptable results. For the first and second experiments, we got 1910 and 2005 nodes respectively.

In the random graph, while looking for 30 extra seed nodes with GP, we were able to attract 1171 nodes, with EMinG, we got 1066. Using, the single discount algorithm, we got 794, with DG we got 549, and with Random we got 240. In the fourth experiment, DG and Random got the worst results. Furthermore, in the randomly generated graph, it was more notorious the not so good results for DG and random seed placement For random seeed placemente, seeds were selected at random and the second phase of EMinG was used to find attrected nodes and extra seeds. The main difference between the first and third experiments is the out-degree node average. In #israel is 1.21, in #immigration is 1.14, in random graph experiment is 2.48 and #syria 1.18.

Although, the first, second and third expriments (Fig. 4) were designed to show the performance of the algorithms, the fourth experiment was done over a Big Data set (1,000,000 nodes) in order to show the scalability of the algorithms. Thus, we noticed that with GP an EMinG we got 671310 and 650010 extra attracted nodes respectively. It is more, with $m > 30$, DG and Random got small increases. Finally, the single discount algorithm had incremental results, but they were much worse when compared with ones by GP and EMinG.

B. Running Time

The final benchmark was done over the running time (Fig. 5). It was possible to see that DG and random seed placement were the fastest algorithms at the four datasets with respect to EMinG, but it was only with an advantage of seconds. For large amount of data EMinG was faster than GP. The interesting fact was that the Single discount algorithm had almost the same results in time when compared with EMinG.

For example, at the $\#israel$ graph in order to find 5 extra seeds there were 300 seconds of difference between both algorithms, DG and EMinG. Still, the nodes' quality, for DG, were not as good as in EMinG. We had a similar result for for the random graph: With DG, we obtained 858 attracted nodes, with EMinG 1033 and 245 for random placement. When we

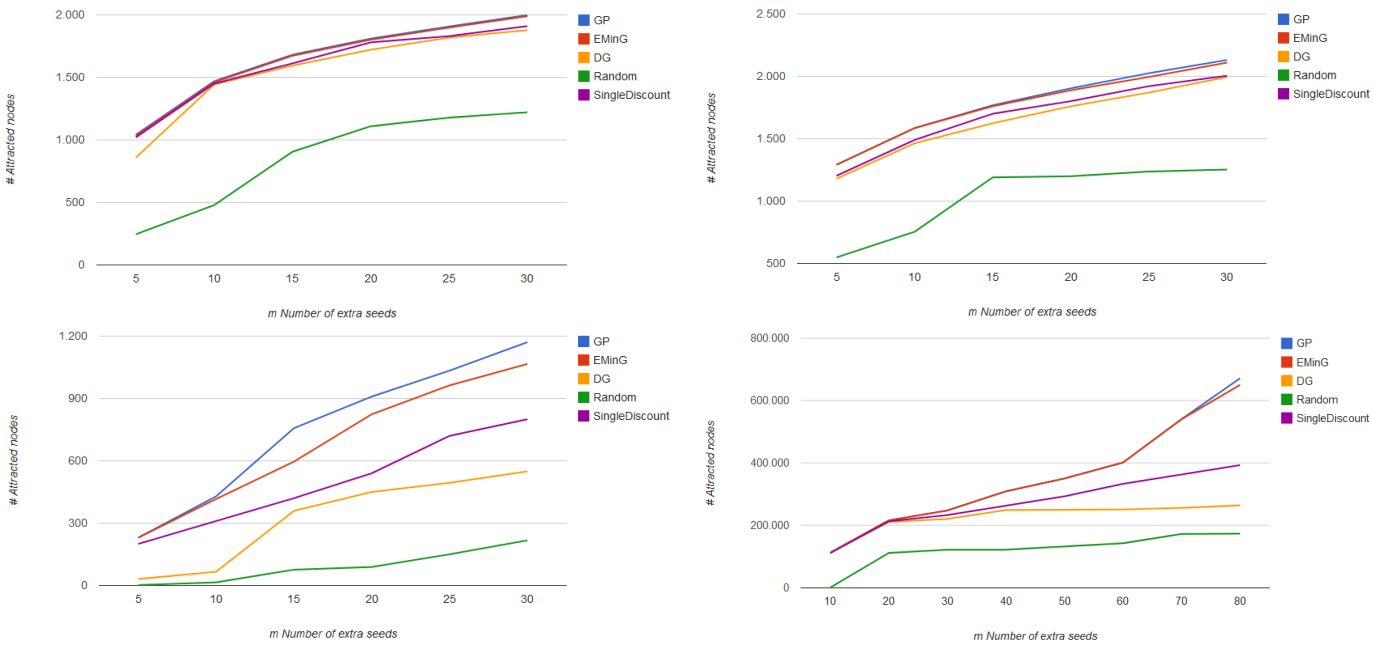

Figure 4. Number of attracted nodes over #israel (4,000 nodes), #immigration (6,307 nodes), Barabasi-Albert (7,500 nodes) and #syria (1,000,000) graphs.

increased the number of seeds to 30 and used DG we got 1878 attracted nodes. Instead with EMinG while looking for 25 extra seeds, the algorithm got 1989 attracted nodes. Finally, with random seed only placemente, there were only 1221 attracted nodes.

Now, the GP's results were slower than EMinG. This is because with GP we need to evaluate every single node's influence. Then, while looking for a small number of extra seeds, it can be seen that it has lower running time. But with 10 or more extra seeds to find, GP's time complexity increases faster than any other algorithm. For example, to find 30 seeds, in the Barabasi-Albert random graph, GP runned for 11171.76 seconds against EMinG that runned only for 4814.04 seconds, in addition that GP got only 9 more attracted nodes. For the same experiment the single discount algorithm runned for 4579.09 seconds, which is less than EMinG, but with 111 attracted nodes less.

For #syria graph, EMinG attracted each node in 0.83 seconds versus GP that attrected each node in 2.31 seconds. Of course DG and random seed placement are faster than EMinG, but as we saw in the Big Data Graph, if we look for more than 30 extra seeds there are no significant changes. As we can see EMinG's computational time is related with the graph size because entropy ranking is calculated over all negative opinion nodes. Finally, The additional time in EMinG is for the greedy choice on top of the $m + m'$ nodes.

VI. Conclusion and Future Work

The influence maximization problem will continue to be studied from different points of view because the increasing interest in studying the interaction among members of social networks. This is increasingly having a heavy impact in application development and gaming industry. The proposed algorithm is based on simple algorithms, and other simple techniques in order to be able to implement an highly scalable application for Big Data. Of course, the algorithm is still only trying to get an approximation to the optimal solution. However, the experiments showed the accuracy and high quality of the results of the proposed algorithm. Additionally, one the characteristics of the algorithm is the use of local information for increasing speed when dealing with large data sets.

In addition, incorporating new ranking algorithms is quite simple because this phase was designed independently in the second phase of the proposed algorithm. Also, heuristics can be incorporated, such as the ones applied in greedy algorithms to reduce the number of operations without loss of quality, or the use of heuristics to choose the seed nodes. In a future implementation, many of the operations could be parallelized. In addition, we are working in developing new methods to analyze IM and mining large data streams to get faster results.

Finally for future applications, we want to apply the proposed algorithm in no-social datasets to measure its effectiveness. For example, in biological, sensor, communication or networking datasets.

Acknowledgment

We appreciate the financial support given by CONACYT. We want to thank computer science laboratory members for the tips, reviews and recommendations over experiments and algorithms that were proposed. Finally, we want to thank people who are developing new algorithms, new methods and new software which are the basis of this work.

References

[1] M. Shindler, A. Wong, and A. W. Meyerson, "Fast and accurate k-means for large datasets," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., 2011, pp. 2375–2383.

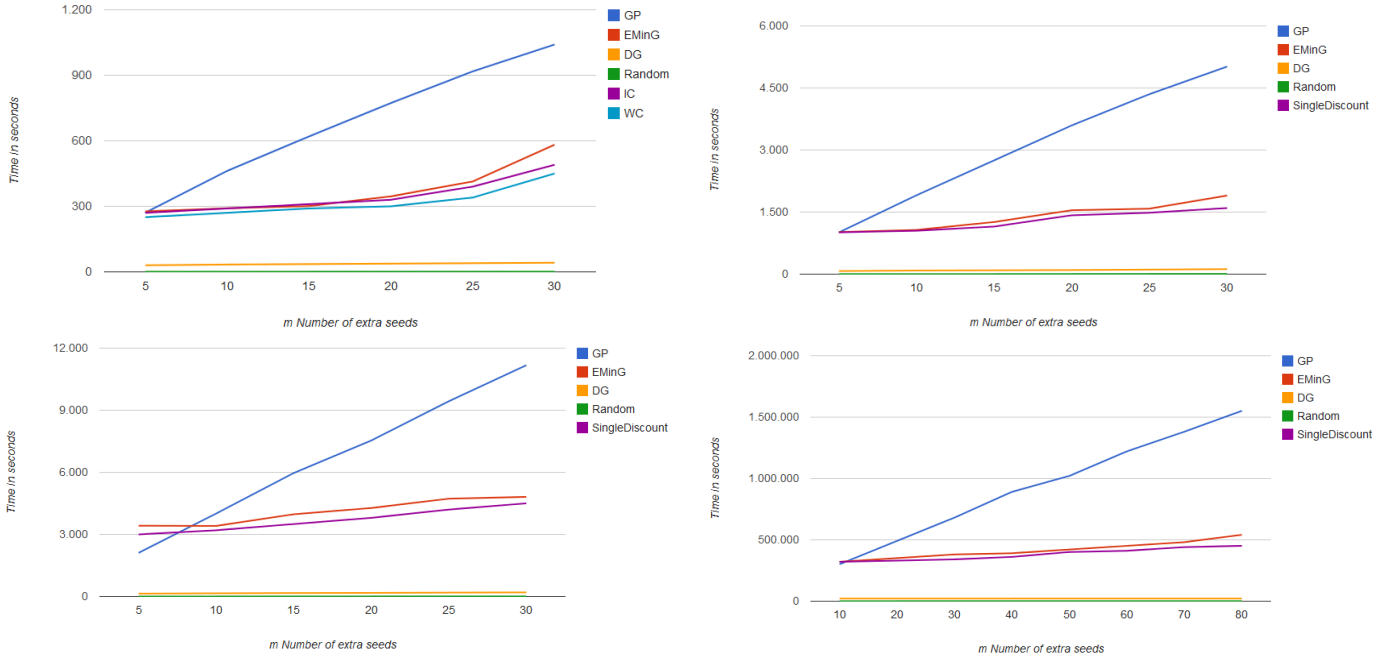

Figure 5. Running time over #israel (4,000 nodes), #immigration (6,307 nodes), Barabasi-Albert (7,500 nodes) and #syria (1,000,000) graphs.

[2] Z. Li, H. Ning, L. Cao, T. Zhang, Y. Gong, and T. S. Huang, "Learning to search efficiently in high dimensions," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., 2011, pp. 1710–1718.

[3] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, May 2011.

[4] A. Metwally, D. Agrawal, and A. E. Abbadi, "An integrated efficient solution for computing frequent and top-k elements in data streams," *ACM Trans. Database Syst.*, vol. 31, no. 3, pp. 1095–1133, Sep. 2006.

[5] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '01. New York, NY, USA: ACM, 2001, pp. 57–66.

[6] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 137–146.

[7] S. Liu, L. Ying, and S. Shakkottai, "Influence maximization in social networks: An ising-model-based approach," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, 2010, pp. 570–576.

[8] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 199–208.

[9] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincn, X. Sun, Y. Wang, W. Wei, and Y. Yuan, "Influence maximization in social networks when negative opinions may emerge and propagate," in *Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, April 28-30, 2011, Mesa, Arizona, USA*, 2011, pp. 379–390.

[10] W. Chen, W. Lu, and N. Zhang, "Time-critical influence maximization in social networks with time-delayed diffusion process," *CoRR*, vol. abs/1204.3074, 2012.

[11] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 807–816.

[12] H. Li, S. S. Bhowmick, and A. Sun, "Cinema: conformity-aware greedy algorithm for influence maximization in online social networks," in *Proceedings of the 16th International Conference on Extending Database Technology*, ser. EDBT '13. ACM, 2013, pp. 323–334.

[13] K. Ohara, K. Saito, M. Kimura, and H. Motoda, "Effect of in/out-degree correlation on influence degree of two contrasting information diffusion models," in *Proceedings of the 5th international conference on Social Computing, Behavioral-Cultural Modeling and Prediction*, ser. SBP'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 131–138.

[14] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Technical Report 1999-66, November 1999, previous number = SIDL-WP-1999-0120.

[15] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, Sep. 1999.

[16] K. P. Chitrapura and S. R. Kashyap, "Node ranking in labeled directed graphs," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, ser. CIKM '04. ACM, 2004, pp. 597–606.

[17] F. Chung, A. Tsiatas, and W. Xu, "Dirichlet pagerank and trust-based ranking algorithms," in *Proceedings of the 8th international conference on Algorithms and models for the web graph*, ser. WAW'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 103–114.

[18] J. Shetty and J. Adibi, "Discovering important nodes through graph entropy the case of enron email database," in *Proceedings of the 3rd international workshop on Link discovery*, ser. LinkKDD '05. ACM, 2005, pp. 74–81.

[19] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact Maximum A Posteriori Estimation for Binary Images," 1989.

[20] L. Onsager, "Crystal statistics. i. a two-dimensional model with an order-disorder transition," *Phys. Rev.*, vol. 65, pp. 117–149, Feb 1944.

[21] S.-W. Son, H. Jeong, and J. D. Noh, "Random field ising model and community structure in complex networks," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 50, no. 3, pp. 431–437, 2006.

[22] K. R. McKelvey and F. Menczer, "Truthy: enabling the study of online social networks." in *CSCW Companion*, A. Bruckman, S. Counts, C. Lampe, and L. G. Terveen, Eds. ACM, 2013, pp. 23–26.

[23] A. D'Andrea, F. Ferri, and P. Grifoni, "An Overview of Methods for

Virtual Social Networks Analysis," in *Computational Social Network Analysis*, ser. Computer Communications and Networks, A. Abraham, A.-E. Hassanien, and V. Snáel, Eds. London: Springer London, 2010, ch. 1, pp. 3–25.

[24] J. Korner, "Fredman-kolmo's bounds and information theory," *SIAM J. Algebraic Discrete Methods*, vol. 7, no. 4, pp. 560–570, Oct. 1986.

[25] T. Leighton and S. Rao, "Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms," *J. ACM*, vol. 46, no. 6, pp. 787–832, Nov. 1999.

[26] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," 2009.

[27] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.