Contingency-Aware Influence Maximization: A Reinforcement Learning Approach

Haipeng Chen[1] **Wei Qiu**[2] **Han-Ching Ou**[1] **Bo An**[2] **Milind Tambe**[1]

[1]Center for Research on Computation and Society, Harvard University
[2]School of Computer Science and Engineering, Nanyang Technological University

Abstract

The influence maximization (IM) problem aims at finding a subset of seed nodes in a social network that maximize the spread of influence. In this study, we focus on a sub-class of IM problems, where whether the nodes are willing to be the seeds when being invited is uncertain, called *contingency-aware IM*. Such contingency aware IM is critical for applications for non-profit organizations in low resource communities (e.g., spreading awareness of disease prevention). Despite the initial success, a major practical obstacle in promoting the solutions to more communities is the tremendous runtime of the greedy algorithms and the lack of high performance computing (HPC) for the non-profits in the field – whenever there is a new social network, the non-profits usually do not have the HPCs to recalculate the solutions. Motivated by this and inspired by the line of works that use reinforcement learning (RL) to address combinatorial optimization on graphs, we formalize the problem as a Markov Decision Process (MDP), and use RL to learn an IM policy over historically seen networks, and generalize to unseen networks with negligible runtime at test phase. To fully exploit the properties of our targeted problem, we propose two technical innovations that improve the existing methods, including state-abstraction and theoretically grounded reward shaping. Empirical results show that our method achieves influence as high as the state-of-the-art methods for contingency-aware IM, while having negligible runtime at test phase.

1 INTRODUCTION

Influence maximization is the problem of finding a subset of seed nodes in a social network that maximize the spread of influence. Originally derived from the viral marketing domain, the majority of IM algorithms [Kempe et al., 2003, Leskovec et al., 2007, Borgs et al., 2014, Tang et al., 2015] focus on settings where nodes are always willing to be the seeds, which may not be the case in many real-world scenarios. For example, recent work [Yadav et al., 2016, 2018, Wilder et al., 2018] provides large-scale applications of IM in public health. More specifically, they use IM algorithms to help spread the awareness of HIV prevention among homeless youth, where the youth leaders when being invited to be the "seed" nodes, may have difficulty joining and thus deviate from the intervention plan. In this sub-class of IM problems called *contingency-aware influence maximization* [Yadav et al., 2018], when a node is invited to become a seed node, there is *uncertainty* in whether it is willing to accept the invitation.

This contigency-aware IM problem has been addressed using Partially observable MDP (POMDP) [Yadav et al., 2016, 2018] and greedy algorithms [Wilder et al., 2018]. Despite their success in the field [Wilder et al., 2021], there is a major limitation in transitioning the solution to more homeless youth shelters and cities – whenever the underlying social network changes, the solution to the IM problem needs to be recomputed, whereas the stakeholders usually do not have the HPCs to perform the computation on their own. Figure 1 shows the runtime of the IM component for the state-of-the-art CHANGE algorithm [Wilder et al., 2018, 2021] on a network with 1000 nodes and 4974 edges when the number of seeds increases (run on a single Intel(R) Core(TM) i9-9820X CPU @ 3.30GHz core, with the sampling frequency of CHANGE set to 50, the influence propagation probability being set to 0.2). CHANGE runs 10 hours even with just 20 seed nodes, presenting a great burden to the low-resource non-profits such as homeless shelters, particularly as they scale-up these applications. In fact, the low-resource computing issue exists in many other works on social network intervention [Srivastava et al., 2019, Rice et al., 2020, Awasthi et al., 2020, Petering et al., 2021] and deploying AI techniques to the public sector in general [Mehr et al., 2017,

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

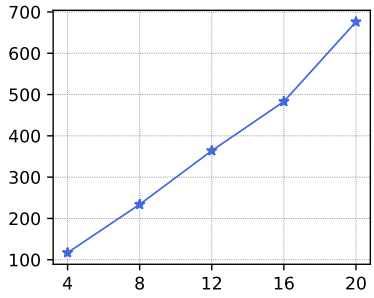


Figure 1: Runtime of CHANGE (in mins), where x-axis is the number of seed nodes.

Mikhaylov et al., 2018, Guo and Li, 2018], especially when low-resource non-profits are the decision makers at stake.

Recently, there have been studies that use RL to learn a generalized policy for a certain combinatorial optimization problems on graphs [Khalil et al., 2017, Nazari et al., 2018, Deudon et al., 2018, Bengio et al., 2020]. The key idea is to decompose the selection of nodes into a sequence, and learn a heuristic policy that selects nodes sequentially. The RL policy is usually trained on a set of seen training graphs, in the hope that it generalizes to unseen test graphs of similar characteristics. To better generalize the trained policy across different graphs, graph embedding techniques, such as Structure to Vector (S2V) [Dai et al., 2016] and Graph Convolutional Networks (GCNs) [Kipf and Welling, 2016] are integrated as part of the RL value functions to extract the graph structure information. Primarily proposed to solve relatively simple problems such as the traveling salesman problem (TSP) and the maximum vertex cover (MVC) problem, recent works [Li et al., 2019, Manchanda et al., 2020, Tian et al., 2020] extend it to the IM problem without considering node uncertainty. Inspired by these works, we propose to address the contingency-aware IM problem using RL.

There are however new challenges in designing an effective RL algorithm for contigency aware IM. First, in previous RL for IM methods, the state (as well as state transition) of the MDP is the nodes that are previously selected, which is deterministic. In our problem, the willingness status of nodes selected before the current step are unknown. When formulating an MDP model, there remains a question of how to define a state that well incorporates the uncertainty information. Second, in these previous works, the immediate reward is set as the marginal contribution of a new node selected at the current time step. This cannot be simply applied in our problem because of the uncertainty in node status. Moreover, it introduces an extremely high variance in the marginal contribution of a new node, and thus renders the RL training much more challenging.

To address the challenges, we propose a new MDP formulation to the underlying problem. Though this formulation preserves the Markovian property, its state is highly sparse and thus makes it hard for RL to learn efficiently. We make the first technical innovation with a state-abstraction component for RL, which compresses the states in a more compact manner, while preserving the uncertainty information. To address the high variance in reward function, we make our second technical innovation by using a novel reward-shaping technique. The reward-shaping component exploits two unique properties in the problem: 1) The probability of a generic node willing to be a seed or not is usually known, which can be learned from historical data; 2) The influence function is submodular. We first use the node willingness probability to express the exact "expected" reward, which we show is computationally infeasible. Using the submodularity property, we then design a surrogate reward function in place of the exact expected reward, with provable worst-case guarantee compared to the exact expected reward.

Summary of contributions. 1) We are the first to address the contingency-aware IM problem using an RL approach. We propose a new MDP formulation for this problem. 2) Our technical contribution is a new RL algorithm that is built upon the line of works that use RL to address combinatorial optimization problems on graphs, while making non-trivial, theoretically grounded adaptations that exploit the problem properties. 3) We conduct extensive experimental evaluations and show that under various settings, RL can perform as good as state-of-the-art greedy IM algorithms from the HIV prevention domain. Ablation study results demonstrate the effectiveness for each of the two novel components. Our code can be found via `https://github.com/Haipeng-Chen/RL4IM-Contingency`.

2 RELATED WORK

Influence maximization The IM problem is first studied by Domingos and Richardson [2001] as an algorithmic problem. Kempe et al. [2003] formulate it as a discrete optimization problem over the graphs, and propose a greedy algorithm to solve the problem, which has a guarantee of $1 - 1/e$. Cost-Effective Lazy Forward (CELF) [Leskovec et al., 2007], Reverse Influence Sampling (RIS) [Borgs et al., 2014] and Influence Maximization via Martingales (IMM) improve the greedy algorithm by more efficient spread estimation techniques. Golovin and Krause [2011] extend the IM problem to the *adaptive* setting, where seed selection is adapted based on observing the influence spread of previously selected nodes. More efficient methods [Han et al., 2018, Sun et al., 2018, Huang et al., 2020] are proposed later on to solve the adaptive IM problem. Adaptive IM differs from our setting in that our focus is to address the *uncertainty* in a node's willingness to participate, which they do not address. Moreover, they observe the influence of the previously selected nodes and select new nodes based on the observation, whereas we do not observe such intermediate influence. Yadav et al. [2018] introduce contigency-aware influence maximization in the context of HIV prevention among homeless youth and solve the challenge using a

POMDP; this solution does not scale beyond very small number of influencers. To remedy this shortcoming, Wilder et al. [2018] develop greedy IM algorithms for contingency aware influence maximization in the field, which is deployed in field test [Wilder et al., 2021]. Different from their work, we introduce learning techniques to address the problem.

ML/RL for combinatorial optimization on Graphs Vinyals et al. [2015], Bello et al. [2016], Graves et al. [2016] make early attempts in using ML/RL to address combinatorial optimization problems on graphs, where they decompose the original combinatorial action into a sequence of individual actions, and propose learning frameworks to learn heuristics for the problems. These approaches do not generalize well among unseen graphs, or are data-inefficient. Khalil et al. [2017] propose to use graph embedding techniques as the value approximator for the Deep Q-Networks (DQN) [Mnih et al., 2013], and therefore their approach generalizes better for graphs out of distribution. Kool et al. [2018] propose an approach that combines attention-based function approximators with policy gradient methods [Williams, 1992]. Li et al. [2018] approximate the solution quality with GCNs [Kipf and Welling, 2016], and use a learning framework based on guided tree search. Joshi et al. [2019] address the problem using a combination of GCNs and beam search. Qiu et al. [2019] combine RL and GCNs to address the road tolling problem in a transportation network. Ou et al. [2021] adapt the idea to address recurrent disease prevention on a social network. Mao et al. [2019] use it to address the scheduling problem in data processing clusters. We refer to Bengio et al. [2020] for a detailed survey on this line of works.

ML/RL for IM Lin et al. [2015], Ali et al. [2018] use RL to do influence maximizaiton in a competitive setting. They do not consider generalization, and the policy is to choose which high-level greedy algorithm to use. Kamarthi et al. [2020] apply RL to explore an unknown graph in the context of influence maximization. This work is different from ours as they use RL to explore the graph structure instead of selecting seeds. Ko et al. [2020] propose an inductive ML approach to estimate the influence spread of unseen networks. Li et al. [2019], Tian et al. [2020], Manchanda et al. [2020] extend the method in [Khalil et al., 2017] to address the IM problem, where reward of a new node is defined as its marginal contribution. Manchanda et al. [2020] aim at solving problem instances with millions of nodes. It uses supervised learning as a preliminary step to predict the individual quality of a node, which introduces large extra computational overhead and effort of hand-crafting the learning pipeline. Because of this, it does not scale to large number of training graphs. Moreover, all these methods do not consider the uncertainty of a node's willingness to be seed, and thus fail to address the challenges that are discussed previously. We will show empirically that directly applying their methods leads to sub-optimal performance.

3 CONTINGENCY-AWARE IM

Our work is motivated by previous works [Yadav et al., 2016, 2018, Wilder et al., 2018] which use influence maximization to spread the awareness of HIV prevention among the homeless youth. An HIV awareness intervention is a day-long class followed by weekly hour-long meetings. Due to limited resources, only a subset of youth will be selected to attend the classes. The trained youth will then act as peer-leaders who further spread the awareness of HIV prevention among the youth social network. An important observation is that when being invited to the training sessions, whether the youth is willing to be present at the classes is unknown until the end of the intervention round. Despite the initial success, a practical challenge that prevents the transitioning of their methods to more homeless youth shelters and cities is the lack of HPC resources for the non-profits in the field. Once the social network changes, the algorithm has to be rerun, without reusing knowledge about historical data. In fact, the *low-resource computing* scenario is ubiquitous in real life, especially for low-resource non-profits which are in urgent need of help from the AI community. We provide a new learning-based perspective of addressing the problem.

3.1 PROBLEM SETUP

Influence spread model We consider a social network $G = (V, E)$ where V and E are respectively the nodes and edges. Each node is either *activated*, meaning the node is influenced, or *inactivated* otherwise. We assume all nodes are initially inactivated unless chosen as the seed node. Two nodes that are connected by an edge $e \in E$ has a probability of influencing each other. We model the influence spread using the prominent Independent Cascade (IC) model [Goldenberg et al., 2001, Kempe et al., 2003]. That is, for each node $v \in V$ that is activated at a certain time, it has a *single* chance of activating its neighbors at the next time, with a probability p. Given a seed node set $S \subseteq V$, the influence spread in the graph G is represented as $I(G, S)$. The *influence maximization* problem aims at finding an optimal set of (usually budget-constrained) seed nodes $S^* \subseteq G$, such that the influence spread is maximized.

Seed selection and node uncertainty As motivated by the multi-round seed selection in the HIV prevention domain, we consider seed selection as multiple rounds $t = 1 \dots T$ of seed nodes selection, where each round selects a mini-batch of B nodes.[1] At each round t, the set of selected seed nodes is represented as S_t. As discussed before, when being selected, each node $v \in S_t$ may not necessarily be willing to act as a seed node. To capture this uncertainty, we denote the

[1]Note that our model is not limited to the multi-round setting, but is a more generalized model that can tackle both single round and multi-round seed selections. We will show empirically that our model and algorithm work well on single round node selection.

probability of a node willing to be seed (when selected as seed node) as q.[2] We assume that this probability is known *a priori*, which can be estimated by using statistics of historical data. The realization of the willingness status of the selected nodes can be observed at the end of each intervention round t. Naturally, the set of nodes O_t who are willing to be seeds at round t is a subset of S_t: $O_t \subseteq S_t$. At the end of t, the history of selection and willingness status of all seeds is denoted as a sequence $H_t = ((S_1, O_1) \dots (S_t, O_t))$.

3.2 MDP FORMULATION

Due to the sequential planning essence of the multi-round IM problem, we formulate it as a discrete time MDP.

Time step A natural way of defining a time step is to treat each intervention round t as a time step. However, in doing so, the action of each time step still consists of B nodes, and thus selecting the optimal action in each round t is still a combinatorial optimization problem with a combination of choices of size $\binom{|V|}{B}$. To avoid this, we define the time step as selecting each individual node. To distinguish the two concepts, we will call each intervention round as a *main step* t, and the selection of each individual node within each main step as a *sub-step* (t, b). We have $t = 1 \dots T$, and $b = 1 \dots B$. The time horizon is thus $T \times B$.

State To fully capture the information of the status of a current sub-step (t, b), we use a binary matrix $X_{t,b} \in \{0, 1\}^{3 \times |V|}$, together with the adjacency matrix G to represent the state $(G, X_{t,b})$.[3] Note that G is fixed over time. Thus we will just use $X_{t,b}$ to refer to the state. Each column $X^v_{t,b}$ of $X_{t,b}$ denotes the status of one node v. In the initial state, $X_{t,b}$ is initialized as all zeros. As the sequence of decision goes, the first element $X^{1,v}_{t,b} = 1$ indicates node v is selected as a seed node and is willing to be seed. The second element $X^{2,v}_{t,b} = 1$ means node v is selected as a seed node and is unwilling to be seed. The third element $X^{3,v}_{t,b} = 1$ means node v is selected at a main step t but the main step is not ended, so that its willingness status remains unknown. In this way, we can compress the history H_t of node selection and realization of nodes' willingness status using a matrix form. Given the Markovian property, the status of the current time step does not depend on the sequence. Moreover, it considers the uncertainty in nodes' willingness status within the current main step. Thus, the state representation does not lose information about the state.

Action There are two types of actions. We define a *sub action* as the selection of a single node at each sub-step (t, b). It is denoted as a one-hot vector $a_{t,b} \in \{0, 1\}^{|V|}$, where there is only one element in $a_{t,b}$ that corresponds to the node being selected, i.e., $\sum_{v=1}^{|V|} a^v_{t,b} = 1$. Correspondingly, a *main action* A_t is defined as the aggregation of all the sub actions in this main step at the end of each main step t:

$$A_t = \sum\nolimits_{b=1}^{B} a_{t,b},\ \forall t = 1 \dots T \tag{1}$$

A_t can be seen as the vector form representation of S_t. Note that $\sum_{v=1}^{|V|} A^v_t = B$.

State transition We omit the description of G as it is fixed over time. Apart from that, there are two types of state transitions. The first type happens at the end of each sub-step (t, b), i.e., whenever a new node is selected, there is:

$$X^3_{t,b+1} = X^3_{t,b} + a_{t,b},\ \forall b = 1 \dots B - 1,\ t = 1 \dots T \tag{2}$$

We can see that this type of state transition is deterministic.

The second type of sate transition happens only at the end of each main step t. It reveals the realization of the willingness status of the selected nodes at the main step. This type of state transition is stochastic, and directly depends on the probability q. To formalize it, we first define $\bar{A}_t \in \{0, 1\}^{|V|}$ as the realization of main action A_t, which can be seen as the vector form of O_t. The v-th element $\bar{A}^v_t = 1$ means that node v is invited and willing to be a seed. It is naturally constrained that for each $v \in V : \bar{A}^v_t \leq A^v_t$. Let a scalar $\bar{B}_t \coloneqq \sum_{v=1}^{|V|} A^v_t$. It means the number of nodes which are willing to be seeds when being selected at main step t. Given the above, we can derive the three dimensions of state $X_{t+1,1}$ of the next time step $(t + 1, b = 1)$ as

$$\begin{aligned} &X^1_{t+1,b=1} = X^1_{t,B} + \bar{A}_t, \\ &X^2_{t+1,b=1} = X^2_{t,B} + A_t - \bar{A}_t,\ \ X^3_{t+1,b=1} = 0 \end{aligned} \tag{3}$$

The probability of this transition is:

$$P(X_{t+1,1}, A_t \to \bar{A}_t | X_{t,B}, A_t) = q^{\bar{B}_t}(1 - q)^{B - \bar{B}_t} \tag{4}$$

Reward The total reward is defined as the total influence that is achieved within the social network G, given the selection of nodes that is represented as $X_{T,B}$. We denote the total accumulated reward as $r(G, X_{T,B})$. This incurs the issue known as reward sparseness, which makes it challenging for RL to learn efficiently. To mitigate this issue, Li et al. [2019], Tian et al. [2020], Manchanda et al. [2020] use the marginal contribution of a new node as the *immediate reward*. Denote the set of seed nodes selected before as S, the *marginal contribution* of a new node v is defined as as $\Delta I(G, S, v) \coloneqq I(G, S \cup \{v\}) - I(G, S)$. This cannot be directly applied to our problem due to the node willingness uncertainty within each intervention round t. For now we denote the immediate reward as a generic notation $r(G, X_{t,b}, a_{t,b})$. We will revisit this issue in Section 4.3 when we introduce our proposed reward shaping technique.

[2] Similar to Yadav et al. [2018], we assume a same q value for all the nodes in this model due to the practical challenge of knowing the exact q value for each node. In Yadav et al. [2018], it is done by using statistics on the historical attendance rate of youths when being invited.

[3] With a bit abuse of notation, we use G to represent both a graph and its adjacency matrix.

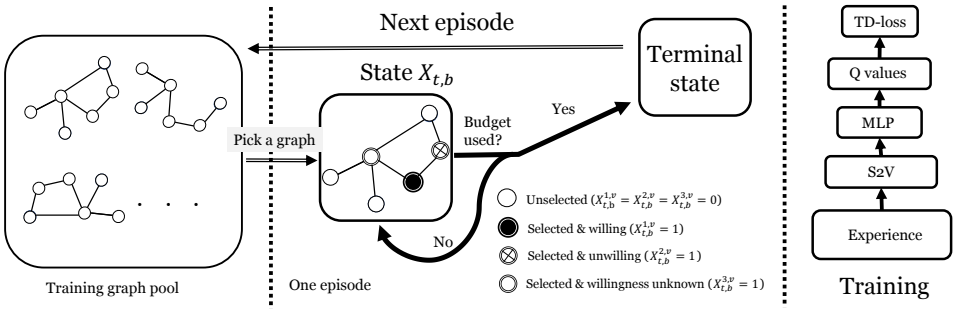


Figure 2: RL4IM training procedure. The graphs on the left are the set of training graphs $\mathcal{G}$. The process starts by randomly selecting a graph $g \in \mathcal{G}$. The sampled graph constructs a new environment. The RL4IM agent then interacts with it. At each time step, it observes the state from the environment, and selects the next node (based on its Q-function) if the budget is not used, or otherwise reaches the terminal state. It then selects the next graph and the training iterates. Meanwhile, the trajectory data are fed into the replay buffer, which is then used to compute the TD loss for updating the Q-function (on the right).

4 RL4IM

Inspired by [Khalil et al., 2017, Nazari et al., 2018, Deudon et al., 2018, Bengio et al., 2020] that use RL and graph embedding to address combinatorial optimization problem on graphs, we design Reinforcement Learning for Influence Maximization (RL4IM), a new RL-based algorithm that addresses the contingency-aware IM problem. RL4IM exploits two significant properties in the underlying problem: 1) The influence function is submodular; 2) The state transition probability, i.e., the probability q of a node willing to be a seed is known *a priori*, which can be estimated using historical data.

4.1 RL4IM ARCHITECTURE

Figure 2 shows the overall architecture of RL4IM. The graphs on the left are the set of training graphs $\mathcal{G}$. The process starts by randomly selecting a graph g from the set of training graphs. Each sampled training graph constructs an *environment*, which defines a new MDP as described in Section 3.2. Given the environment, the RL4IM agent then interacts with it in discrete time steps. At each time step (t, b), it observes the *state* $X_{t,b}$ from the environment, and decides whether the budget of selecting the seed nodes (i.e., $T \times B$) is spent. If not, the RL4IM agent will determine the next seed to select based on its learned *policy* $\pi(a_{t,b}|X_{t,b})$, which is a probability distribution over the feasible action space given the current state $X_{t,b}$ that trades off *exploiting* a node with an estimated high reward and *exploring* nodes that could potentially have higher reward. If the budget is spent, then it reaches the *terminal state* for this *episode*. The RL4IM agent will then select the next graph and the training procedure iterates until its policy reaches convergence.

In Q-learning [Watkins, 1989], the value of a node/action is measured using the *Q-function*. The Q-function is usually estimated using the Bellman equation: $Q(X_{t,b}, a_{t,b}) = r(X_{t,b}, a_{t,b}) + \gamma \arg\max_{a_{t',b'}} Q(X_{t',b'}, a_{t',b'})$, where γ is the discount factor. DQN [Mnih et al., 2013, 2015] improves vanilla Q-learning by using deep neural networks as the *function approximator*, along with other techniques like experience replay, which stores the historical training trajectories in a *replay buffer* $\mathcal{M}$ and updates the Q-function by minimizing the loss function $(\hat{Q} - Q)^2$ with batch data from the replay buffer using gradient descent.

We follow Khalil et al. [2017], Li et al. [2018] and generalize the learned policies to unseen test graphs using graph/node embedding techniques [Dai et al., 2016, Kipf and Welling, 2016] as the *function approximator*. Essentially, graph/node embedding takes input an attributed matrix (the state $X_{t,b}$ and the action $a_{t,b}$ in our case) as well as the adjacency matrix G and maps it to an embedding space. It aggregates the neighborhood information from the adjacency matrix. We omit their explicit form, and represent them with a generic form as $f(X_{t,b}, G)$ and $g(a_{t,b}, G)$. In this way, the Q-function is represented as: $Q(X_{t,b}, a_{t,b}) = MLP(f(X_{t,b}, G), g(a_{t,b}, G))$, where $MLP(\cdot)$ means multi-layer perceptron. Alg. 1 shows the pseudo-code of RL4IM's training process, where the key novel components are state-abstraction and reward shaping.

4.2 STATE ABSTRACTION

We use a $3 \times |V|$ matrix to capture all the information of the current state. Because only a small subset of nodes are selected as seed nodes, the number of 1's is bounded by the total budget $T \times B$. Moreover, the 3rd dimension contains the status of nodes at only one intervention round, and is bounded by the budget B at each intervention round. Therefore, the state matrix is extremely sparse and makes learning rather inefficient. To address this issue, instead of assuming that we do not know about the state-transition model – as typical model-free RL methods do – we exploit the fact that the transition model is actually known. That is,

Algorithm 1: RL4IM training

Initialize replay buffer $\mathcal{M}$, Q-function $Q_\theta(\bar{X}_{t,b}, a_{t,b})$
for *episode: 1 to #episodes* **do**
  Draw a graph $G \in \mathcal{G}$
  Get initial state $X_{1,1} = 0$
  for $t = 1 \dots T$ **do**
    for $b = 1 \dots B$ **do**
      Get abstracted state $\bar{X}_{t,b} \leftarrow X^1_{t,b} + qX^3_{t,b}$
      Get action from policy $a_{t,b} \leftarrow \arg\max Q_\theta(a_{t,b}|\bar{X}_{t,b})$ with probability $1-\varepsilon$ or otherwise random
      Play $a_{t,b}$, get surrogate reward $\tilde{r}(X_{t,b}, a_{t,b})$ with Eq.(11)
      if *b<B* **then** $X_{t',b'} \leftarrow X_{t,b+1}$ with Eq.(2);
      else $X_{t',b'} \leftarrow X_{t+1,1}$ with Eqs.(3)-(4);
      Add new memory to replay buffer: $\mathcal{M} = \mathcal{M} \cup (\bar{X}_{t,b}, a_{t,b}, \tilde{r}(X_{t,b}, a_{t,b}), X_{t',b'})$
      Update θ using sampled memories from $\mathcal{M}$
return $Q_\theta(\bar{X}_{t,b}, a_{t,b})$

we know the probability q that a node is willing to be seed, which can usually be learned from historical data.[4] More specifically, we use a more compact vector $\bar{X}_{t,b} \in \mathcal{R}^{1\times|V|}$, that performs a state abstraction to $X_{t,b}$:

$$\bar{X}_{t,b} = X^1_{t,b} + qX^3_{t,b} \tag{5}$$

By multiplying the 3rd dimension with the probability q, the intuition is to use this prior knowledge to better reflect the "expected" contribution of the corresponding node. Note that in the abstracted state, the information about the nodes that are selected but are unwilling to be seeds are not tracked. To keep track of this information, we maintain a feasible action set that is updated at each time step. The set is initialized as the entire set of nodes of the graph. In every time step, whenever a node is selected, it will be removed from the set so that it is no longer feasible in future time steps.

4.3 REWARD SHAPING

As discussed in Section 3.2, to mitigate the reward sparseness issue, existing works [Li et al., 2019, Tian et al., 2020, Manchanda et al., 2020] use the marginal contribution of a newly selected node as the immediate reward at the current time step. However, this is infeasible in our problem when there is uncertainty about the nodes' willingness status in each main step. A straightforward way of handling it is just to assume that all the nodes in the current main step are willing to be seeds, and calculate the marginal contribution with respect to all these nodes. However, due to submodularity of the influence function, this incurs underestimation of a new node's marginal contribution. As discussed previously, we have prior knowledge about the node willingness status transition probability q. We then use it to explicitly represent the expected marginal contribution of a new node. Recall that $\bar{A}_t$ denotes the realization of the main step action A_t. Let $\bar{A}_{t,b}$ be the realization of sub-steps $a_{t,1}$ to $a_{t,b-1}$ (or equivalently $X^3_{t,b}$, as $X^3_{t,b} = a_{t,1} + \ldots + a_{t,b}$), then $\bar{B}_{t,b} \coloneqq \sum_{v=1}^{|V|} \bar{A}_{t,b}$ means the number of nodes that are selected from $(t,1)$ to (t,b) and are willing to be seeds. Thus, the explicit form of expected marginal contribution $r(X_{t,b}, a_{t,b})$ of action $a_{t,b}$ is:

$$\sum\nolimits_{\beta=0}^{b-1} q^\beta (1-q)^{b-1-\beta} \sum\nolimits_{X^3_{t,b}} \delta I(G, X_{t,b}, a_{t,b}) \Big| \bar{B}_{t,b} = \beta, \tag{6}$$

where $\delta I(G, X_{t,b}, a_{t,b}) = I(G, X_{t,b}, a_{t,b}) - I(G, X_{t,b})$ is the marginal contribution of action $a_{t,b}$ given the current state $X_{t,b}$, and $\sum_{X^3_{t,b}} \delta I(G, X_{t,b}, a_{t,b})$ is the sum of marginal contribution of action $a_{t,b}$ over all possible values of the state's 3rd dimension $X^3_{t,b}$. The condition $\bar{B}_{t,b} = \beta$ specifies that the number of nodes that are willing to be seeds in the realization of $X^3_{t,b}$. We can see that there are $\binom{b}{\beta}$ such terms in the summation.

In practice, the exact influence values $I(G, X_{t,b}, a_{t,b})$ and $I(G, X_{t,b})$ are not known, and must be estimated by running multiple influence spread simulations over the graph G. Therefore, it becomes computationally infeasible to enumerate all the possible combinations of $X^3_{t,b}$ at each sub-step, especially when the budget B at each main step is large. This becomes a major obstacle for RL, as it usually requires a large number of training samples (time steps) to learn.

To overcome this obstacle, we notice that the influence function $I(G, S)$ is usually *submodular* [Kempe et al., 2003], meaning that the marginal contribution of a node v when being added to an existing set of nodes S, is no larger than that when it is added to subset $S' \subseteq S$, i.e.,

$$I(G, S\cup\{v\}) - I(G, S) \leq I(G, S'\cup\{v\}) - I(G, S') \tag{7}$$

At each time step (t,b), we use δI_0 to denote the marginal contribution of an action/node $a_{t,b}$ when no node selected from $(t,1)$ to $(t,b-1)$ is willing to be a seed, and use δI_{b-1} to denote the marginal contribution of an action $a_{t,b}$ when all of these nodes are willing to be seeds. That is, $\delta I_0 = I(G, X_{t,b}, a_{t,b}) - I(G, X_{t,b}) \Big| \bar{B}_{t,b} = 0, \delta I_{b-1} = I(G, X_{t,b}, a_{t,b}) - I(G, X_{t,b}) \Big| \bar{B}_{t,b} = b-1$. Following the submodularity property of the influence function, we have

Lemma 1 *At sub-step (t,b), the marginal contribution of any action is bounded by δI_0 and δI_{b-1}:*

$$\delta I_{b-1} \leq \delta I(G, X_{t,b}, a_{t,b}) \leq \delta I_0 \tag{8}$$

Proof 1 *According to the submorularity property of the influence function, i.e., for any subset $S' \subseteq S$,*

$$I(G, S \cup \{v\}) - I(G, S) \leq I(G, S' \cup \{v\}) - I(G, S')$$

[4] We do not know its realization till the main step ends, though.

Recall that O_t denotes the set of nodes that are willing to be seeds at round t, then $O_1 \cup \ldots \cup O_{t-1}$ denotes the set of nodes that are willing to be seeds before t. We denote $S^{t,b}$ as the set of nodes that are selected in t before b, and $O_{t,b}$ as the set of nodes that are willing to be seeds at round t before b. Because for any $O_{t,b}$, there is $\emptyset \subseteq O_t \subseteq S_{t,b}$. Therefore $O_1 \cup \ldots \cup O_{t-1} \subseteq O_1 \cup \ldots \cup O_{t-1} \cup O_{t,b} \subseteq O_1 \cup \ldots \cup O_{t-1} \cup S_{t,b}$. By definition we have Lemma 1.

Using this property, we design a surrogate marginal contribution function of $\delta I(G, X_{t,b}, a_{t,b})$, where we assume for any realization $\bar{A}_{t,b}$ of $X^3_{t,b}$, the marginal contribution of an action $a_{t,b}$ is the same when $\bar{B}_{t,b} = \beta$, i.e., for any two states $X_{t,b}$ and $X'_{t,b}$ and their corresponding $\bar{B}_{t,b}$ and $\bar{B}'_{t,b}$:

$$\begin{aligned} &\bar{B}_{t,b} = \bar{B}'_{t,b} \Rightarrow \\ &\delta I(G, X_{t,b}, a-t, b) = \delta I(G, X'_{t,b}, a-t, b) \end{aligned} \tag{9}$$

This assumption means the marginal contribution of a new node only depends on *how many* nodes are willing to be seeds, not *which*. We can then denote the marginal contribution as δI_β. We futher assume that $(\delta I_0 \ldots \delta I_\beta \ldots \delta_{b-1})$ is an arithmetic sequence of common difference, i.e.,

$$\delta I_\beta = \delta I_0 + \beta\Delta, \tag{10}$$

where $\Delta \coloneqq (\delta I_{b-1} - \delta I_0)/(b-1)$ is the common difference. Due to Lemma 1, $\Delta \leq 0$.

Theorem 1 *With the assumptions in Eqs.*(9)-(10)*, the surrogate marginal contribution of action $a_{t,b}$ in Eq.*(6) *is:*

$$\tilde{r}(X_{t,b}, a_{t,b}) = (1-q)\delta I_0 + q\delta I_{b-1} \tag{11}$$

Proof 2 *By substituting Eqs.*(9)-(10) *into Eq.*(6)*, we have*

$$\begin{aligned} \tilde{r}(X_{t,b}, a_{t,b}) &= \sum_{\beta=0}^{b-1} q^\beta (1-q)^{b-1-\beta} \binom{b-1}{\beta} \delta I_\beta \\ &= \sum_{\beta=0}^{b-1} q^\beta (1-q)^{b-1-\beta} \binom{b-1}{\beta} (\delta I_0 + \beta\Delta) \\ &= \sum_{\beta=0}^{b-1} q^\beta (1-q)^{b-1-\beta} \binom{b-1}{\beta} \delta I_0 + \\ &\quad \Delta \sum_{\beta=0}^{b-1} q^\beta (1-q)^{b-1-\beta} \binom{b-1}{\beta} \beta \\ &= \delta I_0 + \Delta(b-1)q \\ &= \delta I_0 + q(\delta I_{b-1} - \delta I_0) \\ &= (1-q)\delta I_0 + q\delta I_{b-1} \end{aligned}$$

The 4th equation holds because for arithmetic sequence with common difference, there is $\sum_{\beta=0}^{b-1} q^\beta (1-q)^{b-1-\beta} \binom{b-1}{\beta} = 1$, and $\sum_{\beta=0}^{b-1} q^\beta (1-q)^{b-1-\beta} \binom{b-1}{\beta} \beta = (b-1)q$.

Despite the simple form, we have the following two desirable properties of the surrogate reward function.

Theorem 2 *Using the surrogate marginal contribution in Eq.*(11)*, the computational complexity at each step (t, b) reduces from $\mathcal{O}(2^b)$ to $\mathcal{O}(1)$.*

Proof 3 *Because for each $\bar{B}_{t,b} = \beta$, we need to calculate the marginal contribution $\binom{b-1}{\beta}$ times, the total number of of calculations is then $2 \times \sum_{\beta=0}^{b-1} \binom{b-1}{\beta} = 2^b \sim \mathcal{O}(2^b)$. The number 2 at the LHS of the equation means calculating once for both the minuend and the subtrahend. On the other hand, calculating Eq.*(11) *requires only calculating $2 \times 2 = 4$ influence values, which is of order $\mathcal{O}(1)$ as it is a constant.*

Theorem 3 *The gap between the surrogate immediate reward in Eq.*(11) *and the original reward in Eq.*(6) *is bounded by* $\max\{(q-(1-q)^{b-1})(\delta I_0 - \delta I_{b-1}), (1-q-q^{b-1})(\delta I_0 - \delta I_{b-1})\}$.

Proof 4 *The worst case happens when 1) for all $b' < b-1$, there is $\delta I(G, X_{t,b}, a_{t,b'}) = \delta I_0$, or 2) for all $b' > 1$, there is $\delta I(G, X_{t,b}, a_{t,b'}) = \delta I_{b-1}$. In case 1), the gap between the exact expected marginal influence and our designed approximated one is:*

$$\begin{aligned} &\sum_{\beta=0}^{b-1} q^\beta (1-q)^{b-1-\beta} \binom{b-1}{\beta} [\delta I_0 - \delta I_\beta] \\ &- (1-q)^{b-1}\delta I_0 + (1-q)^{b-1}\delta I_{b-1} \\ &= \delta I_0 - (1-q)\delta I_0 - q\delta I_{b-1} - (1-q^{b-1}(\delta I_0 - \delta I_{b-1}) \\ &= (q-(1-q)^{b-1})(\delta I_0 - \delta I_{b-1}) \end{aligned}$$

In case 2), the gap is:

$$\begin{aligned} &\sum_{\beta=0}^{b-1} q^\beta (1-q)^{b-1-\beta} \binom{b-1}{\beta} [\delta I_\beta - \delta I_{b-1}] \\ &- q^{b-1}\delta I_0 + q^{b-1}\delta I_{b-1} \\ &= (1-q)\delta I_0 + q\delta I_{b-1} - \delta I_{b-1} - q^{b-1}(\delta I_0 - \delta I_{b-1}) \\ &= (1-q-q^{b-1})(\delta I_0 - \delta I_{b-1}) \end{aligned}$$

The bound is then $\max\{(q-(1-q)^{b-1})(\delta I_0 - \delta I_{b-1}), (1-q-q^{b-1})(\delta I_0 - \delta I_{b-1})\}$.

Theorem 2 is critical as it makes the calculation of expected reward in our setting feasible. Meanwhile, Theorem 3 provides a guarantee to the approximation. It is worth noting that though the bound could be arbitrarily bad when $q \to 0$ or $q \to 1$, in practice this is rare. Moreover, the worst cases described in the proof are very extreme cases. Empirical results show that even when $q = 0.2$ or 0.8, RL4IM still practically works well.

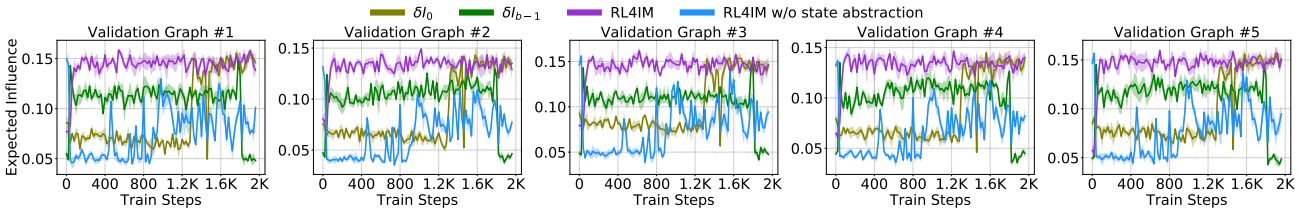


Figure 3: Validation curve on the 5 validation graphs during training for Q1. Shaded area indicates one standard deviation.

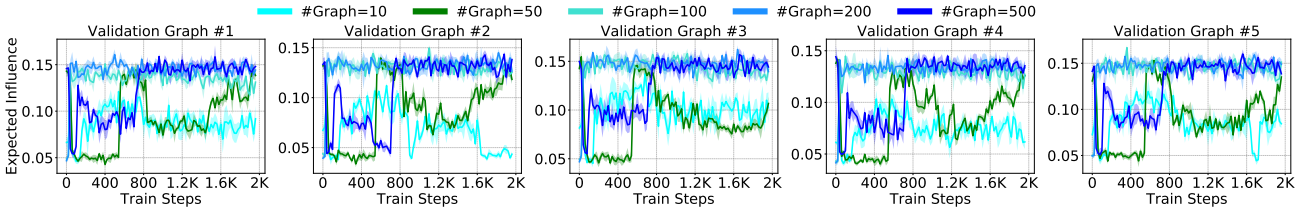


Figure 4: Validation curve on the 5 validation graphs for Q2. Shaded area indicates one standard deviation.

5 EXPERIMENT

5.1 EXPERIMENT SETTINGS

Environment To evaluate the performances of different methods, we generate synthetic graphs using the powerlaw graphs [Onnela et al., 2007], which is the Barabási–Albert (BA) growth model with an extra step that each random edge is followed by a chance of making an edge to one of its neighbors too. The average degree of a node is set to 3. The probability of adding a triangle after adding a random edge is set to 0.05. We will vary 1) the number of training graphs, 2) the willingness probability q, 3) intervention rounds T and the per-round budget B, 4) the graph sizes $|V|$, and evaluate different methods under these varied settings. The belief propagation probability is of IM is set to 0.1. To get an influence number, the IM simulator runs 100 times and returns an average. All experiments are run on a Dell DSS 8440 Cauldron node, with a virtual environment with 2 Intel Xeon Gold 6148 2.4G CPU cores, 5G RAM, 1 NVIDIA Tesla V100 32G GPU, EDR Infiniband.

Baselines The baselines include 1) a greedy algorithm that adaptively selects seeds based on observation of the willingness status of selected nodes in previous rounds. It is part of the CHANGE algorithm [Wilder et al., 2018] that is used in HIV prevention with node willingness uncertainty. 2) S2V-DQN-IM which is adapted from S2V-DQN Khalil et al. [2017] that combines RL with graph embedding. Note that this is the underlying architecture of recent works on RL for IM [Li et al., 2019, Tian et al., 2020]. We have added the state-abstraction component to it as we will show that the version without it barely converges well. The major difference between S2V-DQN-IM and RL4IM is it does not use our reward shaping technique, but estimates the reward using δI_{b-1}, i.e., assuming no uncertainty in a node's willingness. 3) Random which chooses nodes randomly.

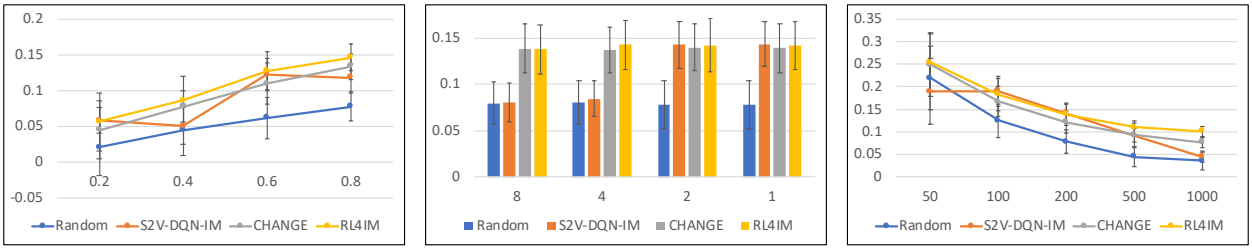
Evaluation setting In evaluation, we first generate a set of training graphs, and then generate another set of 5 held-out graphs that are used as validation set. The validation process will be activated approximately every 20 time steps (i.e., a checkpoint). During validation, it will run 20 episodes for each graph. The averaged reward over $20 \times 5 = 100$ runs will be used as the metric to select the best hyperparameters as well as the model at the best checkpoint. The model selected using the validation set will then be evaluated in the test phase. During testing, 10 graphs will be generated that are unseen either in training or validation graphs. Each method will be run 20 times on a graph, totalling $20 \times 10 = 200$ runs for one problem setting.

For both S2V-DQN-IM and RL4IM, the following parameters are set to the same: memory size is 4096, 2) batch size is 32, 3) maximal training time steps is set to 2000, 4) the discount factor is 0.99, 5) the q-networks is an S2V-based graph embedding layer followed by a 128-neuron MLP layer, 6) the optimizer is Adam [Kingma and Ba, 2015]. The other parameters, such as learning rate, exploration rate ε and its decay rate are optimized from the validation set.

5.2 RESULTS & DISCUSSIONS

The following values are set to default unless being evaluated: $|V| = 200$, $T = 2, B = 4$, #training graphs = 200; $q = 0.6$. We are interested in the following questions.

Q1: How does each new component of RL4IM affect the performance? Figure 3 shows the ablation study results. By removing either state-abstraction or our proposed reward shaping technique, the RL training becomes very unstable, or converges at a sub-optimal point. If we remove state-abstraction, then the state is very sparsely represented, and



(a) vary q (b) vary T (c) vary $|V|$

Figure 5: Performance of different methods on unseen test graphs. The x-axis is the value of the underlying setting, and the y-axis is the expected normalized influence (w.r.t. the number of nodes) averaged over $10 \times 20 = 200$ runs.

thus makes it hard for the graph embedding layer to effectively learn the optimal weights. On the other hand, by using either δI_0 or δI_{b-1} as the reward, it leads to either over or under-estimation (which is the practice of existing works on RL for IM [Li et al., 2019, Tian et al., 2020, Manchanda et al., 2020] that do not consider node uncertainty.

Q2: Does the number of training graphs affect training performance? To see whether and how the number of training graphs affect the performance, we show the validation curve during training, as in Figure 4. It shows that by increasing the number of training graphs, the validation curve tends to converge at a more stable point. This is because with larger training graph pool, the RL agent is exposed to more environments and thus tends to generalize better among unseen graphs. Based on this set of experiments, from now on all the experiments use 200 training graphs.

Q3: Does RL4IM work well for different uncertainty values q? Figure 5(a) shows that when q increases, i.e., when nodes are more likely to be seeds when being selected, the expected influence grows higher. This is intuitive as the expected number of seed nodes becomes larger w.r.t. larger q values. Both RL4IM and CHANGE are better and more stable across different q values, which are approximately twice the values of Random. The performance of S2V-DQN-IM is unstable as it uses a reward function that is far from the expected ground truth value.

Q4: What if we decrease T until 1? In this setting, we fix $T \times B = 8$, and evaluate $T = 8, 4, 2, 1$. Note that when $T = 1$, it is essentially a single round IM problem. Figure 5(b) shows the comparison results. This shows that RL4IM works well for single round IM problem as well. Similarly, CHANGE and RL4IM are the two best methods, while S2V-DQN-IM appears unstable at different settings.

Q5: How does performance of RL4IM vary across different graph sizes? In this set of experiments we vary graph sizes whtin [50, 100, 200, 500, 1000]. Figure 5(c) shows that the normalized influence value decreases when the number of nodes $|V|$ increases. This is because the influence budget is fixed as $2 \times 4 = 8$. When $|V|$ increases, the portion of nodes getting influence decreases. Similarly, CHANGE and RL4IM perform the best among all methods.

From questions 3-5, the key takeaway is that RL4IM works robustly across different settings. RL4IM is slightly better than CHANGE. This is potentially because RL4IM considers proactively the unwillingness of a node in its reward shaping component, whereas CHANGE only reactively adapts to realizations of the willingness status of nodes. Nonetheless, our main argument is that RL4IM, while performing as good as CHANGE, uses negligible runtime during test phase, and is therefore the better alternative for scenarios with low-resource computing. For example, once being trained, the policies are returned by RL within seconds even when the RL policy is run on a normal laptop.

6 CONCLUSION

We study the contingency-aware IM problem where a node's willingness to be a seed is uncertain. The state-of-the-art uses greedy algorithms to address the problem, but its slow run times are a barrier to transitioning this approach to low-resource settings as with non-profits serving marginalized populations. We propose a new learning-based perspective of solution to this problem using RL, so that it can output a seed selection strategy on a laptop within seconds at test time. Our major technical innovation is a theoretically grounded new algorithm, RL4IM, that exploits the properties of the underlying problem. Empirical results show that it matches the influence spread of the state-of-the-art, while having the advantage of negligible runtime during the test phase, a feature that is critical to low-resource non-profits. Our work is an example of RL for social good. We hope to shed some light to broader research areas in the low-resource computing paradigm.

Acknowledgements

This work was supported by the Army Research Office (MURI W911NF1810208). Chen was supported by the Center for Research on Computation and Society.

References

Khurshed Ali, Chih-Yu Wang, and Yi-Shin Chen. Boosting reinforcement learning in competitive influence maximization with transfer learning. In *WI*, pages 395–400, 2018.

Raghav Awasthi, Prachi Patel, Vineet Joshi, Shama Karkal, and Tavpritesh Sethi. Learning explainable interventions to mitigate hiv transmission in sex workers across five states in india. *arXiv preprint arXiv:2012.01930*, 2020.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 2020.

Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *SODA*, pages 946–957, 2014.

Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *ICML*, pages 2702–2711, 2016.

Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *CPAIOR*, pages 170–181, 2018.

Pedro Domingos and Matt Richardson. Mining the network value of customers. In *KDD*, pages 57–66, 2001.

Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3):211–223, 2001.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

Jonathan Guo and Bin Li. The application of medical artificial intelligence technology in rural areas of developing countries. *Health Equity*, 2(1):174–181, 2018.

Kai Han, Keke Huang, Xiaokui Xiao, Jing Tang, Aixin Sun, and Xueyan Tang. Efficient algorithms for adaptive influence maximization. *VLDB*, 11(9):1029–1040, 2018.

Keke Huang, Jing Tang, Kai Han, Xiaokui Xiao, Wei Chen, Aixin Sun, Xueyan Tang, and Andrew Lim. Efficient approximation algorithms for adaptive influence maximization. *The VLDB Journal*, pages 1–22, 2020.

Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

Harshavardhan Kamarthi, Priyesh Vijayan, Bryan Wilder, Balaraman Ravindran, and Milind Tambe. Influence maximization in unknown social networks: Learning policies for effective graph sampling. In *AAMAS*, pages 575–583, 2020.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *NeurIPS*, pages 6348–6358, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Jihoon Ko, Kyuhan Lee, Kijung Shin, and Noseong Park. Monstor: An inductive approach for estimating and maximizing influence over unseen social networks. In *ASONAM*, 2020.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *ICLR*, 2018.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

Hui Li, Mengting Xu, Sourav S Bhowmick, Changsheng Sun, Zhongyuan Jiang, and Jiangtao Cui. Disco: Influence maximization meets network embedding and deep learning. *arXiv preprint arXiv:1906.07378*, 2019.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *NeurIPS*, pages 539–548, 2018.

Su-Chen Lin, Shou-De Lin, and Ming-Syan Chen. A learning-based framework to handle multi-round multi-party influence maximization on social networks. In *KDD*, pages 695–704, 2015.

Sahil Manchanda, Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs. *NeurIPS*, 33, 2020.

Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. Learning scheduling algorithms for data processing clusters. In *SIGCOMM*, pages 270–288, 2019.

Hila Mehr, H Ash, and D Fellow. Artificial intelligence for citizen services and government. *Ash Cent. Democr. Gov. Innov. Harvard Kennedy Sch., no. August*, pages 1–12, 2017.

Slava Jankin Mikhaylov, Marc Esteve, and Averill Campion. Artificial intelligence for the public sector: opportunities and challenges of cross-sector collaboration. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2128):20170357, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Mohammadreza Nazari, Afshin Oroojlooy, Martin Takáč, and Lawrence V Snyder. Reinforcement learning for solving the vehicle routing problem. In *NeurIPS*, pages 9861–9871, 2018.

J-P Onnela, Jari Saramäki, Jorkki Hyvönen, György Szabó, David Lazer, Kimmo Kaski, János Kertész, and A-L Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332–7336, 2007.

Han-Ching Ou, Haipeng Chen, Shahin Jabbari, and Milind Tambe. Active screening for recurrent diseases: A reinforcement learning approach. In *AAMAS*, pages 992–1000, 2021.

Robin Petering, Nicholas Barr, Ajitesh Srivastava, Laura Onasch-Vera, Nicole Thompson, and Eric Rice. Examining impacts of a peer-based mindfulness and yoga intervention to reduce interpersonal violence among young adults experiencing homelessness. *Journal of the Society for Social Work and Research*, 12(1):000–000, 2021.

Wei Qiu, Haipeng Chen, and Bo An. Dynamic electronic toll collection via multi-agent deep reinforcement learning with edge-based graph convolutional networks. In *IJCAI*, pages 4568–4574, 2019.

Eric Rice, Laura Onasch-Vera, Graham Diguiseppi, Chyna Hill, Robin Petering, Nicole Wilson, Darlene Woo, Nicole Thompson, Milind Tambe, Bryan Wilder, et al. Using artificial intelligence to augment network-based, hiv prevention for youth experiencing homelessness. In *APHA's 2020 VIRTUAL Annual Meeting and Expo (Oct. 24-28)*. American Public Health Association, 2020.

Ajitesh Srivastava, Robin Petering, Nicholas Barr, Rajgopal Kannan, Eric Rice, and Viktor K Prasanna. Network-based intervention strategies to reduce violence among homeless. *Social Network Analysis and Mining*, 9(1): 1–12, 2019.

Lichao Sun, Weiran Huang, Philip S Yu, and Wei Chen. Multi-round influence maximization. In *KDD*, pages 2249–2258, 2018.

Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, pages 1539–1554, 2015.

Shan Tian, Songsong Mo, Liwei Wang, and Zhiyong Peng. Deep reinforcement learning-based approach to tackle topic-aware influence maximization. *Data Science and Engineering*, pages 1–11, 2020.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NeurIPS*, pages 2692–2700, 2015.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards, 1989.

Bryan Wilder, Laura Onasch-Vera, Juliana Hudson, Jose Luna, Nicole Wilson, Robin Petering, Darlene Woo, Milind Tambe, and Eric Rice. End-to-end influence maximization in the field. In *AAMAS*, volume 18, pages 1414–1422, 2018.

Bryan Wilder, Laura Onasch-Vera, Graham Diguiseppi, Robin Petering, Chyna Hill, Amulya Yadav, Eric Rice, and Milind Tambe. Clinical trial of an ai-augmented intervention for hiv prevention in youth experiencing homelessness. In *AAAI*, pages 14948–14956, 2021.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

Amulya Yadav, Hau Chan, Albert Xin Jiang, Haifeng Xu, Eric Rice, and Milind Tambe. Using social networks to aid homeless shelters: Dynamic influence maximization under uncertainty. In *AAMAS*, pages 740–748, 2016.

Amulya Yadav, Ritesh Noothigattu, Eric Rice, Laura Onasch-Vera, Leandro Soriano Marcolino, and Milind Tambe. Please be an influencer? contingency-aware influence maximization. In *AAMAS*, pages 1423–1431, 2018.