QUINNRESEARCHGROUP

REINFORCEMENT LEARNING

IN THEORY, IN PRACTICE

Roi Ceren

Snr. Mgr. Data Science @ Cox Auto
roi.ceren@gmail.com
LI: roi-c

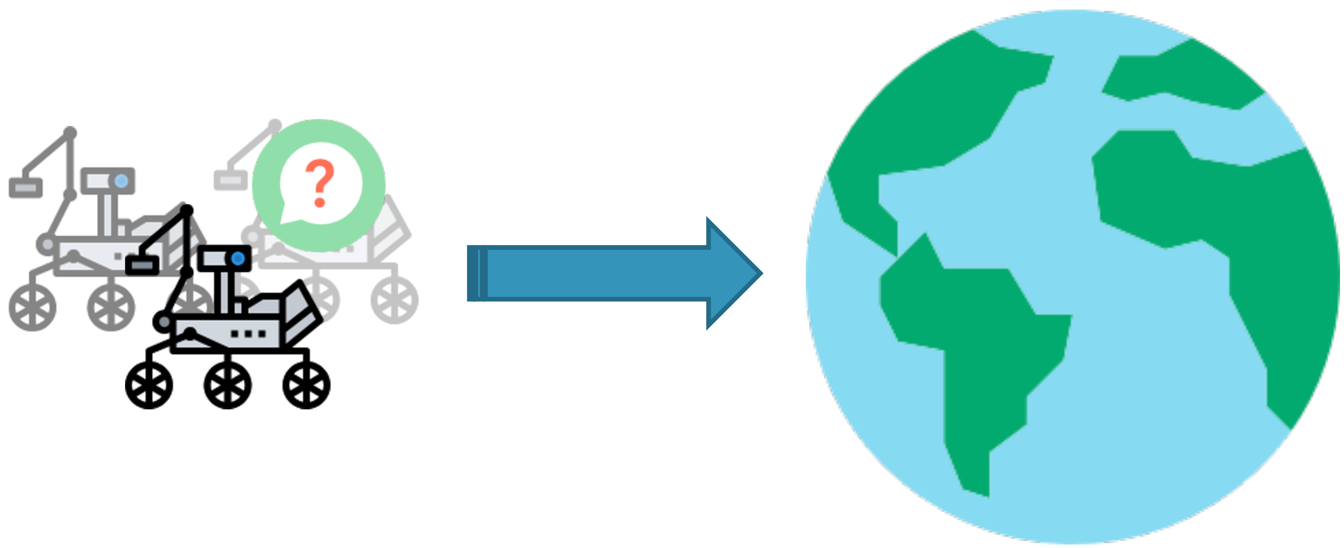
OUTLINE

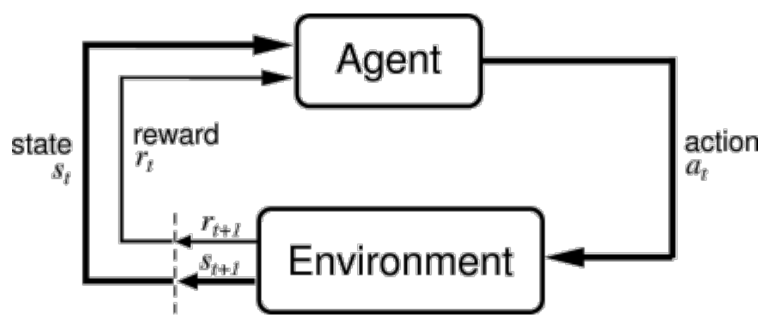
- RL
- Decision Processes
- Q-Learning
- PAC Learning
- Application: RL in Sales Domain
- AWS

REINFORCEMENT LEARNING

IN AGENT-BASED REASONERS

RL BACKGROUND





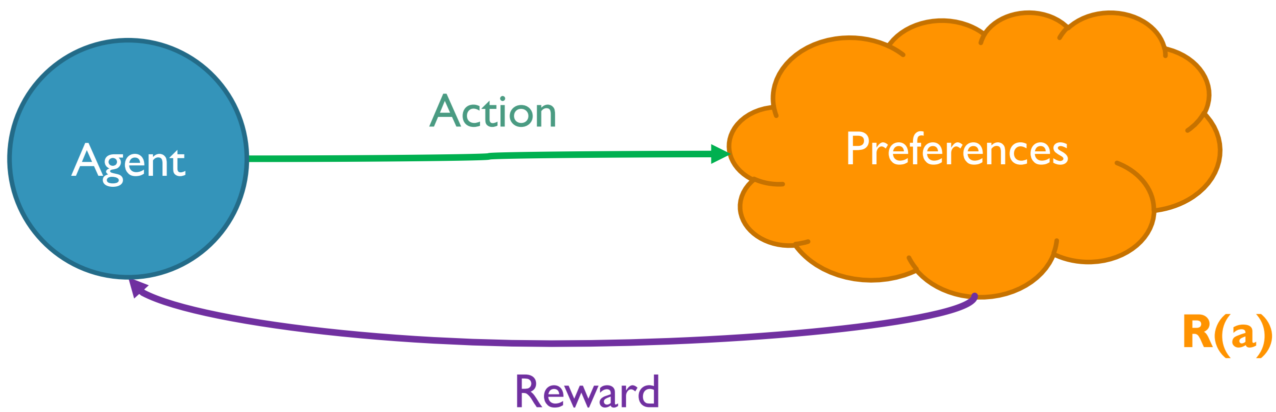
QUINNRESEARCHGROUP

Credit: MASPlan.org, flaticon.com

DECISION PROCESSES

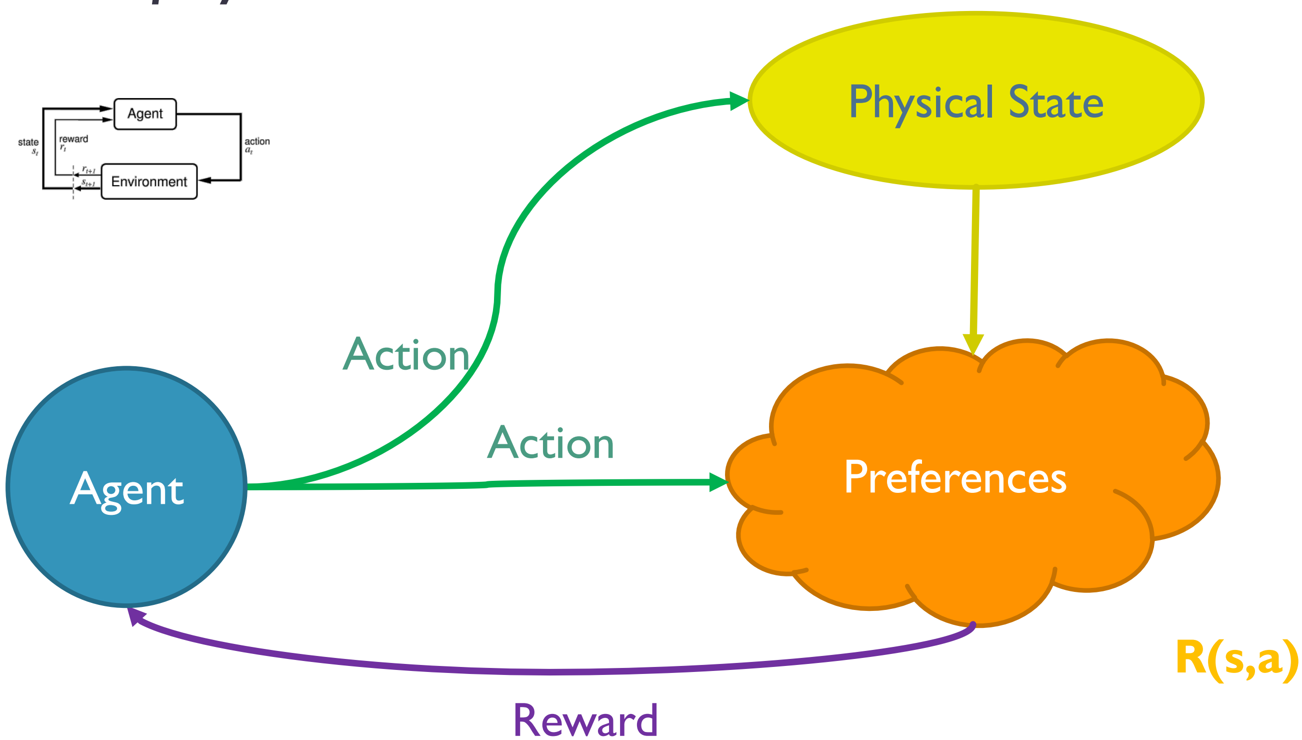
Decision problem: how to optimize behavior to maximize reward?

- Choose the action that has the best expected outcome

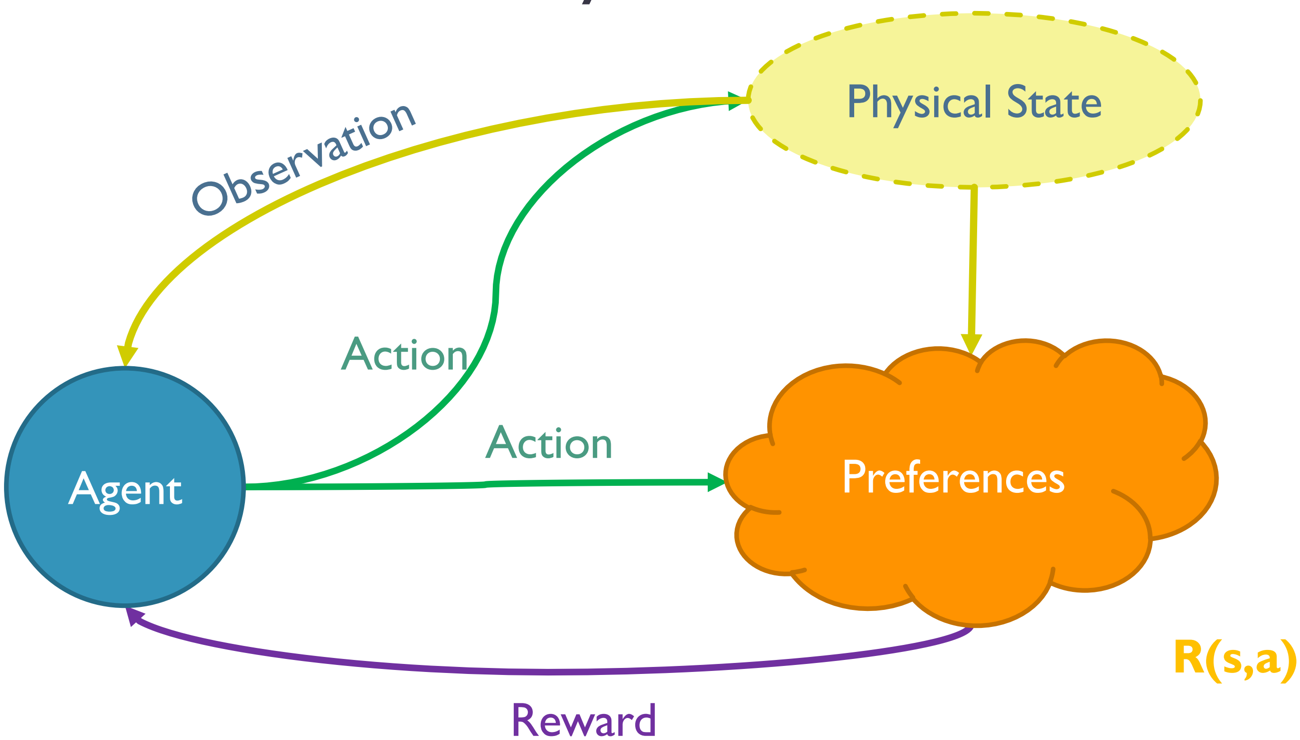


DECISION PROCESSES

Rewards may be based on more than just the action, but also the *physical state*

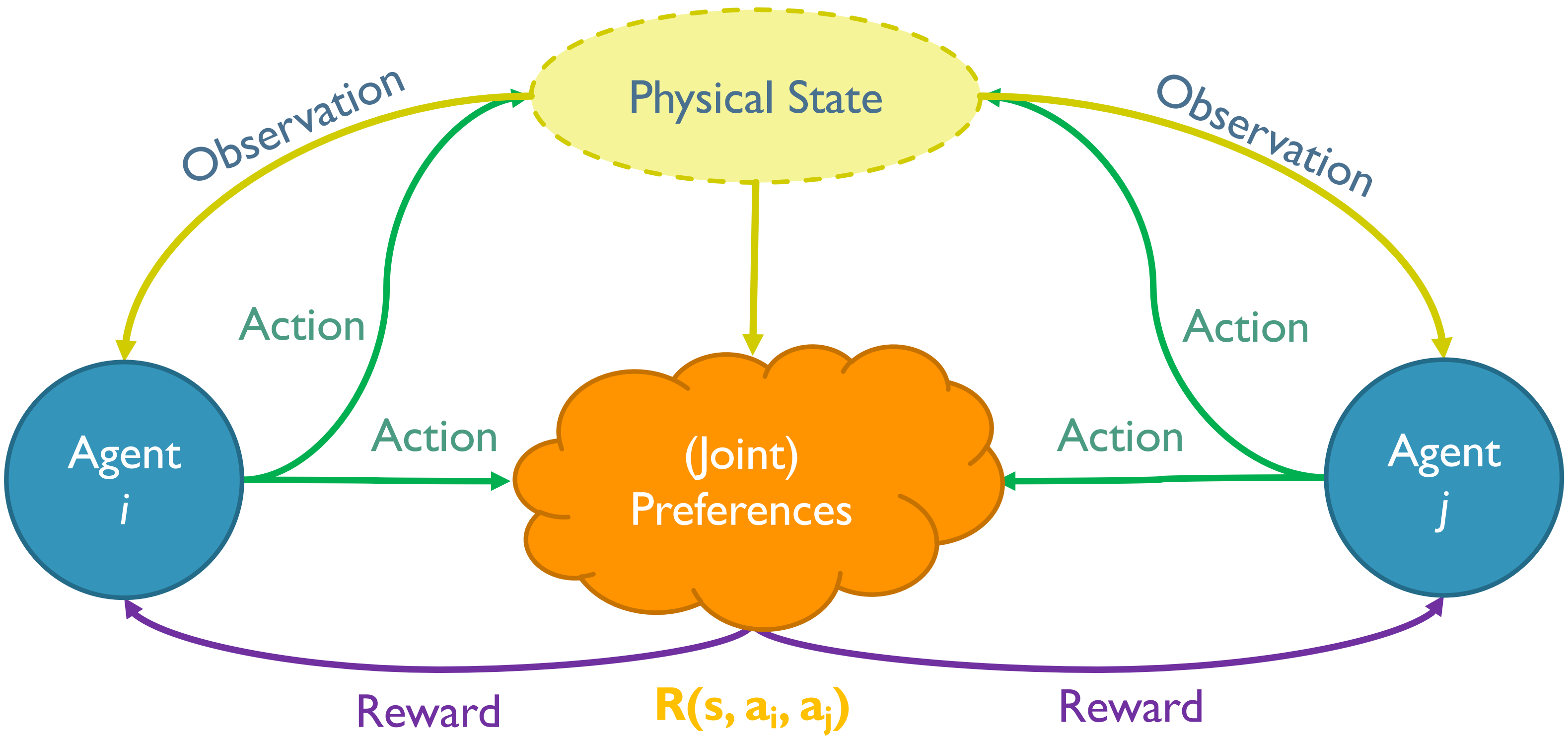


Sometimes the physical state is unknown, but the agent gets a clue as to where they are



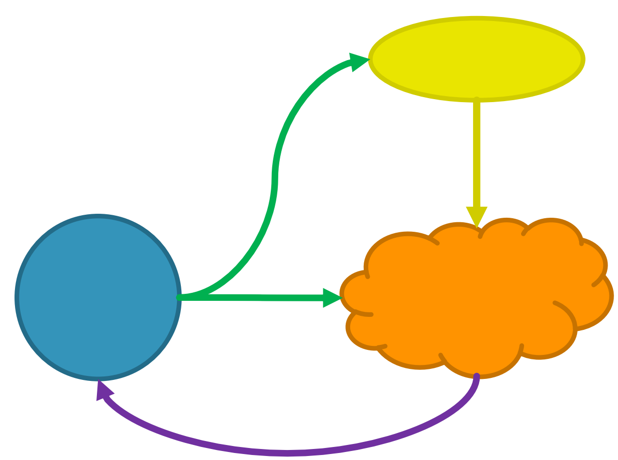
DECISION PROCESSES

In the *multiagent* setting, additional agents affect the reward for each agent and the state



DECISION PROCESSES

- The **Markov Decision Process** (MDP)
  <S,A,T,R>
  - **S:** Set of physical states
  - **A:** Set of actions
  - **T: S x A x S → [0,1]**: State transition function
  - **R: S x A → R**: Reward function



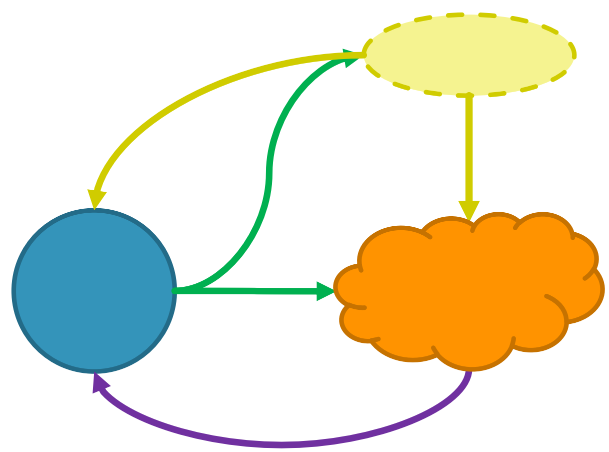
DECISION PROCESSES

- The **Partially Observable MDP** (POMDP) <S,A,T,Ω,O,R>
  - **Ω:** Set of observations
  - **O: S x A x Ω→[0,1]**: Likelihood of an observation from a state

- **States** are unknown!

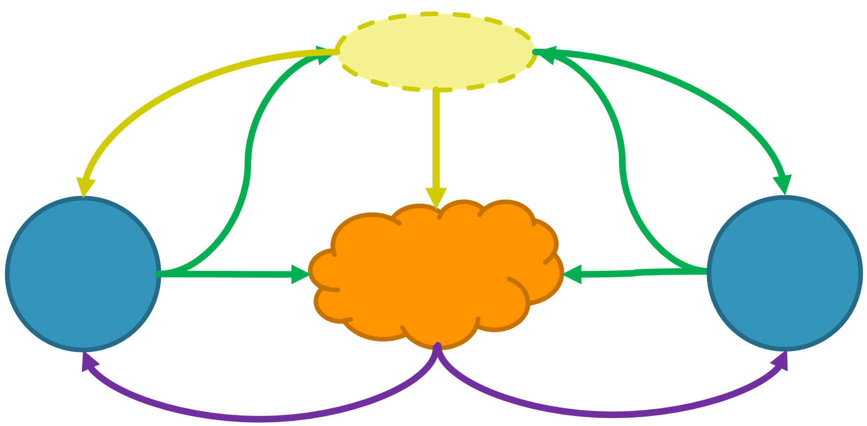
$$\boldsymbol{b_i^t(s)} = \beta O(o^t, s, a^{t-1}) \sum_{s' \in S} b_i^{t-1}(s') T(s, a, s')$$

$$\boldsymbol{R(b, a)} = \sum_{s \in S} b(s) R(s, a)$$



DECISION PROCESSES

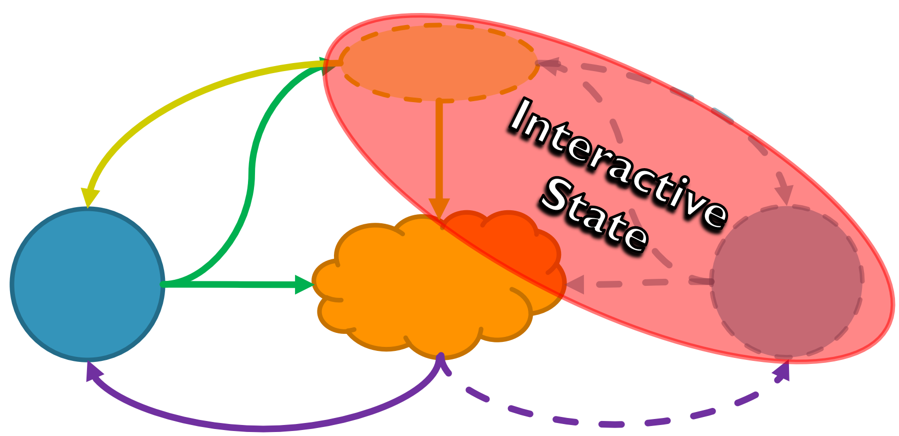
- The **Multiagent POMDP** (MPOMDP)
  <S,A,T,Ω,O,R>
  - *Cooperative:* Agents get identical, often joint, rewards
  - **A**: Joint action of all agents
  - **O,T**: Maps joint actions and state to new state
- *Functionally, the MPOMDP may be solved as POMDP, where the action space is increased by agents exponentially*



DECISION PROCESSES

- The **Interactive POMDP** (IPOMDP)
  <IS,A,T,Ω,O,R>
  - *Non-cooperative*: Individual, potentially competitive rewards
    - **R** is an *individual* reward, still based on the *joint* action
  - **IS:** Interactive state (state and model of opponent)
  - Opponent might act without considering others (**subintentional**) or might also be modeling their opponents (**intentional**)

- Enhanced Uncertainty
  - Model of opponent includes location *and* behavior
  - Complicates **R**



DECISION PROCESSES

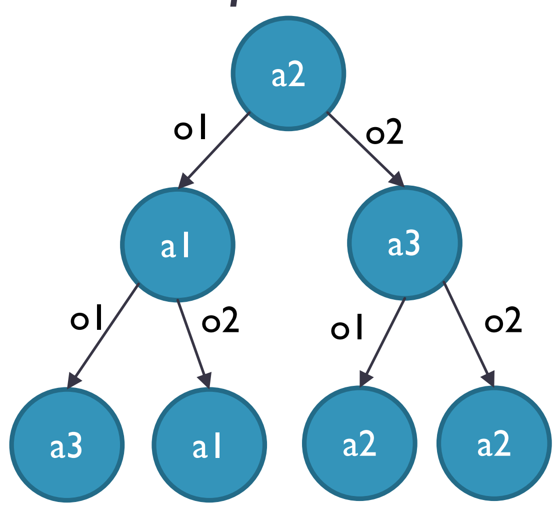
Policies in MDPs map states to actions

s1	s2	s3
a1	a3	a1

POMDPs may map states or single observations to actions

∅	o1	o2
a2	a1	a3

A policy can also be a *sequence* of observations to actions



horizon 3 policy tree

Q-LEARNING

- *Temporal difference learning* models
  - **TD(0)**

$$V(s;\alpha) = (1-\alpha)V(s) + \alpha\big(r(s) + \gamma \cdot V(s')\big)$$

  - $\boldsymbol{\alpha}$**:** Learning rate
  - $\boldsymbol{\gamma}$**:** Discount factor
- *On-policy:* Calculates value based on following a given strategy

Q-LEARNING

- Off-policy **Q-learning** considers actions
  - Future rewards consider the *best* action

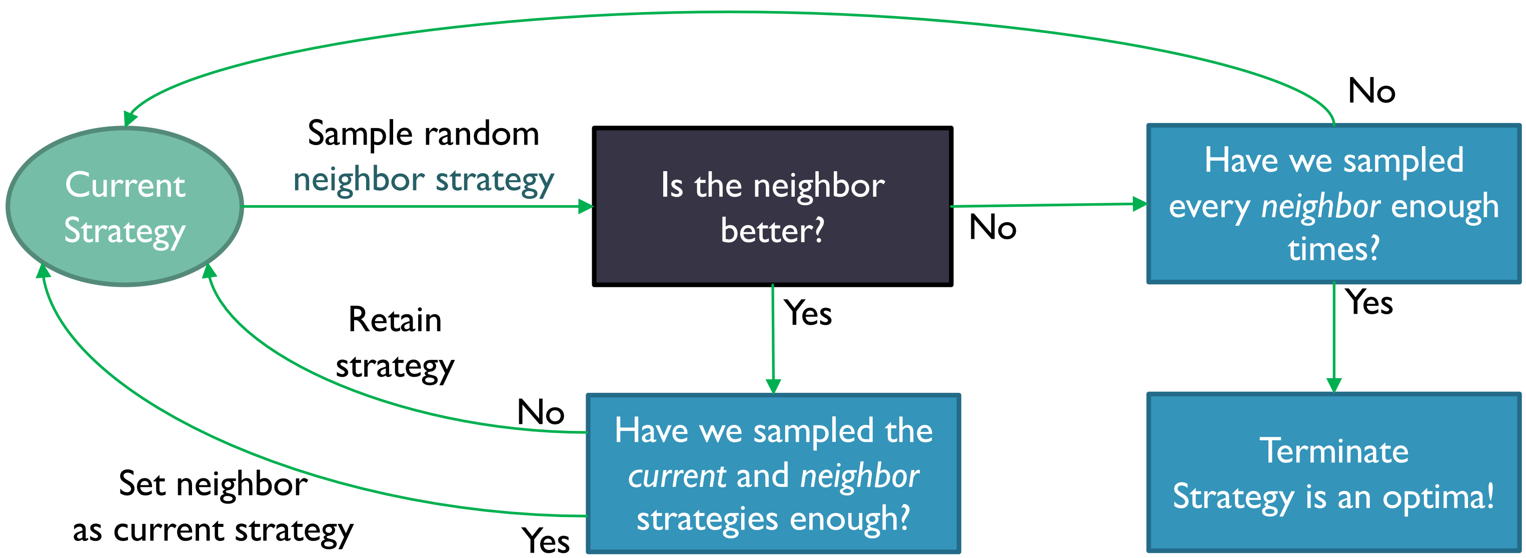
$$Q(s,a;\alpha) = (1-\alpha)Q(s,a) + \alpha\left(r(s,a) + \gamma \cdot \max_{\boldsymbol{a'}} Q(s', \boldsymbol{a'})\right)$$

- On-policy **S**tate **A**ction **R**eward **S**tate **A**ction follows a policy

$$Q(s,a;\pi,\alpha) = (1-\alpha)Q(s,a) + \alpha(r(s,a) + \gamma \cdot Q(s', \boldsymbol{\pi(s)}))$$

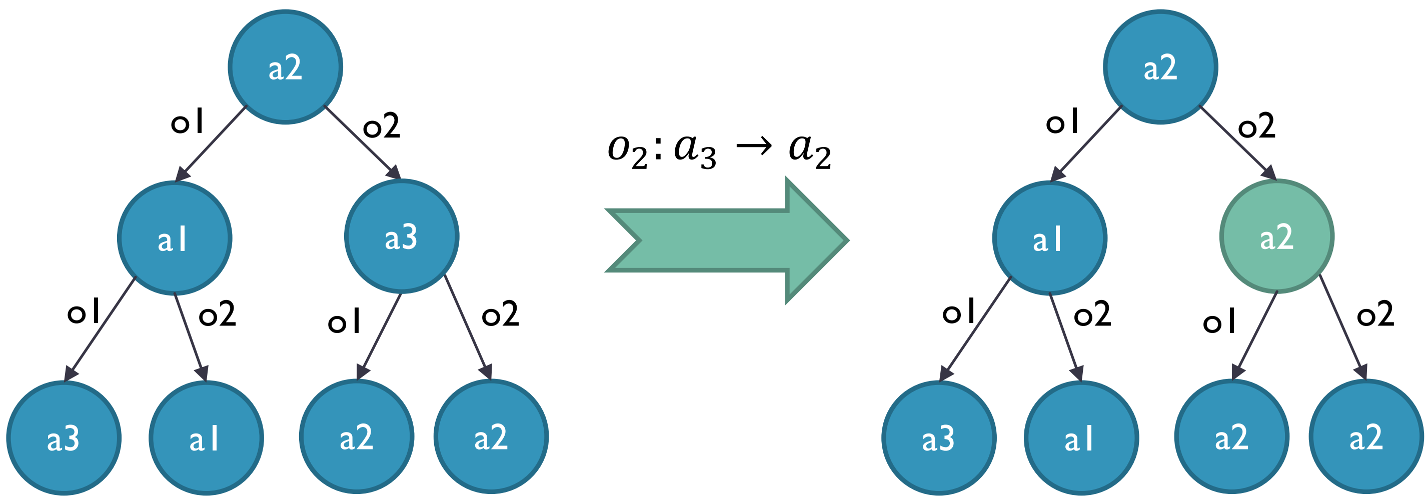
Q-LEARNING: MONTE CARLO EXPLORING STARTS

- **Monte Carlo Exploring Starts for POMDPs** (MCES-P)
  Theodore Perkins (AAAI 2002)

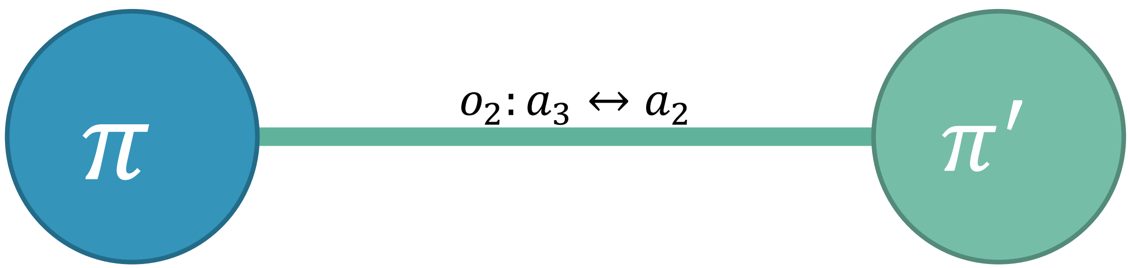


Q-LEARNING: MONTE CARLO EXPLORING STARTS

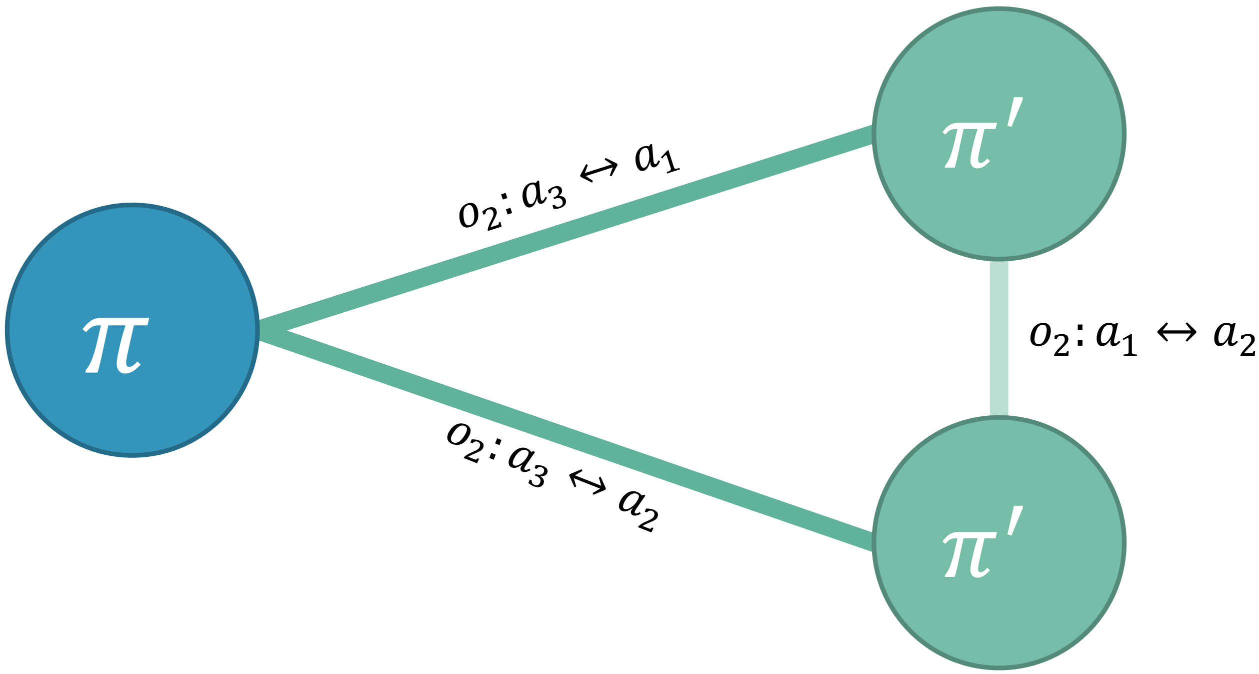
Random observation sequence replaced with a *random* action



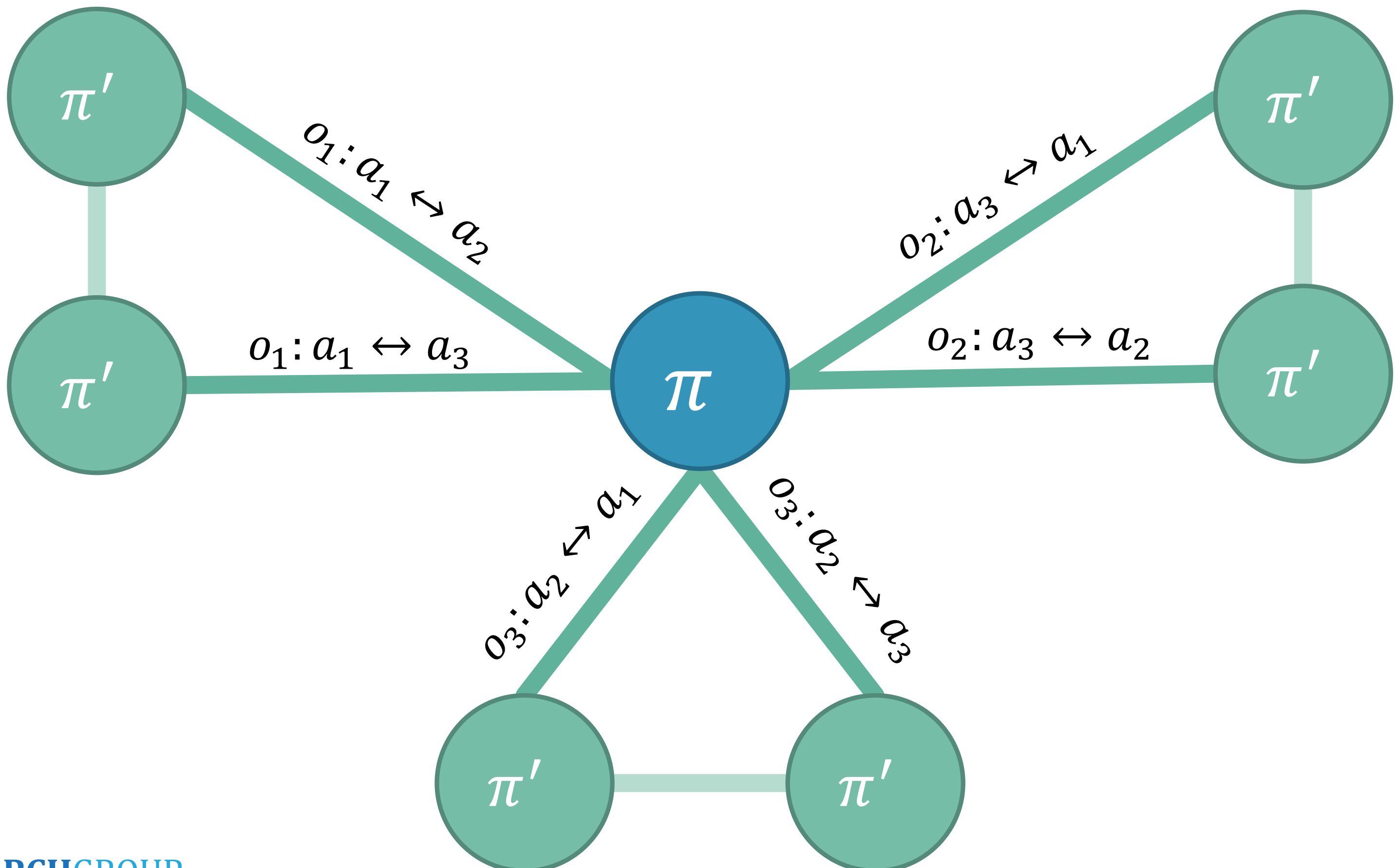
Q-LEARNING: MONTE CARLO EXPLORING STARTS



Q-LEARNING: MONTE CARLO EXPLORING STARTS

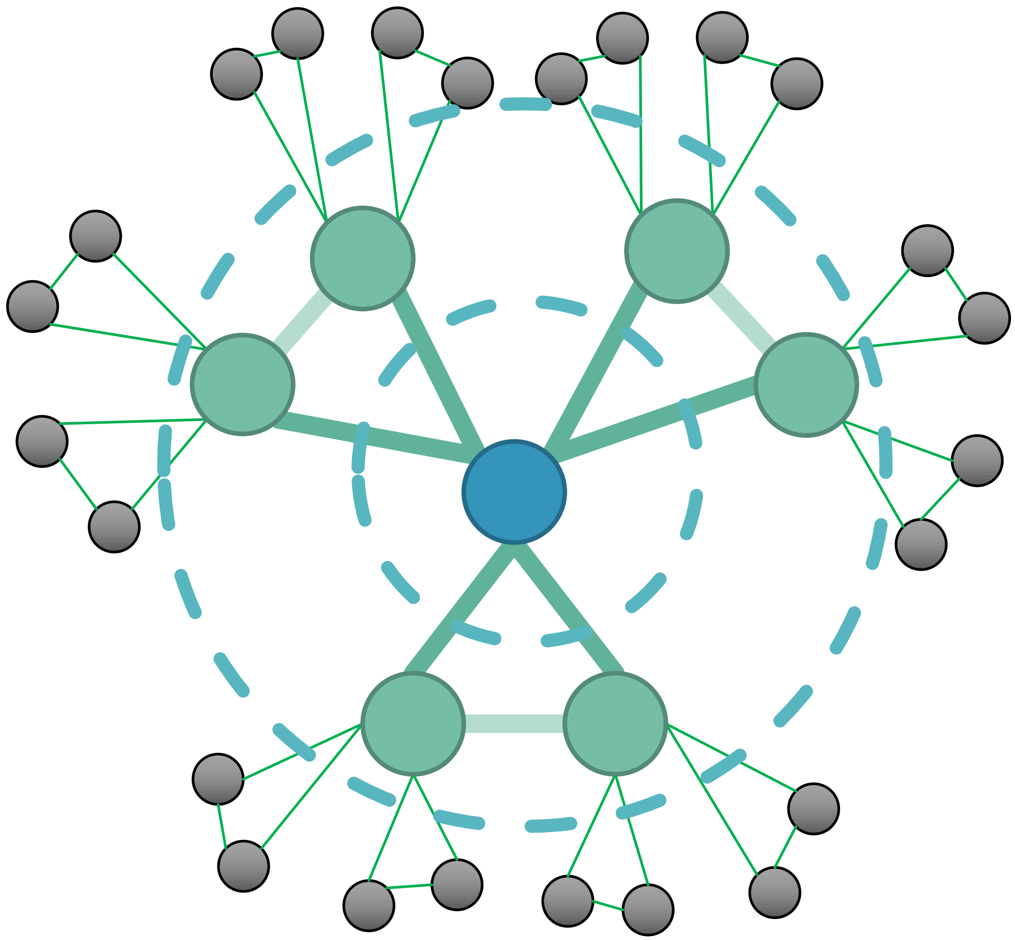


Q-LEARNING: MONTE CARLO EXPLORING STARTS



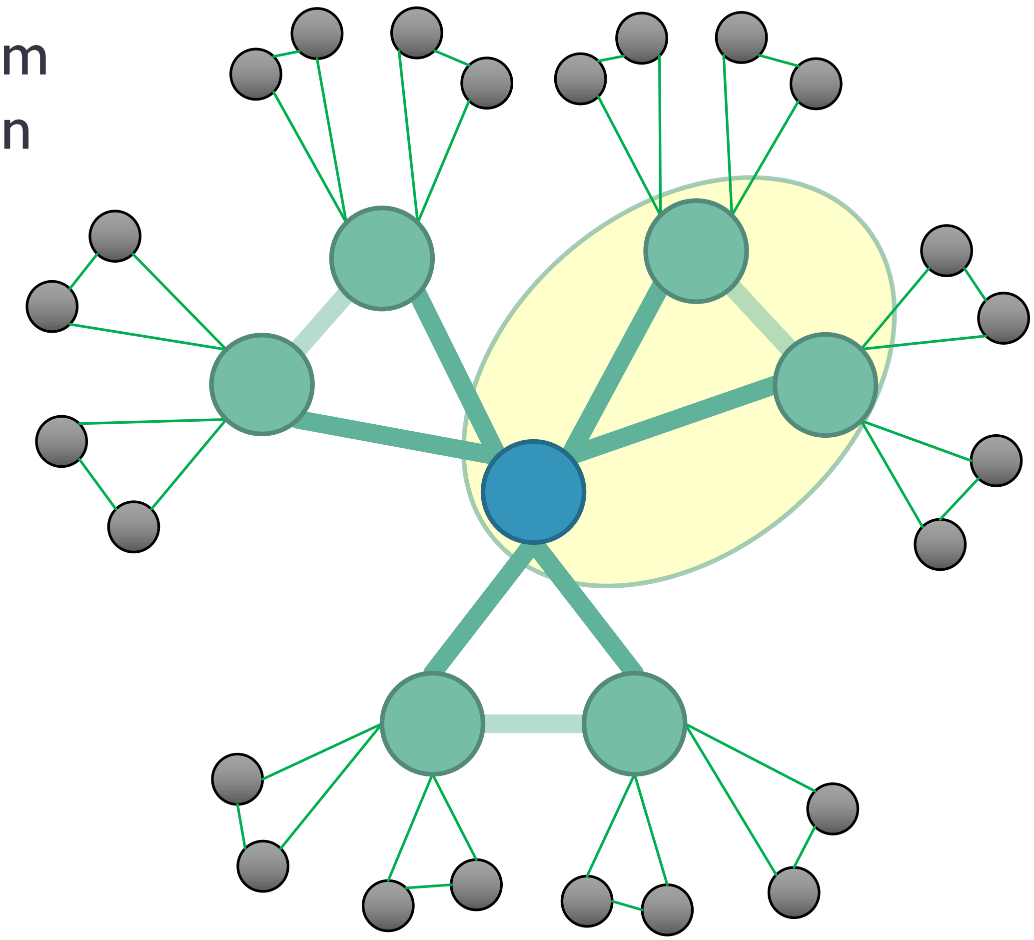
Q-LEARNING: MONTE CARLO EXPLORING STARTS

Local Neighborhood



Q-LEARNING: MONTE CARLO EXPLORING STARTS

Pick random observation

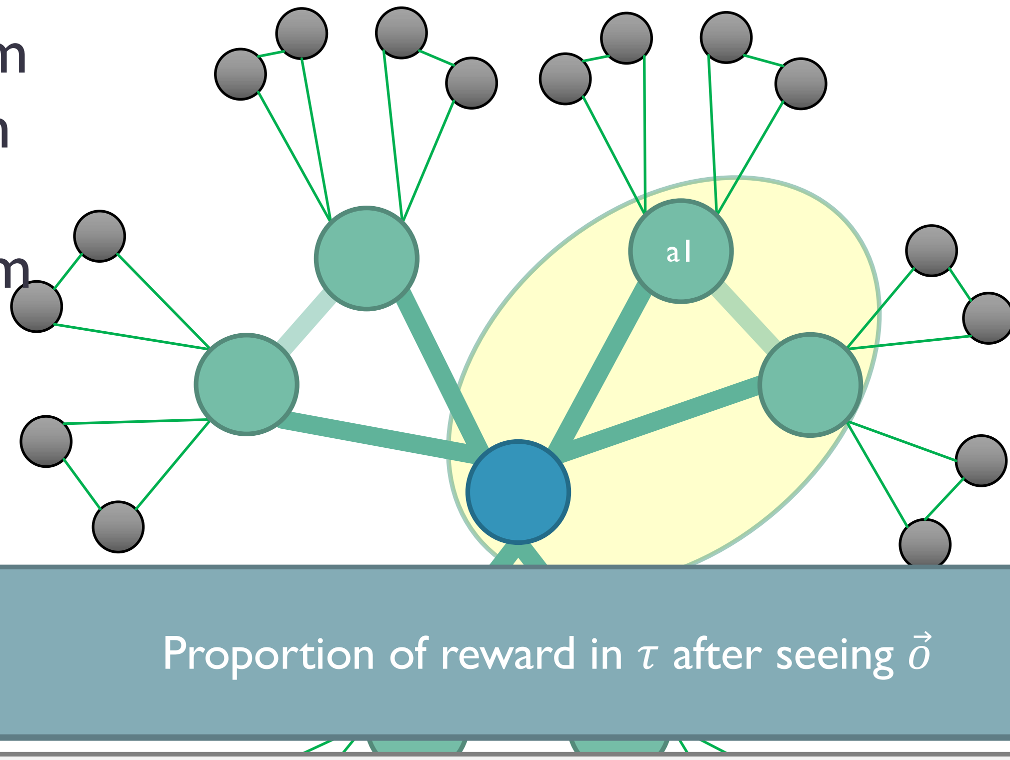


Q-LEARNING: MONTE CARLO EXPLORING STARTS

Pick random observation

Pick random action

Simulate



Proportion of reward in τ after seeing $\vec{o}$

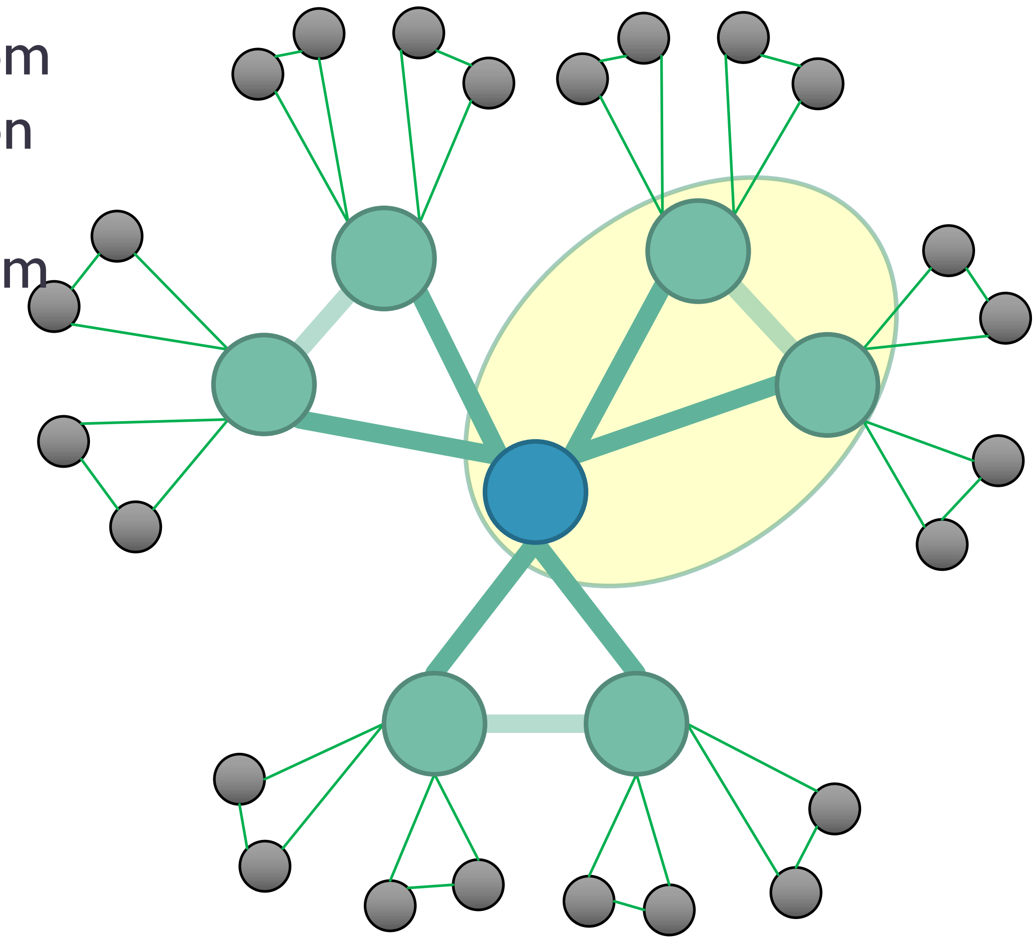
$$Q_{\pi\leftarrow(\vec{o},a)} \leftarrow \left(1-\alpha(m, c_{\vec{o},a})\right)Q_{\pi\leftarrow(\vec{o},a)} + \alpha\left(m, c_{\vec{o},a}\right)\cdot R_{post-\vec{o}}(\tau)$$

Q-LEARNING: MONTE CARLO EXPLORING STARTS

Pick random observation

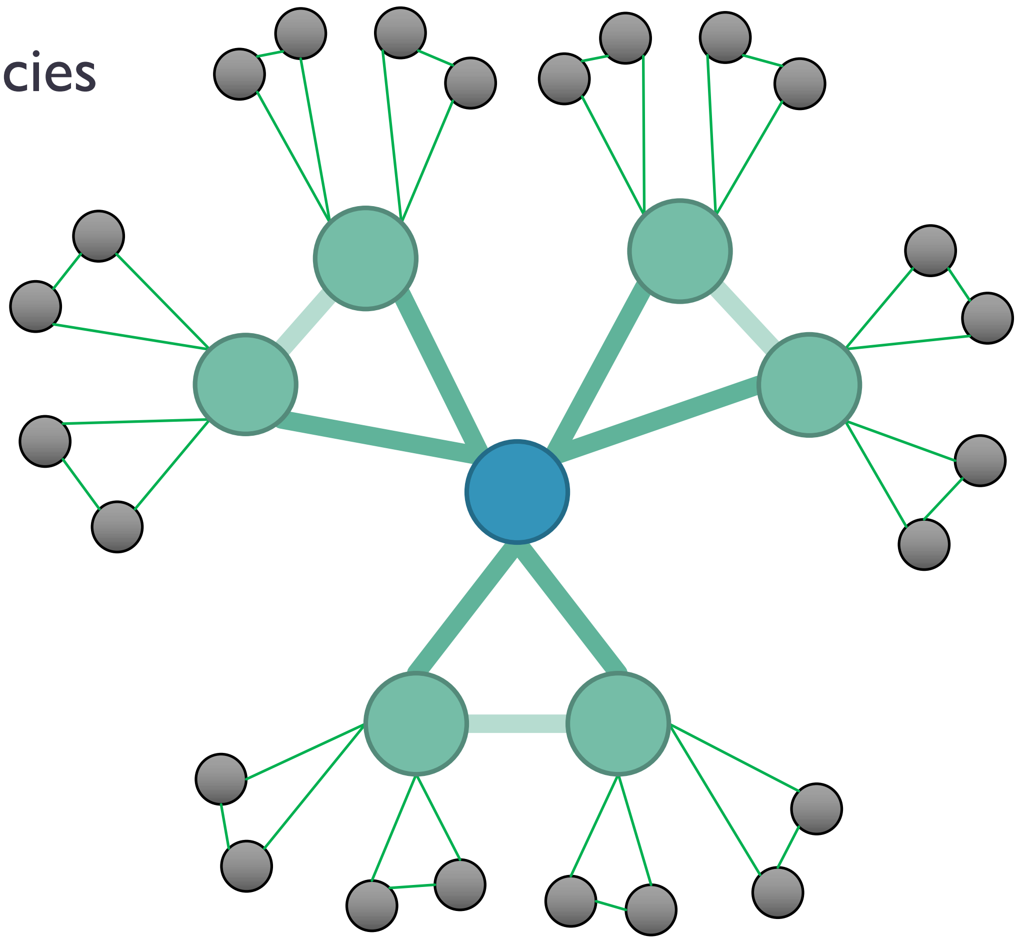
Pick random action

Simulate



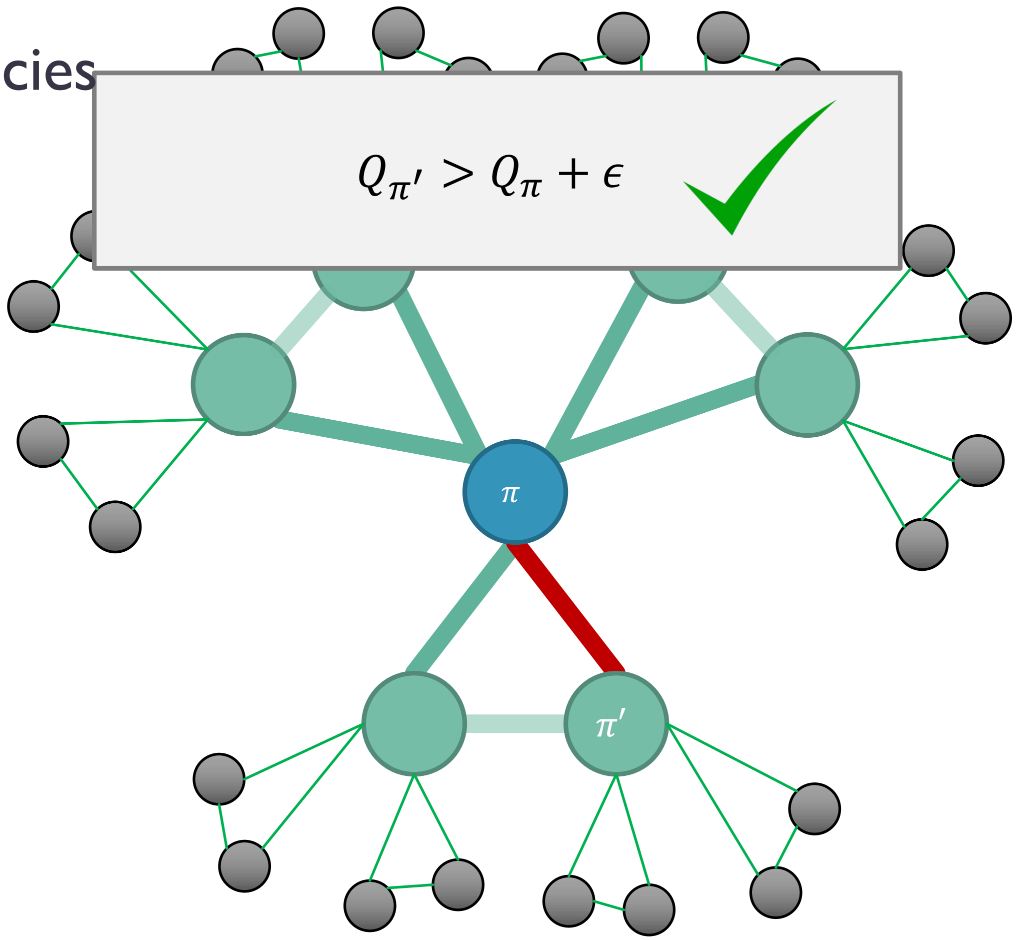
Q-LEARNING: MONTE CARLO EXPLORING STARTS

Sample policies
k times



Q-LEARNING: MONTE CARLO EXPLORING STARTS

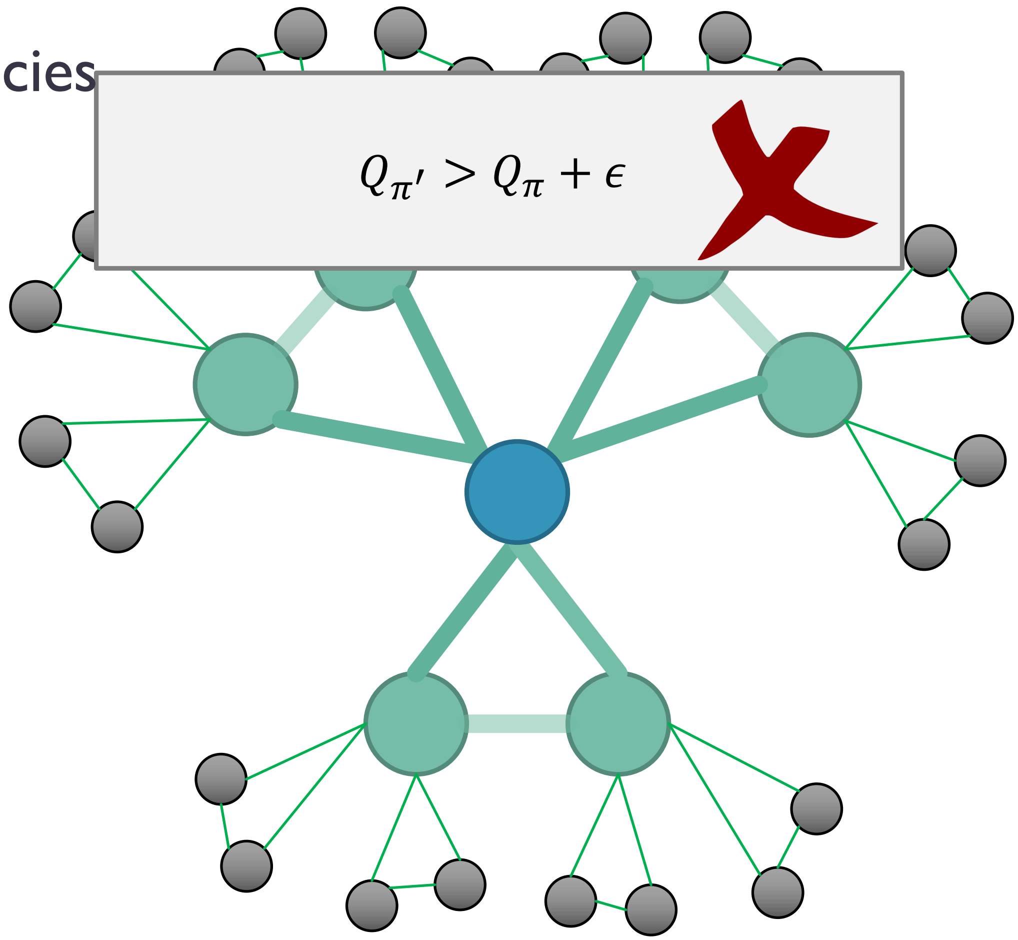
Sample policies
k times



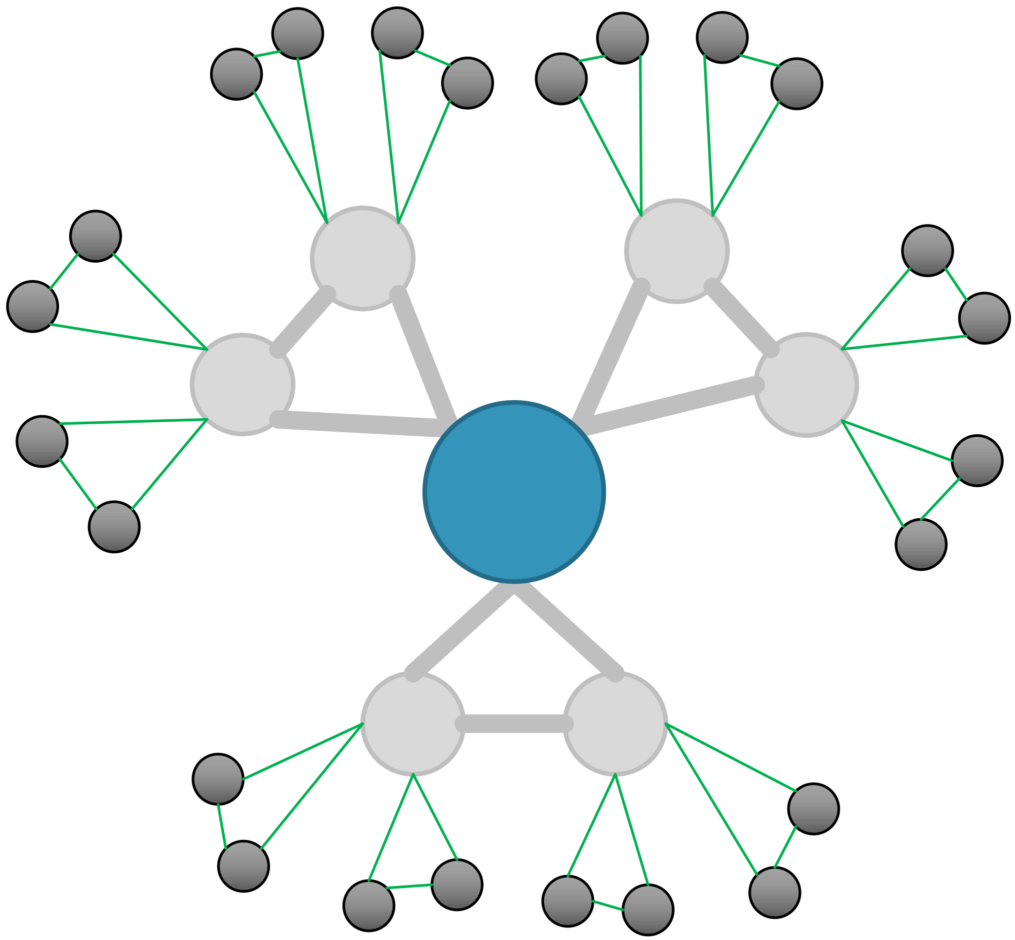
Q-LEARNING: MONTE CARLO EXPLORING STARTS

Sample policies
k times

$$Q_{\pi'} > Q_\pi + \epsilon$$



Q-LEARNING: MONTE CARLO EXPLORING STARTS



PAC LEARNING

- Picking a sample count

Low Samples **High Samples**

Inaccurate Q-values
Cheap to run

Accurate Q-values
Expensive to run

PAC LEARNING

- Derive *k*!
  - *Probably Approximately Correct* (PAC) *Learning*
  - The probability of the sample average deviating from the true mean by more than $\epsilon > 0$ can be bound by probabilistic error $\delta \in (0,1)$

Hoeffding's Inequality

$$\Pr(|\bar{X} - \mu| > \epsilon) \leq 2 \cdot \exp\left\{-2k\left(\frac{\epsilon}{\Lambda}\right)^2\right\} = \delta$$

Sample count

Value bound

QUINNRESEARCHGROUP

PAC LEARNING

- ϵ and δ determine sample counts

$$k_m = \left\lceil 2\left(\frac{\Lambda(\pi)}{\epsilon}\right)^2 \ln \frac{2|N|}{\delta_m} \right\rceil$$

  - $\boldsymbol{m}$: current transformations
  - $\boldsymbol{N}$: neighbor policies
  - $\delta_m = \frac{6\delta}{m^2\pi^2}$

$$\boldsymbol{\Lambda(\pi',\pi)} \triangleq \max_{\tau}(Q_\pi - Q_{\pi'}) - \min_{\tau}(Q_\pi - Q_{\pi'}) \leq \boldsymbol{2T(R_{max} - R_{min})}$$

$$\boldsymbol{\Lambda(\pi)} = \max_{\pi' \in neighbor(\pi)} \Lambda(\pi',\pi)$$

PAC LEARNING

- We can transform early by modifying ϵ

$$\epsilon(m,p,q) = \begin{cases} \Lambda(\pi,\pi')\sqrt{\frac{1}{2p}\ln\left(\frac{2(k_m-1)N}{\delta_m}\right)} & \textbf{if } p = q < k_m \\ \frac{\epsilon}{2} & \textbf{if } p = q = k_m \\ \infty & \text{otherwise} \end{cases}$$

- Terminate when k_m samples of each neighbor is taken *or* for all neighbor policies:

$$Q_{\vec{o},a} < Q_{\vec{o},\pi(\vec{o})} + \epsilon - \epsilon(m, c_{\vec{o},a}, c_{\vec{o},\pi(\vec{o})})$$

PAC LEARNING

- Then, with probability $1 - \delta$, MCESP+PAC
  1. Transforms to π that are *guaranteed* better than the current policy
  2. Terminates with a π that is an ϵ-local optima
     - No neighbor is better than the last policy by more than ϵ

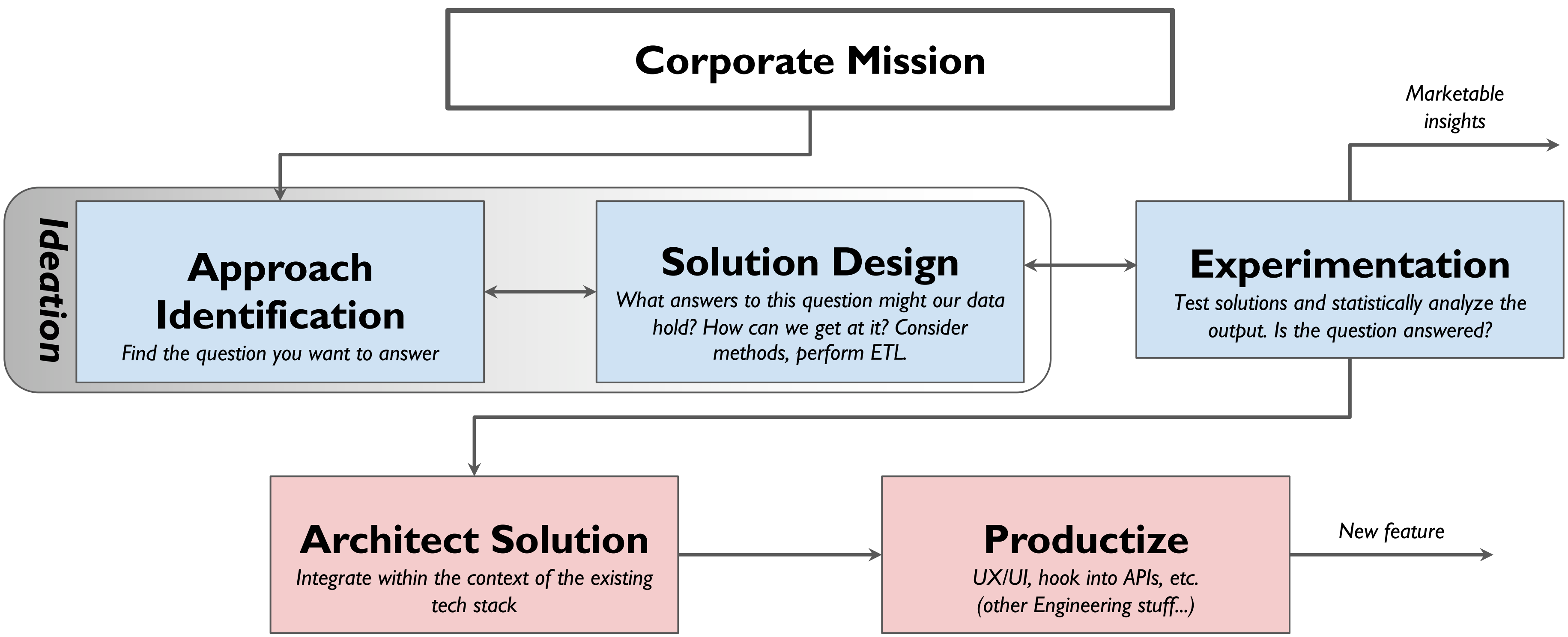
https://arxiv.org/pdf/1901.01325.pdf

REINFORCEMENT LEARNING

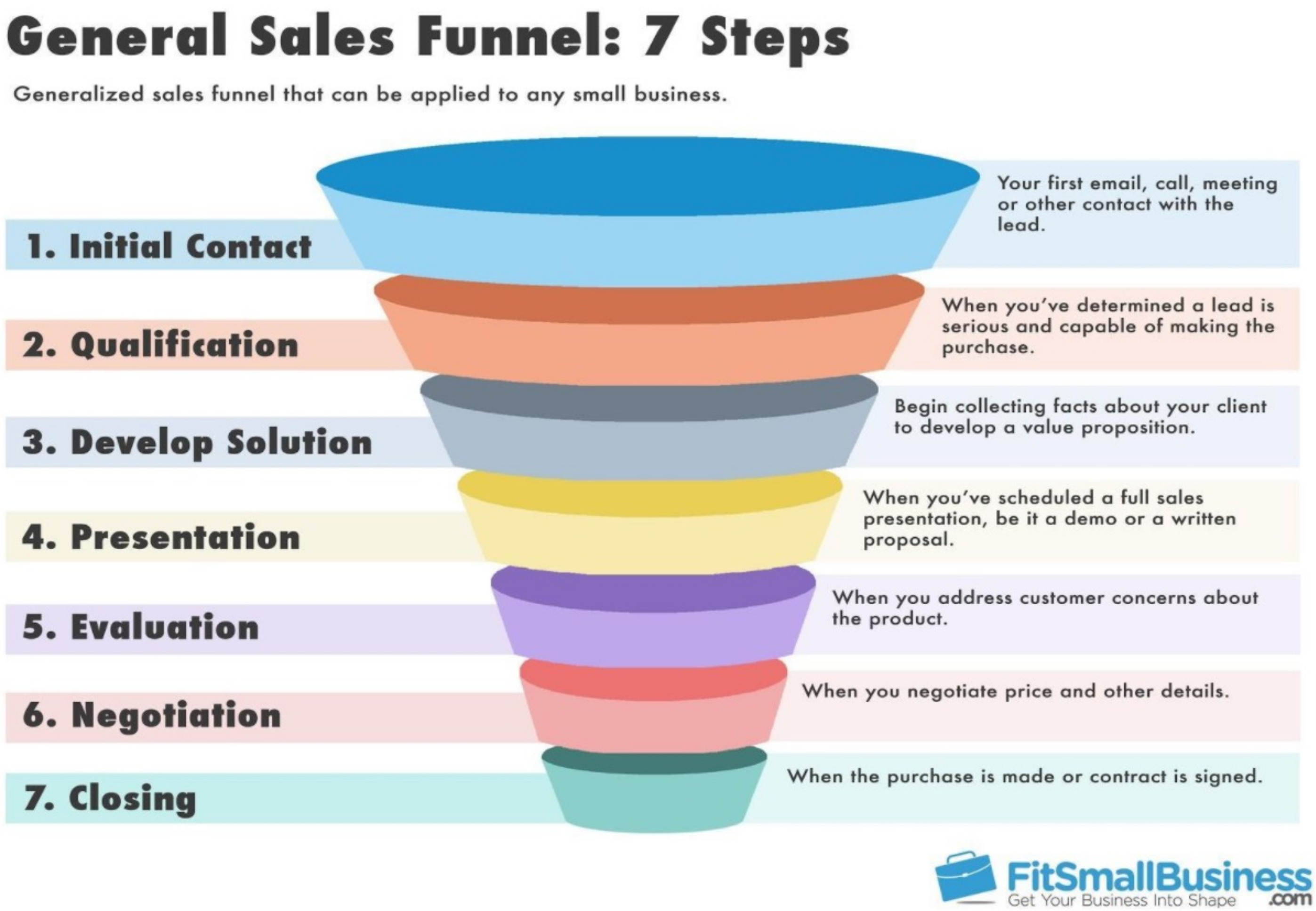
IN THE SALES DOMAIN

DATA SCIENCE PROCESS

Ask interesting questions, get interesting answers.



THE SALES DOMAIN



SALES ENGAGEMENT

- Calendar
- Owler/Crystal Knows
- Dialer
- **Cadences**

- Meetings
- Opportunities

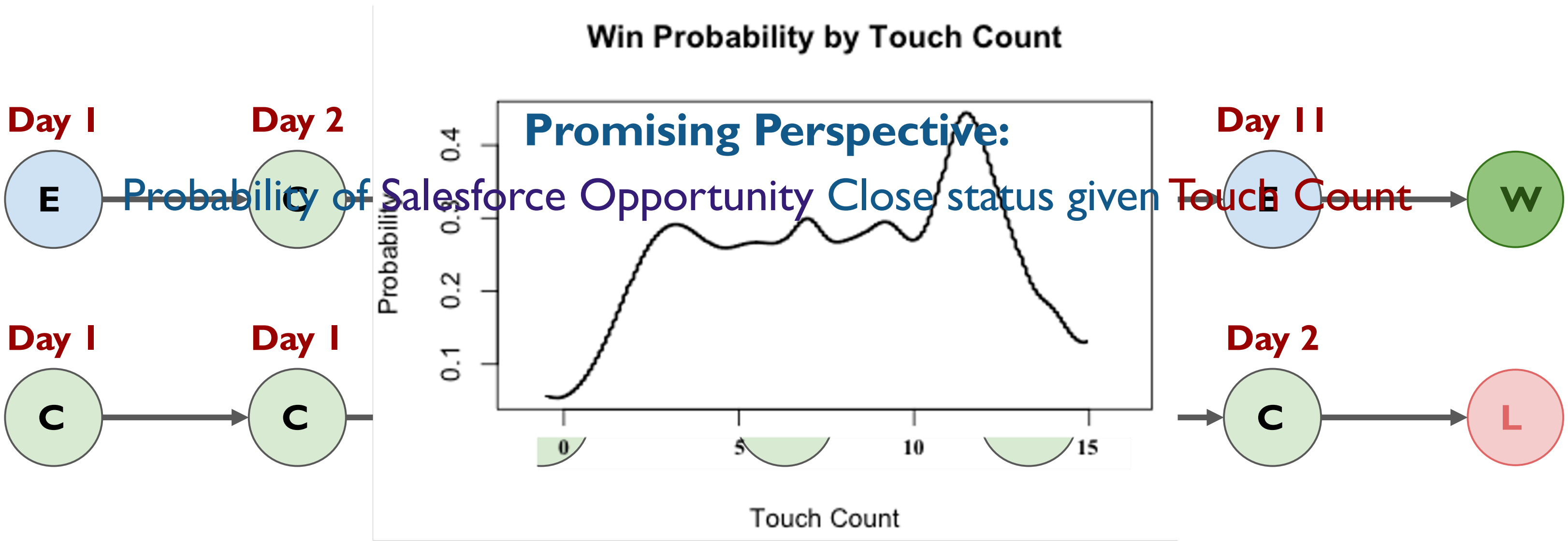
DERIVED CADENCES - THE JOURNEY

Interesting Question:

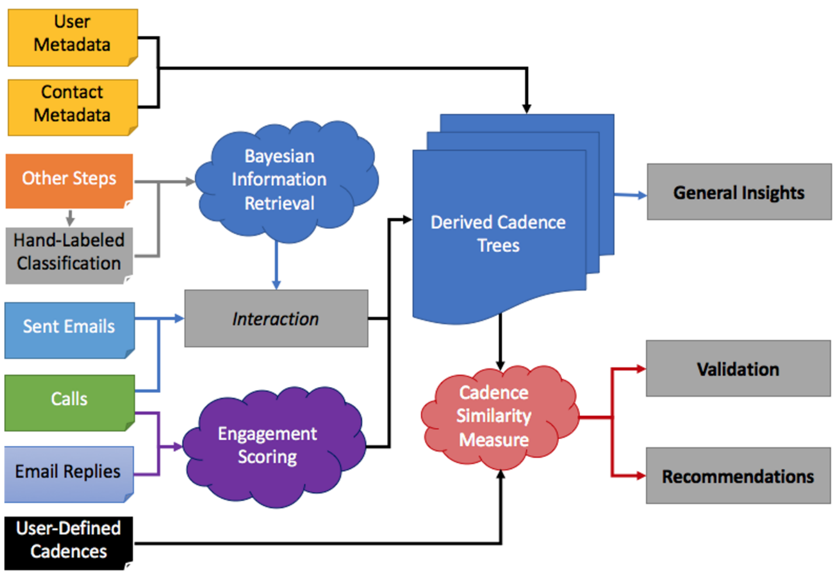
What's the optimal number of touches?

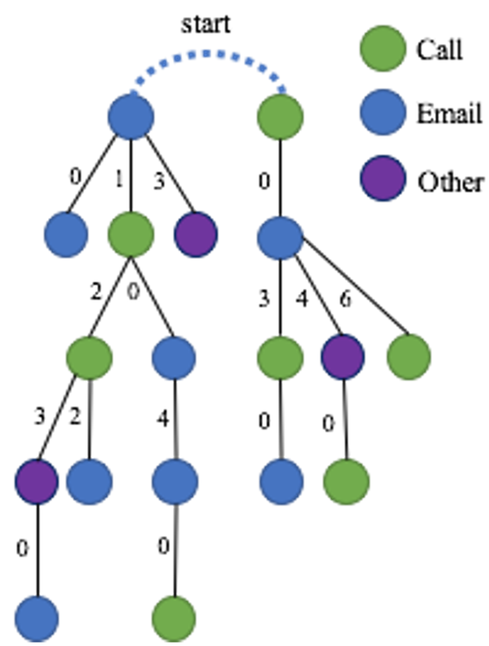
Promising Perspective:

Probability of Salesforce Opportunity Close status given Touch Count



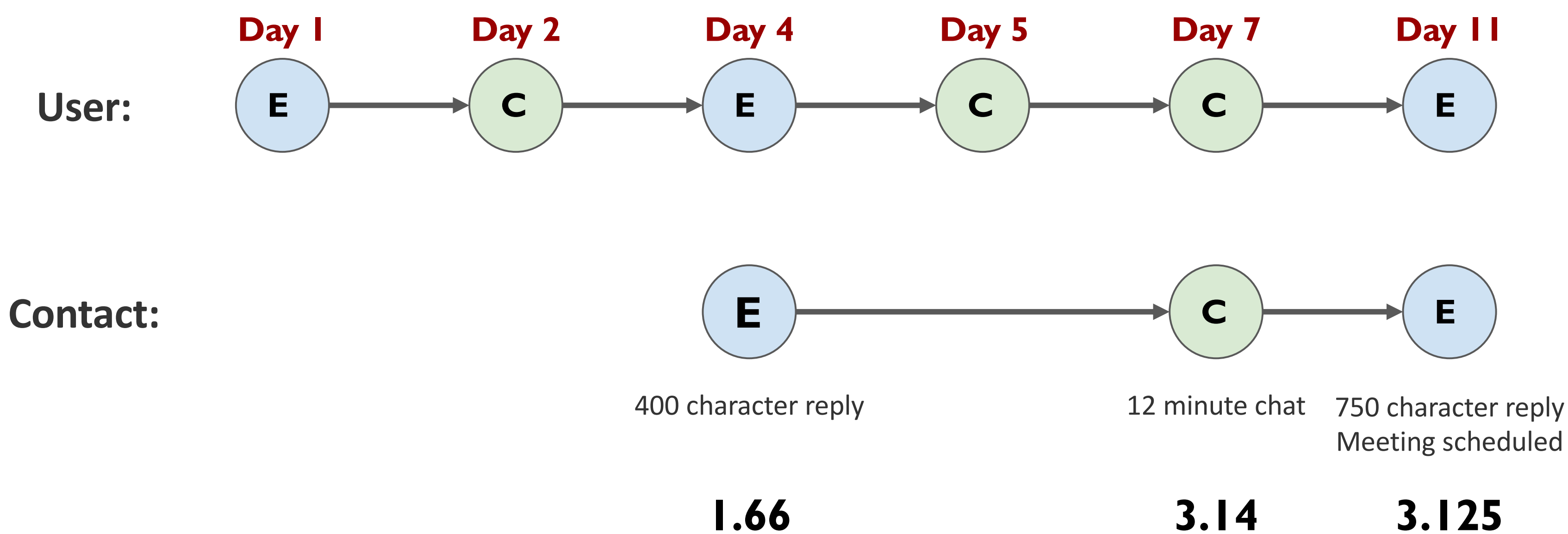
DERIVED CADENCES - THE PROCESS





DERIVED CADENCES - THE PROCESS

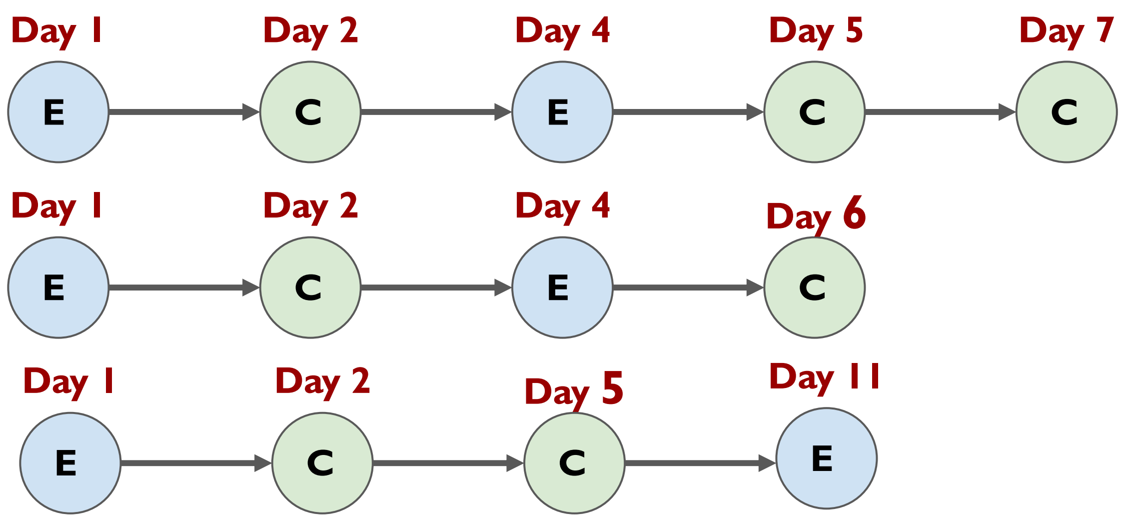
Optimizing Time Value

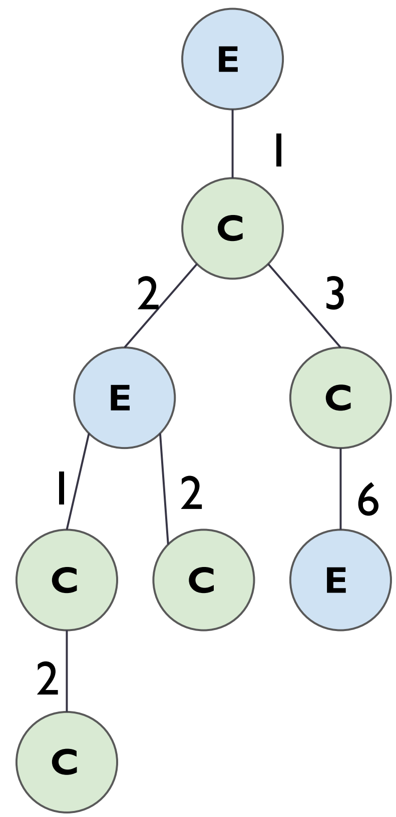


$$ES(t) = \begin{cases} \frac{length(t)}{240} & \text{if } t = Email \\ \frac{duration(t)}{3.81} & \text{if } t = Call \end{cases}$$

DERIVED CADENCES - THE PROCESS

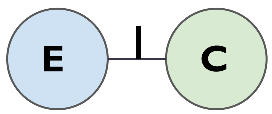
Aggregating Behavioral Data





DERIVED CADENCES - THE PROCESS

Aggregating Behavioral Data



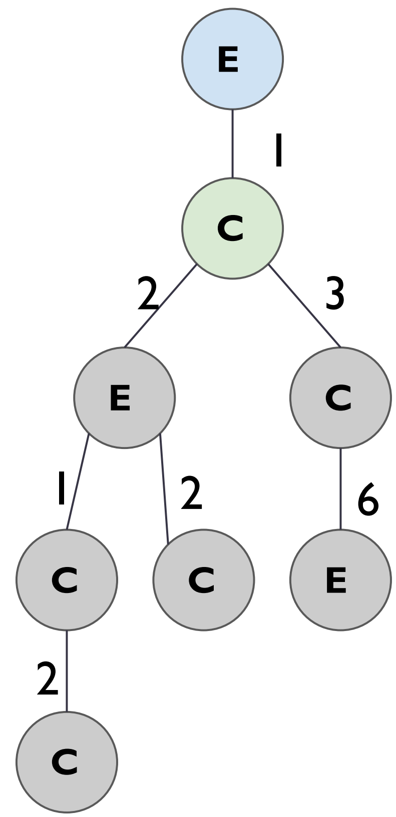
Moving Average

$$\bar{E}S_n = \bar{E}S_{n-1} + \frac{ES_n - ES_{n-1}}{n}$$

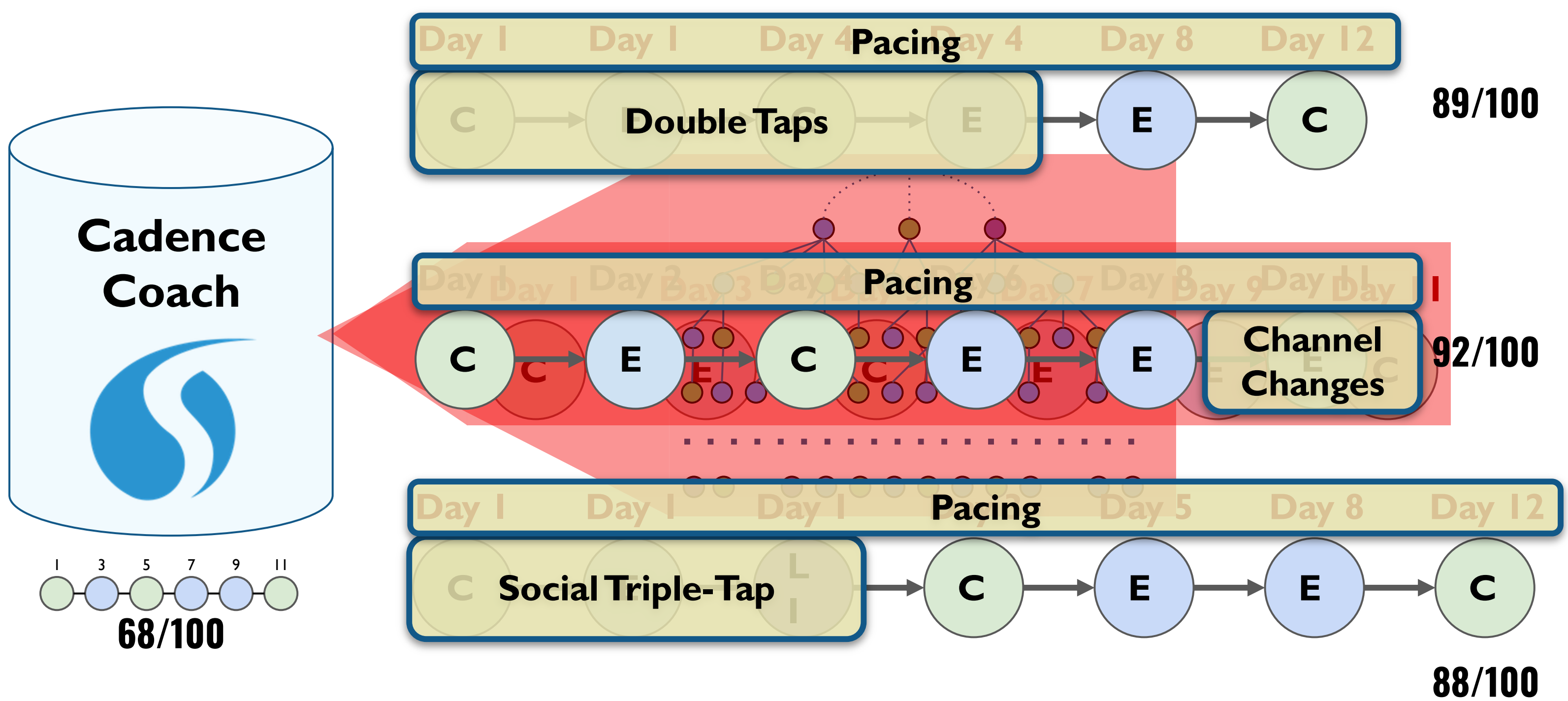
Moving Variance

$$M_n = M_{n-1} + \frac{ES_n - M_{n-1}}{n}$$

$$V_n = V_{n-1} + \frac{ES_n - M_{n-1}}{ES_n - M_n}$$



DERIVED CADENCES - ACTIONABLE INSIGHTS



QUINNRESEARCHGROUP

FIN

AWS

A TASTE

COMMON STACK