Rank Refinement: An Algorithmic Framework with Applications to Diversity Aware Influence Maximization

Prateek Yadav
CSE, IIT Madras
prateekyadav.iisc@gmail.com

Arun Rajkumar
CSE, IIT Madras
arunr@cse.iitm.ac.in

ABSTRACT

Several real-world problems including network influence maximization, rank aggregation, etc. can be posed as problems that output a *good* ranking of a set of items. We introduce a unifying algorithmic framework based on a novel notion called *rank refinement of monotonic set functions* to tackle such problems. We prove that under very mild monotonicity assumptions the proposed algorithm converges to a stable ranking. We show that IMRank, a highly scalable influence maximization algorithm can be derived as a special case of our framework. By careful choice of the monotonic set functions, we derive novel generalizations of IMRank that balances the influence and diversity of the top-ranked nodes without compromising on scalability. We provide extensive experimental analysis on both synthetic data-sets based on stochastic block models and large scale real-world data-sets to demonstrate the efficacy of the proposed framework.

1 INTRODUCTION

We consider settings where the input is typically modelled as a graph on an underlying set of nodes and the output is a permutation or ranking of the nodes which has some desirable property. Several natural problems including rank aggregation, seed selection in influence maximization, etc., can be modelled in this fashion. As a concrete application, we consider the influence maximization seed selection problem. Here, given a input graph and pairwise probabilities of influence among nodes, the goal is to output a set of top K *influential nodes* which on activation results in the largest influence spread. Typical algorithms for this problem try to exploit the structure of the network and the nature of the probability of influence to come up with a set of good nodes to activate. However, most of these algorithms either run into scalability issues when the network is large or are too simple as heuristics (top degree nodes) which may be highly sub-optimal. Another important problem that arises in this application is the notion of *diversity in seed selection*. For instance, if a product company wishes to choose top 10 social media influencers, it is only fair to choose a set of people who are diverse and represent all sections of the population. However, simply trying to maximize diversity might lead to severe loss of influence. Thus, a natural question to ask in this scenario is *how can one develop efficient, scalable algorithms for the seed selection problem which respect the underlying diversity in the population without compromising much on influence?*

In this work, we present a general algorithmic framework which we call as *Rank Refinement* and show how it can be used to tackle the above problem. While our framework can potentially be useful in general settings which involve ranking on graphs, we will restrict our focus to the application of influence maximization in this work. For this application, we will show that DAIM - Diversity Aware Influence Maximization, a concrete algorithm that we propose to tackle this problem emerges as a natural instantiation of our general rank refinement framework.

By rank refinement, we refer to an iterative scheme which starts with a reasonable guess for a ranking of interest and refines it over iterations. We show that under a very general monotonicity assumption of set functions, our algorithm converges to a stable ranking. A specific instantiation of the rank refinement framework recovers a well known scalable ranking based seed selection algorithm known as IMRank [10]. Viewing IMRank using the rank refinement lens proves to be extremely beneficial in establishing several interesting properties of the algorithm which were not explicitly known before. For instance, we show that the refinement procedure of IMRank is a linear map of the all ones vector where the linear map can be viewed as the transpose of a Markov chain transition probability matrix.

Conference'20, June 2020, Portland, OR, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

More importantly, our analysis of IMRank using the rank refinement lens reveals a fundamental trade-off between what we term as *influence resistance* and *influence capacity* of nodes in a network. We show that the IMRank algorithm tries to balance this trade-off in a specific fashion which may not be always optimal. Developing this further, we propose a novel algorithm known as DAIM - Diversity Aware Influence Maximization which in practice balances this trade-off in a much better fashion without compromising on the influence spread.

We test our proposed algorithm on several real world data-sets and synthetic data-sets and analyze the sensitivity for our algorithm to several interesting problem parameters. Overall, our experiments reveal that one can expect to achieve significant gain in diversity of seed selection without much loss in influence (or even gain in influence in certain cases) in practice.

1.1 Summary of Contributions

(1) We propose a general iterative rank refinement scheme based on monotonicity of set functions and prove the convergence of our algorithm to a stable ranking
(2) We demystify the workings of a popular scalable ranking based algorithm for seed selection known as IMRank which we show to be a special case of the rank refinement framework.
(3) We propose DAIM- Diversity Aware Influence Maximization, an algorithm which balances both influence resistance and influence capacity of nodes and hence leads to better diversity
(4) We conduct extensive experiments on real world and synthetic data-sets to understand why and how the proposed algorithm DAIM is a better alternative to IMRank for applications which seek diversity.

2 RELATED WORK

The task of influence maximisation was introduced by [18], where the aim is to select a set of seed nodes such that given a propagation model the spread is maximised. [18] proposed the Independent Cascade (IC) and linear threshold (LT) models for the spread of information over networks. There are three broad categories of algorithms which are used for the task the influence maximisation. The first class of algorithms perform Monte Carlo Simulations for simulating the Influence spread. The first method in this class was proposed by [18] where the problem of Influence maximisation was proposed and the greedy algorithm which in each iteration selects the node which provides the maximum marginal gain was analyzed. [13, 20] exploited the sub-modularity property to develop efficient algorithms which are much faster than the greedy algorithm. These methods come with theoretical guarantees on solution qualities but suffer in compute time.

The second class of algorithms are based on sampling techniques. [3, 28, 29] perform sampling to estimate the influence of node and use the estimates to perform seed selection. They construct reverse reachable set from the sampled nodes and then a greedy max covering algorithm selects the seed nodes. [11, 24] generate several snapshots of the graph and estimate the influence by averaging over these snapshots.

The third class of algorithms perform some sort of approximate scoring of nodes which are then used for the seed selection. The key idea in [7–9, 12, 13, 17] is that the influence of a node on other nodes is a function of the number of paths between these two nodes. These models are efficient and scalable but there is no guarantee on the quality of the solution obtained. We on the other hand consider a ranking based approach to the problem.

There have been many recent algorithms which look at various aspects of the influence maximization problem. [6, 15] propose robust algorithms for influence maximisation, [19] propose an online algorithm for influence maximisation, [23] propose a scalable algorithm which can work on distributed platforms like MapReduce, [25] use graph neural networks for learning latent social representation and use them for influence prediction.

The work that is most related to ours is that of [10]. Here, the authors propose a ranking based method for approximating influence scores for each node in an iterative manner called IMRank. Given any initial ranking they employ a Last-to-First allocation (LFA) strategy (See Section on Preliminaries for further details) to approximate the marginal influence of the nodes in the network. The scores are then used to refine the ranking. They provide a proof of convergence for their algorithm and empirically show that the solution obtained by IMRank as very close to the greedy solution and it is also scalable to very large graphs. Our proposed Rank Refinement framework is significantly general and incorporates the IMRank algorithm as a special case. We show that our methods can provide a way to trade of between maximizing influence or propagating influence to diverse communities.

Community based approaches: [14, 26] proposed a community based approach in which the influence propagation process has two steps. In the first part, the seed nodes are divided among different communities and in the second part the influence propagates within communities which are independent of each other. In contrast, the algorithms we propose do not have this two step structure. More recently [5] incorporated the topology of the graph to model user diversity. They define two objective functions, namely capital and diversity and then perform targeted Influence maximisation. They provide approaches based on local and global diversity

to exploit the structural information from the diffusion subgraph given a target node. Our setting is different from theirs and do not have a target node which we wish to influence. [4] exploited the community structure to select seed nodes in communities and computed their local spreads in order to minimize the number of seed nods required for influence maximisation. In our setting, the community structure is not assumed by the algorithm. [22] proposed an algorithm which constructs a tree based index that computes user's community based influences and the employ maximum influence arborescence (MIA) model to approximate the influence spread. [27] partition the network and then select most influential nodes from each partition based on their local influences. [16] builds a latent variable model which captures community level topic interest and then they infer community-to-community influence strength based on the topic interest. Finally, they propose a heuristic based algorithm to mine influential nodes in the community. None of the above algorithms employ a ranking based approach to selecting diverse nodes while our focus is on developing ranking based procedures.

3 PRELIMINARIES

Let $[n] = \{1, \ldots, n\}$ and let $[a, b] = \{a, a+1, \ldots, b\}$ for some integers $a < b$. let $G = ([n], P)$ denote a directed graph on n nodes where $P_{ij} \in [0, 1]$ denotes the probability of node i influencing node j. We assume that there are no self edges i.e., $P_{ii} = 0 \ \forall i$. Let $\mathbb{S}_n$ denote the set of permutations on $[n]$ where if $\sigma \in \mathbb{S}_n, \sigma(i)$ denotes the rank of node i according to σ. We will use the terms permutation and ranking interchangeably. For a (score) vector $\mathbf{s} \in \mathbb{R}^n$, we say that $\sigma = \textit{argsort}(\mathbf{s})$ if $\sigma(i) < \sigma(j) \implies s_i >= s_j \ \forall i, j$. We say that $\sigma = \textit{argstabsort}(\mathbf{s})$ if $\sigma = \textit{argsort}(\mathbf{s})$ and $\sigma(i) < \sigma(j)$ whenever $s_i = s_j$ for some $i < j$ i.e., σ is a *stable* sorting of $\mathbf{s}$.

Given a positive integer $k \in [n]$, the influence maximization problem is to choose a set $S \subseteq [n]$ s.t. $|S| = k$ for which the expected size of the *influenced* nodes is maximized according to certain underlying influence spread model. One of the most popular and well studied influence spread model is the Independent Cascade (IC) model introduced in [18]. In this model an initial set of k seed nodes is activated. In the first step, every seed nodes get one chance to activate each of it's neighbours where the neighbour j of a seed node i gets activated with probability P_{ij}. In the next step, the nodes that were activated in the previous step get one chance to activate each of their inactive neighbours. The process continues until there are no more active nodes. It is well known that a greedy algorithm for the IC model gives a $(1 - \frac{1}{e})$ approximation to the expected spread.

3.1 The IMRank Algorithm

The greedy algorithm, while attractive in terms of it's approximation guarantee, is not practical for large scale graphs owing to the large number of Monte Carlo simulations needed to find the top k nodes [18]. Several approaches have been proposed to remedy this problem [3, 7, 11, 13, 20, 23, 29]. The line of work which we will be interested in is the ranking approach to select the top K seed nodes. Surprisingly the work in this area has been quite sparse and the current state of the art ranking based approach is the IMRank algorithm of [10]. Given a graph G, the IMRank algorithm starts by selecting a *good* initial ranking $\sigma_0 \in \mathbb{S}_n$ of the nodes. The algorithms proceeds in iterations and at every round, a *last-to-first* allocation (LFA) strategy described below is followed to update a ranking σ_t to a new ranking σ_{t+1}. The algorithm terminates when $\sigma_{t+1} = \sigma_t$ for some t.

Algorithm 1 IMRank Algorithm [10]

Input: Graph $G = ([n], P)$ and an initial ranking σ_0
Initialize $\mathbf{s}_i = 1 \ \forall i$
repeat
  for $i = n$ to 1 **do**
    for $j = 1$ to i **do**
      $\mathbf{s}(\sigma(j)) = \mathbf{s}(\sigma(j)) + P_{\sigma(j)\sigma(i)} * \mathbf{s}(\sigma(i))$
      $\mathbf{s}(\sigma(i)) = \mathbf{s}(\sigma(i)) * (1 - P_{\sigma(j)\sigma(i)})$
    end for
  end for
  $\sigma_{t+1} = \textit{argsort}(\sigma_t)$
until $\sigma_{t+1} \neq \sigma_t$
Output: σ_t

The LFA strategy for IMRank is as follows: An initial score of 1 is assigned to each node in $[n]$. The score of a node i $\mathbf{s}_i^{\sigma_t}$ w.r.t $\sigma \in \mathbb{S}_n$ is computed in a last to first manner i.e., from $i = n$ to 1 as follows. In the turn for computing score of node i, every node j that is ranked above i in σ (i.e., $\sigma_t(j) < \sigma_t(i)$) *takes away* P_{ji} fraction of node i's current score and adds it to it's own score, thus reducing the score of node i by a fraction of $(1 - P_{ji})$. The nodes follow the order according to σ_t to take away from i where the higher ranked nodes take away from i before the lower ranked nodes. Note that nodes ranked lesser than i according to σ_t do not get a chance to take away from i. Once the score for every node has been computed according to the LFA strategy described above, the updated permutation is computed as $\sigma_{t+1} = \textit{argsort}(\sigma_t)$.

[10] use a potential function based argument to show that the IMRank algorithm converges for the LFA strategy described above. The algorithm is easily scalable for large scale graphs and converges in a very few iterations in practice. As noted in [2], there do exist algorithms which trade off scalability to the size of the spread.

Algorithm 2 RR-Generic: Rank Refinement of Monotonic Set Functions

Input: $F = \{f_1, \ldots, f_n\}$ where each $f_i : 2^{[n]} \rightarrow \mathbb{R}_+$ is a monotonic set function. A initial ranking $\sigma_0 \in \mathbb{S}_n$
repeat
  $\mathbf{s}_F^{\sigma_t}(i) = f_i(B^{\sigma_t}(i))\ \forall i$
  $\sigma_{t+1} = argstabsort(\mathbf{s}_F^{\sigma_t})$
until $\sigma_{t+1} \neq \sigma_t$
Output: $\sigma_t \in \mathbb{S}_n$

4 RANK REFINEMENT OF MONOTONIC FUNCTIONS

In this section, we present an abstract and generic iterative procedure called *Rank Refinement-Generic* (Algorithm 2) to obtains rankings from a set of set functions. Concrete realizations of this procedure will lead us to interesting algorithms for the influence maximization problem with far reaching generalizations of the IMRank algorithm described earlier. We begin with the following definitions

Definition 4.1. Let $\sigma \in \mathbb{S}_n$. Define the *before* set of i in σ as $B^\sigma(i) = \{j : \sigma(j) < \sigma(i)\}$

Definition 4.2. Let $f : 2^{[n]} \rightarrow \mathbb{R}_+$. We will call f a *monotonic set function* if for any $E \subseteq E^{'} \subseteq [n]$, $f(E) \geq f(E^{'})$

Definition 4.3. Let $F = \{f_1, \ldots, f_n\}$ be a set of n monotonic set functions defined on $2^{[n]}$. A score function $\mathbf{s}_F^\sigma : [n] \rightarrow \mathbb{R}_+$ is said to be *induced by F with respect to σ* if

$$\mathbf{s}_F^\sigma(i) = f_i(B^\sigma(i))\ \forall i$$

With the above definitions in hand, we describe a iterative algorithm, which we call Rank Refinement- Generic, to refine a ranking at every iteration. The algorithm is shown in Algorithm 2. Our main result of this section is to show that the iterative procedure described in Algorithm 2 terminates after a finite number of steps.

Theorem 4.4. *Let $F = \{f_1, \ldots, f_n\}$ be a set of monotonic set functions and let $\mathbf{s}_F^{\sigma_0}$ be the score function induced by F w.r.t σ_0. Then the sequence $\{\sigma_0, \sigma_1, \ldots\}$ computed by the RR-Generic algorithm has the property that there exists a $t < \infty$ such that $\sigma_{t+1} = \sigma_t$.*

Proof. Assume for the sake of contradiction that the RR-Generic algorithm does not converge. As there are only a finite ($n!$) number of permutations, the only way the algorithm cannot converge is when it cycles in a loop i.e., $\exists k > 2$ such that

$$\forall t \in [1, k-1], \quad \sigma_t = argstabsort(\mathbf{s}_F^{\sigma_{t-1}}), \quad \sigma_k = \sigma_0$$

We can assume that the loop begins and ends at σ_0. This is without loss of generality as the only other case is when the loop begins at some σ_p after $p > 0$ iterations of the algorithm. In such a case, we can assume that the algorithm was initialized with $\sigma_0 = \sigma_p$. A pictorial representation of the key idea of what follows in the proof is given in **??**.

Define $m \in [n]$ to be such that $\forall j > m$, $\sigma_t(j) = \sigma_0(j)\ \forall t \in [1, k-1]$. If such an m does not exist, then let $m = n$. Define $\ell(\sigma) = j$ where $\sigma(j) = m$. We first show that the following is true:

$$\mathbf{s}_F^\sigma(\ell(\sigma)) \leq \mathbf{s}_F^\rho(\ell(\rho))\ \forall \rho \in \mathbb{S}_n \tag{1}$$

To see why this is true, note that

$$\mathbf{s}_F^\sigma(\ell(\sigma)) = f(B^\sigma(\ell(\sigma)) = f(E)$$

where $E = \{j : \sigma(j) < m\}$. On the other hand,

$$\mathbf{s}_F^\rho(\ell(\rho)) = f(B^\rho(\ell(\sigma)) = f(E^{'})$$

where $E^{'} = \{j : \rho(j) < \rho(\ell(\sigma))\}$. However, by our choice of m, it must be the case that $\ell(\sigma) < m$ and hence $E \subseteq E^{'}$.

From Equation 1 we see that by choice of m, there must exist a ranking σ_r, $r \in [1, k]$ such that $\ell(\sigma_r) \neq \ell(\sigma_{r+1})$ (If not, then $\sigma_1 = \sigma_0$ and the algorithm converges). Without loss of generality, assume that $\sigma_r = \sigma_0$. This can be done because if a cycle exists among $\{\sigma_0, \ldots, \sigma_k = \sigma_0\}$, then a cycle also exists among $\{\sigma_r, \sigma_{r+1}, \ldots, \sigma 0, \ldots, \sigma_r\}$.

From the previous argument, we know that $\ell(\sigma_0) \neq \ell\sigma_{r+1})$. Moreover, as $\sigma_0(\ell(\sigma_0)) = m$, it must be the case that $\sigma_1(\ell(\sigma_0)) < m$. Let v be the smallest index when $\ell(\sigma_v) = \ell(\sigma_0)$. The following series of inequalities must then hold:

$$\begin{aligned}
\mathbf{s}_F^{\sigma_{v-1}}(\ell(\sigma_v)) &= \mathbf{s}_F^{\sigma_{v-1}}(\ell(\sigma_v)) \quad \text{(By definition)} \\
&\geq \mathbf{s}_F^{\sigma_0}(\ell(\sigma_0)) \quad \text{(By choice of } v) \\
&> \mathbf{s}_F^{\sigma_0}(\ell(\sigma_1)) \\
&\geq \mathbf{s}_F^{\sigma_1}(\ell(\sigma_1)) \\
&\geq \mathbf{s}_F^{\sigma_1}(\ell(\sigma_2)) \\
&\ldots \\
&\geq \mathbf{s}_F^{\sigma_{v-1}}(\ell(\sigma_v))
\end{aligned}$$

But this cannot happen because the inequality in the third line is strict. Hence we arrive at a contradiction. Thus the RR-Generic algorithm has to necessarily converge. □

Corollary 4.5. *The RR-Generic Algorithm converges when F is replaced by $F^t = \{f_1^t, \ldots, f_n^t\}$ where in the t-th iteration, f_i^t depends on σ^t i.e., $f_i^t : (2^{[n]} \times \sigma^t) \rightarrow \mathbb{R}_+$ and $f_i^t(E, \sigma^t) \geq f_i^t(E^{'}, \sigma^t)$ whenever $E \subseteq E^{'}\ \forall i, t$.*

Proof. Note that the proof of RR-generic uses the score function which depends on σ^t. It is easy to check that with the monotonicity property as stated in the theorem, the proof is identical to that of Theorem 4.4 □

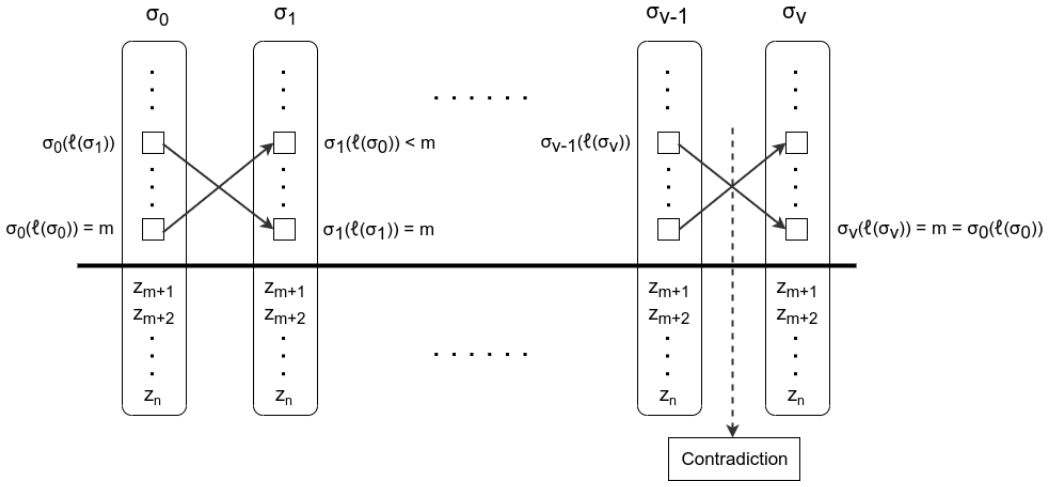

Figure 1: A pictorial representation of the key idea in the Proof of Theorem 4.4

4.1 Examples of RR-Generic Monotonic Functions

We present a few simple instantiations of the RR-Generic algorithm to illustrate the flexibility that comes with the abstraction.

Corollary 4.6. *Let $F = \{f_1, \ldots, f_n\}$ and $f_i(E) = n - |E|$ for all i and for all $E \subseteq [n]$. Then, for any $\sigma_0 \in \mathbb{S}_n$, the RR-generic algorithm converges in a single step and outputs σ_0*

The above corollary illustrates that the output of the RR-generic algorithm crucially depends on the initialization and could potentially have $n!$ possible outputs for certain choice of F. However, as the corollary below states, the output can also be the same independent of the initial permutation for certain choices of F.

Definition 4.7. Given a $G = ([n], P)$, define $\mathbf{b}_i = \sum_j P_{ij}$ as the *Borda* score associated with node i. Define σ_B to be the Borda ranking i.e., $\sigma_B = argstabsort(\mathbf{b})$.

Corollary 4.8. *Let $F = \{f_1, \ldots, f_n\}$ and $f_i(E) = \sum_{j \in E}(1 - P_{ji}) + \sum_{j \notin E} P_{ij}$ for all i and for all $E \subseteq [n]$. Then, for any $\sigma_0 \in \mathbb{S}_n$, the RR-generic algorithm converges in a single step and outputs the Borda ranking σ_B*

The above corollary, while straightforward, immediately suggests that if one comes up with a reasonable set of monotonic functions F, then the RR-generic algorithm can be used to find a useful ranking efficiently. While the Borda ranking is widely used for rank-aggregation, our focus in this work will be on coming up with monotonic functions which will be useful in influence maximization. We discuss this in detail in the next section.

5 IMRANK: INSIGHTS AND A NOVEL ANALYSIS

In this section, we begin by showing that the state of the art ranking based influence maximization algorithm IMRank can be derived as a special case of the RR-Generic algorithm.

Theorem 5.1. *The IMRank algorithm is a special case of the RR-Generic algorithm.*

Proof. We prove this by showing that the score function of a node in IMRank increases monotonically if the node is pushed above in the ranking. Specifically fix a $\sigma \in \mathbb{S}_n$ and consider any node i. The score assigned by IMRank depends both on the set $B^\sigma(i)$ and $A^\sigma(i) = \{[n] \setminus \{B^\sigma(i) \cup i\}\}$. Now consider a different permutation $\rho \in \mathbb{S}_n$ such that $B^\rho(i) \subset B^\sigma(i)$ and the relative ordering of all other nodes remain the same w.r.t ρ and σ. It follows that $A^\rho(i) \supset A^\sigma(i)$. Thus the number of nodes i *takes away* score from is greater in ρ than in σ. Also, the number of nodes i *gives away* it's score is lesser in ρ than σ. As the relative ordering of ρ and σ is same for every other node, this proves that the score $\mathbf{s}^\sigma(i) \leq \mathbf{s}^\rho(i)$.

Now we can choose f_i $\forall i$ such that $f_i^t(B^{\sigma_t}) = \mathbf{s}^{\sigma_t}(i)$ $\forall i, t$ i.e., the score computed for the i-th node by IMRank in the t-th iteration w.r.t σ^t. The result follows from Corollary 4.5. □

While the proof of the IMRank algorithm depended on the submodularity of the LFA allocation strategy, we find here that much lesser is actually required. Specifically, the key property needed for the convergence of the IMRank algorithm is the monotonicity of the LFA strategy and not submodularity.

We now deconstruct the properties of the LFA strategy further which will lead us to novel algorithms for diversity

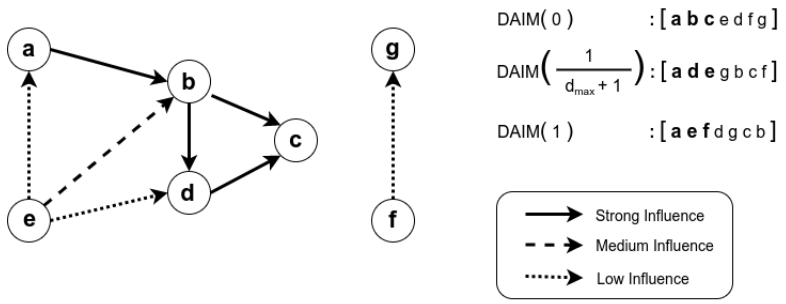

Figure 2: A toy example illustrating different criteria for seed node selection. DAIM(1) corresponds to preferring nodes which are diverse, DAIM(0) corresponds to nodes which have high Influence capacity, DAIM($\frac{1}{d_{\max}+1}$) corresponds to nodes which balance both as done by IMRank. Refer text for explanation regarding the nomenclature.

aware influence maximization that we will describe in the next section.

Theorem 5.2. *Let $\mathbf{s}_{in} \in \mathbb{R}^n$ be any score vector and let $\mathbf{s}_{out} \in \mathbb{R}^n$ be the score vector of each node after the LFA strategy i.e.,*

$$\mathbf{s}_{out} = LFA(\mathbf{s}_{in})$$

The mapping LFA as described above is linear i.e., there exists a $\mathbf{Q} \in \mathbb{R}^{n \times n}$ such that

$$\mathbf{s}_{out} = \mathbf{Q}.\mathbf{s}_{in}$$

Furthermore $\mathbf{Q}^T$ is a valid transition probability matrix of a Markov chain.

Proof. Note that in the IMRank algorithm (Algorithm 1), the scores of items gets updated from the last ranked to the first ranked (according to σ) and once the score of an item is updated in it's turn, it does not change after that. Define the matrix Q^{ij} associated with a directed edge (i, j) as the matrix which is identical to the identity matrix except in the j-th column where $Q^{ij}(i, j) = P_{ij}$ and $Q^{ij}(i, i) = 1 - P_{ij}$. Given a σ, the LFA strategy can then been seen as repeatedly pre-multiplying the score vector $\mathbf{s}_{in}$ with the matrices corresponding to the edges according to σ chosen by the IMRank Algorithm in a last to first manner. Thus the LFA mapping is a product of these simple edge matrices and hence is a linear mapping. Furthermore as each of these simple edge matrices can be viewed as the transpose of a transition probability of a Markov chain on n nodes (since the values are all between $[0, 1]$ and every column sum to 1), their product is also the transpose of a Markov chain transition probability matrix. □

The above theorem posits that the LFA mapping is linear and in fact the transpose of a Markov chain transition matrix. We can further understand this mapping as follows: As the mapping $\mathbf{Q}$ is linear, the i-th column of $\mathbf{Q}$ is simply given by $Q.\mathbf{e}_i$ where $\mathbf{e}_i$ is the i-th standard basis vector. The interpretation of this is as follows: If a score of 1 is given to node i and 0 to every other node, the score left with each node after the LFA strategy is given by the i-th column of $\mathbf{Q}$. In particular for every $j \neq i$, Q_{ji} represents the amount of score *taken away* by item j from item i during the LFA strategy and Q_{ii} is the amount of score that remains with item i after the LFA strategy.

In light of the above discussion, a node i is *important* in terms of maximizing influence in a network if the following two intuitive properties hold:

- **Influence Resistance** - The node cannot be *directly* influenced easily by other nodes i.e., influence resistance Q_{ii} is large.
- **Influence Capacity** - The node influences, directly or indirectly, several other nodes i.e., Q_{ij} is large.

The score given to a node by the IMRank algorithm is given by $\mathbf{s}_i = \sum_j Q_{ij} = Q_{ii} + \sum_{j \neq i} Q_{ij}$ can now be seen as weighing these two requirements with equal importance.

6 DAIM - DIVERSITY AWARE INFLUENCE MAXIMIZATION

As mentioned in the previous section,IMRank algorithm balances the *Influence Resistance* and the *Influence Capacity* properties assuming that these are equally important. However, when the graph has a community structure, this may not always be the best selection criteria especially when one is interested in diversity of the seed nodes picked. We illustrate this with the simple toy example in Figure **??**. As can be seen, the example consists of a network with 7 nodes with three different types of influences on edges, *low, medium* and *high*. The graph has two disjoint *communities* corresponding to the nodes $\{a, b, c, d, e\}$ in one community and $\{f, g\}$ in another. We consider three different selection criteria for this graph: one based on Influence Resistance (corresponding to DAIM(1)), one based on Influence Capacity (Corresponding

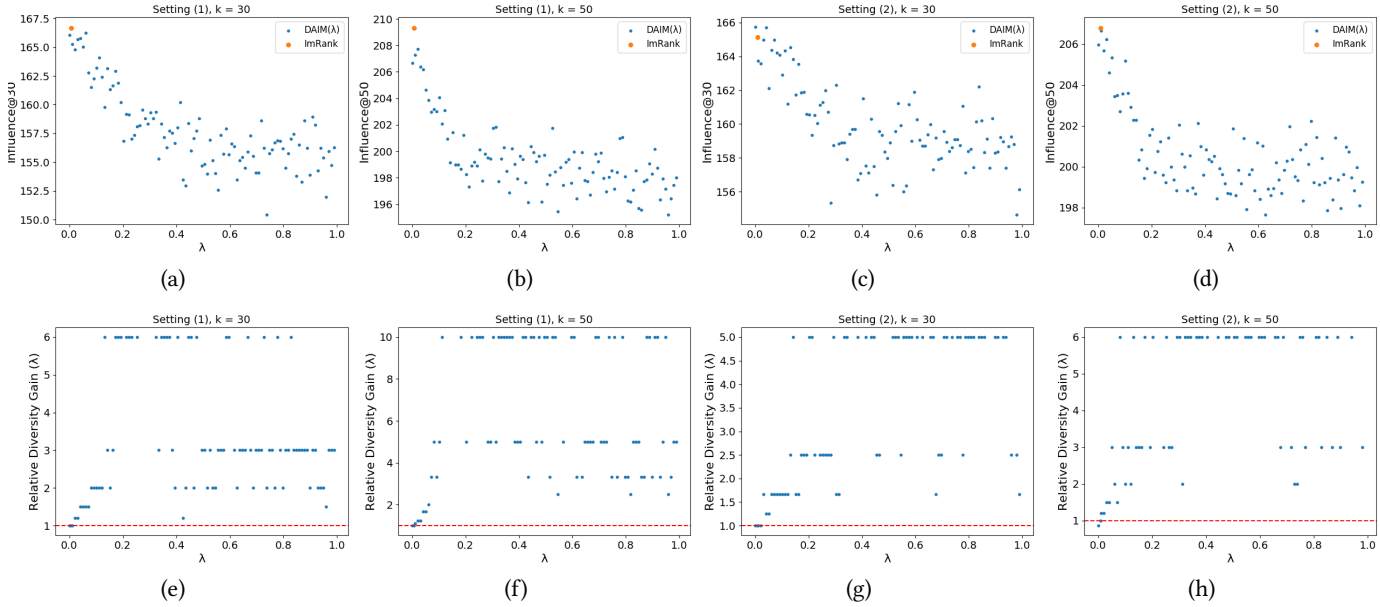

Figure 3: Effect of λ on Influence and Diversity for settings (1) and (2) corresponding to the stochastic block model experiments.

to DAIM(0)) and one that balances these both as done by IMRank algorithm (corresponding to DAIM($\frac{1}{d_{\max}+1}$). We shall explain the algorithm and the rationale behind these nomenclature shortly. For now, notice that an algorithm which tries to pick nodes purely based on high influence resistance results in picking the nodes $\{a, e, f\}$ as the top-3 nodes i.e., it picks nodes from both communities whereas the other two criteria pick nodes only from one community. Furthermore among those that pick from the same community, the one that uses just the Influence capacity picks the node $\{b\}$ among the top 3 whereas the one that balances both as done by IMRank picks the node $\{e\}$. Notice that node b individually has better influencing capacity over it's neighbours, it has less influence resistance (strongly influenced by node $\{a\}$). On the other hand, node $\{e\}$ has much weaker influence over nodes in the community but has higher influence resistance (not influenced by any other node). Thus IMRank prefers node $\{e\}$ over node $\{b\}$ at the top of the list.

As mentioned in the introduction, it may be critical to select seed nodes which could potentially trade away a bit of influence for gaining diversity. While the basic IMRank algorithm does not have a way to incorporate this, our insights gained from the analysis suggests the following procedure. *Compute influence resistance and influence capacity for each node and re-order nodes in an iterative fashion according to a convex combination of these two quantities.* This is precisely what we do in the next section where we present our proposed algorithm DAIM - Diversity Aware Influence Maximization.

6.1 The DAIM Algorithm

Algorithm 3 DAIM Algorithm

Input: Graph $G = ([n], P)$, an initial ranking σ_0, Diversity parameter $\lambda \in [0, 1]$
Initialize $\mathbf{s}_i = 1 \; \forall i$
repeat
  Compute the LFA map $\mathbf{Q}^{\sigma_t} \in [0, 1]^{n \times n}$
  Let $\mathbf{r}_i^t = \mathbf{Q}_{ii}^{\sigma_t} \; \forall i$
  Let $\mathbf{c}_i^t = \sum_j \mathbf{Q}_{ij}^{\sigma_t} \; \forall i$
  $\mathbf{s}^{\sigma_t} = \lambda . d_{\max} \mathbf{r}^t + (1 - \lambda)\mathbf{c}^t$
  $\sigma_{t+1} = argstabsort(\mathbf{s}^{\sigma_t})$
until $\sigma_{t+1} \neq \sigma_t$
Output: σ_t

In this section, we present the main algorithm of this paper - the Diversity Aware Influence Maximization algorithm. The algorithm is presented in Algorithm 3. The input to the algorithm is a graph $G = ([n], P)$ along with an initial ranking σ_0 and a diversity parameter λ. We refer to $d_{\max}$ to be the maximum (unweighted) out-degree of the graph G. We refer to the algorithm as DAIM(λ) when it is run with a certain choice of λ.

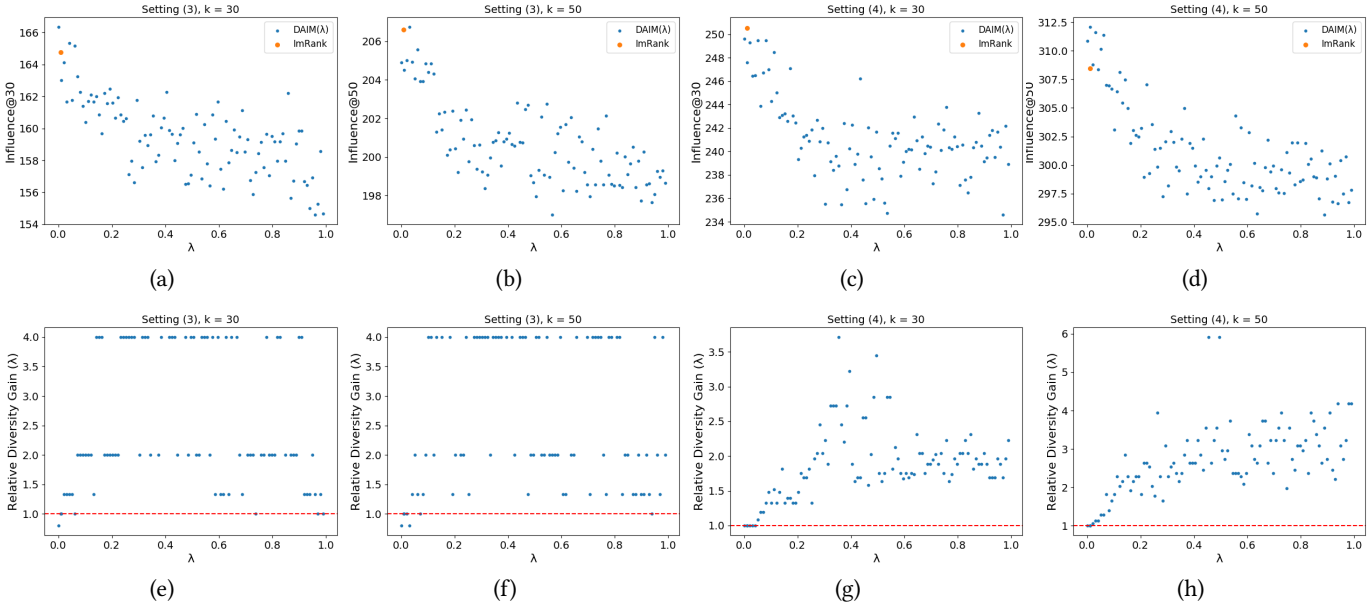

Figure 4: Effect of λ on Influence and Diversity for settings (3) and (4) corresponding to the stochastic block model experiments.

The algorithm starts with an initial ranking σ_0 and iteratively refines it as follows. At iteration t, the algorithm computes the linear LFA map matrix $\mathbf{Q}^{\sigma_t}$ w.r.t σ_t (refer Theorem 5.2). Two score vectors are obtained from this matrix: the influence resistance scores correspond to the diagonal entries of $\mathbf{Q}^{\sigma_t}$ and the influence capacity corresponds to the sum of the non-diagonal entries in each row of $\mathbf{Q}^{\sigma_t}$. As these two quantities are in different scales, they diagonal entries are scaled by $d_{\max}$ which is the maximum possible value of the sum of the non-diagonal entries in $\mathbf{Q}^{\sigma_t}$. The algorithm now computes a new score vector which is a convex combination of the influence resistance score and the influence capacity score of each node. A new ranking is obtained by sorting the scores and the algorithm proceeds to the next iteration.

We first show that the DAIM algorithm converges for any choice of λ.

Theorem 6.1. *Algorithm DAIM (Algorithm 3) converges for any choice of $\lambda \in [0, 1]$ as input.*

Proof. The proof of this amounts to showing that both the influence resistance and influence capacity scores have the monotonicity property. This will then imply that any convex combination of scores will also retain the monotonicity property. The theorem then follows as an application of Corollary 4.5 to the case where the functions f_i are defined to produce the convex combination of the influence resistance and influence capacity score vectors. To see why the influence resistance score vector $\mathbf{r}^t$ has the monotonicity property, observe that if a node i is moved higher up in the ranking keeping other relative positions fixed, then the number of nodes which *take away* from i decreases and hence the score increases. Similarly the number of items from which i takes away increases and hence the influence capacity score increases. Thus any convex combination of these score vectors would also have the monotonicity property. □

The following corollary is almost immediate

Corollary 6.2. *The DAIM rank algorithm recovers the IMRank algorithm when run with $\lambda = \frac{1}{d_{\max}+1}$*

Proof. Observe that for the choice of λ as in the statement of the theorem, the scores are a scaled version of the row sums of $\mathbf{Q}^{\sigma_t}$ where the scaling factor is given by $\frac{d_{\max}}{d_{\max}+1}$. As a constant scaling does not affect the *argstabsort* routine, the result follows. □

6.2 Computational Complexity

Algorithm 4 Influence Resistance Computation Algorithm

Input: Graph $G = ([n], P)$, a ranking σ
for $i = 1 : n$ **do**
  $\mathbf{r}_i = \Pi_{j \in B^{\sigma}(i)}(1 - P_{ji})$
end for
Output: $\mathbf{r} \in \mathbb{R}^n$

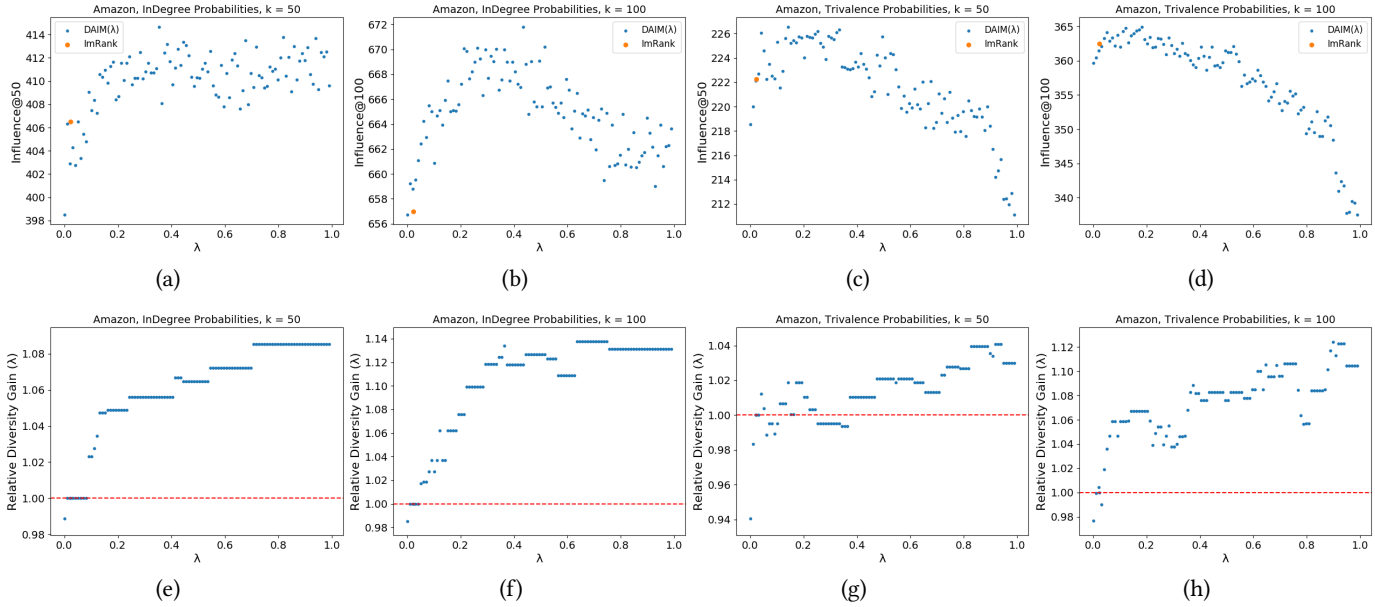

Figure 5: Effect of λ on Influence and Diversity for the Amazon dataset

The main advantage of IMRank over it's other seed selection counterparts is it's computational advantage. In practice, IMRank is perhaps the fastest algorithm available to pick reasonable quality seed nodes that approximate the computationally expensive greedy algorithm. While DAIM was proposed with the dual objective of balancing diversity and influence in a better fashion, it is not clear from the way it is presented as to how it compares computationally w.r.t IMRank. This may look like an issue as we seem to compute the matrix $\mathbf{Q}^{\sigma_t} \in [0,1]^{n\times n}$ at every iteration. However, in practice, this bottleneck can be sidestepped by a careful implementation of the algorithm. In particular, in our experiments, for each iteration t, we run IMRank to obtain the scores $\mathbf{s}_{im}^t$ for each node and obtain the influence resistance scores using algorithm 4. As $\mathbf{s}_{im}^t = (\mathbf{r}^t + \mathbf{c}^t)\frac{d_{\max}}{d_{\max}+1}$, we can obtain $\lambda\mathbf{r}^t + (1-\lambda)\mathbf{c}^t$ easily. As computing the influence resistance takes at most $O(nK)$ where K is the maximum in-degree of any node, the DAIM algorithm can be implemented as efficiently as the IMRank algorithm.

7 EXPERIMENTAL RESULTS

We present the experimental evidence on several real world data-sets and synthetic data-sets to test the performance of the proposed DAIM algorithm. We begin by describing the data-sets which were used in our experiments.

7.1 Data-sets:

We use both real world and synthetic data-set in our experiments.

Synthetic: For the synthetic data-sets, we work with stochastic block models [1] under 4 different settings as listed below. We chose SBMs for our experiments due to the natural community structure in these graphs.

(1) Two clusters, cluster 1 with 400 nodes where an edge is present with probability of 0.3 for each pair, cluster 2 with 100 nodes with edge probability of 0.2. The inter cluster edge probability was set to 0.05.
(2) Same edge probabilities as (i) but the cluster sizes of 100 and 400 nodes respectively.
(3) Same edge probabilities as (i) but the clusters of size 250 nodes each.
(4) 10 clusters with sizes given by the vector [200, 175, 150, 125, 100, 90, 70, 50, 30, 10]. Each intra cluster edge is present with probability 0.2 and each inter cluster edge with probability 0.05.

In each of the above settings, a graph was generated according the edge probabilities and fixed. The influence of node i over node j was fixed using the in-degree of node j as follows: $P_{ij} = \frac{1}{n_j}$ where n_j is the in-degree of node j.

Real World: The following real world data-sets were used in our experiments. We chose these because of the availability of ground truth communities in each of these data-sets.

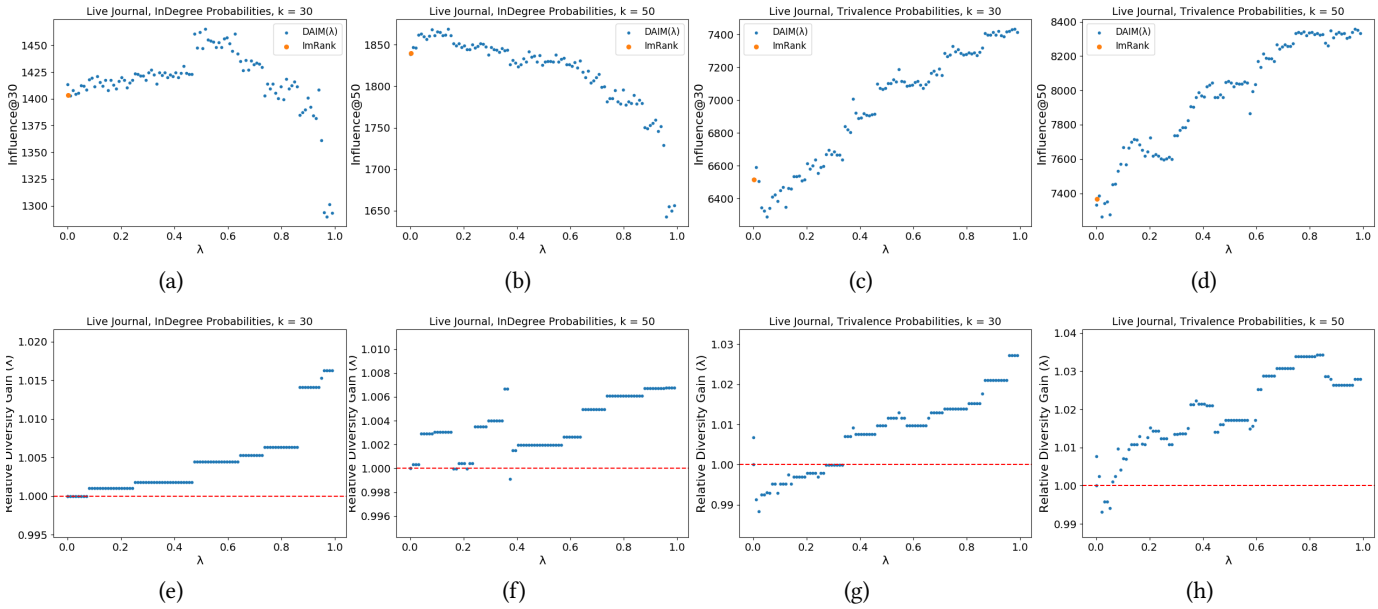

Figure 6: Effect of λ on Influence and Diversity for the DBLP dataset

Data-set	Nodes	Edges	Communities
Amazon	3074	16584	256
Live Journal	51474	860532	4605
DBLP	74907	358552	4605

Table 1: Statistics of Real-world Data-sets used in our experiments

(1) **Amazon [21]:** Here the nodes are products and an edge between two nodes is present if the products were purchased together. The product category were provided by Amazon.
(2) **Live Journal [21]** Here the nodes are online bloggers and an edge indicates a friendship relation between the bloggers. Communities are user defined groups.
(3) **DBLP [21]** Here the nodes were authors and an edge indicate that the authors published a paper together. The conference venue/journals were communities containing all authors who published in the venue.

In all the above data-sets, we filtered out nodes which were part of only one community so that the validity of the results can be tested unambiguously. Also if the graph was undirected, we made it directed by introducing edges in both directions. Further details about these data-sets is given in Table 1

In each of the above data-sets, we worked with two different type of influences probabilities (standard in IM literature). The first one is the in-degree based structure which is same as that explained under the synthetic data-set section. Secondly, we also consider the **trivalence model** where for each directed edge (i, j) node i influences node j in one of the three modes: low, medium, strong. In all our experiments, we set probability of low to be 0.0025, medium to be 0.025 and strong to be 0.25.

7.2 Experiment Settings:

We performed the following experiments for all the settings described above

(1) Effect of varying λ in DAIM on Influence
(2) Effect of varying λ in DAIM on diversity

In (1) above, the influence was computed by running Monte Carlo simulations for the greedy algorithm on the top K nodes suggested by each value of λ. We tested with 100 values of λ for each experiment where the λ values were equally spaced in $[0, 1]$. In addition, we always added the λ that corresponded to IMRank as well.

In (2) above, the diversity in the seed set is measured as follows: Let the ground truth community fractions be given be the vector $\mathbf{f} \in [0, 1]^L$ where L is the number of communities and $\sum_i \mathbf{f}_i = 1$ where $\mathbf{f}_i$ is the fraction of nodes belonging to community i. We compute a similar fraction for the seed nodes. For a fair allocation algorithm that respects the diversity of the population, the seed node fraction is expected to be close to that of the ground truth than an allocation that does not respect this diversity. Let the seed node fraction for

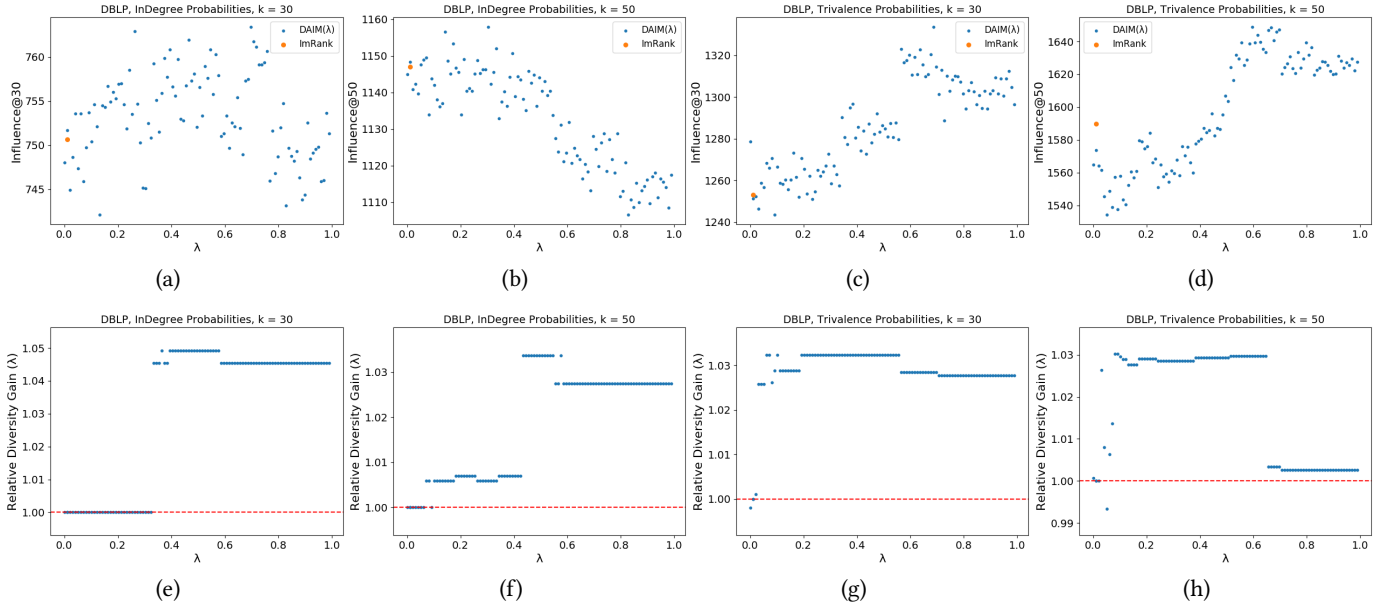

Figure 7: Effect of λ on Influence and Diversity for the Live Journal dataset

IMRank be given by f_{imrank} and for DAIM(λ) be given by $\mathbf{f}_\lambda$. We measure how better or worse a particular choice of λ is with respect to IMRank as follows:

$$\text{Relative Diversity Gain of } \lambda = \frac{\|\mathbf{f}_{imrank} - \mathbf{f}\|}{\|\mathbf{f}_\lambda - \mathbf{f}\|}$$

Thus if the relative diversity gain is greater than 1, then the choice of λ is better in terms of capturing the diversity in the underlying node population.

7.3 Results:

Synthetic Data: We first present the result for synthetic experiments on the stochastic block models (SBM). As described before, we consider 4 different types of SBM settings. The results for the first two settings are in Figure 3 and those for the last two settings are in Figure 4. We show results for two different value for the size of seed nodes $K = 30$ and $K = 50$ respectively [1]. In each of the plot, we also highlight the IMRank algorithm: for the influence plot, the IMRank performance is indicated by a large sized (orange) dot and for the diversity plot a (red) dotted line is shown corresponding to the diversity relative gain equal to 1 (i.e., same as IMRank).

As can be observed from these figures,with increase in λ a drop of atmost 10% is seen in terms of influence of the seed nodes. However with respect to diversity, a relative gain of around 400% is observed. bhis suggests that for SBM models, DAIM rank with a reasonably large value of λ (around 0.5) provides much better diversity gain with almost negligible loss of influence. Recall that the λ corresponding to IMRank is equal to $\frac{1}{d_{\max}+1}$ which is close to 0.

Real-world Data-sets: We next present our results on the real-world data-sets described earlier. Note that while these data-sets had underlying ground truth communities, the algorithm is unaware of the same. In general, it would not be fair to expect the real world data-sets to perfectly follow SBM structure especially with the number of communities in the order of thousands in some data-sets.

We present the results for two different choices of K for each of the three data-sets. For the Amazon data-set, we report results for $K = 50$ and 100 (Figure 5) and for the DBLP (Figure 6) and Live Journal (Figure 7 data-sets, we report results on $K = 30$ and $K = 50$.

For all the three data-sets, when the influence probabilities are based on in-degree of nodes, the behaviour mimics that of SBMs. In particular, as λ increases, one observes a drop in influence while a gain in diversity. Interestingly, and perhaps surprisingly, in the case when the influence probabilities follow the trivalence model, we get the best of both worlds i.e., with increase in λ, both the influence and the diversity increase for both the DBLP and Live Journal data-sets. In some cases (see for instance $K = 30$ for Live Journal), the gain in influence is significant and is in the order of 1000 nodes compared to IMRank while the gain in diversity is around 3% which is also significant given the size of these networks. This clearly suggests that the proposed DAIM

[1]Other values of K were tried and similar results were observed and hence have been omitted

algorithm is robust to picking diverse nodes independent of the model of influence (trivalence or in-degree) and can potentially lead to improvement in influence as well under certain conditions.

In practice, the above experiments suggest that for real world data, a reasonable choice for λ is again 0.5 which empirically can be justified to provide improvement in diversity with either minimial loss of influence or even gain in influence for certain models.

8 CONCLUSION

In this work, we presented a novel rank refinement of monotonic set functions based framework to obtain rankings from an initial ranking. We showed that the popular IMRank algorithm can be viewed as a special case of our framework and proposed DAIM - a diversity aware influence maximization algorithm. Extensive experimental evidence on real and synthetic data-sets establish the superiority of the proposed method in contrast to existing ranking based algorithms.

We believe the rank refinement framework can be analyzed as a general tool for several related problems including the problem of rank aggregation. In terms of applications, we would like to investigate properties of this framework for such problems as part of future work. On the theoretical front, we would like to understand the fixed points of the rank refinement scheme in terms of quality of approximating the desirable property in a graph. We also wish to consider the theoretical aspects of rate of convergence for the proposed framework.

REFERENCES

[1] Emmanuel Abbe. Community detection and stochastic block models: recent developments, 2017.

[2] Akhil Arora, Sainyam Galhotra, and Sayan Ranu. Debunking the myths of influence maximization: An in-depth benchmarking study. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 651–666, New York, NY, USA, 2017. ACM.

[3] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 946–957, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics.

[4] Arastoo Bozorgi, Saeed Samet, Johan Kwisthout, and Todd Wareham. Community-based influence maximization in social networks under a competitive linear threshold model. *Knowledge-Based Systems*, 134:149 – 158, 2017.

[5] A. Caliò, R. Interdonato, C. Pulice, and A. Tagarelli. Topology-driven diversity for targeted influence maximization with application to user engagement in social networks. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2421–2434, Dec 2018.

[6] Wei Chen, Tian Lin, Zihan Tan, Mingfei Zhao, and Xuren Zhou. Robust influence maximization. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 795–804, New York, NY, USA, 2016. ACM.

[7] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 1029–1038, New York, NY, USA, 2010. ACM.

[8] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 199–208, New York, NY, USA, 2009. ACM.

[9] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 88–97, Washington, DC, USA, 2010. IEEE Computer Society.

[10] Suqi Cheng, Huawei Shen, Junming Huang, Wei Chen, and Xueqi Cheng. Imrank: Influence maximization via finding self-consistent ranking. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 475–484, New York, NY, USA, 2014. ACM.

[11] Suqi Cheng, Huawei Shen, Junming Huang, Guoqing Zhang, and Xueqi Cheng. Staticgreedy: Solving the scalability-accuracy dilemma in influence maximization. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 509–518, New York, NY, USA, 2013. ACM.

[12] Sainyam Galhotra, Akhil Arora, and Shourya Roy. Holistic influence maximization: Combining scalability and efficiency with opinion-aware models. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 743–758, New York, NY, USA, 2016. ACM.

[13] Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 47–48, New York, NY, USA, 2011. ACM.

[14] L. Hajdu, M. Krász, and A. Bota. Community based influence maximization in the independent cascade model. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 237–243, Sep. 2018.

[15] Xinran He and David Kempe. Robust influence maximization. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 885–894, New York, NY, USA, 2016. ACM.

[16] Huimin Huang, Hong Shen, Zaiqiao Meng, Huajian Chang, and Huaiwen He. Community-based influence maximization for viral marketing. *Applied Intelligence*, 49(6):2137–2150, Jun 2019.

[17] Kyomin Jung, Wooram Heo, and Wei Chen. Irie: Scalable and robust influence maximization in social networks. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ICDM '12, pages 918–923, Washington, DC, USA, 2012. IEEE Computer Society.

[18] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[19] Siyu Lei, Silviu Maniu, Luyi Mo, Reynold Cheng, and Pierre Senellart. Online influence maximization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 645–654, New York, NY, USA, 2015. ACM.

[20] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 420–429, New York, NY, USA, 2007. ACM.

[21] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[22] Xiao Li, Xiang Cheng, Sen Su, and Chenna Sun. Community-based seeds selection algorithm for location aware influence maximization. *Neurocomputing*, 275:1601 – 1613, 2018.

[23] Brendan Lucier, Joel Oren, and Yaron Singer. Influence at scale: Distributed computation of complex contagion in networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 735–744, New York, NY, USA, 2015. ACM.

[24] Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-Ichi Kawarabayashi. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 138–144. AAAI Press, 2014.

[25] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'18)*, 2018.

[26] Jiaxing Shang, Shangbo Zhou, Xin Li, Lianchen Liu, and Hongchun Wu. Cofim: A community-based framework for influence maximization on large-scale networks. *Knowledge-Based Systems*, 117:88 – 100, 2017. Volume, Variety and Velocity in Data Science.

[27] Shashank Sheshar Singh, Kuldeep Singh, Ajay Kumar, and Bhaskar Biswas. Coim: Community-based influence maximization in social networks. In Ashish Kumar Luhach, Dharm Singh, Pao-Ann Hsiung, Kamarul Bin Ghazali Hawari, Pawan Lingras, and Pradeep Kumar Singh, editors, *Advanced Informatics for Computing Research*, pages 440–453, Singapore, 2019. Springer Singapore.

[28] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1539–1554, New York, NY, USA, 2015. ACM.

[29] Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 75–86, New York, NY, USA, 2014. ACM.