Activity Report: Targeted Network Discovery using a POMDP formulation

Zachary Sunberg
Summer 2015

***Abstract*—This report gives a short overview of the investigation into a POMDP approach to a Social Network exploration problem conducted during the summer of 2015. I attempted to use a Monte Carlo Tree Search approach to solve the following problem online. Consider a social network with each member assigned to one of M communities and edges between members generated according to a stochastic block model. Suppose that we are only given knowledge of a small subset of the edges and the community labels of a small subset of the vertices. We are also permitted to sequentially probe T of the vertices, learning the neighbors of each probed vertex. The goal is to correctly guess the community label of a single target vertex given the network structure observed by probing.**

I. Background

Exploration of social networks is a topic of active research. Many strategies have been proposed for sampling large graphs to accurately measure the properties of such graphs [1], [2], inferring community structure with incomplete information [3], [4], [5], [6], learning about individuals based on features and network relationships [7], [8], and selecting individual vertices to query to learn specific information about the graph [9], [10].

This work is most closely related to [10]. While these researchers attempted to identify graph characteristics such as the size of the largest connected component, vertex with the highest PageRank, and community structure based on a heuristic probing strategy, we attempt the less ambitious task of identifying the community affiliation of a single vertex. However, we adopt a principled approach modeling the problem as a Partially Observed Markov Decision Process (POMDP).

The POMDP framework is a system for modelling decision-making problems with uncertainty [11]. In a POMDP, a policy determines an action to take based on incomplete knowledge of the true system state. This incomplete knowledge is known as the belief. A reward is received every time an action is taken, and the value of the reward is determined by the action and true system state. The policy seeks to maximize this reward by judiciously selecting actions.

The general problem of gathering information by probing a network is similar to a POMDP in that the true state (network structure) is unknown. However, it is difficult to state the problem as a POMDP in a strict sense because the goal of probing is to minimize the uncertainty in the belief, but the reward in a true POMDP is determined by the true state rather than the belief. Nevertheless, there has been some written about problems that have POMDP structure except that the reward is a function of the belief state rather than the true state. Such problems are known as ρPOMDPs [12]. The goal in such problems is to minimize the uncertainty (evaluated as the entropy for example) in the belief. Unfortunately, for network problems, the state of the network is usually too large for calculation of the exact belief after taking an action and receiving an observation to be computationally practical. For example, the set of all possible network structures for an undirected graph with N vertices has a cardinality of $O\left(2^{N^2}\right)$ (each pair of vertices can either be connected or disconnected). Since the set of possible belief states is $[0,1]^{O\left(2^{N^2}\right)}$, even simply storage of the complete belief state is problematic.

Another way to evaluate the success of an exploratory probing policy is to have the policy make a guess at the answer to a question about the network after probing [13]. If the policy is usually able to answer this question correctly, then it is effective at exploring for the information in the question. For this research, the question answered by the policy after probing is "**Which community does the target vertex belong to?**". A potential real world example that this problem could represent is linking a terrorist to a terrorist organization.

Recent research explored the use of a Markov Decision Process (MDP) model for controlling the spread of infection on a network [14]. Monte Carlo Tree Search (MCTS) was the most successful of the approaches used, so we used a variant of it on this problem. In the following sections, the problem formulation is given, followed by a description of the solution approach.

II. Problem Description

The objective is to efficiently explore a network by probing vertices to determine the network affiliation of a target vertex. The true network, $G_{\text{true}} = (\mathcal{V}_{\text{true}}, \mathcal{E}_{\text{true}})$, is randomly generated and consists of $N = 100$ vertices. Each vertex is assigned uniformly randomly to one of $M = 5$ groups. This mapping from vertices to communities will be referred to as $\mathcal{C}_{\text{true}} : \mathcal{V}_{\text{true}} \to \{1 \dots M\}$. Edges are generated according to a stochastic block model (SBM) [15], that is, two vertices within the same community are connected with a probability of $p_{\text{inter}} = 0.3$ and vertices not in the same community are connected with a probability of $p_{\text{intra}} = 0.01$. An example network generated according to this method is shown in Figure 1.

The decision making agent is initially given partial information about the network. The graph $G_{\text{samp}} = (\mathcal{V}_{\text{samp}}, \mathcal{E}_{\text{samp}})$ is randomly sampled from G_{true} with $|\mathcal{V}_{\text{samp}}| = N_0 = 10$ and $|\mathcal{E}_{\text{samp}}| = E_0 = 10$. Also a subset of $\mathcal{C}_{\text{true}}$, $\mathcal{C}_{\text{samp}}$, is revealed, specifically the affiliations of all of the vertices in $\mathcal{V}_{\text{samp}}$ are

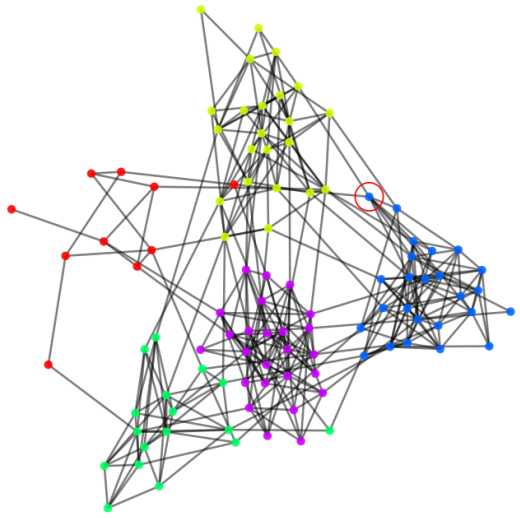

Fig. 1. An example social network. Community membership is indicated by the vertex's color. The target node is indicated by a red circle.

given. The agent also knows the number of vertices and communities in the network and statistical information used to generate the network, that is p_{inter} and p_{intra}. In addition, the agent may choose T vertices sequentially, and, after observing a vertex, gain full knowledge of all of the edges connected to that vertex.

After observing all T vertices, the final action taken by the agent is to guess the answer to the question, and a reward is given if the answer is correct. This problem may be stated as a POMDP. A POMDP is defined by the tuple (S, A, T, R, O, Z):

- The **state space**, S, is the set of all possible social networks involving the N members of the network and their community affiliations. The true state is the true social network and community mapping, that is $s = (G_{\text{true}}, \mathcal{C}_{\text{true}})$.
- The **action space**, A, is the set of all possible actions at each step of the simulation. For each time step in $1..T$, the action is the choice of which vertex to query from the list of known vertices. At time $T+1$, the action is a guess of the community label of the target vertex.
- The **transition probability distribution**, $T : S \times A \times S \to [0, 1]$ encodes how the state changes by specifying the probability of state s transitioning to s' given action a. In this problem, the state remains constant, so the probability distribution is simply an indicator function, $T(s, a, s') = \mathbf{1}(s = s')$. However, it would be reasonable for the social network to change slowly over time. This can be easily modeled in the POMDP framework and should be investigated in the future.
- The **reward function**, $R : S \times A \to \mathbb{R}$ returns the reward gained by executing action a in state s. Since the only reward is given for guessing the correct community affiliation of the target vertex at the end of the problem, it is defined as

$$R(s_t, a_t) = \begin{cases} 100 \cdot \mathbf{1}(a_t = \mathcal{C}_{\text{true}}(\text{target})) & \text{if } t = T+1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

  In future problem formulations, there may also be costs associated with measuring particular vertices.
- The **observation space**, O, is the set of all possible observations that could be produced after taking action a in state s. At each time step, the observation is the set of neighbors of the vertex probed as specified by the action.
- The **observation distribution**, $Z : A \times S \times O \to [0, 1]$ yields the probability of observation o given that action a was taken and the resulting state is s'. Since the observations are deterministic, this is an indicator function, $Z(a, s', o) = \mathbf{1}(o = \text{neighbors}_{s'}(a))$. Noisy measurements could be incorporated into the model easily by spreading out this distribution.

The objective of a POMDP is to find an optimized policy, π^*, that will specify an action given the initial information about the problem and the sequence of actions and observations received up to the current time. Let I_t be the information vector at time t, that is, the information known about the network at the beginning of the problem, and the sequence of actions and observations up to time t,

$$I_t = (G_{\text{samp}}, \mathcal{C}_{\text{samp}}, a_0, o_1, a_1, \ldots a_{t-1}, o_t). \tag{2}$$

The optimization objective is

$$\begin{aligned} \underset{\pi}{\text{maximize}} \quad & R\left(s_{T+1}, \pi(I_T)\right) \\ \text{subject to} \quad & I_{t+1} = (I_t, \pi(I_t), o_t), \quad t = 0 \ldots T-1 \\ & I_0 = (G_{\text{samp}}, \mathcal{C}_{\text{samp}}) \end{aligned}$$

III. Solution Approach

In order to evaluate the effectiveness of using a POMDP approach for this problem, we compare it to a simple heuristic which will be described first, followed by the description of the POMDP solver.

A. Heuristic

The heuristic approach selects vertices to probe without considering the future consequences, that is, it is a myopic policy. It simply probes the vertices with the highest observed degree, a strategy that was reported to be successful for different tasks in [10]. This is not a particularly good approach for the task at hand; it would likely be easy to specify a much better heuristic. However it serves as a baseline with which to compare strategies.

Specifically, at steps 1 through T, this heuristic policy considers all of the vertices and edges that have been revealed and chooses to probe the unprobed vertex with the highest revealed degree, that is, the largest number of revealed incident edges. At step $T+1$, the policy must make a guess of the target vertex's affilitation. To do this, it first considers each neighbor, n, of the target vertex. If n is unlabeled, the policy

labels it with the label that is most common in the neighbors of n, breaking ties randomly. If n has no labeled neighbors, it is left unlabeled. Once all neighbors of the target are labeled, the target label is guessed to be the most common label among the target neighbors with ties again broken randomly.

B. Monte Carlo Tree Search

The POMDP solution approach is Monte Carlo Tree Search (MCTS) [16] on the belief state space. This is an online approach, that is, it carries out the computations to determine the best action at time t *after* all of the observations up to time t have been received instead of calculating the best response to all potential observations beforehand (an offline solution). In order to do this, the algorithm uses Monte Carlo simulations to construct a tree with alternating layers of nodes corresponding to actions and observations. Each observation node corresponds to a belief, and each action node has a corresponding value estimate, V_a. This value estimate represents the expected reward accumulated by taking the action and following the optimal policy in the future. At each iteration, a new set of action nodes is added to the tree. Rollout simulations using a heuristic rollout policy provide unbiased estimates of the value. Details of the algorithm can be found in the journal article describing the POMCP algorithm [17], though this algorithm is not strictly POMCP because it does not use the particle filter belief updates described in the paper.

The expansion of the tree is carried out in an optimistic manner, i.e. the actions that yield higher rewards are explored more thoroughly. However, if the action with with the highest value were chosen to be explored every time, the algorithm would not perform well because of under-exploration. In order to mitigate this problem, the "Upper Confidence Bound for Trees" (UCT) modification is used [16]. In this approach, the algorithm explores the action that has the highest score according to

$$V_a + c\sqrt{\frac{\log(N_o)}{N_a}}, \tag{3}$$

where c is an adjustable parameter controlling the greediness of the exploration, N_o is the number of times that the parent observation node of the action node has been observed, and N_a is the number of times that the action node has been chosen.

Typically, instead of using the action-observation history, I, directly as feedback, the policy makes its decision based on a belief representation, b, that is a probability distribution over S that is a sufficient statistic for the action-observation history (that is, it contains the same information as the history) [11]. As mentioned before, an exact belief distribution over all possible graph structures is much too large to be practical. Because of this, we use a belief representation that consists of a list of all the edges and vertices observed in the initial sample and the subsequent probes.

In order to run the Monte Carlo simulations, this belief is "sampled" by generating a complete network (state) in the following manner. First the edges in the original sample, $\mathcal{E}_{\text{samp}}$, and the edges observed by probing are added to the sampled network. Next, the community affiliation of the vertices *that are neighbors of the known community nodes in* $\mathcal{C}_{samp}$ is inferred using p_{intra}, p_{inter}, and Bayes rule. The community affiliations of the remaining vertices are assigned randomly, and the remaining edges are generated according to the stochastic block model with p_{intra} and p_{inter}.

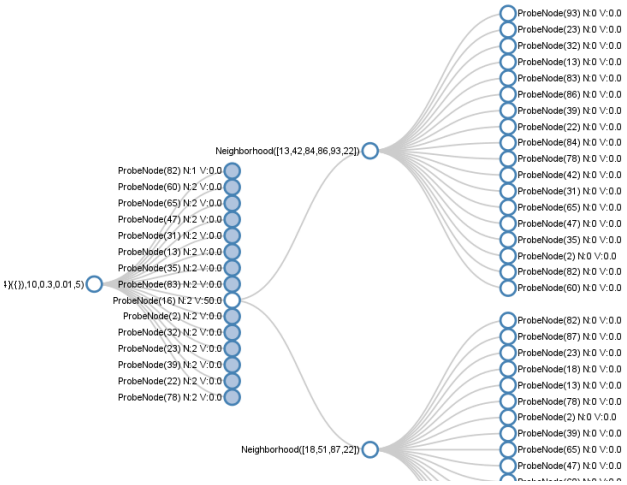

Fig. 2. Example MCTS Tree after 30 iterations. The node on the left is the root node. The first column of nodes correspond to actions. All of these nodes have been explored (N¿0 for all of them), but their subtrees are not shown for clarities sake. The displayed subtree shows the next level of observation nodes and action nodes.

When simulations are continued beyond the depth of the tree, a rollout policy must be used to determine the actions for the simulation. The rollout policy used in our work is the highest degree heuristic described in Section III-A.

IV. Results

In a set of 100000 tests, the heuristic was able to guess the correct target vertex affiliation about 41% of the time (random guesssing would result in a 20% success rate.

Small mistakes in the problem implementation were repeatedly discovered, so unfortunately none of the results produced by the MCTS solver can be reported with confidence in their correctness. At the end of the research period, the MCTS solver was not scoring better than the expected score for random guessing (20).

A visualization of the tree that MCTS constructed for one of the problems is shown in Figure 2. This visualization was captured after only a small number (30) tree exploration iterations, so it is much smaller than the trees actually used for decision making.

The code used for this project is available at https://github.com/sisl/NetworkDiscovery.jl.

V. Future Work

There are a variety of directions for this work to be carried into. The first task would be deeper investigation into why MCTS is not working correctly for the current problem. The heuristic could also be improved and tailored more specifically to this problem for better performance. For example, a heuristic that queries the neighbors of the target

node would probably work very well. The problem could also be made more difficult within the current framework by allowing the network to change slowly over time or producing noisy observations. Finally the POMDP approach may be extended to other problems, for example the goal could be to probe nodes for the purpose of guessing the labels of as many nodes as possible.

REFERENCES

[1] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636, 2006.

[2] C. Seshadhri, Ali Pinar, and Tamara G. Kolda. Wedge sampling for computing clustering coefficients and triangle counts on large graphs. *Statistical Analysis and Data Mining*, 7(4):294–307, 2014.

[3] R Caceres, K Carter, and Jeremy Kun. A Boosting Approach to Learning Graph Representations. *arXiv preprint arXiv:1401.3258*, 2014.

[4] Jialu Liu and Jiawei Han. On Integrating Network and Community Discovery. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 117–126, 2015.

[5] Arun S. Maiya and Tanya Y. Berger-Wolf. Sampling community structure. *Proceedings of the 19th international conference on World wide web - WWW '10*, pages 701–710, 2010.

[6] Se-Young Yun and Alexandre Proutiere. Community detection via random and adaptive sampling. *arXiv preprint*, pages 1–39, 2014.

[7] Mustafa Bilgic and Lise Getoor. Active Learning for Networked Data. *Computer*, 411(29-30):2712–2728, 2010.

[8] Jj Pfeiffer Iii, J Neville, and Pn Bennett. Active sampling of networks. *Proceedings of the Workshop on Mining and Learning with Graphs (MLG-2012)*, 2012.

[9] Xuezhi Wang, Roman Garnett, and Jeff Schneider. Active search on graphs. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, page 731, 2013.

[10] Sucheta Soundarajan, Tina Eliassi-rad, Brian Gallagher, and Ali Pinar. Strategies for Probing Incomplete Graphs. 2014.

[11] Mykel J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Practice*. MIT Press, 2015.

[12] M Araya-López, O Buffet, and V Thomas. A POMDP Extension with Belief-dependent Rewards. *Nips*, (i):1–21, 2010.

[13] L Mihaylova, T Lefebvre, H Bryunincks, and J De Schutter. A Comparison of Decision Making Criteria and Optimization Methods for Active Robotic Sensing. pages 316–324, 2003.

[14] Christopher Ho, Mykel J. Kochenderfer, Vineet Mehta, and Rajmonda S. Caceres. Control of Epidemics on Graph. In *IEEE Conference on Decision and Control*, 2015.

[15] Yuchung J Wang and George Y Wong. Stochastic Blockmodels for Directed Graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987.

[16] CB Browne and Edward Powley. A survey of monte carlo tree search methods. *IEEE Transactions on computational intelligence and ai in games*, 4(1):1–43, 2012.

[17] D Silver and J Veness. Monte-Carlo Planning in Large POMDPs. In *Advances in neural information processing systems*, pages 2164—-2172, 2010.