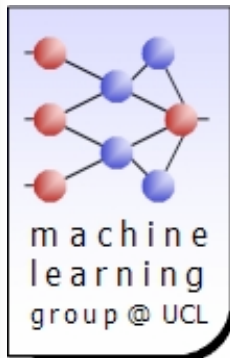


Reinforcement learning and Markov decision processes: A short, informal, introduction

Marco Saerens (UCL)



Université catholique de Louvain

Université partenaire de l'Académie universitaire 'Louvain'





Introduction to reinforcement learning

- We define the « reinforcement learning » problem
- Or the deterministic/stochastic « shortest path problem »



Reinforcement learning

- We have a set of states, $S = \{1, 2, \dots, n\}$
 - $s_t = k$ means that the process, or system, is in state k at time t
- In each state $s = k$, we have a set of admissible control actions, $U(k)$
 - So that $u(k) \in U(k)$ is a control action available at state k
 - The action only depends on the current state



Reinforcement learning

- When we choose action $u(s_t)$ at state s_t ,
 - A **bounded cost** $c(u(s_t) | s_t) < \infty$ is incurred
 - The system **jumps** to state $s_{t+1} = k'$ with a probability distribution:

$$P(s_{t+1} = k' | s_t = k, u(k) = i) = p(k' | k, i)$$

- Which only depends on the current state k and not on the past states (Markov assumption)
- The probability distribution is **independent** of the past

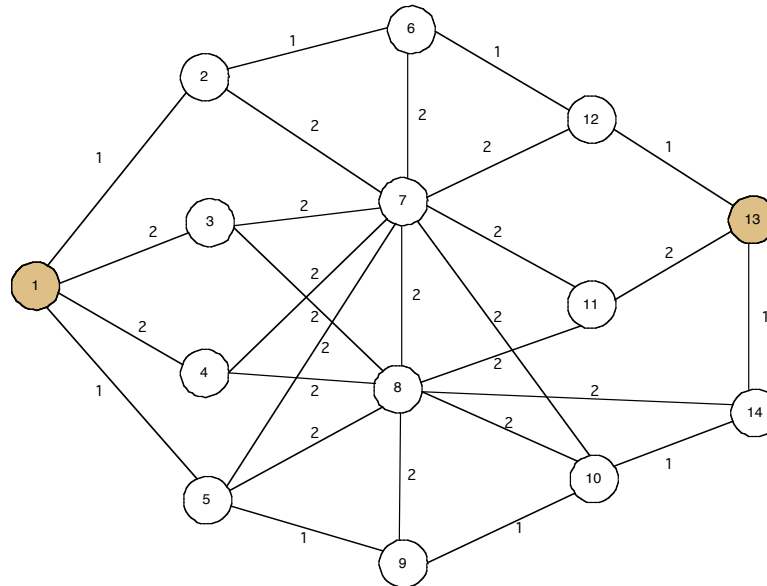


Reinforcement learning

- If the problem is **deterministic**, the probability distribution reduces to a Kronecker delta
- We suppose the network of states does not contain any negative cycle

Reinforcement learning

- The **policy** π is defined as the set of actions $\pi = \{u(1), u(2), \dots\}$
- Which will be chosen as **optimal** in a certain sense





Reinforcement learning

- The goal is to reach a destination state, $s = d$
- From some initial state, $s_0 = k_0$
- While minimizing the total expected cost

$$V_{\pi}(s_0 = k_0) = E \left[\sum_{t=0}^{\infty} c(u(s_t)|s_t) \mid s_0 = k_0, \pi \right]$$

- The expectation is taken on the random variables



Reinforcement learning

- Thus an optimal policy π^* is given by

$$V^*(k_0) = \min_{\pi} \{V_{\pi}(k_0)\}$$



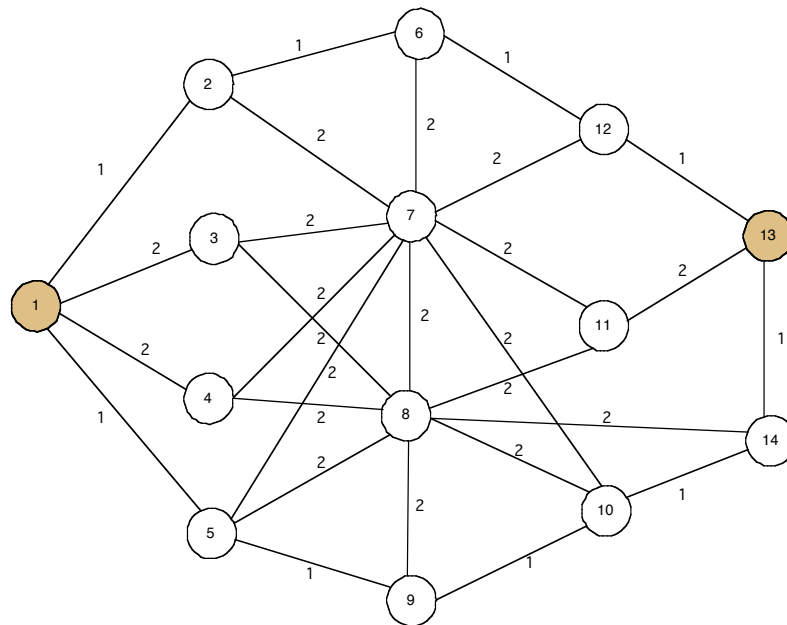
Reinforcement learning

- In other words, we have to determine the **best policy** π that minimizes $V_\pi(k_0)$
 - That is, the best action associated to each state
- We will see that the **optimal expected cost** can be computed thanks to the recurrence relations
 - Value-iteration algorithm

$$\begin{cases} V(k) \leftarrow \min_{i \in U(k)} \left\{ \sum_{k'} p(k'|k, i) [c(i|k) + V(k')] \right\}, & k \neq d \\ V(d) \leftarrow \text{reward}, & \text{where } d \text{ is the destination state} \end{cases}$$

Reinforcement learning

- Thus, we route the agents as fast as possible through the network





Reinforcement learning

- Here is a heuristic derivation of the optimality conditions
- This policy will be called the **optimal policy**

$$V^*(k_0) = \min_{\{u(s_0), u(s_1), \dots\}} \left\{ E \left[\sum_{t=0}^{\infty} c(u(s_t) | s_t) \mid s_0 = k_0, \pi \right] \right\}$$



Reinforcement learning

$$V^*(k_0) = \min_{\{u(s_0), u(s_1), \dots\}} \left\{ \sum_{k_1, k_2, \dots} P(s_1 = k_1, s_2 = k_2, \dots | s_0 = k_0, \pi) \times \left[\sum_{t=0}^{\infty} c(u(k_t) | k_t) \right] \right\}$$

We try to particularize/factorize with respect to $t = 0$ by computing the conditional probability and using the Markov property

$$V^*(k_0) = \min_{\{u(s_0), u(s_1), \dots\}} \left\{ \sum_{k_1, k_2, \dots} P(s_2 = k_2, s_3 = k_3, \dots | s_1 = k_1, \pi) P(s_1 = k_1 | s_0 = k_0, u(s_0) = i_0) \times \left[c(u(k_0) | k_0) + \sum_{t=1}^{\infty} c(u(k_t) | k_t) \right] \right\}$$

Reinforcement learning

$$V^*(k_0) = \min_{u(s_0)} \min_{\{u(s_1), u(s_2), \dots\}} \left\{ \sum_{k_1} P(s_1 = k_1 | s_0 = k_0, u(k_0) = i_0) \sum_{k_2, k_3, \dots} P(s_2 = k_2, s_3 = k_3, \dots | s_1 = k_1, \pi) \right. \\ \left. \times \left[c(u(k_0) | k_0) + \sum_{t=1}^{\infty} c(u(k_t) | k_t) \right] \right\}$$

$$V^*(k_0) = \min_{u(s_0)} \left\{ \sum_{k_1} P(s_1 = k_1 | s_0 = k_0, u(k_0) = i_0) \right. \\ \left. \times \left[c(u(k_0) | k_0) + \underbrace{\min_{\{u(s_1), u(s_2), \dots\}} \sum_{k_2, k_3, \dots} P(s_2 = k_2, s_3 = k_3, \dots | s_1 = k_1, \pi) \sum_{t=1}^{\infty} c(u(k_t) | k_t)}_{V^*(k_1)} \right] \right\}$$



Reinforcement learning

- Thus, we obtain:

$$V^*(k_0) = \min_{i_0} \left\{ \sum_{k_1} p(k_1 | k_0, i_0) [c(i_0 | k_0) + V^*(k_1)] \right\}$$

= Bellman's equations



Reinforcement learning

- There are many generalizations of this technique
 - For instance, there is not always a final destination state
- In that case, we can minimize
 - The average cost per time unit
 - The total discounted cost



Reinforcement learning

- Total discounted cost:

$$V_{\pi}(k_0) = E \left[\sum_{t=0}^{\infty} \alpha^t c(u(s_t)|s_t) \mid s_0 = k_0, \pi \right], \text{ with } 0 < \alpha < 1$$

- Which leads to the value-iteration algorithm

$$V(k) \leftarrow \min_{i \in U(k)} \left\{ \sum_{k'} p(k'|k, i) [c(i|k) + \alpha V(k')] \right\}$$



Reinforcement learning

- The average cost per time unit criterion:

$$V_{\pi}(k_0) = \lim_{N \rightarrow \infty} E \left[\frac{1}{N} \sum_{t=0}^N c(u(s_t)|s_t) \mid s_0 = k_0, \pi \right]$$

- Which is somewhat more complex to analyze