Reinforcement Learning For Adaptive Traffic Signal Control With Limited Information

Machine Learning (CS 229) Final Project, Fall 2015

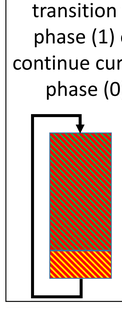
Jeff Glick (jdglick@stanford.edu), M.S. Candidate, Department of Management Science and Engineering, Stanford University

Motivation

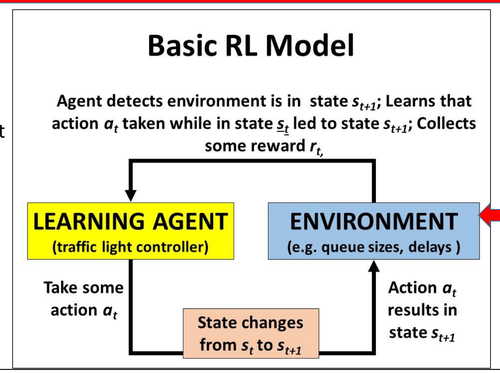
Recent research has effectively applied reinforcement learning to adaptive traffic signal control problems. Learning agents learn best with a high level of intelligence about what state the environment is in to determine the right actions for a given state. Most researchers have provided this access to perfect state information. However, in a real-life deployment, this level of intelligence would require extensive physical instrumentation. This study explores the possibility of training a control agent that has only access to information from a limited number of vehicles using cell phone geo-location data in the interest of comparing performance against legacy fixed phase timing policy and under regimes where the agent has access to perfect information.

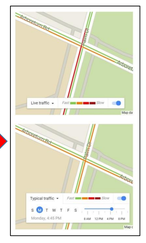
Reinforcement Learning Cycle

Action: Every s seconds, transition to phase (1) or continue current phase (0)

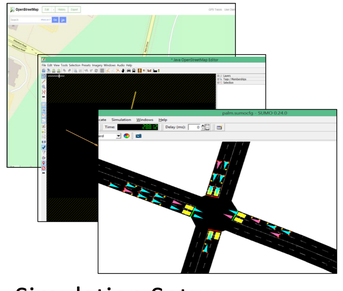


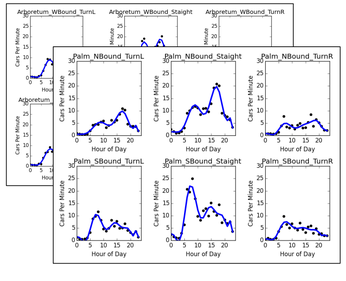
Basic RL Model





Simulation Build & Data Generation



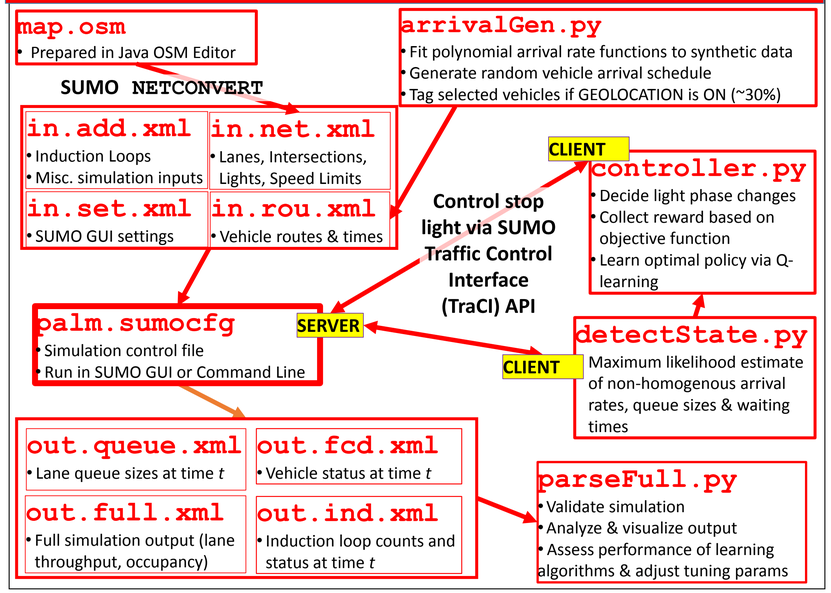


Simulation Setup:

- Open Street Map and Java Open Street Map Editor
- Simulation for Urban Mobility (SUMO)

- Using realistic, variable arrival and turn rates for a single 8-phase 4-way intersection (`arrivalGen.py`)

Simulation Architecture & Learning Pipeline



Algorithm & Key Parameters

- Q-learning develops **quality-values** $\boldsymbol{Q(s,a)}$ for each pair $\boldsymbol{(s,a)}$ which is an estimate for the true value $\boldsymbol{V(s,a)}$
- Continuous **asynchronous updating** for **on-line learning;** assume infinite visits to states for convergence
- **Q-learning update**:

$$\boldsymbol{Q(s,a) := (1 - \alpha)Q(s,a) + \alpha(r + \gamma \max_{a' \in A} Q(s',a'))}$$

- **States:** Discretized state space; number of states for problem:

$$(\#\ light\ phases) * [(\#\ queue\ sizes) * (\#\ waiting\ times)]^{(\#\ edges)}$$

- **Learning Rate** α**:** Initially $\alpha = 1$ ignores previous experience; As $\alpha \to 0$, we are weighting previous experience more heavily

$$\alpha(s,a) = \frac{1}{n(s,a)} = 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$$

- **Discount factor** γ**:** Use $\gamma \approx 1$ to prevent myopic decision making
- **Control policy**: Given a state s, try action a with probability:

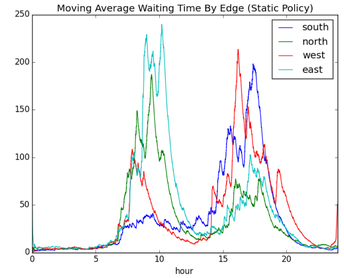
$$p_s(a) = \frac{e^{Q(s,a)/\tau}}{\sum_{a \in A} e^{Q(s,a)/\tau}}$$

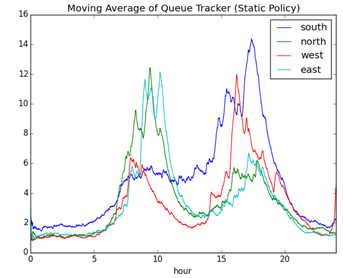
(soft max distribution) where τ controls exploration; if τ is large, actions are chosen with $\approx$ equal $p_s(a)$. As $\tau \to 0$, policy becomes deterministic and choose $\max_{a \in A} Q(s,a)$ with probability of 1.

- **Reward** r**:** determined by objective function:

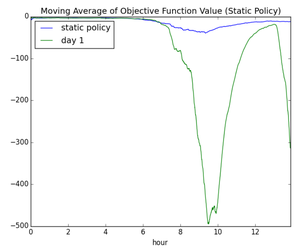
$$r = \sum_{edge\ i=1}^{4} \beta_q (queue\ size)_i^{\theta_q} + \beta_w (waiting\ time)_i^{\theta_w}$$

Initial Results

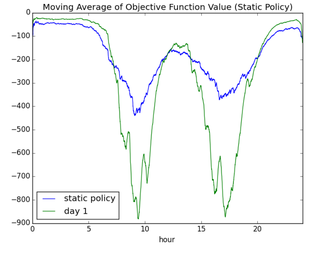




- Validated traffic dynamics; Selected bucket thresholds for discrete queue sizes and waiting times



- Queues blowing up
- Learning rate α shrinking quickly
- Crude discretization (most of 25k states not being visited)
- Challenges with volatility
- Reward should = change in objective function (reward improvement)



- Throttled learning rate $\alpha(s,a)$; system still performing better during off-peak;
- Some improvement by increasing bucket thresholds, delaying the progression of the learning rate
- Increased τ (important when rewards are negative)
- Still performance issues; MDP assumption may not hold

Next Steps

- Continue to experiment with learning strategy, parameters and objective function; improve discretization
- Work on state detection problem (limited information); learn arrival rates or use hour of day in the state space
- Change arrival rate dynamics to test robustness of process

Acknowledgements: Michael Bennon (Stanford Global Projects Center), Allen Huang (CS229 Student), Jesiska Tandy (CS229 Student)