Please be an Influencer? Contingency-Aware Influence Maximization

Amulya Yadav
University of Southern California
amulyaya@usc.edu

Ritesh Noothigattu
Carnegie Mellon University
riteshn@cmu.edu

Eric Rice
University of Southern California
ericr@usc.edu

Laura Onasch-Vera
University of Southern California
onaschve@usc.edu

Leandro Marcolino
Lancaster University
l.marcolino@lancaster.ac.uk

Milind Tambe
University of Southern California
tambe@usc.edu

ABSTRACT

Most previous work on influence maximization in social networks assumes that the chosen influencers (or *seed nodes*) can be influenced with certainty (i.e., with no contingencies). In this paper, we focus on using influence maximization in public health domains for assisting low-resource communities, where *contingencies* are common. It is very difficult in these domains to ensure that the seed nodes are influenced, as influencing them entails contacting/convincing them to attend training sessions, which may not always be possible. Unfortunately, previous state-of-the-art algorithms for influence maximization are unusable in this setting. This paper tackles this challenge via the following four contributions: (i) we propose the Contingency Aware Influence Maximization problem and analyze it theoretically; (ii) we cast this problem as a Partially Observable Markov Decision Process and propose CAIMS (a novel POMDP planner) to solve it, which leverages a natural action space factorization associated with real-world social networks; and (iii) we provide extensive simulation results to compare CAIMS with existing state-of-the-art influence maximization algorithms. Finally, (iv) we provide results from a real-world feasibility trial conducted to evaluate CAIMS, in which key influencers in homeless youth social networks were influenced in order to spread awareness about HIV.

INTRODUCTION

The influence maximization problem is an NP-Hard combinatorial optimization problem [9], which deals with finding a set of K influential seed nodes in a social network to optimally spread influence in the network according to some pre-specified diffusion model. It is a practically relevant problem with numerous potential applications in the real world, especially in public health domains involving low-resource communities. For example, it has been used to prevent smoking among teenagers [23], and to promote healthier lifestyles among risky populations [13]. Recently, influence maximization algorithms were used to spread awareness about HIV among homeless youth with great results [26].

Recently, several efficient algorithms have been proposed (and deployed in the real-world) to solve influence maximization problems [2, 5, 22, 24]. Most of these algorithms rely on the following key assumption: seed nodes can be influenced with certainty. Unfortunately, in most public health domains, this assumption does not hold as "influencing" seed nodes entails training them to be "*peer leaders*" [23]. For example, seed nodes promoting HIV awareness among homeless youth need to be trained so that they can communicate information about supposedly private issues in a safe manner [18]. This issue of training seed nodes leads to two practical challenges. First, it may be difficult to contact seed nodes in a timely manner (e.g., contacting homeless youth is challenging since they rarely have fixed phone numbers, etc). Second, these seed nodes may decline to be influencers (e.g., they may decline to show up for training sessions). In this paper, we refer to these two events as contingencies in the influence maximization process.

Unsurprisingly, these contingencies result in a wastage of valuable time/money spent in unsuccessfully contacting/convincing the seed nodes to attend the training. Moreover, the resulting influence spread achieved is highly sub-optimal, as very few seed nodes actually attend the training session, which defeats the purpose of conducting these interventions. Clearly, contingencies in the influence maximization process need to be considered very carefully.

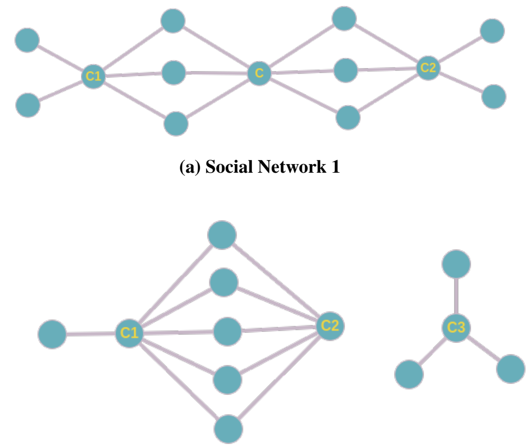
In this paper, we propose a principled approach to handle these inevitable contingencies via the following contributions. First, we introduce the **C**ontingency **A**ware **I**nfluence **M**aximization (or CAIM) problem to handle cases when seed nodes may be unavailable, and analyze it theoretically. The principled selection of alternate seed nodes in CAIM (when the most preferred seed nodes are not available) sets it apart from any other previous work in influence maximization, which mostly assumes that seed nodes are always available for activation. Second, we cast the CAIM problem as a Partially Observable Markov Decision Process (POMDP) and solve it using CAIMS (**CAIM S**olver), a novel POMDP planner which provides an adaptive policy which explicitly accounts for contingency occurrences. CAIMS is able to scale up to real-world network sizes by leveraging the community structure (present in most real-world networks) to factorize the action space of our original POMDP into several smaller community-sized action spaces. Further, it utilizes insights from social network literature to represent belief states in our POMDP in a compact, yet accurate manner using Markov networks. Our simulations show that CAIMS outperforms state-of-the-art influence maximization algorithms by ~60%. Finally, we evaluate CAIMS's usability in the real-world by using it to train a small set of homeless youth (the seed nodes) to spread awareness about HIV among their peers. This domain is an excellent testbed for CAIMS,

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. Andre, S. Koenig (eds.), July 2018, Stockholm, Sweden
© 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.
https://doi.org/doi

as the transient nature of homeless youth increases the likelihood of the occurrence of contingencies [15].

Related Work In addition to the work on influence maximization highlighted in the introduction, [21] is related to our work as it solves an orthogonal problem: how to incentivize people in order to be influencers? Unlike us, they solve a mechanism-design problem where nodes have private costs, which need to be paid for them to be influencers. However, in our domains of interest, monetary gains/losses are not the reason behind nodes getting influenced or not. Instead, nodes do not get influenced because of contingencies.

We also discuss work in POMDP planning, since we cast CAIM as a POMDP. SARSOP [11] is a state-of-the-art offline POMDP solver but it does not scale up to larger state spaces. [20] proposed POMCP which use Monte-Carlo tree search in online planning, but it does not scale up to larger action spaces. As a result, FV-POMCP [1, 17] was proposed which relies on a factorized action space to scale up to larger problems. In our work, we complement their advances to build CAIMS, which leverages insights from social network theory to factorize action spaces in a provably "*lossless*" manner, and to represent beliefs in an accurate manner.



(a) Social Network 1

(b) Social Network 2

Figure 1: Examples illustrating harm in overprovisioning

CAIM MODEL & PROBLEM

We motivate our discussion of the CAIM problem by focusing on a particular public health domain: preventing HIV spread among homeless youth. In this domain, youth are extremely susceptible to HIV infection due to high-risk activities that they engage in, e.g., unprotected sex, etc. [4]. In order to reduce the spread of HIV, several non-profit agencies called "homeless shelters" conduct intervention training camps to train influential homeless youth as "peer leaders", so that they can spread awareness about HIV in the *friendship based social network* of homeless youth, via peers in their social circles [13].

Unfortunately, homeless shelters do not have the resources to train all homeless youth in the social network as peer leaders. Moreover, as behavioral problems of homeless youth makes managing larger groups difficult [14], intervention training camps (interventions for short) can only include a small number (∼5-6) of youth.

In practice, the shelter officials typically only have 4-5 days to locate/invite the desired youth to be trained. However, the transient nature of homeless youth (i.e., no fixed postal address, phone number, etc) makes contacting the chosen peer leaders difficult for homeless shelters. Further, most youth are distrustful of adults, and thus, they may decline to be trained as peer leaders [12]. As a result of these "*contingencies*", the shelter officials are often forced to conduct their intervention with very few peer leaders in attendance, despite each official spending 4-5 days worth of man hours in trying to find the chosen peer leaders [26]. Moreover, the peer leaders who finally attend the intervention are usually not influential seed nodes. This has been the state of operations even though peer-led interventions have been conducted by social workers for almost a decade now.

To avoid this outcome, ad-hoc measures have been proposed [26], e.g., contacting many more homeless youth than they can safely manage in an intervention. However, one then runs the risk that lots of youth may agree to be peer leaders, and shelter officials would have to conduct the intervention with all these youth (since it's unethical to invite a youth first and then ask him/her not to come to the intervention), even if the total number of such participants exceeds their maximum capacity [16]. This results in interventions where the peer leaders may not be well trained, as insufficient attention is given to any one youth in the training. Note that if contingencies occurred infrequently, then inviting a few extra nodes (over the maximum capacity) may be a reasonable solution. However, as we show in the real-world feasibility trial conducted by us, contingencies are very common (*∼80%, or 14 out of 18 invitations in the real-world study resulted in contingencies*), and thus, overprovisioning by a small number of nodes is not an option. An ad-hoc fix for this over-attendance, is to first select (say) twice the desired number of homeless youth, invite them one at a time, and stop as soon as the desired number of homeless youth have accepted the invitation. We study this algorithm further after describing our influence model.

Social Networks We represent friendship based social networks as undirected graphs $G = (\boldsymbol{V}, \boldsymbol{E})$, where each *node* $v \in \boldsymbol{V}$ represents a person in the social network and an *edge* $e = (A, B) \in \boldsymbol{E}$ between two nodes A and B (say) represents that nodes A and B are friends. Each edge $e \in \boldsymbol{E}$ has a propagation probability $p(e)$ associated with it, which represents the probability that a node which is influenced (has information) will pass on that influence to their neighbor. Influence spreads using the independent cascade model [9], in which all nodes that get influenced at time t get a **single** chance to influence their un-influenced neighbors at time $t + 1$. This graph G with all relevant $p(e)$ values represents a *friendship based social network* and serves as an input to the CAIM problem.

Overprovisioning May Backfire Let K denote the number of nodes (or homeless youth) we want at the intervention. Now, suppose we overprovision by a factor of 2 and use the algorithm mentioned before. In particular, instead of searching for the optimal set of K seed nodes, the algorithm finds the optimal set of $2K$ seed nodes and then influences the *first* K of these nodes that accept the invitation. Under contingencies, this algorithm is expected to perform better than the algorithm without overprovisioning. Surprisingly, this is not the case. In particular, overprovisioning may make things worse.

Two key ideas behind this are: (i) No K-sized subset of the optimal set of $2K$ nodes may be as good as the optimal set of K nodes (this indicates that we may not be looking for the right nodes when we search for the optimal set of $2K$ nodes), and (ii) An arbitrary K-sized subset of the optimal set of $2K$ nodes (obtained because we stick to the *first* K nodes that accept the invitation) may perform arbitrarily bad.

We consider two examples that concretize these facts. For simplicity of the examples, we assume that influence spreads only for one round, number of nodes required for the intervention is $K = 1$ and the propagation probability $p(e)$ is 0.5 for every edge. Firstly, consider the example social network graph in Figure 1a. Suppose C and $C1$ are nodes that are regularly available, and are likely to accept the invitation. Now, let's find the best single node to influence for maximum influence spread. We don't need to consider nodes other than $\{C1, C, C2\}$ since they're obviously suboptimal. For the remaining nodes, we have $\mathcal{I}(C1) = 5 * 0.5 = 2.5$, $\mathcal{I}(C) = 6 * 0.5 = 3$ and $\mathcal{I}(C2) = 5 * 0.5 = 2.5$, and so the best single node to influence is C. Now, suppose we overprovision by a factor of 2, and try to find the optimal set of 2 nodes for maximum influence spread. The influence values are $\mathcal{I}(\{C1, C\}) = \mathcal{I}(\{C2, C\}) = 5 * 0.5 + 3 * 0.75 = 4.75$ and $\mathcal{I}(\{C1, C2\}) = 10 * 0.5 = 5$. So, the optimal set of 2 nodes to influence is $\{C1, C2\}$. But, since we need only one node, we would eventually be influencing either $C1$ or $C2$, giving us an expected influence of 2.5. On the other hand, if we did not overprovision, we would go for node C (the best single node to influence) and have an expected influence of 3. This example demonstrates that no K-sized subset of the optimal set of $2K$ nodes may be as good as the optimal set of K nodes. Note that, for clarity, the example considered here was small and made simple, and hence the difference between 3 and 2.5 may seem small. But, the example can be extended such that the difference is arbitrarily larger.

Secondly, consider the example social network graph of Figure 1b. Again, for simplicity, we assume that influence spreads only for one round, number of nodes required for the intervention is $K = 1$ and the propagation probability $p(e)$ is 0.5 for every edge. Like before, let's find the best single node to influence for maximum influence spread. We don't need to consider nodes other than $\{C1, C2, C3\}$ since they're obviously suboptimal. For the remaining nodes, we have $\mathcal{I}(C1) = 6 * 0.5 = 3$, $\mathcal{I}(C2) = 5 * 0.5 = 2.5$ and $\mathcal{I}(C3) = 3 * 0.5 = 1.5$, and so the best single node to influence is $C1$. Now, suppose we overprovision by a factor of 2, and try to find the optimal set of 2 nodes for maximum influence spread. The influence values are $\mathcal{I}(\{C1, C2\}) = 1 * 0.5 + 5 * 0.75 = 4.25$, $\mathcal{I}(\{C2, C3\}) = 8 * 0.5 = 4$ and $\mathcal{I}(\{C1, C3\}) = 9 * 0.5 = 4.5$. So, the optimal set of 2 nodes is $\{C1, C3\}$, and it would be selected by the overprovisioning algorithm. But, as mentioned before, we stop once we find the first node that accepts the invitation. In case $C1$ is the first node encountered and it accepts the invitation, then there's an expected influence of 3, but if $C3$ is the first such node, the expected influence would be as low as 1.5. On the other hand, the standard algorithm would directly go for $C1$ giving an expected influence of 3.

On a different note, suppose in this second example, node $C1$ is unavailable (because say it declines the invitation). In this case, the overprovisioning algorithm would have to go for $C3$ (the only other node in the optimal set of 2 nodes), leading to an expected influence of 1.5. However, an adaptive solution, would look for node $C1$ and after finding that its unavailable, would go for the next best node which is node $C2$. This gives an adaptive solution an expected influence of 2.5.

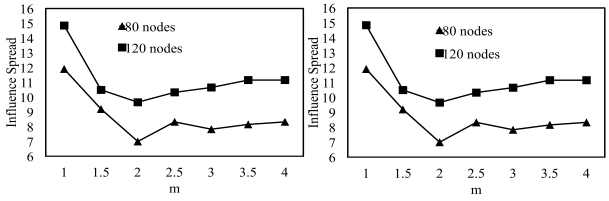


Figure 2: The Harm in Overprovisioning

We further investigate the impact of overprovisioning by measuring the performance of the Greedy algorithm [9] (the gold standard in influence maximization) under varying levels of overprovisioning. Figures 2a and 2b compare influence spread achieved by Greedy on stochastic block model (SBM) and preferential attachment (PA) networks [19], respectively, as it finds the optimal set of $m * K$ nodes ($K = 2$) to invite (i.e., overprovision by factor m) and influence the first K nodes that accept the invitation. The x-axis shows increasing m values and the y-axis shows influence spread. This figure shows that in both SBM and PA networks of different sizes, overprovisioning hurts, i.e., optimizing for larger seed sets in anticipation of contingencies actually hurts influence spread. Overprovisioning's poor performance reveals that simple solutions do not work, thereby necessitating careful modeling of contingencies, as we do in CAIM.

Problem Setup Given a *friendship based social network*, the goal in CAIM is to invite several network nodes for the intervention until we get K nodes who agree to attend the intervention. The problem proceeds in T sequential sessions, where T represents the number of days that are spent in trying to invite network nodes for the intervention. In each session, we assume that nodes are either available or unavailable for invitation. This is because on any given day (session), homeless youth may either be present at the shelter (i.e., available) or not (i.e., unavailable). We assume that only nodes which are available in a given session can accept invitations in that session. This is because homeless youth frequently visit shelters, hence we utilize this opportunity to issue invitations to them if we see them at the shelter.

Let $\phi^t \in \{0, 1\}^N$ (called a realization) be a binary vector which denotes the availability or unavailability (for invitation) of each network node in session $t \in [1, T]$. We take a Bayesian approach and assume that there is a known prior probability distribution $\mathbf{\Phi}$ over realizations ϕ^t such that $p(\phi^t) := \mathcal{P}[\mathbf{\Phi} = \phi^t]$. In our domain, this prior distribution is represented using a Markov Network. We assume that the realization ϕ^t for each session $t \in [1, T]$ is drawn i.i.d. from the prior distribution $\mathbf{\Phi}$, i.e., the presence/absence of homeless youth at the shelter in every session $t \in [1, T]$ is assumed to be an i.i.d. sample from $\mathbf{\Phi}$. We further assume that while the prior distribution $\mathbf{\Phi}$ is provided to the CAIM problem as input, the complete i.i.d. draws from this distribution (i.e., the realizations $\phi^t \ \forall t \in [1, T]$) are not observable. This is because while querying the availability of a small number of nodes (~3-4) is feasible, querying each node in the social

network (which can have 150-160 nodes) for each session/day (to completely observe ϕ^t) requires a lot of work which is not possible with the shelters limited resources [13].

In each session $t \in [1, T]$, a maximum of L actions can be taken, each of which can be of three possible types: *queries*, *invites* and *end-session* actions. Query action $q_{\boldsymbol{a}}$ in session $t \in [1, T]$ ascertains the availability/unavailability of a subset of nodes $\boldsymbol{a}$ ($\|\boldsymbol{a}\| \leqslant Q_{max}$, the maximum query size) in session t with certainty. Thus, query actions in session t provide partial *observations* about the realization of nodes ϕ^t in session t. On the other hand, invite action $m_{\boldsymbol{a}}$ invites a subset of nodes $\boldsymbol{a} \subset V$ ($\|\boldsymbol{a}\| \leqslant K$) to the intervention. Upon taking an invite action, we *observe* which invited nodes are present (according to ϕ^t) in the session and which of them accepted our invitation. Each invited node that is present accepts the invitation with a probability ϵ. We refer to the nodes that accept our invitation as "*locked* nodes" (since they are guaranteed to attend the intervention). Finally, we can also take an *end-session* action, if we choose not to invite/query any more nodes in that session.

The observations received from query and invite actions (*end-session* action provides no observation) taken in a session allows us to update the original prior distribution Φ to generate a posterior distribution $\Phi_t^{pos}(i)\ \forall i \in [0, L]$ for session t (where i actions have been taken in session t so far). These posteriors can then be used to decide future actions that need to be taken in a session. Note that for every session t, $\Phi_t^{pos}(0) = \Phi$, i.e., at the beginning of each session, we start from the original prior distribution Φ and then get new posteriors every time we take an action in the session.

Informally then, given a friendship based social network $G = (\boldsymbol{V}, \boldsymbol{E})$, the integers T, K, L, Q_{max} and ϵ, and prior distribution Φ, the goal of CAIM is to find a policy for choosing L sequential actions for T sessions s.t. the expected influence spread (according to our influence model) achieved by the set of locked nodes (i.e., nodes which finally attend the intervention) is maximized.

Let $\boldsymbol{Q} = \{q_{\boldsymbol{a}} \text{ s.t. } 1 \leqslant \|\boldsymbol{a}\| \leqslant Q_{max}\}$ denote the set of all possible query actions that can be taken in any given session $t \in [1, T]$. Similarly, let $\boldsymbol{\mathcal{M}} = \{m_{\boldsymbol{a}} \text{ s.t. } 1 \leqslant \|\boldsymbol{a}\| \leqslant K\}$ denote the set of all possible invite actions that can be taken in any given session $t \in [1, T]$. Also, let $\boldsymbol{\mathcal{E}}$ denote the end-session action. Let $\mathcal{A}_i^t \in \boldsymbol{Q} \cup \boldsymbol{\mathcal{M}} \cup \boldsymbol{\mathcal{E}}$ denote the i^{th} action ($i \in [1, L]$) chosen by CAIM's policy in session $t \in [1, T]$.

Upon taking action $\mathcal{A}_i^t$ ($i \in [1, L], t \in [1, T]$), we receive observations which allow us to generate posterior distribution $\Phi_t^{pos}(i)$. Denote by $\boldsymbol{M}_i^t$ the set of all locked nodes after the i^{th} action is executed in session t. Denote by Δ the set of all possible posterior distributions that we can obtain during the CAIM problem. Denote by Γ all possible sets of locked nodes that we can obtain during the CAIM problem. Finally, we define CAIM's policy $\Pi : \Delta \times \Gamma \times [0, L] \times [1, T] \to \boldsymbol{Q} \cup \boldsymbol{\mathcal{M}} \cup \boldsymbol{\mathcal{E}}$ as a function that takes in a posterior distribution, a set of locked nodes, the number of actions taken so far in the current session, and the session-id as input, and outputs an action $\mathcal{A}_i^t$ for the current timestep.

PROBLEM 1. ***CAIM Problem*** *Given as input a social network $G = (\boldsymbol{V}, \boldsymbol{E})$ and integers T, K, L, Q_{max} and ϵ, and a prior distribution Φ (as defined above), denote by $\mathcal{R}(\boldsymbol{M}_L^T)$ the expected total influence spread (i.e., number of nodes influenced) achieved by nodes in $\boldsymbol{M}_L^T$ (i.e., locked nodes at the end of T sessions). Let $\mathbb{E}_{\boldsymbol{M}_L^T \sim \Pi}[\mathcal{R}(\boldsymbol{M}_L^T)]$ denote the expectation over the random variable $\boldsymbol{M}_L^T$, where $\boldsymbol{M}_L^T$ is updated according to actions recommended by policy $\Pi(\Phi_T^{pos}(L-1), \boldsymbol{M}_{L-1}^T, L-1, T)$. More generally, in session $t \in [1, T]$, $\boldsymbol{M}_i^t\ \forall i \in [0, L]$ is updated according to actions recommended by policy $\Pi(\Phi_t^{pos}(i-1), \boldsymbol{M}_{i-1}^t, i-1, t)$. Then, the objective of CAIM is to find an optimal policy $\Pi^* = argmax_{\Pi} \mathbb{E}_{\boldsymbol{M}_L^T \sim \Pi}[\mathcal{R}(\boldsymbol{M}_L^T)]$.*

We now theoretically analyze the CAIM problem. Due to lack of space, some proofs are in the appendix[1].

LEMMA 1. *The CAIM problem is NP-Hard.*

PROOF. Consider an instance of the CAIM problem with prior probability distribution Φ that is the realization ϕ_* with probability 1, where ϕ_* is a vector of all 1s. Such a problem reduces to the standard influence maximization problem, wherein we need to find the optimal subset of K nodes to influence to have maximum influence spread in the network. But, the standard influence maximization problem is an NP-Hard problem, making CAIM NP-Hard too. □

Some NP-Hard problems exhibit nice properties that enable approximation guarantees for them. [6] introduced adaptive submodularity, the presence of which would ensure that a simple greedy algorithm provides a $(1-1/e)$ approximation w.r.t. the optimal CAIM policy. However, we show that while CAIM can be cast into the adaptive stochastic optimization framework of [6], our objective function is not adaptive submodular, because of which their Greedy algorithm does not have a $(1-1/e)$ approximation guarantee.

LEMMA 2. *The objective function of CAIM is not adaptive submodular.*

These theorems show that CAIM is a computationally hard problem and it is difficult to even obtain any good approximate solutions for it. In this paper, we model CAIM as a POMDP.

POMDP MODEL

States A POMDP state consists of four entities $s = \langle \phi, \boldsymbol{M}, numAct, sessID \rangle$. Here, $sessID \in [1, T]$ identifies the session we are in. Also, $numAct \in [0, L]$ determines the number of actions that have been taken so far in session $sessID$. $\boldsymbol{M}$ denotes the set of locked nodes so far (starting from the first session). Finally, ϕ is the node realization ϕ^{sessID} in session $sessID$. In our POMDP model, states with $sessID = T$ and $numAct = L$ are terminal states, since they represent the end of all sessions.

Actions A POMDP action is a tuple $a = \langle S, type \rangle$. Here, $type$ is a symbolic character which determines whether a is a query action (i.e., $type = q$), an invite action (i.e., $type = i$) or an *end-session* action (i.e., $type = e$). Also, $\boldsymbol{S} \subset \boldsymbol{V}$ denotes the subset of nodes that is queried ($type = q$) or invited ($type = i$). If $type = q$, the size of subset $\|\boldsymbol{S}\| \in [1, Q_{max}]$. Similarly, if $type = i$, $\|\boldsymbol{S}\| \in [1, K]$. Finally, if $type = e$, subset $\boldsymbol{S}$ is empty.

Observations Upon taking a query action $a = \langle S, q \rangle$ in state $s = \langle \phi, \boldsymbol{M}, numAct, sessID \rangle$, we receive an observation that is completely determined by state s. In particular, we receive the observation $o_q = \{\phi(v)\ \forall v \in \boldsymbol{S}\}$, i.e., the availability status of each node in $\boldsymbol{S}$. And, by taking an invite action $a = \langle S, i \rangle$ in state $s = \langle \phi, \boldsymbol{M}, numAct, sessID \rangle$,

[1] Appendix is available at https://www.dropbox.com/s/d9p6fal7kf5tfn9/appendix.pdf

we receive two kinds of observations. Let $\mathbf{\Gamma} = \{v \in \boldsymbol{S} \text{ s.t. } \phi(v) = 1\}$ denote the set of available nodes in *invited set* $\boldsymbol{S}$. First, we get observation $o_i^1 = \{\phi(v)\ \forall v \in \boldsymbol{S}\}$ which specifies the availability status of each node in invited set $\boldsymbol{S}$. We also get an observation $o_i^2 = \{b(v)\ \forall v \in \mathbf{\Gamma}\}$ for each available node $v \in \mathbf{\Gamma}$, which denotes whether node v accepted our invitation and joined the locked set of nodes ($b(v) = 1$) or not ($b(v) = 0$). Finally, the *end-session* action does not generate any observations.

Rewards We only get rewards when we reach terminal states $s' = \langle \phi, \boldsymbol{M}, numAct, sessID \rangle$ with $sessID = T$, $numAct = L$. The reward attained in terminal state s' is the expected influence spread (as per our influence model) achieved by influencing nodes in the locked set $\boldsymbol{M}$ of s'.

Transition And Observation Probabilities Due to our exponential sized state and action spaces, maintaining transition and observation probability matrices is not feasible. Hence, we follow the paradigm of large-scale online POMDP solvers [20] by using a generative model $\Lambda(s, a) \sim (s', o, r)$ of the transition and observation probabilities. This generative model allows us to generate on-the-fly samples from the exact distributions $T(s'|s, a)$ and $\Omega(o|a, s')$ at very low computational costs. In our generative model, the state undergoes transitions as follows. On taking a query action, we reach a state s' which is the same as s except that $s'.numAct = s.numAct + 1$. On taking an invite action $\langle \boldsymbol{S}, i \rangle$, we reach s' which is the same as s except that $s'.numAct = s.numAct + 1$, and $s'.\boldsymbol{M}$ is $s.\boldsymbol{M}$ appended with nodes of $\boldsymbol{S}$ that accepted the invitation. Note that the binary vector ϕ stays unchanged in either case (since the session does not change). Finally, on taking the end-session action, we start a new session by transitioning to state s' s.t., $s'.numAct = 0$, $s'.sessID = s.sessID + 1$, $s'.\boldsymbol{M} = s.\boldsymbol{M}$ and $s'.\phi$ is resampled i.i.d. from the prior distribution $\mathbf{\Phi}$. Note that the components $\boldsymbol{M}$, $numAct$ and $sessID$ of a state are fully observable.

Initial Belief State The prior distribution $\mathbf{\Phi}$, along with other completely observable state components (such as $sessID = 1$), $numAct = 0$, and an empty locked set $\boldsymbol{M} = \{\}$) forms our initial belief state.

CAIMS: CAIM SOLVER

Our POMDP algorithm is motivated by the design of FV-POMCP, a recent online POMDP algorithm [1]. Unfortunately, FV-POMCP has several limitations which make it unsuitable for solving the CAIM problem. Thus, we propose CAIMS, a Monte-Carlo (MC) sampling based online POMDP algorithm which makes key modifications to FV-POMCP, and solves the CAIM problem for real-world sized networks. Next, we provide a brief overview of POMCP, and its extension FV-POMCP.

POMCP POMCP [20] uses UCT based Monte-Carlo tree search (MCTS) [3] to solve POMDPs. At every stage, given the current belief state b, POMCP incrementally builds a UCT tree that contains statistics that serve as empirical estimators (via MC samples) for the POMDP Q-value function $Q(b, a) = R(b, a) + \sum_z P(z|b, a) max_{a'} Q(b', a')$. The algorithm avoids expensive belief updates by maintaining the belief at each UCT tree node as an unweighted particle filter (i.e., a collection of all states that were reached at that UCT tree node via MC samples). In each MC simulation, POMCP samples a start state from the belief at the root node of the UCT tree, and then samples a trajectory that first traverses the partially built UCT tree, adds a node to this tree if the end of the tree is reached before the desired horizon, and then performs a random rollout to get one MC sample estimate of $Q(b, a)$. Finally, this MC sample estimate of $Q(b, a)$ is propagated up the UCT tree to update Q-value statistics at nodes that were visited during this trajectory. Note that the UCT tree grows exponentially large with increasing state and action spaces. Thus, the search is directed to more promising areas of the search space by selecting actions at each tree node h according to the UCB1 rule [10], which is given by: $a = argmax_a \hat{Q}(b_h, a) + c\sqrt{log(N_h + 1)/n_{ha}}$. Here, $\hat{Q}(b_h, a)$ represents the the Q-value statistic (estimate) that is maintained at node h in the UCT tree. Also, N_h is the number of times node h is visited, and n_{ha} is the number of times action a has been chosen at tree node h (POMCP maintains statistics for N_h and $n_{ha} \forall a \in A$ at each tree node h). While POMCP handles large state spaces (using MC belief updates), it is unable to scale up to large action sizes (as the branching factor of the UCT tree blows up). We validate POMCP's poor scale-up performance in our experiments.

FV-POMCP FV-POMCP extends POMCP to deal with large action spaces. It assumes that the action space of the POMDP can be factorized into a set of ℓ factors, i.e., each action a can be decomposed into a set of sub-actions $a_l \forall l \in [1, \ell]$. Under this assumption, the value function of the original POMDP is decomposable into a set of overlapping factors. i.e., $Q(b, a) = \sum_{l \in [1, \ell]} \alpha_l Q_l(b, a_l)$, where α_l ($\forall l \in [1, \ell]$) are factor-specific weights. FV-POMCP maintains a single UCT tree (similar to standard POMCP), but it differs in the statistics that are maintained at each node of the UCT tree. Instead of maintaining $\hat{Q}(b_h, a)$ and n_{ha} statistics for every action in the global (unfactored) action space at tree node h, it maintains a set of statistics that estimates the values $\hat{Q}_l(b_h, a_l)$ and n_{ha_l} $\forall l \in [1, \ell]$.

Joint actions are selected by the UCB1 rule across all factored statistics, i.e., $a = argmax_a \sum_{l \in [1, \ell]} \hat{Q}_l(b_h, a_l) + c\sqrt{log(N_h + 1)/n_{ha_l}}$. This maximization is efficiently done using variable elimination (VE) [7], which exploits the action factorization appropriately. Thus, FV-POMCP achieves scale-up by maintaining fewer statistics at each tree node h, and by using VE to find the maximizing joint action.

However, there are two limitations which makes FV-POMCP unsuitable for solving CAIM. First, the VE procedure used in FV-POMCP (as described above) may return an action (i.e., a set of nodes) which is infeasible in the CAIM problem (e.g., the action may have more than K nodes). We elaborate on this point later. Second, FV-POMCP uses unweighted particle filters to represent belief states, which becomes highly inaccurate with exponentially sized state spaces in CAIM. We address these limitations in CAIMS.

CAIMS

CAIMS is an online Monte-Carlo sampling based POMDP solver that uses UCT based Monte-Carlo tree search to solve the CAIM problem. Similar to FV-POMCP, CAIMS also exploits action factorization to scale up to large action spaces. We now explain CAIMS's action factorization.

Action Factorization Real world social networks generally exhibit a lot of community structure, i.e., these networks are composed of several tightly-knit communities (partitions), with very few edges

going across these communities [19]. This community structure dictates the action factorization in CAIMS. As stated before, the POMDP model has each action of the form $\langle S, type \rangle$, where S is a subset of nodes (that are being queried or invited). This (sub)set S can be represented as a boolean vector $\vec{S}$ (denoting which nodes are included in the set). Let $Q_q(\vec{S})$ denote the Q-value of the query action $\langle S, q \rangle$, $Q_i(\vec{S})$ denote the Q-value of the invite action $\langle S, i \rangle$ and let Q_e denote the Q-value of the end-session action $\langle \{\}, e \rangle$. Now, suppose the real-world social network is partitioned into ℓ partitions (communities) $P_1, P_2, \cdots P_\ell$. Let $\vec{S}_{P_x}$ denote the sub-vector of $\vec{S}$ corresponding to the x^{th} partition. Then, the action factorization used is: $Q_q(\vec{S}) = \sum_{x=1}^{\ell} Q_q^{P_x}(\vec{S}_{P_x})$ for query actions and $Q_i(\vec{S}) = \sum_{x=1}^{\ell} Q_i^{P_x}(\vec{S}_{P_x})$ for invite actions.

Intuitively, $Q_i^{P_x}(\vec{S}_{P_x})$ can be seen as the Q-value of inviting only nodes given by $\vec{S}_{P_x}$ (and no other nodes). Now, if querying/inviting nodes of one partition has negligible effect/influence on the other partitions, then the Q-value of the overall invite action $\langle S, i \rangle$ can be approximated by the sum of the Q-values of the sub-actions $\langle S_{P_x}, i \rangle$. The same holds for query actions. We now show that this action factorization is appropriate for CAIM as it introduces *minimal* error into the influence spread calculations for stochastic block model (SBM) networks, which mimic many properties of real-world networks [19].

THEOREM 3. *Let $\mathcal{I}(S)$ denote the expected influence in the whole network when nodes of set S are influenced, and we have one round of influence spread. For an SBM network with n nodes and parameters (p, q) that is partitioned into ℓ communities, the difference between the true and factored expected influences can be bounded as $\mathbb{E}\left[\max_S \left|\mathcal{I}(S) - \sum_{x=1}^{\ell} \mathcal{I}(S_{P_x})\right|\right] \leq qn^2\left(1 - \frac{1}{\ell}\right)p_m$, where $p_m = \max_{e \in E} p(e)$ is the maximum propagation probability. Note that the (outer) expectation is over the randomness in the SBM network model.*

This action factorization allows us to maintain separate Q-value statistics $(\hat{Q}_{type}^{P_x}(\vec{S}_{P_x})\ \forall type \in \{q, i, e\})$ for each factor (i.e., network community) at each node of the UCT tree maintained by CAIMS. However, upon running MC simulations in this UCT tree, we acquire samples of only Q_{type} (i.e., rewards of the joint *un-factored* actions). We learn factored estimates $Q_{type}^{P_x}$ from estimates Q_{type} of the *un-factored* actions by using mixture of experts optimization [1], i.e. we estimate the factors as $\hat{Q}_{type}^{P_x}(\vec{S}_{P_x}) = \alpha_{P_x}\mathbb{E}[Q_{type}(\vec{S})|\vec{S}_{P_x}]$, where this expectation is estimated by using the empirical mean. Please refer to [1] for more details. We now describe action selection in the UCT tree.

Action Selection At each node in the UCT tree, we use the UCB1 rule (over all factors) to find the best action. Let $n_{h\vec{S}_{P_x}}^{q}$ (or $n_{h\vec{S}_{P_x}}^{i}$) denote the number of times a query (or invite) action with sub-action $\vec{S}_{P_x}$ has been taken from node h of the UCT tree. Let N_h denote the number of times tree node h has been visited. The best query action to be taken is given as $\langle S_q, q \rangle$, where $\vec{S}_q = argmax_{\|\vec{S}\|_1 \leq Q_{max}} \sum_{x=1}^{\ell} \hat{Q}_q^{P_x}(b_h, \vec{S}_{P_x}) + c\sqrt{log(N_h+1)/n_{h\vec{S}_{P_x}}^{q}}$. Similarly, the best invite action to be taken is given as $\langle S_i, i \rangle$, where $\vec{S}_i = argmax_{\|\vec{S}\|_1 \leq K-|M|} \sum_{x=1}^{\ell} \hat{Q}_i^{P_x}(b_h, \vec{S}_{P_x}) + c\sqrt{log(N_h+1)/n_{h\vec{S}_{P_x}}^{i}}$ (where M is the set of locked nodes at tree node h). Let V_q and V_i denote the value attained at the maximizing query and invite actions, respectively. Finally, let V_e denote the value of the end-session action, i.e. $V_e = \hat{Q}_e + c\sqrt{log(N_h+1)/n_h^e}$ where n_h^e is the number of times the end-session action has been taken from tree node h. Then, the values V_q, V_i and V_e are compared and the action corresponding to $max(V_q, V_i, V_e)$ is chosen.

Improved VE Note that the UCB1 equations to find maximizing query/invite actions (as described above) are of the form $argmax_{\|\vec{a}\|_1 \leq z} \sum_{x=1}^{\ell} f_x(\vec{a}_x)$ (where $\vec{a} \in \{0,1\}^n$). Unfortunately, plain application of VE (like FV-POMCP) to this results in infeasible solutions which may violate the L-1 norm constraint. Thus, FV-POMCP's VE procedure may not produce feasible solutions for CAIM.

CAIMS addresses this limitation by using two adjustments. First, we incorporate this L-1 norm constraint as an additional factor in the objective function: $argmax_{\vec{a} \in \{0,1\}^n} \sum_{x=1}^{\ell} f_x(\vec{a}_x) + f_c(\vec{a})$. This constraint factor f_c's scope is all the n variables (as it represents a global constraint connecting actions selected across all factors), and hence it can be represented using a table of size $O(2^n)$ in VE. Unfortunately, the exponentially sized table of f_c eliminates any speed-up benefits that VE provides, as the induced width of the tree formed (on running VE) will be n, leading to a worst possible time-complexity of $O(2^n)$.

To resolve this, CAIMS leverages a key insight which allows VE to run efficiently even with the additional factor f_c. The key idea is that, if all variables of a community are eliminated at once, then both (i)f_c; and (ii) the factors derived from a combination of f_c and other community-specific factors during such elimination, can be represented very concisely (using just tables of size $z+1$ elements), instead of using tables of size $O(2^n)$. This fact is straightforward to see for the original constraint factor f_c (as f_c's table only depends on $\|\vec{a}\|_1$, it has value 0 if $\|\vec{a}\|_1 \leq z$ and $-\infty$ otherwise). However, it is not obvious why this holds for derived factors, which need to maintain optimal assignments to community-specific variables, for every possible combination of *un-eliminated* variable values (thereby requiring $O(2^n)$ elements). However, it turns out that we can still represent the derived factors concisely. The key insight is that even for these derived factors, all variable assignments with the same L-1 norm have the same value (Lemma 4). This allows us to represent each of these derived factors as a table of only $z+1$ elements (as we need to store one unique value when the L-1 norm is at most z, and we use $-\infty$ otherwise).

In more detail, the exact procedure of the modified VE algorithm is as follows. For the forward pass, we compute $\max_{\vec{a}} \sum_{x=1}^{\ell} f_x(\vec{a}_x) + f_c(\vec{a})$. We know that f_c depends only on the L-1 norm of $\vec{a}$, so let us represent it as such, i.e. as $f_c(\|\vec{a}\|_1)$. Also note that, the communities are disjoint, because of which each action bit a_i (of action $\vec{a}$) appears in the argument of exactly one factor f_x (other than the constraint factor f_c).

As mentioned before, we eliminate all variables of a community at once. So, to eliminate the first block of variables, we compute $\max_{\vec{a}_1} f_1(\vec{a}_1) + f_c(\|\vec{a}\|_1) = \psi_1(\|\vec{a}_{-1}\|_1)$, where $\vec{a}_{-1}$ denotes all action bits of $\vec{a}$ except those in $\vec{a}_1$. Note that, in the RHS of this expression,

we use $\|\vec{a}_{-1}\|_1$ as opposed to $\vec{a}_{-1}$ itself because the LHS (before computing the max) depends only on $\vec{a}_1$ and $\|\vec{a}_1\|_1 + \|\vec{a}_{-1}\|_1$. Also, note that for $\|\vec{a}_{-1}\|_1 > z$, we have $\|\vec{a}\|_1 > z$ making $f_c(\|\vec{a}\|_1)$ and $\psi_1(\|\vec{a}_{-1}\|_1)$ equal to $-\infty$.

To make this more concrete, Table 1 shows how ψ_1 is exactly computed. Here, $v_i^{(x)}$ denotes the maximum value of f_x when exactly i bits of $\vec{a}_x$ are 1, and s_x denotes the number of bits in $\vec{a}_x$.

$\|\vec{a}_{-1}\|_1$	$\psi_1(\|\vec{a}_{-1}\|_1)$
0	$\max\left(v_0^{(1)} + f_c(0), v_1^{(1)} + f_c(1), \cdots v_{s_1}^{(1)} + f_c(s_1)\right)$
1	$\max\left(v_0^{(1)} + f_c(1), v_1^{(1)} + f_c(2), \cdots v_{s_1}^{(1)} + f_c(s_1+1)\right)$
$\vdots$	$\vdots$
z	$v_0^{(1)} + f_c(z)$
$> z$	$-\infty$

Table 1: Factor obtained on (first) block elimination

Apart from computing the maximum objective value (forward pass), we also need to compute the maximizing assignment of the problem (backward pass). For this, we maintain another function $\mu_1(\|\vec{a}_{-1}\|_1)$ which keeps track of the value of $\vec{a}_1$ at which this maximum is attained (for each value of $\|\vec{a}_{-1}\|_1$), i.e. $\mu_1(v) = argmax_{\vec{a}_1}\left[f_1(\vec{a}_1) + f_c(\|\vec{a}_1\|_1 + v)\right]$. After eliminating variables of the first community, we are left with $\max_{\vec{a}_{-1}} \sum_{x=2}^{\ell} f_x(\vec{a}_x) + \psi_1(\|\vec{a}_{-1}\|_1)$. We repeat the same procedure and eliminate $\vec{a}_2$ by computing $\max_{\vec{a}_2} f_2(\vec{a}_2) + \psi_1(\|\vec{a}_{-1}\|_1)$, to obtain $\psi_2(\|\vec{a}_{-1,-2}\|_1)$. Note that, again, ψ_2 depends only on the L-1 norm of the remaining variables. Also, for $\|\vec{a}_{-1,-2}\|_1 > z$, ψ_2 becomes $-\infty$. In a similar way, this holds for the remaining generated factors, giving Lemma 4.

LEMMA 4. *Let $\psi_i(\vec{v})$ denote the i^{th} factor generated during CAIMS's VE. Then, $\psi_i(\vec{v}_1) = \psi_i(\vec{v}_2)$ if $\|v_1\|_1 = \|v_2\|_1$. Further $\psi_i(\vec{v}) = -\infty$ if $\|v\|_1 > z$.*

Once we complete the forward pass, we are left with $\psi_\ell(0)$ which is the maximum value of the objective function. Then, as in standard VE, we backtrack and use the μ_x functions to obtain the maximizer $argmax_{\vec{a}} \sum_{x=1}^{\ell} f_x(\vec{a}_x) + f_c(\|\vec{a}\|_1)$, i.e. $\mu_\ell(0)$ gives us the value of $\vec{a}_\ell$, then $\mu_{\ell-1}(\|\vec{a}_\ell\|_1)$ gives us the value of $\vec{a}_{\ell-1}$, $\mu_{\ell-2}(\|\vec{a}_\ell\|_1 + \|\vec{a}_{\ell-1}\|_1)$ gives us the value of $\vec{a}_{\ell-2}$ and so on.

Observe that to compute the i^{th} derived factor, we needed to compute $\max_{\vec{a}_i} f_i(\vec{a}_i) + \psi_{i-1}(\|\vec{a}_{-1,-2,\cdots-(i-1)}\|_1) = \psi_i(\|\vec{a}_{-1,-2,\cdots-i}\|_1)$. And for this, we just need to compute $v_s^{(i)}$ for each $s = 0, 1, \cdots s_i$, as evident from Table 1. This takes time $O(2^{s_i})$, where s_i denotes the size of the i^{th} community. Hence, the compact representations allow CAIMS to efficiently run VE in time $\sum_{i=1}^{\ell} O(2^{s_i})$ even after adding the global constraint factor f_c (Lemma 5). In fact, this is the best one can do, because any algorithm will have to look at all values of each community-specific factor in order to solve the problem.

LEMMA 5. *CAIMS's VE has time-complexity $\sum_{i=1}^{\ell} O(2^{s_i})$, where s_i is the size of the i^{th} factor (community). There exists no procedure with better time complexity.*

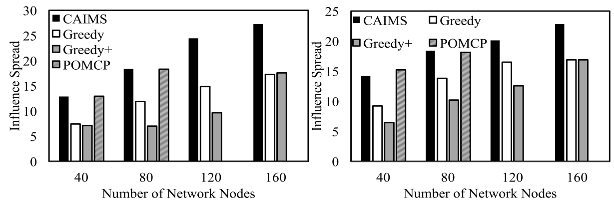
Markov Net Beliefs FV-POMCP uses unweighted particle filters to represent beliefs, i.e. a belief is represented by a collection of states (also known as particles), wherein each particle has an equal probability of being the true state. Unfortunately, due to CAIM's exponential state-space, the particle filter representation of belief states becomes highly inaccurate which leads to losses in solution quality.

To address this limitation, CAIMS makes the following assumption: availability/unavailability of network nodes is positively correlated with the availability/unavailability of their neighboring nodes in the social network. This assumption is reasonable because homeless youth usually go to shelters with their friends [15]. Thus, the confirmed availability of one homeless youth increases the likelihood of the availability of his/her friends. Under this assumption, the belief state in CAIM can be represented using a Markov Network. Formally, the belief is given as $b = \langle \mathcal{N}, \boldsymbol{M}, numAct, sessID \rangle$, where $\mathcal{N}$ is a Markov Network representing our belief of the true realization ϕ (note that the other three components of a state are observable). With the help of this Markov Network, we maintain *exact* beliefs throughout the POMCP tree of CAIMS. As mentioned before, the prior distribution Φ that serves as part of the initial belief state is also represented using a Markov Network $\mathcal{N}_0$. This prior can be elicited from field observations made by homeless shelter officials, and can be refined over multiple runs of CAIMS. *In our simulations, the social network structure $G = (\boldsymbol{V}, \boldsymbol{E})$ is used as a surrogate for the Markov network structure, i.e., the Markov network only has potentials over two variables/nodes (one potential for each pair of nodes connected by an edge in social network G).* Thus, we start with the initial belief as $\langle \mathcal{N}_0, \{\}, 0, 1 \rangle$. Upon taking actions $a = \langle S, type \rangle$ and receiving observations o, the belief state can be updated by conditioning the Markov network on the observed variables (i.e., by conditioning the presence/absence of nodes based on observations received from past query actions taken in the current session). This helps us maintain exact beliefs throughout the POMCP tree efficiently, which helps CAIMS take more accurate decisions.

EVALUATION

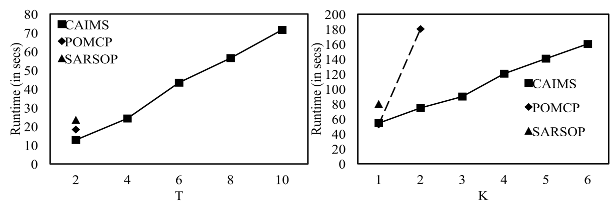
We show simulation results on artificially generated (and real-world) networks to validate CAIMS's performance in a variety of settings. We also provide results from a real-world feasibility study involving 54 homeless youth which shows the real-world usability of CAIMS. For our simulations, all the networks were generated using NetworkX library [8]. All experiments are run on a 2.4 GHz 8-core Intel machine having 128 GB RAM. Unless otherwise stated, we set $L = 3$, $Q_{max} = 2$, $K = 2$, and all experiments are averaged over 50 runs. *All simulation results are statistically significant under t-test ($\alpha = 0.05$).*

Baselines We use two different kinds of baselines. For influence maximization solvers, we use Greedy [9], the gold-standard in influence maximization as a benchmark. We subject Greedy's chosen nodes to contingencies drawn from the same prior Φ distribution that CAIMS uses. We also compare against the overprovisioning variant of Greedy (Greedy+) where instead of selecting K nodes, we select $2K$ nodes and influence the first K nodes that accept the invitation. This was proposed as an ad-hoc solution in [26] to tackle



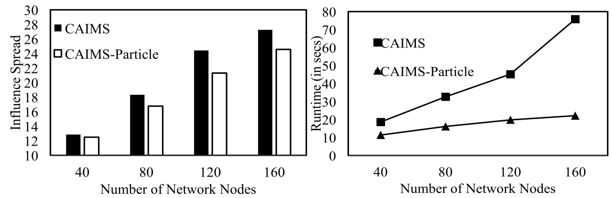
(a) SBM Networks (b) PA networks

Figure 3: Influence Spread Comparison



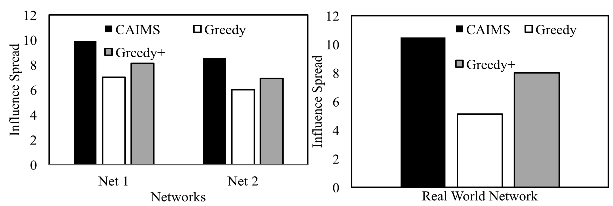
(a) Scale up in T (b) Scale up in K

Figure 4: Scale Up Results



(a) Influence Spread (b) Runtime

Figure 5: Value of using Markov Networks



(a) Homeless Youth Networks (b) Feasibility Trial

Figure 6: Real World Experiments

contingencies, and hence, we compare CAIMS against this. We also compare CAIMS against state-of-the-art POMDP solvers such as SARSOP and POMCP. Unfortunately, FV-POMCP cannot be used for comparison as its VE procedure is not guaranteed to satisfy the K budget constraint used inside CAIMS.

Solution Quality Comparison Figures 3a, 3b and 6a compares influence spread of CAIMS, Greedy, Greedy+ and POMCP on SBM ($p = 0.4, q = 0.1$), Preferential Attachment (PA) ($n = 5$) and real-world homeless youth networks (used in [25]), respectively. We select $K = 2$ nodes, and set $T = 6, L = 3$ for CAIMS. The X-axis shows the size of the networks and the Y-axis shows the influence spread achieved. Figures 3a and 3b show that on SBM and PA networks, POMCP runs out of memory on networks of size 120 nodes. Further, these figures also show that CAIMS significantly outperforms Greedy and Greedy+ on both SBM (by ~73%) and PA networks (by ~58%). Figure 6a shows that even on real-world networks of homeless youth (which had ~160 nodes each) , POMCP runs out of memory, while CAIMS outperforms Greedy and Greedy+ by ~25%. This shows that state-of-the-art influence maximization solvers perform poorly in the presence of contingencies, and a POMDP based method (CAIMS) outperforms them by explicitly accounting for contingencies. Figures 3a and 3b also show that Greedy+ performs worse than Greedy.

Scale up Having established the value of POMDP based methods, we now compare CAIMS's scale-up performance against other POMDP solvers. Figures 4a and 4b compares the runtime of CAIMS, POMCP and SARSOP on a 100 node SBM network with increasing values of T and K respectively. The X-axis shows T (or K) values and the Y-axis shows the influence spread. Figure 4a shows that both POMCP and SARSOP run out of memory at $T = 2$ sessions. On the other hand, CAIMS scales up gracefully to increasing number of sessions. Similarly, Figure 4b ($T = 10$) shows that SARSOP runs out of memory at $K = 1$, whereas POMCP runs out memory at $K = 2$, whereas CAIMS scales up to larger values of K. These figures establish the superiority of CAIMS over its baselines as it outpeforms them over a multitude of parameters and network classes.

Markov Nets We illustrate the value of Markov networks to represent belief states in CAIMS. We compare CAIMS with and without Markov nets (in this case, belief states are represented using unweighted particle filters) on SBM networks of increasing size. Figure 5a shows influence spread comparison between CAIMS and CAIMS-Particle (the version of CAIMS which uses unweighted particle filters to represent belief states). Figure 5b shows runtime comparison of CAIMS and CAIMS-Particle on the same SBM networks. These figures shows that using a more accurate representation for the belief state (using Markov networks) improved solution qualities by ~15% at the cost of ~3X slower runtime. However, the loss in speed due to Markov networks is not a concern (as even on 160 node networks, CAIMS with Markov networks runs in ~75 seconds).

Real World Trial We conducted a real-world feasibility trial to test out CAIMS with a homeless shelter in a large American city. We enrolled 54 homeless youth from this shelter into our trial and constructed a friendship based social network for these youth (using social media contacts). The prior Φ was constructed using field observations made by shelter officials. We then executed policies generated by CAIMS, Greedy and Greedy+ on this network ($K = 4$, $Q_{max} = 4$ and $L = 3$) on three successive days ($T = 3$) in the shelter to invite homeless youth to attend the intervention. In reality, 14 out of 18 invitations (~80%) resulted in contingency events, which illustrates the importance of accounting for contingencies in influence maximization. Figure 6b compares influence spread (in simulation) achieved by nodes in invited sets selected by CAIMS, Greedy and Greedy+. This figure shows that CAIMS is able to spread 31% more influence as compared to Greedy and Greedy+.

CONCLUSION

This paper presents CAIMS, a contingency-aware influence maximization algorithm for selecting key influencers in a social network. Specifically, this paper makes the following five contributions: (i) we propose the Contingency-Aware Influence Maximization problem and provide a theoretical analysis of the same; (ii) we cast this problem as a Partially Observable Markov Decision Process (POMDP); (iii) we propose CAIMS, a novel POMDP planner which leverages a natural action space factorization associated with real-world social networks; (iv) we provide extensive simulation results to compare CAIMS with existing state-of-the-art influence maximization algorithms; and (v) we test CAIMS in a real-world feasibility trial which confirms that CAIMS is indeed a usable algorithm in the real world.

REFERENCES

[1] Christopher Amato and Frans A Oliehoek. 2015. Scalable Planning and Learning for Multiagent POMDPs.. In *Proceedings of the AAAI Conference*.

[2] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing Social Influence in Nearly Optimal Time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*. SIAM, 946–957.

[3] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[4] CDC. 2013. HIV Surveillance Report. www.cdc.gov/hiv/pdf/g-l/hiv_surveillance_report_vol_25.pdf. (March 2013).

[5] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F Werneck. 2014. Sketch-based Influence Maximization and Computation: Scaling up with guarantees. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 629–638.

[6] Daniel Golovin and Andreas Krause. 2011. Adaptive Submodularity: Theory and Applications in Active Learning and Stochastic Optimization. *Journal of Artificial Intelligence Research* 42 (2011), 427–486.

[7] Carlos Guestrin, Daphne Koller, and Ronald Parr. 2002. Multiagent planning with factored MDPs. In *Advances in neural information processing systems*. 1523–1530.

[8] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11 – 15.

[9] David Kempe, 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*. ACM, 137–146.

[10] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based Monte-Carlo Planning. In *Machine Learning: ECML 2006*. Springer, 282–293.

[11] Hanna Kurniawati, David Hsu, and Wee Sun Lee. 2008. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces.. In *Robotics: Science and systems*, Vol. 2008. Zurich, Switzerland.

[12] Norweeta G Milburn, Eric Rice, Mary Jane Rotheram-Borus, Shelley Mallett, Doreen Rosenthal, Phillip Batterham, Susanne J May, Andrea Witkin, and Naihua Duan. 2009. Adolescents Exiting Homelessness over two years: The Risk Amplification and Abatement Model. *Journal of Research on Adolescence* 19, 4 (2009), 762–785.

[13] Eric Rice. 2010. The Positive Role of Social Networks and Social Networking Technology in the Condom-using Behaviors of Homeless Young People. *Public health reports* 125, 4 (2010), 588.

[14] Eric Rice, Anthony Fulginiti, Hailey Winetrobe, Jorge Montoya, Aaron Plant, and Timothy Kordic. 2012. Sexuality and Homelessness in Los Angeles public schools. *American Journal of Public Health* 102 (2012).

[15] Eric Rice and Harmony Rhoades. 2013. How Should Network-based Prevention for Homeless Youth be Implemented? *Addiction* 108, 9 (2013), 1625.

[16] Eric Rice, Eve Tulbert, Julie Cederbaum, Anamika Barman Adhikari, and Norweeta G Milburn. 2012. Mobilizing Homeless Youth for HIV Prevention: a Social Network Analysis of the Acceptability of a face-to-face and Online Social Networking Intervention. *Health education research* 27, 2 (2012), 226.

[17] Katt Sammie, Frans Oliehoek, and Christopher Amato. 2017. Learning in POMDPs with Monte Carlo Tree Search. In *ICML17*. 1819–1827.

[18] John A Schneider, A Ning Zhou, and Edward O Laumann. 2015. A new HIV Prevention Network Approach: Sociometric Peer Change Agent Selection. *Social Science & Medicine* 125 (2015), 192–202.

[19] C Seshadhri, Tamara G Kolda, and Ali Pinar. 2012. Community Structure and Scale-free Collections of Erdős-Rényi Graphs. *Physical Review E* 85, 5 (2012), 056109.

[20] David Silver and Joel Veness. 2010. Monte-Carlo Planning in large POMDPs. In *Advances in Neural Information Processing Systems*. 2164–2172.

[21] Yaron Singer. 2012. How to win friends and influence people, truthfully: influence maximization mechanisms for social networks. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 733–742.

[22] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: Near-Optimal Time Complexity meets Practical Efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 75–86.

[23] Thomas W Valente and Patchareeya Pumpuang. 2007. Identifying Opinion Leaders to Promote Behavior Change. *Health Education & Behavior* (2007).

[24] Bryan Wilder, Amulya Yadav, Nicole Immorlica, Eric Rice, and Milind Tambe. 2017. Uncharted but not Uninfluenced: Influence Maximization with an uncertain network. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1305–1313.

[25] Amulya Yadav, Hau Chan, Albert Xin Jiang, Haifeng Xu, Eric Rice, and Milind Tambe. 2016. Using social networks to aid homeless shelters: Dynamic influence maximization under uncertainty. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 740–748.

[26] Amulya Yadav, Bryan Wilder, Eric Rice, Robin Petering, Jaih Craddock, Amanda Yoshioka-Maxwell, Mary Hemler, Laura Onasch-Vera, Milind Tambe, and Darlene Woo. 2017. Influence maximization in the field: The arduous journey from emerging to deployed application. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 150–158.