Summary of the article: Influence Maximization/ Near-Optimal Time Complexity Meets Practical Efficiency

The application domain of that subject is the viral marketing (to choose the less influential individuals). The subject of this article is the TIM algorithm. They determine the complexity of that algorithm with a probability due to an approximation.

The advantages of the TIM algorithm is that allows to use 2 of the models, the IC(independent cascade) and the LT(linear trashold) due to the fact that it contains novel heuristics.

Variables:

Social Network: G

Constant: k

M: probabilistic model that captures how the node G may influence the behavior.

Historical part & breakthrough until the article

Kempe & al was the first to bring a greedy approach of the problem because he considers it as combinatory optimization problem.

The model was accepted due to the facility of application even if there was a very expensive.

Then the objective was to reduce the complexity of the problem in order to offer better solution to that problem.

They were based on the heuristic method and then their limitation was based on the Kempe's ratio.

Borgs proposed an improvement of the algorithm for IC.

There is still no solution without non-trivial approximation, the algorithm must be based on heuristic approximation even if the results can be worse than the optimum.

Contribution of the article

Article present Two-phase Influence Maximization(TIM)-> an algorithm that respect the approximation & that can practice influence maximization.

The approximation of the results is the same than on the other heuristic approximation with the same probability of getting a result.

The complexity is equals to $O((k+l)(m+n)\log n/\varepsilon^2)$ in the expectation. Not very far from the Borg explanation.

The advantages of that algorithm is that it can support general cascade model whereas other model only supports a specific version of the cascade model.

Preliminaries

Based on the independent cascade. Each edge is associated with a propagation probability.

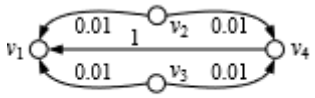

Figure 1: Social network G.

Example of a graph.

First, we activate a node then we go to the inactived neighbor of a node and then check if a node is already activated because the activation can't stop.

Using the Monte Carlo method to have a good approximation of the value of E to estimate E.

Define the value of r to 10000

Problem of the estimation came from the fact that we can't expect the result.

Use the RIS(reverse influence sampling)

Based on 2 methods:

- We invert the graph to show if two nodes are linkable.

- We choose randomly a node for the sampling. (RR)

Then we check if it appears in our path and then we add it to our path.

We generate randomly and if a node appear frequently that mean it has many influence and it is one of the most important node of our graph.

The generation stop when we reach a predefined threshold. To optimize the value, we only increase the value of the threshold.

TIM algorithm is based on 2 steps, the first one is to consider the parameter estimation & the node selection.

The sample is based on the RIS algorithm, but we choose the root. The fact that we choose 0 as value is to ensure the effectiveness of the algorithm. The code for the node selection is as below:

We generate a value and implements the RR and then they choose the value and return all the nodes that have been chosen.

Algorithm 1 NodeSelection (G, k, θ)

1: Initialize a set $\mathcal{R} = \emptyset$.
2: Generate θ random RR sets and insert them into $\mathcal{R}$.
3: Initialize a node set $S_k^* = \emptyset$.
4: **for** $j = 1$ to k **do**
5: Identify the node v_j that covers the most RR sets in $\mathcal{R}$.
6: Add v_j into S_k^*.
7: Remove from $\mathcal{R}$ all RR sets that are covered by v_j.
8: **return** S_k^*

The generation of each RR is based on a random BFS.

We stack the node in an empty queue and we do a coin flip to find the first nodes and add it to the queue.

Then we extract the value of the queue in a second queue. If it considers as not visited, then we add into the second and we dequeue to make the list empty and when the queue is empty we got our results.

Then there is the parameter estimation:

Algorithm 2 KptEstimation (G, k)

for $i = 1$ to $\log_2 n - 1$ **do**
  Let $c_i = (6\ell \log n + 6 \log(\log_2 n)) \cdot 2^i$.
  Let $sum = 0$.
  for $j = 1$ to c_i **do**
    Generate a random RR set R.
    $\kappa(R) = 1 - \left(1 - \frac{w(R)}{m}\right)^k$
    $sum = sum + \kappa(R)$.
  if $sum/c_i > 1/2^i$ **then**
    return $KPT^* = n \cdot sum/(2 \cdot c_i)$
return $KPT^* = 1$

This is an estimation of the KPT, we used it to know how the subject is measured, that means the quality of the previous algorithm and the recovery. The smallest value is equals to 1.

To compute the new value of our KPT.

Algorithm 3 RefineKPT $(G, k, KPT^*, \varepsilon')$

Let $\mathcal{R}'$ be the set of all RR sets generated in the last iteration of Algorithm 2.
Initialize a node set $S'_k = \emptyset$.
for $j = 1$ to k **do**
  Identify the node v_j that covers the most RR sets in $\mathcal{R}'$.
  Add v_j into S'_k.
  Remove from $\mathcal{R}'$ all RR sets that are covered by v_j.
Let $\lambda' = (2 + \varepsilon')\ell n \log n \cdot (\varepsilon')^{-2}$.
Let $\theta' = \lambda'/KPT^*$.
Generate θ' random RR sets; put them into a set $\mathcal{R}''$.
Let f be the fraction of the RR sets in $\mathcal{R}''$ that is covered by S'_k.
Let $KPT'' = f \cdot n/(1 + \varepsilon')$
return $KPT^+ = \max\{KPT'', KPT^*\}$

We redefine the value of our KPT between the two steps. It reduces the complexity with a k factor.

Comparison with others algorithm:

RIS has an awful complexity compared to the TIM, and the TIM algorithm have a better asymptotic performance

Greedy: In the approximation given by the article, we get a worse complexity.