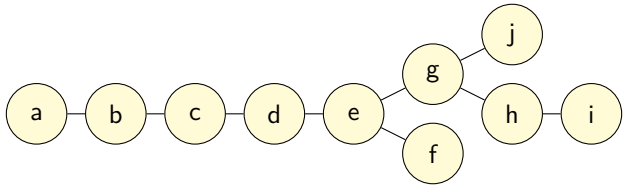
Control of Epidemics on Graphs

Christopher Ho, Mykel J. Kochenderfer, Vineet Mehta, and Rajmonda S. Caceres

July 16, 2015

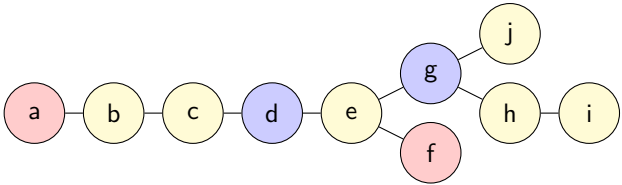
The Lincoln Laboratory portion of this work was sponsored by the Office of the Assistant Secretary of Defense for Research and Engineering under Air Force contract number FA8721-05-C-0002. Interpretations, opinions, and conclusions are those of the authors and do not reflect the official position of the United States government. The Stanford portion of this work was sponsored by the Office of the Assistant Secretary of Defense for Research and Engineering.



Susceptible-Infected-Susceptible (SIS) Model

- Each node is either susceptible or infected
- β_{ij}: infection transmission probability from node i to node j
- δ_i: recovery probability at node i

Targeted Control



Control Node

Target Node

Collateral Node

Objectives:

- Minimize control effort at control nodes
- Maximize infection of target nodes
- Minimize infection at collateral nodes

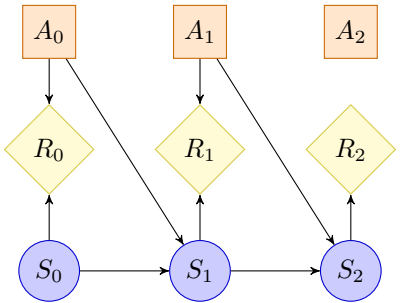
Applications:

- developing defensive strategies in social, biological, and financial networks
- designing biomedical therapeutic intervention strategies
- luring cyber attacks to “honey pots” for study

Outline

- Problem formulation
- Solution methods
- Experiments
- Conclusions

Markov Decision Process (MDP)



- Agent chooses action A_t at time t based on observing state S_t
- Markov assumption: State evolves probabilistically based on current state and action
- Defined by transition model $T(s' \mid s, a)$ and reward model $R(s, a)$
- Objective is to maximize sum of rewards $R_0, \ldots$

MDP Formulation

- V set of nodes, V_T set of targets, V_C set of controls
- $N(i)$ set of neighbors of nodes i (including itself)
- $\mathcal{S} = \times_{i=1}^{n} \mathcal{S}_i$ with $\mathcal{S}_i = \{0, 1\}$ (1 if infected, otherwise 0)
- $\mathcal{A} = \times_{i=1}^{n} \mathcal{A}_i$ with $\mathcal{A}_i = \{0, 1\}$ if $i \in V_C$, otherwise $\mathcal{A}_i = \{0\}$
- $T(\vec{s'} \mid \vec{s}, \vec{a}) = \prod_i T(s'_i = 1 \mid s_{N(i)}, a_i) =$
  $\prod_i \left[(1 - a_i)(1 - s_i) \prod_{j \in N(i)} (1 - s_j \beta_{ji}) + (1 - a_i)\delta_i s_i \right]$
- $R(\vec{s}, \vec{a}) = -||1 - s_{V_T}||_1 - \lambda_1 ||s_{V \setminus V_T}||_1 - \lambda_2 ||\vec{a}||_1$

Policies and Utilities

- A policy π specifies what action to execute from every possible state
- Action to execute from state s according to π is denoted $\pi(s)$
- Expected utility (sum of rewards) for executing π when starting from s is denoted $U^{\pi}(s)$
- Optimal policy π^* is one that maximizes expected utility $\pi^*(s) = \arg\max_{\pi} U^{\pi}(s)$

Computing Utility Function

- Incrementally compute expected utility after k steps of executing π
- $U_0^\pi(s) = 0$
- $U_1^\pi(s) = R(s, \pi(s))$
- ...
- $U_k^\pi(s) = R(s, \pi(s)) + \sum_{s'} T(s' \mid s, \pi(s)) U_{k-1}^\pi(s')$

Computing Optimal Policies

- Policy iteration is one way to compute an optimal policy π^*
- The algorithm starts with any policy π_0 and iterates the following steps
  1. Policy evaluation: given π_i compute U^{π_i}
  2. Policy improvement: compute new policy from U^{π_i}
     $\pi_{i+1}(s) = \arg\max_a [R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) U^{\pi_i}(s')]$
- Algorithm terminates when there is no more improvement
- Since every step leads to improvement and there are finitely many policies, algorithm terminates at optimal solution

Fundamental problem for graph control:
State space is exponential in number of nodes

Solution Methods

- Approximate Linear Programming
- Mean Field Approximate Policy Iteration
- Model Predictive Control
- Monte Carlo Tree Search

Experimental Configuration

Network structure:

- k-regular
- Erdős-Rényi
- Power-Law
- Stochastic Block Model (SBM)

Choice of Control Nodes:

- Maximum Out-Degree
- Approximate Power Dominating Set
  (low in-degree nodes, break ties in favor of high out degree)
- Uniform random

Parameters:

- Single target homogeneous diffusion ($\beta_{ij} = 0.2$)
- Three target homogeneous diffusion ($\beta_{ij} = 0.2$)
- Three target preferential path heterogeneous diffusion
  ($\beta_{ij} = 0.8$ on shortest path from a control to target and 0.1 elsewhere)

Approximate Linear Programming (ALP)

Approach:

- Assume utility function is additive $U(\vec{s}) = \sum_i U_i(s_i)$
- Solve for each local value function (and policy) using n linear programs (LPs)
- Each LP has 3 variables and $2 \times |S_{N(i)}| \times |A_i|$ constraints

Results:

- Scalability and simplicity has made this a standard method for graph-based MDPs
- Can work well depending on the problem
- For targeted control, it only sees additional cost for control, resulting in "do nothing" policies

Mean Field Approximate Policy Iteration (MFAPI)

Approach:

- Assume utility function is additive over neighborhoods $U(\vec{s}) = \sum_i U_i(s_{N(i)})$
- Solve for each local value function (and policy) using policy iteration
- Approximate local transition probabilities, $p_i(s_i' \mid s_{N(i)}, a_i)$ as $p_i(s_i' \mid s_i, a_i)$ (mean field approximation) for local policy evaluation rule

Results:

- Slower than ALP, but still relatively fast so long as $|N(i)|$ is small
- Similar to pure diffusion heuristic; just infect all uninfected control nodes
- Since targets are > 2 steps away from controls, controls do not observe status of targets
- Results in unnecessary control

Model Predictive Control (MPC)

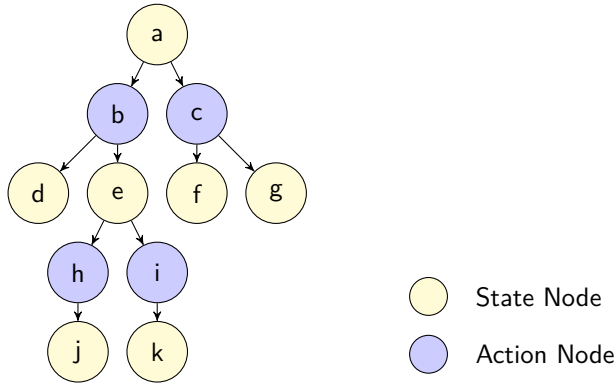
Approach:

- Attempts to find optimal "open-loop"' sequence of actions up to some horizon
- Infection probabilities evolve deterministically over time according to $P(s_i(t+1)=0) = (1-s_i(t))(1-a_i(t))\prod_j(1-s_j(t)\beta_{ji}) + s_i(t)(1-a_i(t))\delta_i$
- Used ISRES algorithm implemented in NLopt package
- Allowed to run 20 seconds from three random initial guesses
- Five-step lookahead horizon
- After first action is sequence is taken, repeat optimization

Results:

- Generally infects at first time step (when there is currently no infection), then does nothing
- Performance limited because plans do not account for future information it will receive

Monte Carlo Tree Search (MCTS)



- MCTS involves building a variable-depth search tree incrementally
- Associated with each node is a count of the times visited and the expected value
- Search traverses tree based on upper confidence bound
  (computed from counts and expected value)

Sparse Upper Confidence Tree (sUCT)

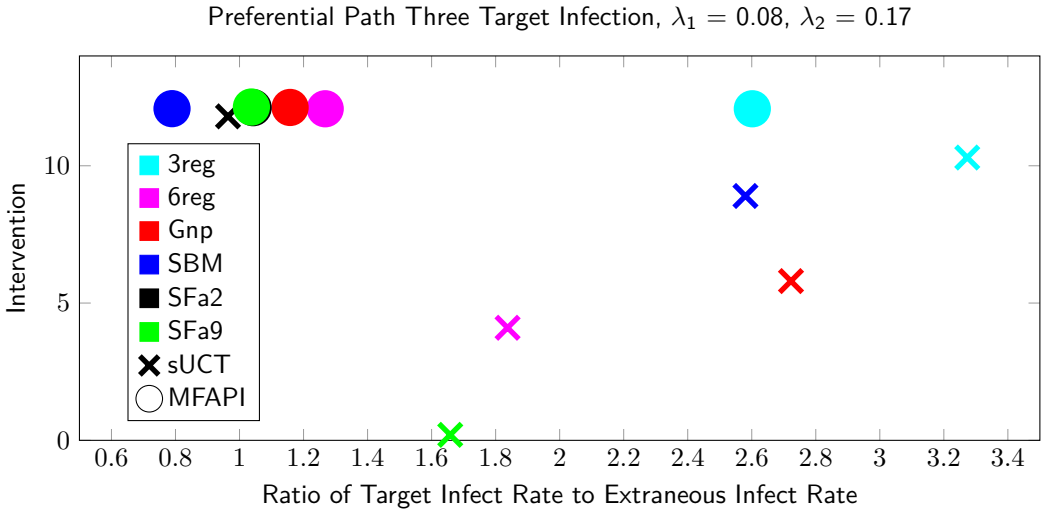
Approach

- To better handle cases with large branching factors, sUCT samples a limited number of next states
- Average action values from current state over 3 independent trees
- For each tree, 160 trajectories were sampled with a rollout depth of 20 time steps following a heuristic policy

Results

- Provided superior results over basic MCTS using the same total number of trajectories with minimal overhead

Performance using Max-Degree



Discussion

- For scale-free networks, the choice of control nodes does not significantly affect performance
- For graphs with community structure, picking control nodes from the PDS sets (central to communities) seems to be most effective

Run Time

Method	Max Degree	100 nodes	200 nodes	400 nodes
sUCT		5.8 ± 1.21	15.3 ± 0.58	48.8 ± 1.63
MFAPI	3	1.05 ± 0.46	1.93 ± 0.96	2.62 ± 2.15
MFAPI	4	4.17 ± 1.34	6.23 ± 3.58	12.07 ± 9.23
MFAPI	5	12.34 ± 2.24	23.95 ± 2.47	68.66 ± 8.3
ALP	3	1.5 ± 0.09	1.8 ± 0.19	2.14 ± 0.38
ALP	4	1.7 ± 0.34	3.3 ± 0.47	4.37 ± 0.80
ALP	5	2.2 ± 0.49	4.37 ± 1.38	9.04 ± 1.77

Note that the run time for MCTS is insensitive to max degree because it considers the global state information without any assumptions or decompositions applied

Conclusions

- Local policies and solution techniques scale well, but are not expressive enough for targeted control
- While MCTS does not scale well, it provides a first step towards a more scalable solution for graph control

Future Work

- Solution techniques that can handle larger action spaces (e.g., edge control)
- Scalable solution techniques
- Partial observability of node states (e.g., missing or noisy node information)