Supervisors: Peter Battaglia and Michal Valko

Graph Networks For Influence Maximization

Hugo Cisneros
hugo.cisneros@ens-paris-saclay.fr

Hind Dadoun
hind.dadoun@ens-paris-saclay.fr

Abstract

Recent approaches using neural networks capable of operating on graph-structured data have had successes on several tasks involving graph manipulations. We consider the Influence Maximization (IM) problem in the Independent Cascade model (IC). The *offline* version of the problem, where one seeks a maximal seed set of nodes maximizing influence with knowledge of transition probabilities between nodes has been well explored. This paper studies possible applications of graph networks to solving IC-IM in order to approximate and compare the performance with existing heuristics for solving IM problems. For this purpose, we train a graph network with both PMIA and the weighted degree heuristic and analyze the results.

1 Introduction

1.1 Influence maximization

When dealing with social networks such as Facebook and Twitter, an essential topic is the propagation of ideas and influence. This question is studied by marketers trying to maximize the visibility of products or campaign managers wanting to persuade voters. Specifically, given a budget of *influencers* (or *seeds*) and a network of people capable of influencing each other, one wants to choose targets (or nodes in a graph) that yields the best *influence spread* or expected number of people affected by the campaign.

Kempe, Kleinberg, and Tardos (2003) were among the first to devise the modern formulation of the Influence Maximization problem as the task of finding a *small* set of seed nodes in a social network that maximizes the spread of influence under some influence models they propose. Here we focus specifically on the Independent Cascade model, originally proposed by Goldenberg, Libai, and Muller (2001), which consists in a discrete process where an initial set of active nodes are given a single chance to activate their currently inactive neighbors through independent sampling of a Bernoulli variable. The process is repeated until no more activation is possible in the graph.

The *offline* IM problem assumes that influence probabilities between each node of the graph are known from the beginning and the task is to find the maximal set of seed nodes. This problem was proven to be #P-hard but its solution can be efficiently approximated with approximate algorithms (Chen, C. Wang, and Y. Wang, 2010).

1.2 Graph networks

Learning from graph-structured data is a difficult problem that has seen a surge in interest in the past years. It has been studied extensively for representation learning of graphs (Hamilton, Ying, and Leskovec, 2017; Xu et al., 2018), with application to semi-supervised classification (Kipf and Welling, 2016) or predicting the temporal evolution of a complex system (P. Battaglia et al., 2016).

Graph networks are designed to process relational data while being invariant to the structure, number of nodes and edges of the graph. Thus, they enable working with much more complex systems than traditional neural networks.

2 Related work

In this section, we first provide some details about the KK-Greedy algorithm. Secondly we present the recent works done to improve the efficiency of KK-Greedy. Finally, we give a more detailed presentation of graph networks, some of their successes and Deep mind's graph net framework that we will be working with.

2.1 KK-Greedy

A greedy algorithm follows the problem solving heuristic of making the locally optimal choice at each stage with the intent of finding a global optimum.

KK-Greedy algorithm uses the properties of sub-modular and monotone functions defined as follow

Definition 1. *If Ω is a finite set, a sub-modular function is a set function f : $2^{\Omega} \rightarrow \mathbb{R}$, which satisfies the following condition :*
For every S, T $\subseteq \Omega$ we have that $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$.

Algorithm 1 Greedy(k, f)

```
initialize S = ∅
for i = 1 to k do
    select u = arg max_{w∈V−S}(f(S ∪ {w}) − f(S))
    S = S ∪ {U}
end for
output S
```

Algorithm 1 shows the pseudo-code implementation of a greedy algorithm for influence maximization.

For any sub-modular and monotone function f with $f(\emptyset) = 0$, the algorithm iteratively selects new seed u that maximizes the incremental change of f into the seed set S until k seeds are selected. The influence spread of S, which is the expected number of activated nodes given seed set S, is denoted as $\sigma_I(S)$. Kempe, Kleinberg, and Tardos (2003) proved that influence maximization on the IC model is monotone and sub-modular, and therefore the approximate Greedy algorithm produces near-optimal solutions and solves the influence maximization problem with an approximation ratio of $1 - 1/e$.

2.2 More recent work

The main issue with the KK-Greedy, is that there is no efficient way to compute $\sigma_I(S)$ given a set S. Chen, C. Wang, and Y. Wang (2010) point out that the computation is of the influence spread is #P-hard. They propose a heuristic scheme to use local arborescence structures spanning from each node to approximate the influence propagation (MIA) and show that the influence spread in the MIA model is sub-modular. Therefore, a simple greedy algorithm can guarantee an influence spread within (1 - $\frac{1}{e}$) of the optimal solution in the MIA model.
Let $P = (u = p_1, ..., p_m = v)$ be a path from node u to node v. We denote $pp(P)$ the propagation probability of the whole path.

$$pp(P) = \prod_{i=1}^{m-1} pp(p_i, p_{i+1}).$$

where $pp(p_i, p_{i+1})$ is the propagation probability from node p_i to node p_{i+1}.

The authors use the maximum influence path, which corresponds to the path from u to v with maximum propagation probability, and is denoted $MIP_G(u, v)$. The definition can be written

$$MIP_G(u, v) = \arg\max\{pp(P) | P \in \mathcal{P}(G, u, v)\})$$

Where $\mathcal{P}(G, u, v)$ is the set of all paths from u to v.

If we transform the propagation probability $pp(u, v)$ to $-log(pp(u, v))$, the MIP is the shortest path from u to v which can be computed efficiently with algorithms such as Dijkstra's algorithm.

Now, for a given node v, we want to estimate the influence to v from other nodes and the influence from v to other nodes. These two quantities are symmetric :

$$MIIA(v,\theta) = \bigcup_{\substack{v \in V \\ pp(MIP_G(u,v)) \geq 0}} MIP_G(u,v)$$

$$MIOA(v,\theta) = \bigcup_{\substack{u \in V \\ pp(MIP_G(v,u)) \geq 0}} MIP_G(v,u)$$

They denote the activation probability for any node u in $MIIA(v,u)$ as $ap(u,S,MIIA(v,u))$ which is the probability that u is activated when the seed set is S and influence is propagated in $MIIA(v,\theta)$. The authors propose an algorithm to compute recursively this quantity. Finally they denote $\sigma_M(S)$ the influence spread of S in the MIA model:

$$\sigma_M(S) = \sum_{v \in V} ap(v,S,MIIA(v,\theta))$$

With this method, they point out that computing $\sigma_M(S)$ is polynomial-time. Together with Algorithm 1, they already have a polynomial-time approximation algorithm.To achieve a better approximation to the IC model, they propose a MIA model in which the influence of a seed is not blocked by other seeds. Let $S = (s_1, .., s_m)$ where s_i are seeds and $S^i = S\ (s_i)$. The maximum influence path is then computed in $G(S^i)$ the sub-graph of G induced by $V : S^i$. This extension is called the prefix excluding MIA (PMIA) model which is the algorithm we will use to train our graph network.

2.3 Graph networks

The motivation behind graph networks is that the standard deep learning toolkit does not offer component on arbitrary relational structure. Whereas this information can increase performance if the task has considerable relational structure. For instance, models in the graph neural network family showed great success in different domains such as visual scene understanding, learning dynamics of physical systems and predicting chemical properties of molecules. (P. W. Battaglia et al., 2018) recently presented their graph networks (GN) framework, that aims at unifying the different architecture of graph neural networks under a same paradigm. The main unit of computation in the GN framework is the GN block, a module that takes a graph as input, performs computations over the structure, and returns a graph as output. Entities are represented by the graph's nodes, relations by the edges, and system-level properties by global attributes. This graph-to-graph input/output interface ensures that the output of one GN block can be passed as input to another, which allows the construction of complex architectures by composing GN blocks. The authors present the encode-process-decode architecture which takes an input graph, transforms it into a latent representation using an encoder (also a GN block) and applies multiple times a shared core block for processing and finally output a graph using a decoder block. The advantage of this framework is that it is very flexible and can easily be adapted to our problem, while taking advantage of the powerful properties of graph networks.

3 Graph networks for Influence Maximization

We define in this section the formal statement of the IM problem under the IC model. Given a directed graph $G = (V, E)$ with $N_V = |V|$ the number of nodes and $N_E = |E|$ the number of edges, the transition probabilities can be described as a probability weight function $w : E \to [0, 1]$ representing for an edge (v_1, v_2), the activation probability of v_2 knowing that v_1 was activated.

Given a integer $K < N_V$, one needs to choose a set $S \subset V$ of cardinality K, these nodes will be *activated*, and will propagate influence according to the IC model. A realization of the IC model diffusion process is obtained by independently sampling from Bernoulli random variables $\mathbf{w}(e) \sim \text{BERNOULLI}(w(e))$ for each edge $e \in E$.

We say, following notations from Wen et al. (2016), that a node v_2 is reachable from v_1 if and only if there exists a directed path $p = (e_1, ..., e_k)$ such that $\forall\ 1 \leq i \leq k, \mathbf{w}(e_i) = 1$. The node v_2 is *influenced* if v_1 is contained in the seed set S and v_2 is reachable from v_1. The total number of influenced nodes for a given realization is $f(S, \mathbf{w})$.

The objective of IM is to maximize the expected number of influenced nodes, that is finding $S^* = \arg\max_{S;|S|=K} \mathbb{E}[f(S, \mathbf{w})]$.

We will use one of the algorithms proposed by Chen, C. Wang, and Y. Wang (2010) that is able to approximate the set S^* to train a graph network to predict a set of maximum influence nodes in a graph.

Graph networks take graphs as input and return graphs as outputs. The input graph has edge-(E), node-(V), and global-level (u) attributes, and the output graph has the same structure, but updated attributes (P. W. Battaglia et al., 2018). In this context, we set the edge attributes to the transition probabilities $\mathbf{w}(e)$ of the graph. Since we expect the architecture to be better at approximating additive relations rather than multiplicative ones, and because influence propagates in a multiplicative way with respect to the transition probabilities, we also add the negative log-transition probability $(-\log \mathbf{w}(e))$ as an attribute of the edges. This is based on the observation that the propagation probability between two nodes is the shortest path if the edges have negative log-transition probability as weights. We also use the node attributes as an indicator of whether the node belongs to the set of maximum influence nodes in a target example.

4 Experiments

We took inspiration from the demonstration of shortest-path in the graph net repository[1] to adapt it to the IM problem. Since the task is taking place in an offline setting, the edge attributes are not updated during the processing of the graph. The graph network is trained with a set of training examples generated as binomial graphs with random number of nodes and random probability of having an edge between any two nodes uniformly sampled between 0.5 and 0.6. The ground truth maximum influence set of nodes is then generated using an oracle that we fisrt choose to be PMIA. This algorithm is know to be sub-optimal, but has better results in terms of expected influence spread than a simple greedy algorithm Chen, C. Wang, and Y. Wang, 2010 in general and is much faster.

A trade-off between efficiency of the oracle algorithm and speed was necessary since training a graph network has to be done in a lot of iterations, which wouldn't be practically possible with a slow target creation process.

To encourage the network to output K nodes $\in S$, we add to a standard cross entropy term for the nodes labels the following additional loss term

$$\mathcal{L}_{\text{card}} = ||\text{card}(\{p > 0.5 \mid p \in f(G)\}) - K||_2^2$$

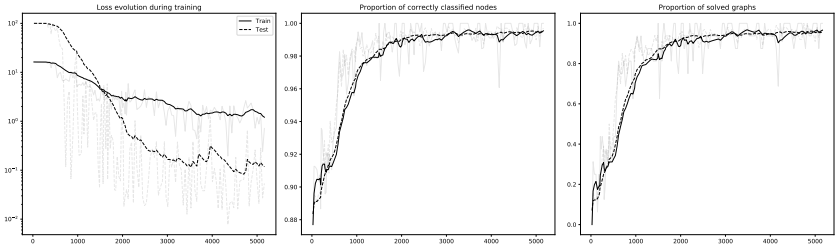
Where $f(G)$ is the output of the graph network for an input graph G, that is a set of probabilities of being in the set S (one for each node) and K is the target number of seed nodes.

We have empirically observed that this loss term helped the network output exactly the number of nodes with high probability as necessary. This was proven especially useful for experiments where we tried predicting sets with more than 3 nodes.

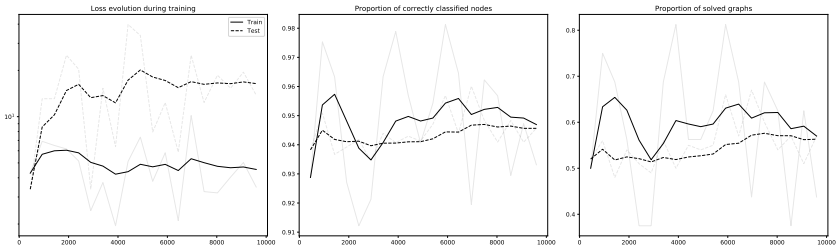
The graph network architecture we chose is similar to the shortest-path demo, with an Encoder-Processor-Decoder structure. The nodes, edges and global attributes are encoded by separate trainable multi-layer perceptrons (MLP). The encoded representation is further processed a fixed number of times by a graph network, where node information is aggregated from in and out-edges. The loss is computed at each step of the message passing process (not only the last) in order to encourage the network to solve the problem in as few steps as possible and not rely on the fixed number of steps to compute a solution. A message passing architecture seems well adapted to a influence spread problem, and we expect the network to learn to estimate influence by spreading information in the graph. It however degraded the performance greatly to aggregate node information from in or out-edges only. We ran the experiments on randomly generated binomial graphs with a set range of number of nodes and random probability between 0.5 and 0.6 of having an edge between any two nodes. Edge transition probabilities are also randomly sampled from a uniform distribution $U(0, 0.1)$ that we treat as the ground truth probabilities. This range of probabilities is used by Wen et al. (2016) and is guided by empirical evidence (Goyal, Bonchi, and Lakshmanan, 2010; Barbieri, Bonchi, and Manco, 2013).

[1] `https://github.com/deepmind/graph_nets/blob/master/graph_nets/demos/shortest_path.ipynb`

We first tried to solve the relatively simple task of finding the node with highest influence in the graph. The sub-optimality of PMIA is relatively low in that setting, especially for small values of the parameter θ, and the target graphs it generates are accurate. Figure 1a shows the train and test loss, accuracy and proportion of solved examples for this experiment. We can see that the network quickly learns to output results that closely match the output of PMIA.



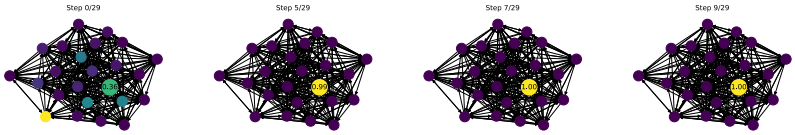
(a) 1 seed, 10 message passing steps and between 10 and 25 nodes per graph and PMIA heuristic.



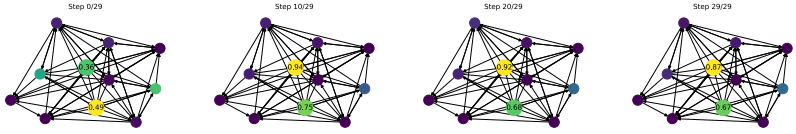
(b) 1 seed, 10 message passing steps and between 10 and 25 nodes per graph with weighted degree heuristic.

Figure 1: Loss, number of correctly predicted node and proportion of solved examples.

As shown on the Figure, the task of finding 1 seed node is relatively easy to achieve, and a graph network is able to predict accurately the maximum influence node. Figure 2a shows a graph from the testing set with output probabilities at several steps of the message-passing process. All nodes outputs are initialized at 0 and the network refines its prediction of the maximum influence node at each step.



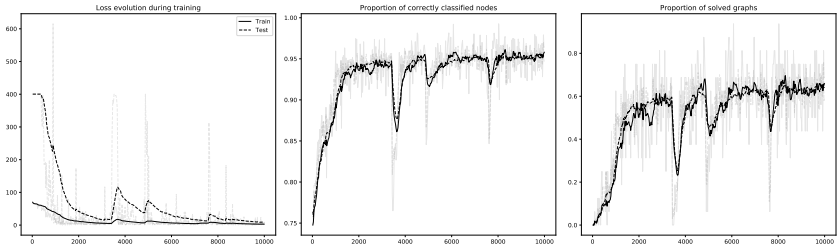
(a) 1 seed



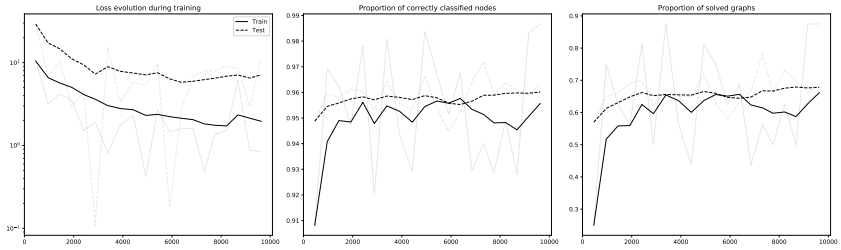
(b) 2 seeds

Figure 2: Sample graphs from the test set, along with the output from the graph network at several steps for the results of Figures 1a and 3a

Naturally, as the number of seed nodes to be predicted increases, the task becomes more difficult and the probability that the oracle gave sub-optimal predictions also increases. For these reasons, overall performance decreases. For example, Figures 2b and 3a show that the network is less efficient at correctly predicting the two most influential nodes with high confidence, and the maximal proportion of solved graphs doesn't go above 70% on average.



(a) 2 seed nodes, 30 message passing steps, between 10 and 25 nodes per graph with PMIA heuristic.



(b) 2 seed nodes, 30 message passing steps and between 10 and 25 nodes per graph with weighted degree heuristic.

Figure 3: Loss, number of correctly predicted node and proportion of solved examples.

Although PMIA was built for time efficiency, label generation is still the main bottleneck in the training of the network, which prevents one from training for a very large number of steps and/or work with larger graphs. The Weighted Degree heuristic which consists in selecting k seeds that have maximum total out-connection weight, is naturally much faster than PMIA at computing a set of maximum influence nodes. If the network is able to learn from such a heuristic, it would allow for much more thorough training on larger scale examples. This heuristic, although sub-optimal like PMIA, has a very advantageous computational complexity of $O(N_V)$ where N_V is the number of nodes in the graph. Figure 4 shows a benchmark of both algorithms on a fixed graph with increasing number of seed nodes and on graphs of increasing size with fixed number of seed nodes to select. It notably highlight the overall polynomial complexity of PMIA compared to the linear complexity of the weighted degree heuristic.

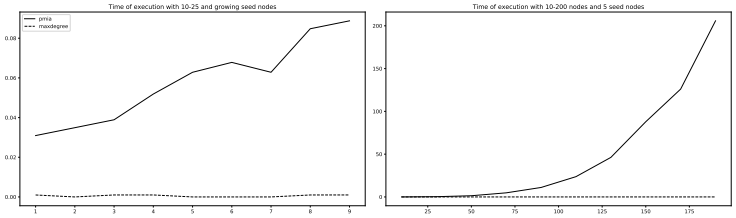


Figure 4: Comparison of execution time in seconds for PMIA and the weighted degree heuristic. First plot shows the execution time with a fixed set of nodes and an increasing number of seeds. Second plot shows the execution time for 5 seeds and a graph of increasing size.

The weighted degree heuristic is frequently used for selecting seeds in influence maximization and Kempe, Kleinberg, and Tardos (2003)'s experimental results show it works quite well in practice compared to other heuristics. We started by replacing the PMIA algorithm by the Weighted Degree heuristic to try experiments with larger graphs.

Influence spread, although hard to compute exactly, can be estimated as a mean of evaluating performance of a IM algorithm by simulating the random process of the IC model. More specifically, the process of influence spread is simulated 500 time for each graph from the test set. Edge outcomes are pseudo-randomly determined every time by sampling from a Bernoulli random variable with parameter the transition probability of the edge. This method was for example used by Kempe,

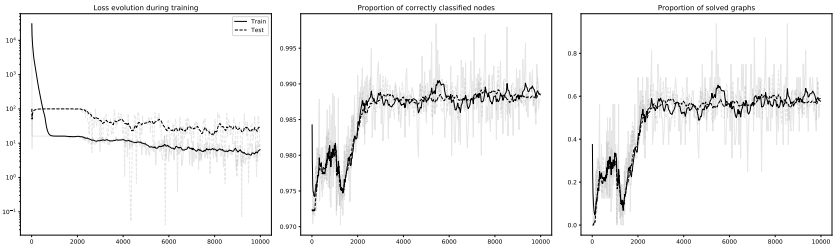
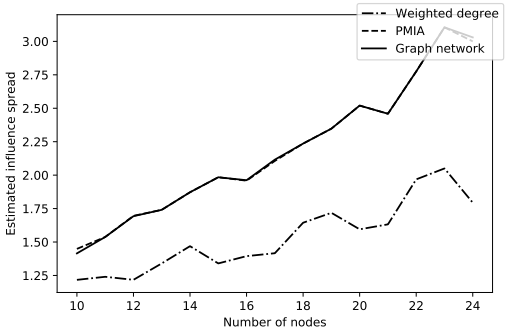
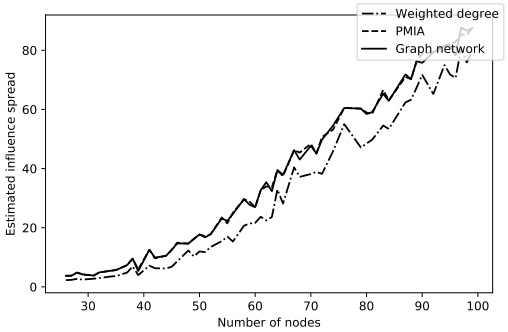


Figure 5: Loss, number of correctly predicted node and proportion of solved examples for 2 seed nodes, 10 message passing steps and between 50 and 100 nodes per graph using Weighted Degree heuristic.

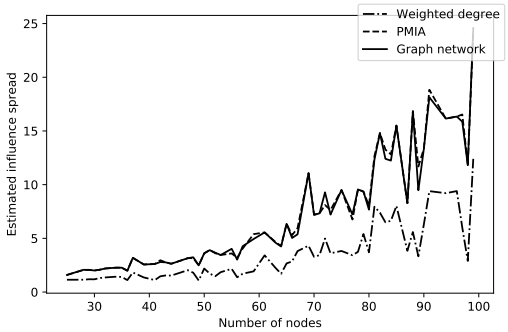
Kleinberg, and Tardos (2003) to estimate the efficiency of their algorithms. We use it here to estimate the performance of a graph network trained on a specific type of graphs to evaluate the ability of the network to generalize to data unobserved during training. This is done for a graph trained to estimate the maximum influence node of a graph, corresponding to the training curves of Figure 1a. We evaluate its performance on graph with similar rate of edges than the training set, with nodes in the range 10-25 and in the range 25-100. The experiments with larger will allow us to estimate how well the network generalizes to larger graphs when trained on graphs of a given size.



(a) Influence spread for 10-25 nodes (same distribution as training set)



(b) Influence spread for 25-100 nodes, same edge rate

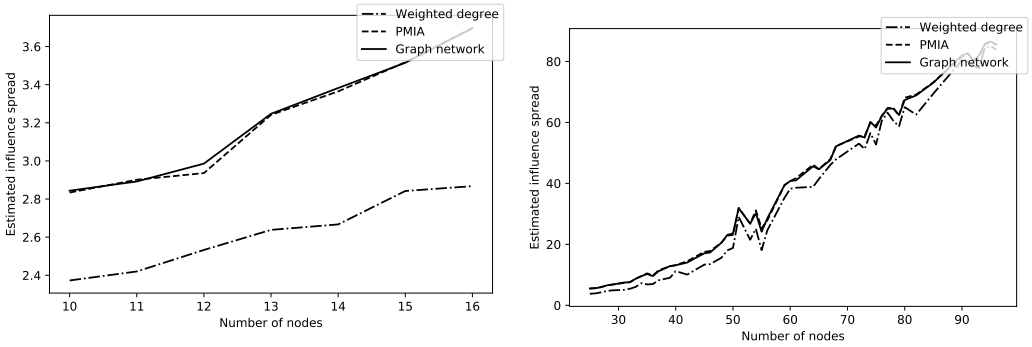


(c) Influence spread for 25-100 nodes, lower edge rate

Figure 6: Estimated influence spread for a graph trained to predict the maximum seed node and the other two heuristics.

Figure 6a shows the estimated influence spread as a function of the number of nodes for graphs of sizes that are within the range of the training set. The network is able to closely match the performances of the PMIA algorithm. This is expected since the network is able to predict very accurately the maximum influence node in the graph. Figure 6b shows the same setup with graphs bigger than the input graph (up to 4 times bigger). Figure 6c corresponds to a setup where graph were made a lot shallower than in the input distribution (rate (0.2, 0.3) against (0.5, 0.6) for the inputs, see

the first paragraph of section 4 for details about the rate). It also appears on Figure 7 that even on a task were the performance of the graph network was lower (60% graphs solved), the estimated influence spread is still comparable to the target heuristic of the graph.



(a) Influence spread for 10-25 nodes (same distribution as training set)

(b) Influence spread for 25-100 nodes, same edge rate

Figure 7: Estimated influence spread for a graph trained to predict the 2 maximal seed nodes and the other two heuristics.

It seems that the network was able to very accurately match and reproduce the performance of PMIA, even for data that was previously unseen and which is coming from a radically different distribution than the input one. This is very promising regarding graph networks, since it could mean that with a restricted training, a network is still able to achieve significant generalization. The generalization property observed here is similar to observations already made regarding the properties of graph networks for combinatorial optimization problems (Khalil et al., 2017; Kool, Hoof, and Welling, 2018) that can be explained in part by graph networks' entity- and relation-centric organization (P. W. Battaglia et al., 2018).

5 Conclusion

From the experiments presented here, it appears that the main limitation to using graph networks for solving the influence maximization problems lies in the fact that it is both time consuming and hard to generate labeled training examples that are of sufficiently high quality for letting the network learn efficiently. The main trade-off is between speed of label generation (number of training examples that can be created) and the gap between generated examples and the optimal solution.

The influence maximization problem is NP-Hard and the existing methods for computing solutions are either efficient but inaccurate, or accurate but hardly suitable for a large scale setting. However, as the experiments showed, graph networks seem to need both accurate and computationally efficient methods for training. Even if such a method existed, it wouldn't allow the network to learn more than the working solution it has been shown and wouldn't exceed the performance of the oracle. A way to cope with this problem could be to give the network solved and labeled graphs based on real-world data, but there doesn't exist any labeled dataset for this kind of tasks since the problem itself is NP-hard and influence spread cannot be reliably observed in a real-world network such as a social network.

However, as we've showed in our experiments, graph net seems to be capable of reaching the same performance as PMIA. We believe it may be less computationally expensive once it is trained, which could accelerate the IMlinUCB algorithm proposed by Wen et al. (2016) while still achieving the same results.These assumptions need computers with high processing power to be properly verified.

References

Barbieri, Nicola, Francesco Bonchi, and Giuseppe Manco (2013). "Topic-aware social influence propagation models". In: *Knowledge and Information Systems* 37.3, pp. 555–584. ISSN: 0219-3116. DOI: 10.1007/s10115-013-0646-6 (cit. on p. 4).

Battaglia, Peter W. et al. (2018). "Relational inductive biases, deep learning, and graph networks". In: *arXiv:1806.01261 [cs, stat]*. arXiv: 1806.01261 (cit. on pp. 3, 4, 8).

Battaglia, Peter et al. (2016). "Interaction Networks for Learning about Objects, Relations and Physics". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., pp. 4502–4510 (cit. on p. 1).

Chen, Wei, Chi Wang, and Yajun Wang (2010). "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*. the 16th ACM SIGKDD international conference. Washington, DC, USA: ACM Press, p. 1029. ISBN: 978-1-4503-0055-1. DOI: 10.1145/1835804.1835934 (cit. on pp. 1, 2, 4).

Goldenberg, Jacob, Barak Libai, and Eitan Muller (2001). "Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth". In: *Marketing Letters* 12.3, pp. 211–223. ISSN: 1573-059X. DOI: 10.1023/A:1011122126881 (cit. on p. 1).

Goyal, Amit, Francesco Bonchi, and Laks V.S. Lakshmanan (2010). "Learning Influence Probabilities in Social Networks". In: *Proceedings of the Third ACM International Conference on Web Search and Data Mining*. WSDM '10. New York, NY, USA: ACM, pp. 241–250. ISBN: 978-1-60558-889-6. DOI: 10.1145/1718487.1718518 (cit. on p. 4).

Hamilton, William L., Rex Ying, and Jure Leskovec (2017). "Representation Learning on Graphs: Methods and Applications". In: *arXiv:1709.05584 [cs]*. arXiv: 1709.05584 (cit. on p. 1).

Kempe, David, Jon Kleinberg, and Éva Tardos (2003). "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*. the ninth ACM SIGKDD international conference. Washington, D.C.: ACM Press, p. 137. ISBN: 978-1-58113-737-8. DOI: 10.1145/956750.956769 (cit. on pp. 1, 2, 6).

Khalil, Elias et al. (2017). "Learning Combinatorial Optimization Algorithms over Graphs". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 6348–6358 (cit. on p. 8).

Kipf, Thomas N. and Max Welling (2016). "Semi-Supervised Classification with Graph Convolutional Networks". In: *arXiv:1609.02907 [cs, stat]*. arXiv: 1609.02907 (cit. on p. 1).

Kool, Wouter, Herke van Hoof, and Max Welling (2018). "Attention Solves Your TSP, Approximately". In: *arXiv:1803.08475 [cs, stat]*. arXiv: 1803.08475 (cit. on p. 8).

Wen, Zheng et al. (2016). "Online Influence Maximization under Independent Cascade Model with Semi-Bandit Feedback". In: *arXiv:1605.06593 [cs, math, stat]*. arXiv: 1605.06593 (cit. on pp. 3, 4, 8).

Xu, Keyulu et al. (2018). "Representation Learning on Graphs with Jumping Knowledge Networks". In: *arXiv:1806.03536 [cs, stat]*. arXiv: 1806.03536 (cit. on p. 1).

Appendices

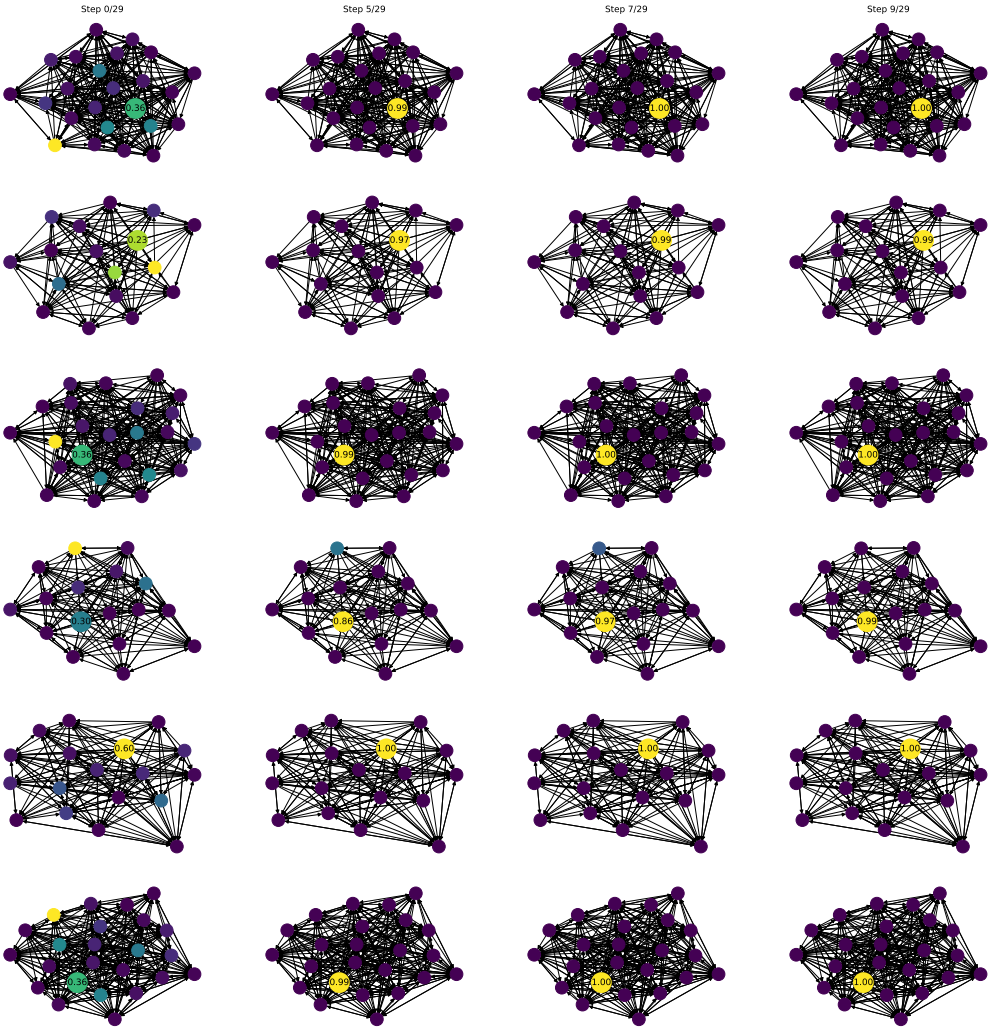


Figure 8: 1 seed

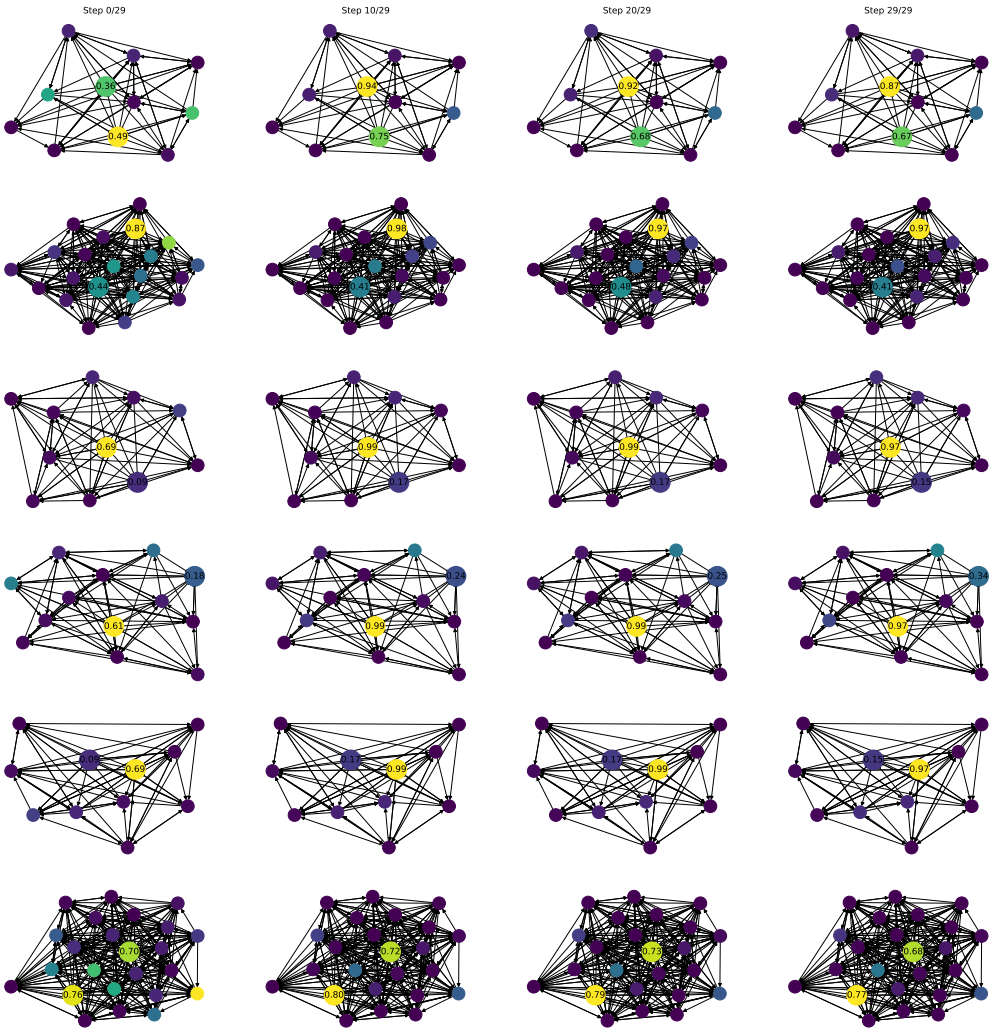


Figure 9: 2 seeds