Spring 2024

Dynamic programming

Solving deterministic finite horizon MDPs

Cathy Wu

1.041/1.200 Transportation: Foundations and Methods

Readings

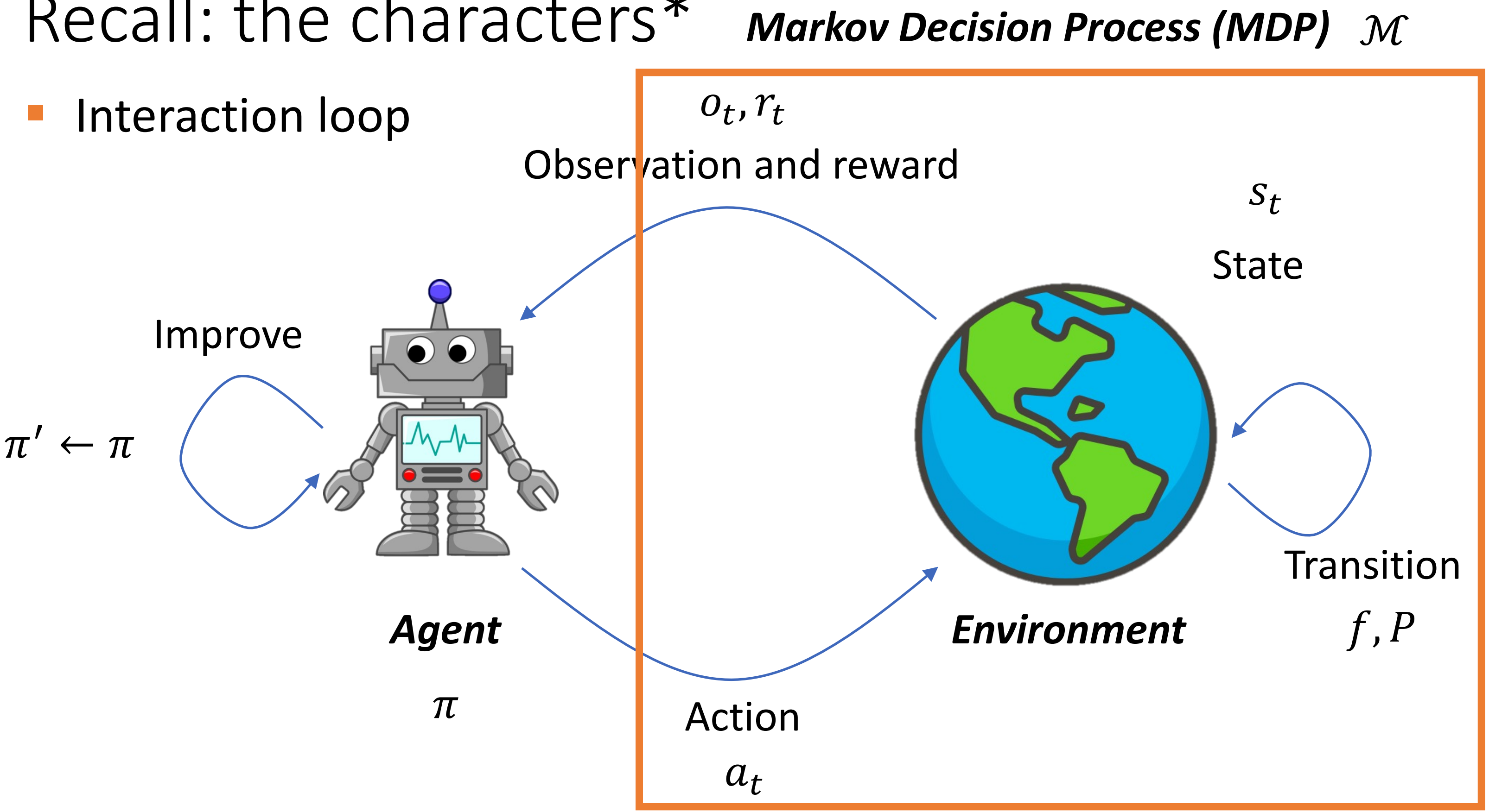
1. Bradley, Stephen P., Arnoldo C. Hax, and Thomas L. Magnanti. **Applied mathematical programming**. Addison-Wesley (1977). Chapter 11: Dynamic Programming. [URL]

Outline

1. Shortest path problems
2. Optimal capacity expansion problem

Recall: the characters*

- Interaction loop



Goal: maximize reward over time (returns, cumulative reward)

* pun intended

Optimization Problem

- Our goal: solve the MDP

Definition (Optimal policy and optimal value function)

The solution to an MDP is an optimal policy π^* satisfying

$$\pi^* \in \arg\max_{\pi \in \Pi} V^{\pi}$$

where Π is some policy set of interest.

The corresponding value function is the optimal value function

$$V^* = V^{\pi^*}$$

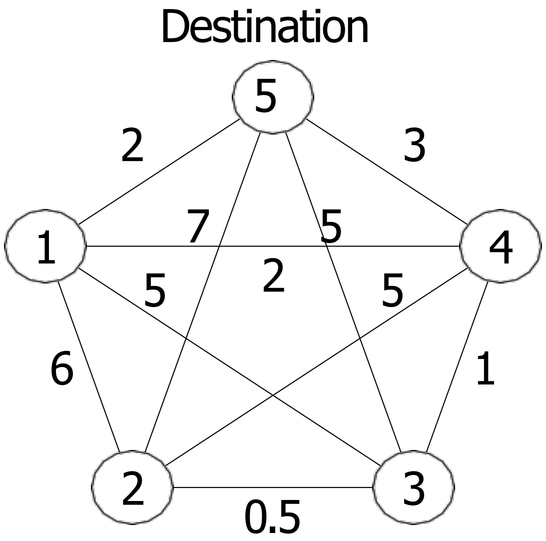
Assume for now: finite horizon problems, i.e. $T < \infty$

Deterministic vs stochastic sequential problems

- A deterministic policy is a special case of a stochastic policy when $\pi(a|s)$ is a unit spike at $a = \pi(s)$ for all $s \in \mathcal{S}$ (and 0 otherwise).
- A deterministic transition is a special case of a stochastic transition when $p(s'|s, a)$ is a unit spike at $\mathrm{s}' = \mathrm{f_t}(\mathrm{s}, \mathrm{a})$ for all $s \in \mathcal{S}, a \in A$ (and 0 otherwise).

That is, a deterministic sequential decision problem is a special case of a stochastic sequential problem. It can still be modeled within the MDP framework.

Example: Shortest Path Problem

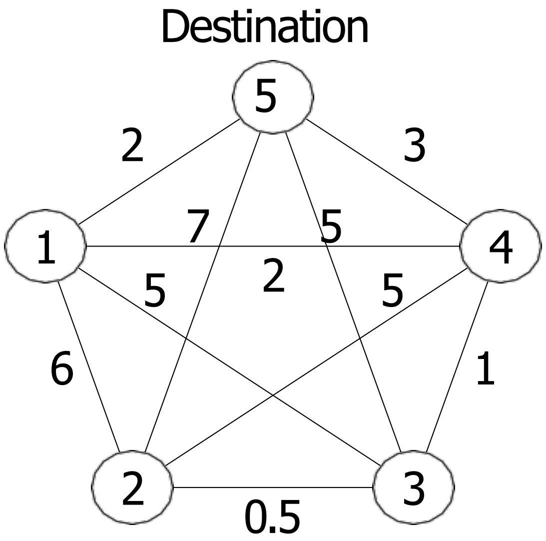


Destination is node 5.

Sequential decision problem

- Start state s_0: city 2
- Action a_0: take link between city 2 and city 3
- State s_1: city 3
- Action a_1: take link between city 3 and city 5
- State s_2: city 5
- …

Solving Shortest Path

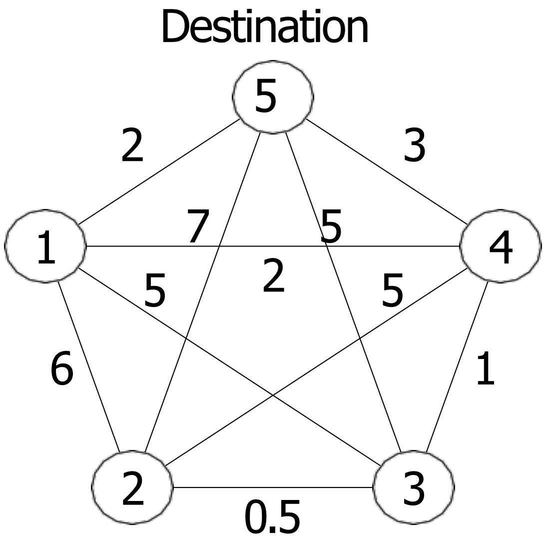


Destination is node 5.

Assumption: all cycles have non-negative length.

- **Naive approach:** enumerate all possibilities.
  - From a starting city s_0, choose any remaining city $(N - 1$ choices). Choose any next remaining city $(N - 2$ choices). ...
    Until there is only 1 option remaining.
  - Add up the edge costs.
  - Select the best sequence (lowest total cost).
  - $O(N!)$. ☹

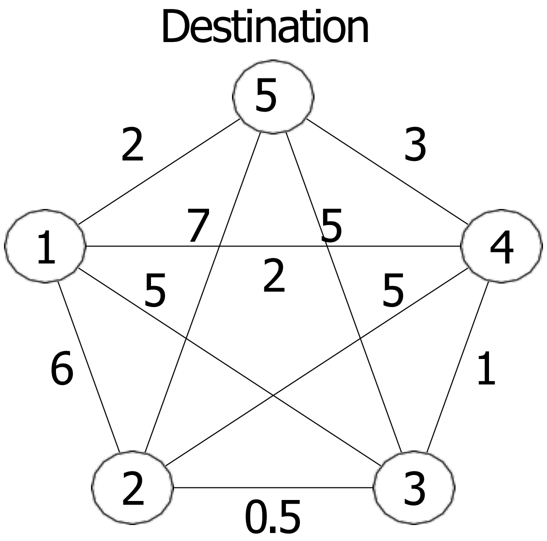
Solving Shortest Path



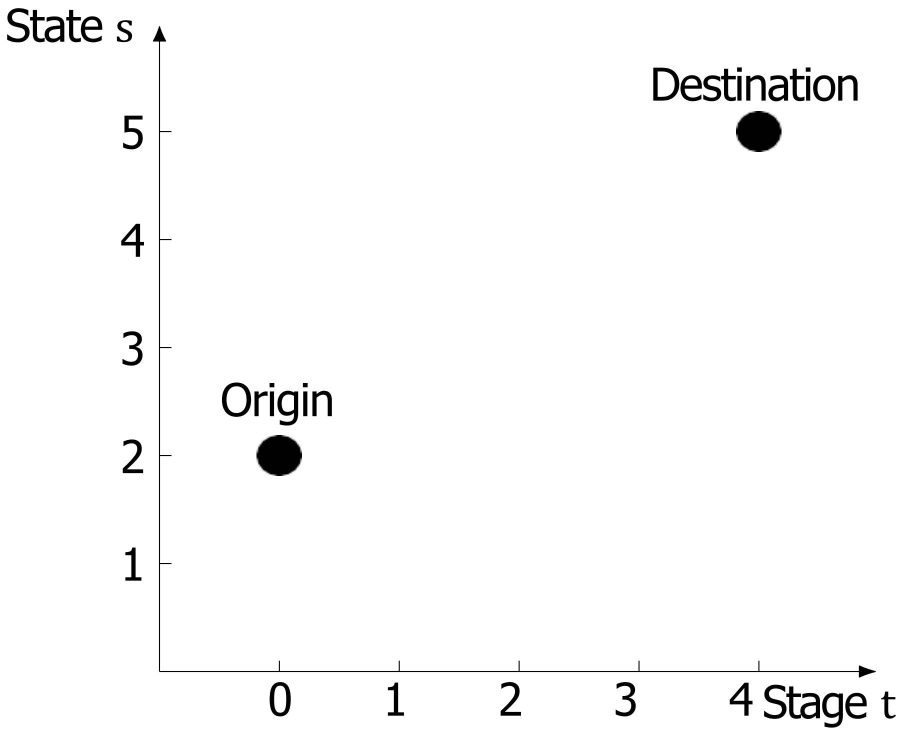
Destination is node 5.

- **Issue**: repeated calculations of subsequences.
  - Dynamic programming: divide-and-conquer, or **the principle of optimality**.
  - Overall problem would be much easier to solve if a part of the problem were already solved.
  - Break a problem down into subproblems.

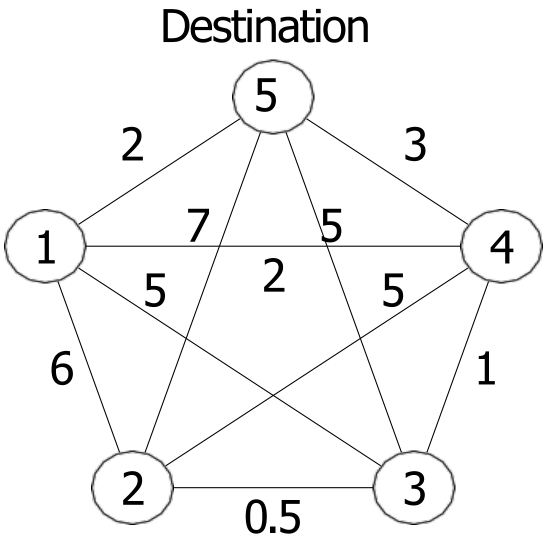
Solving Shortest Path

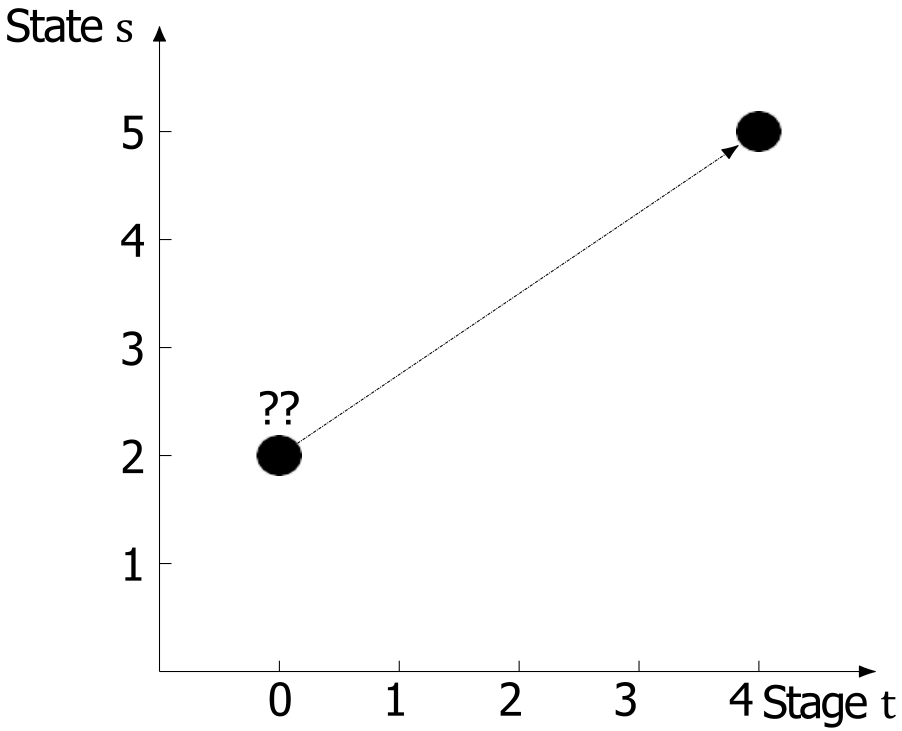


Destination is node 5.



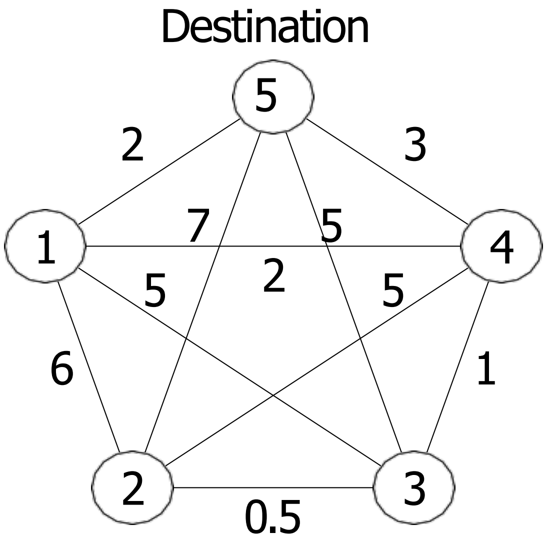
Solving Shortest Path



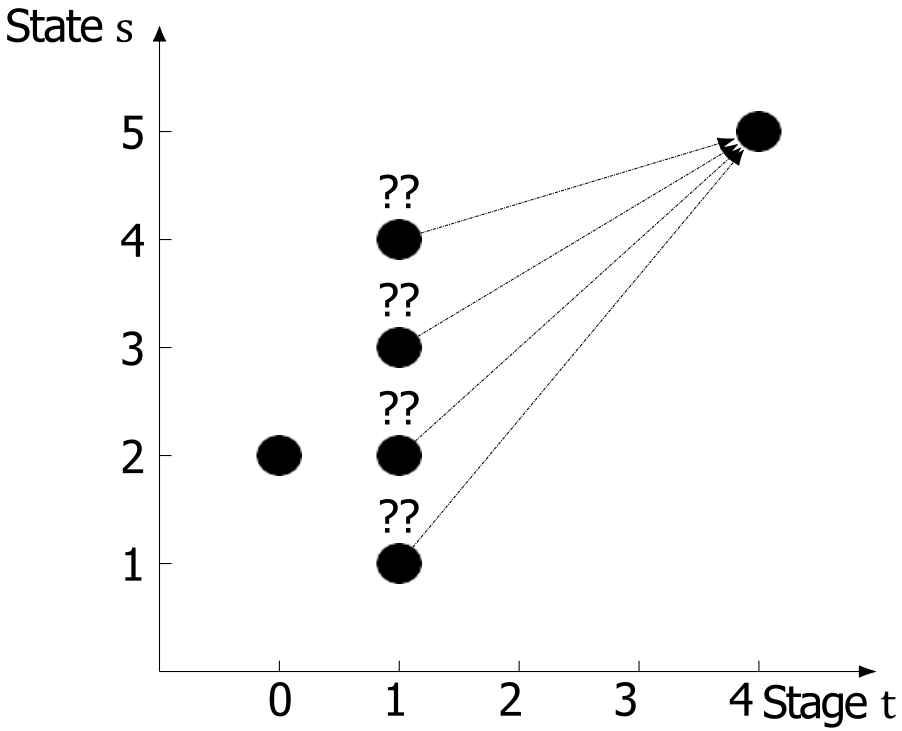


Destination is node 5.

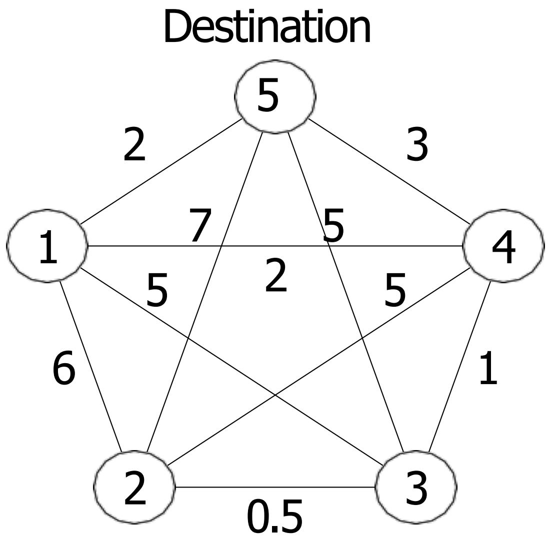
Solving Shortest Path



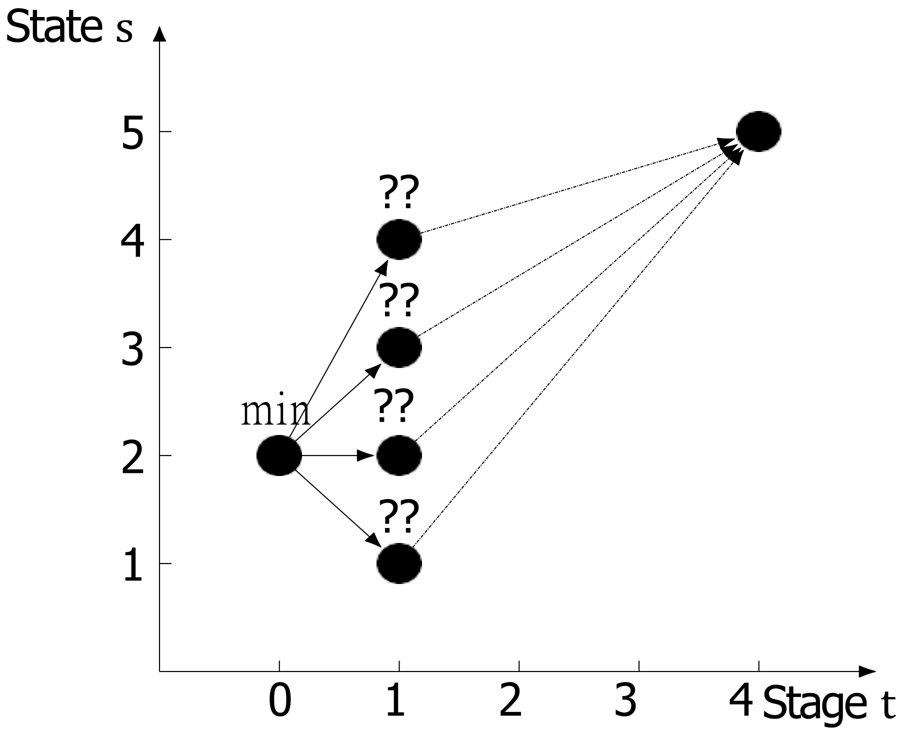
Destination is node 5.



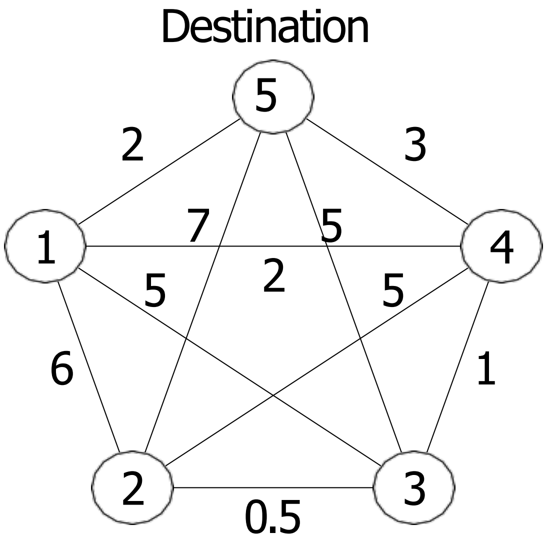
Solving Shortest Path



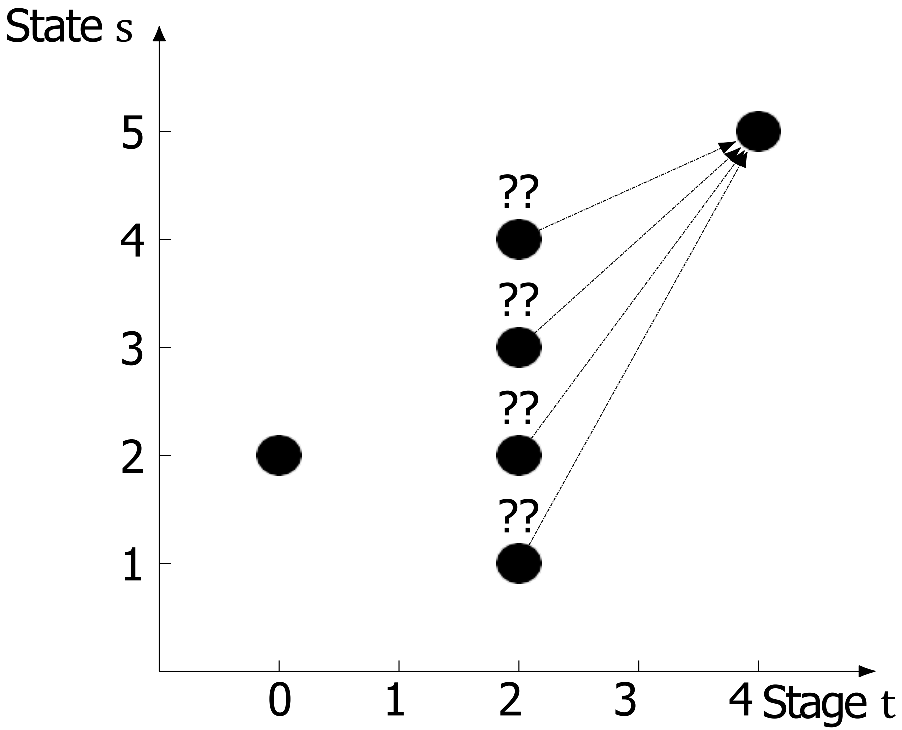
Destination is node 5.



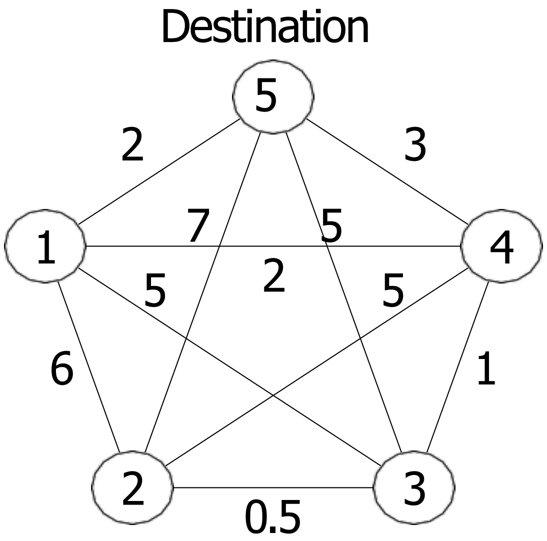
Solving Shortest Path



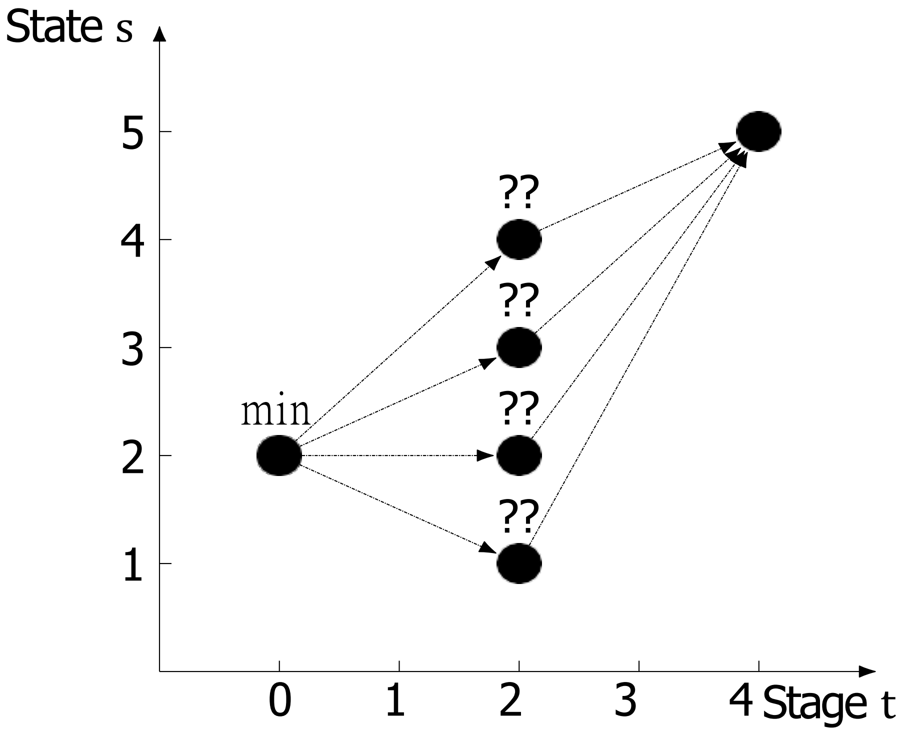
Destination is node 5.



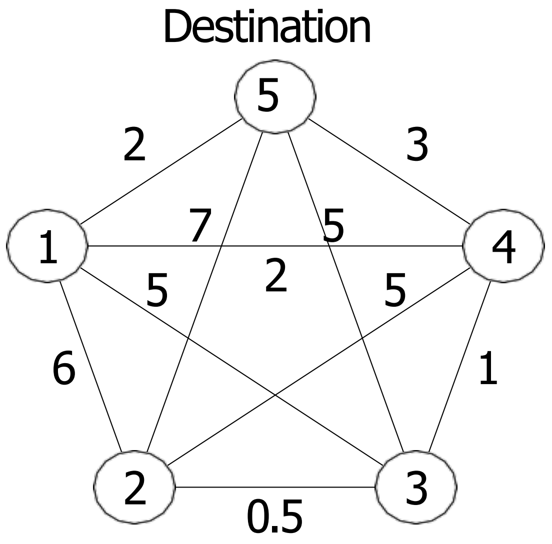
Solving Shortest Path



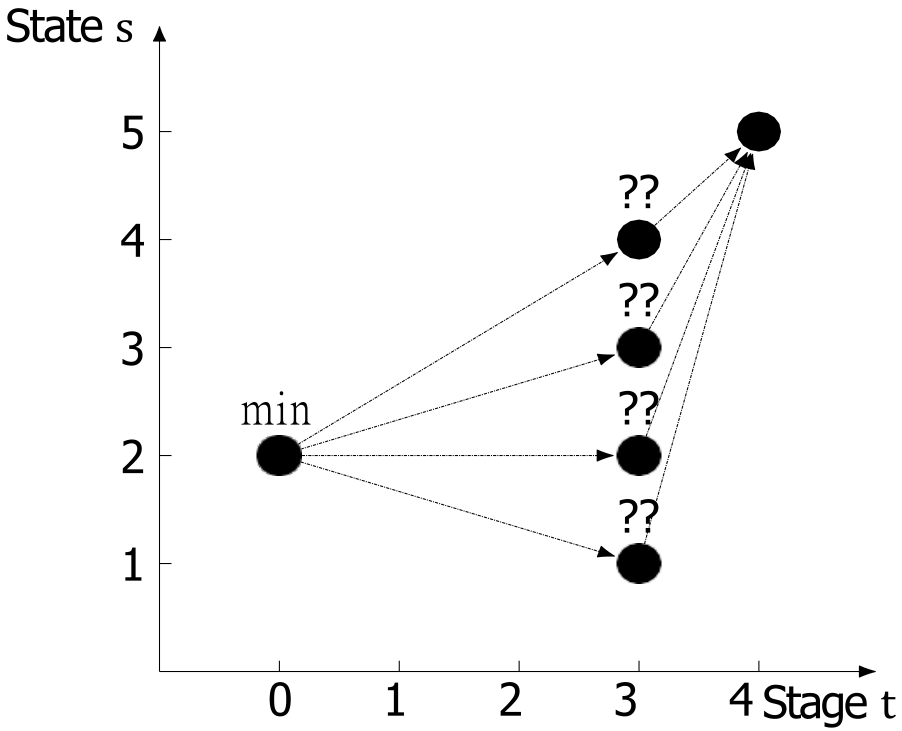
Destination is node 5.



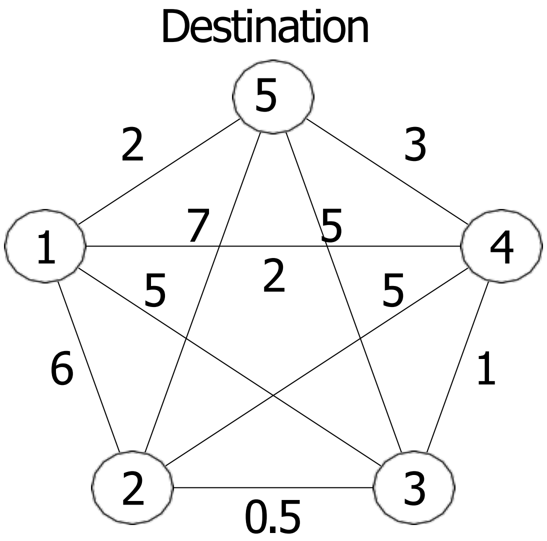
Solving Shortest Path



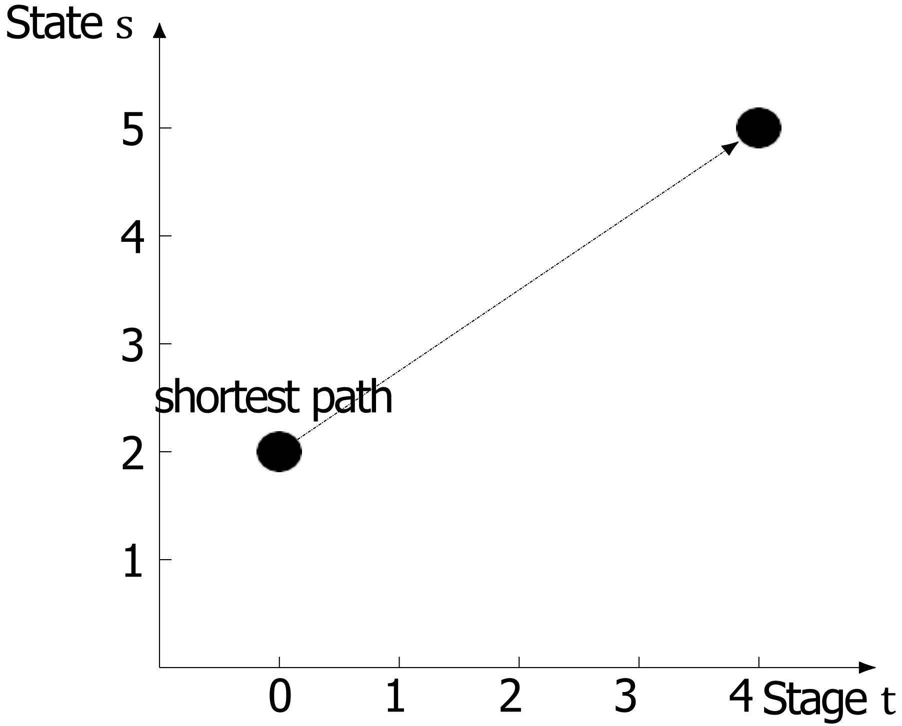
Destination is node 5.



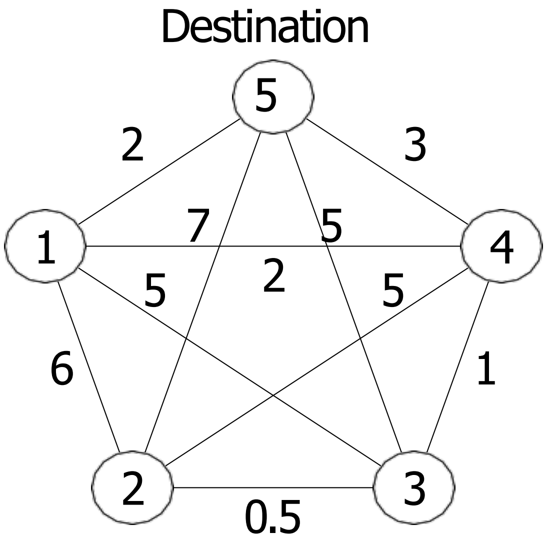
Solving Shortest Path



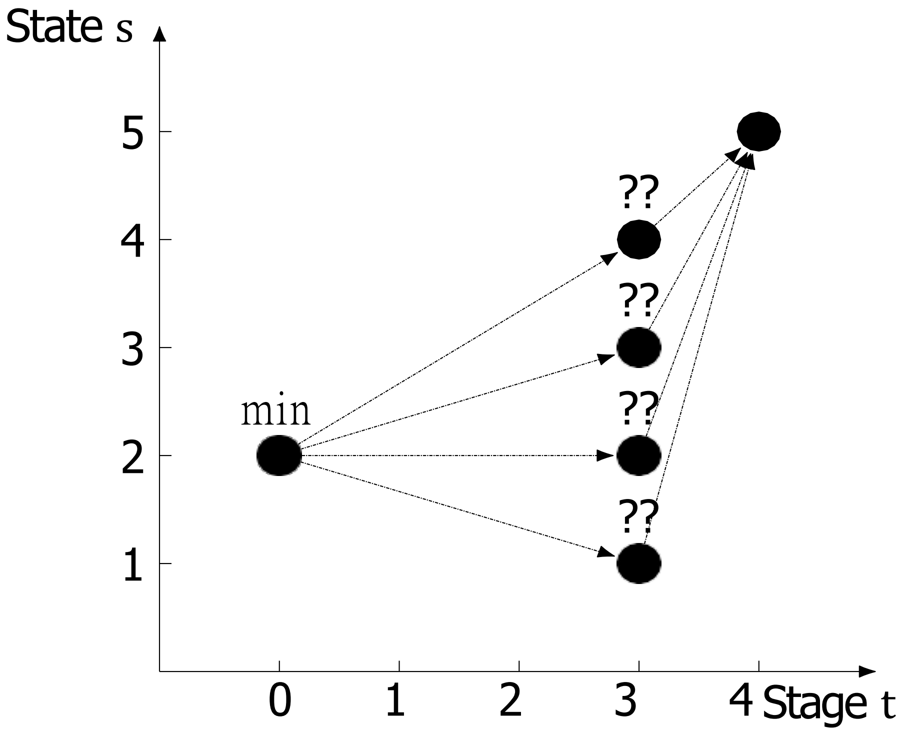
Destination is node 5.



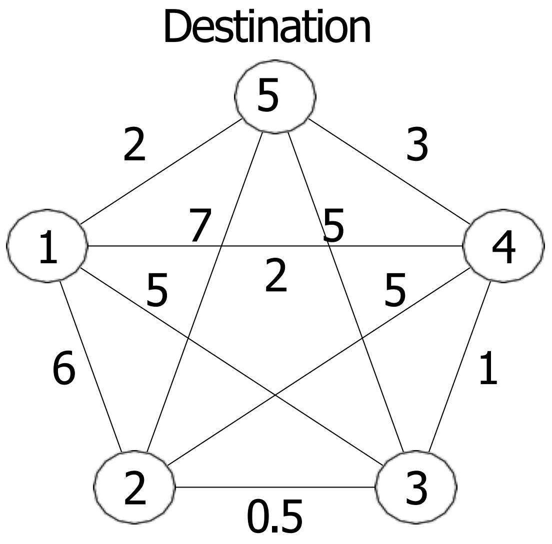
Solving Shortest Path



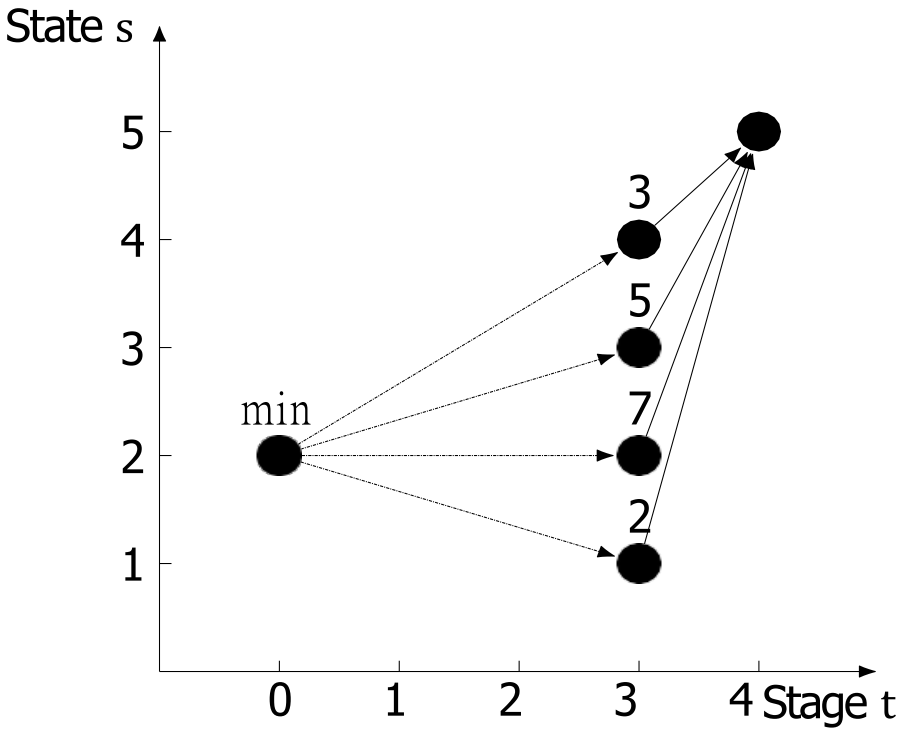
Destination is node 5.



Solving Shortest Path

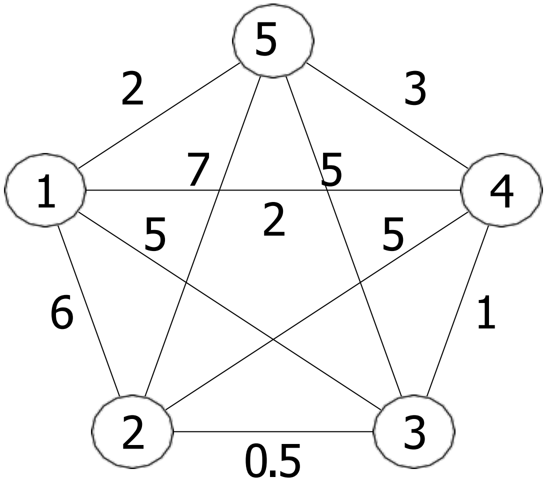


Destination is node 5.

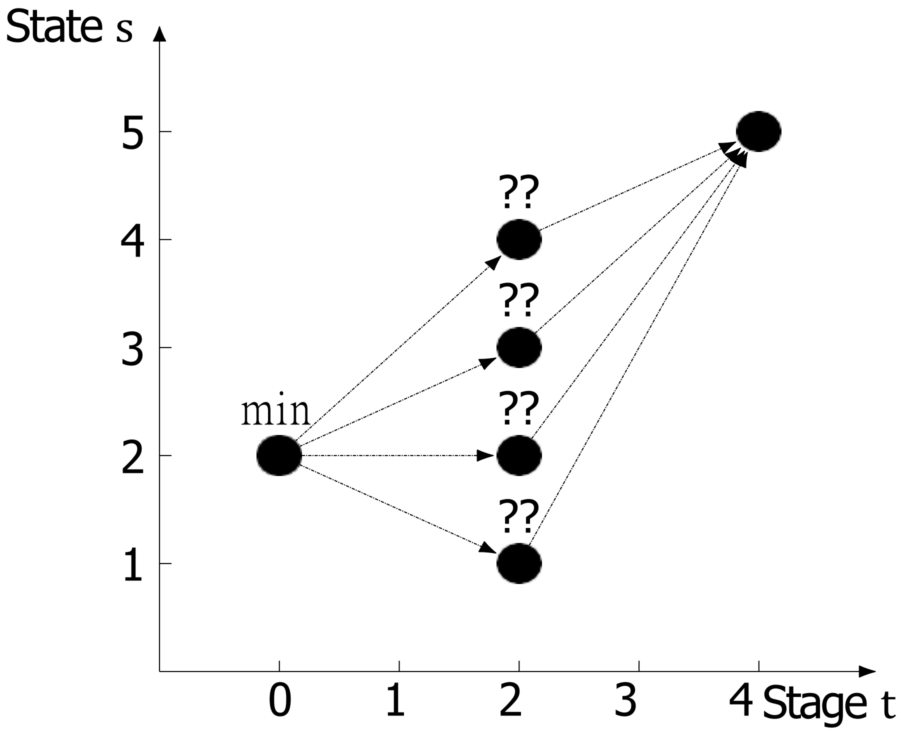


Solving Shortest Path

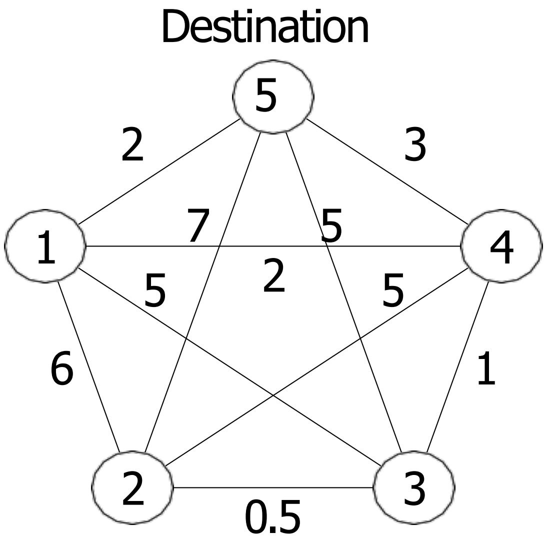
Destination



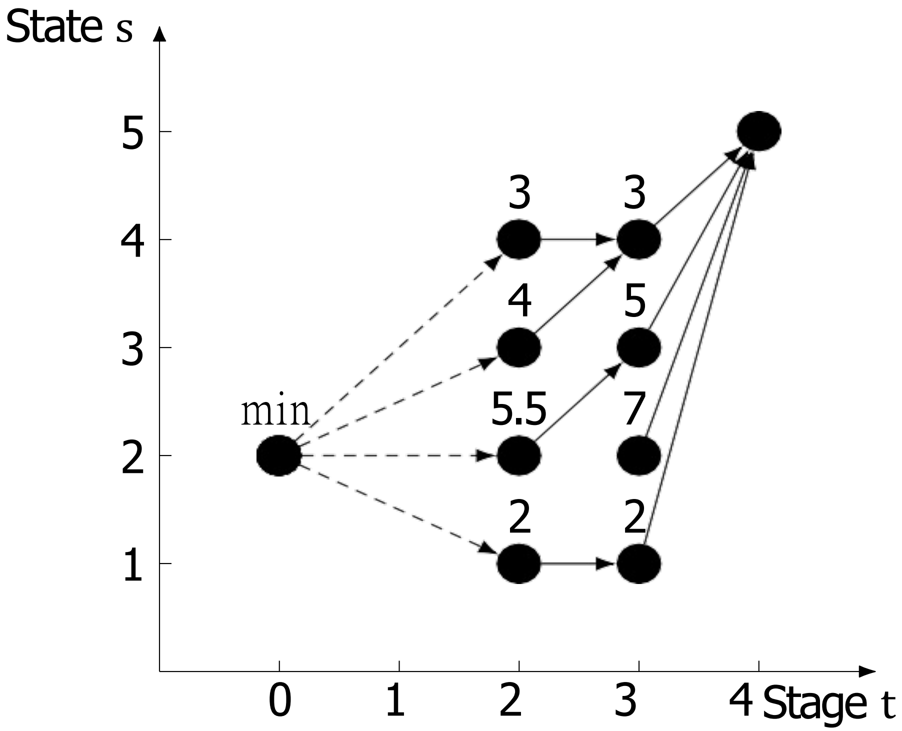
Destination is node 5.



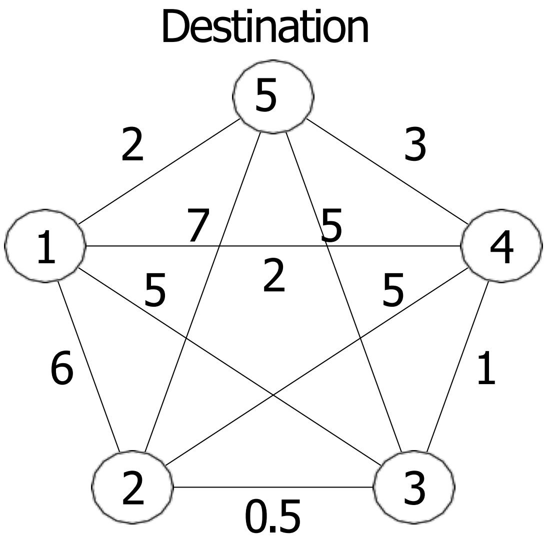
Solving Shortest Path



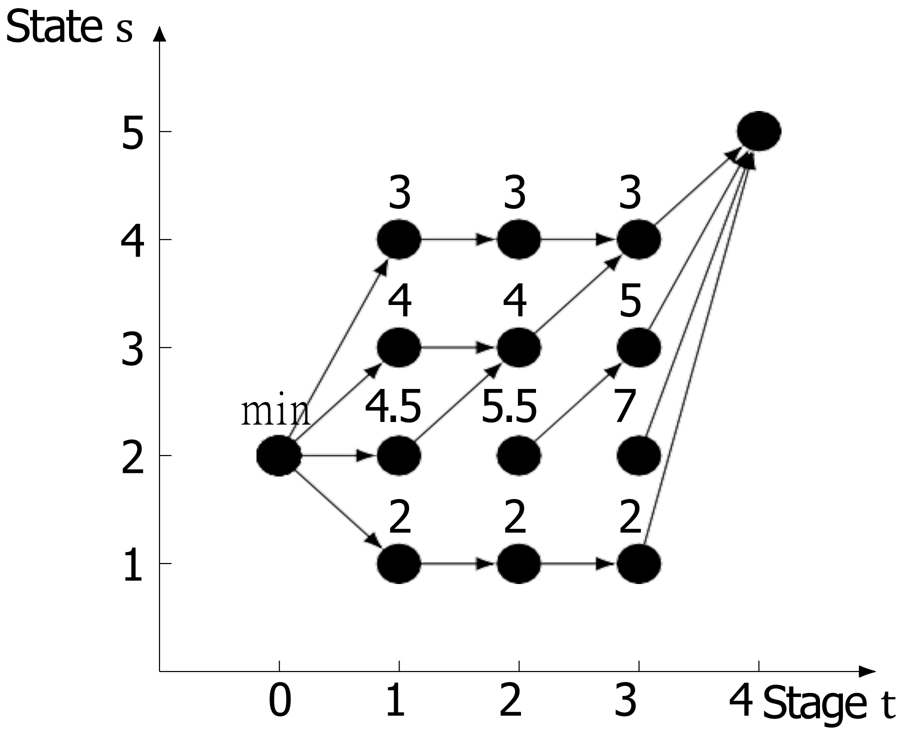
Destination is node 5.



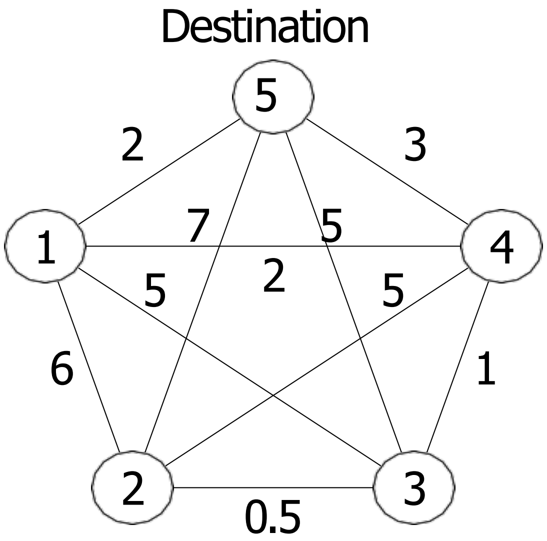
Solving Shortest Path



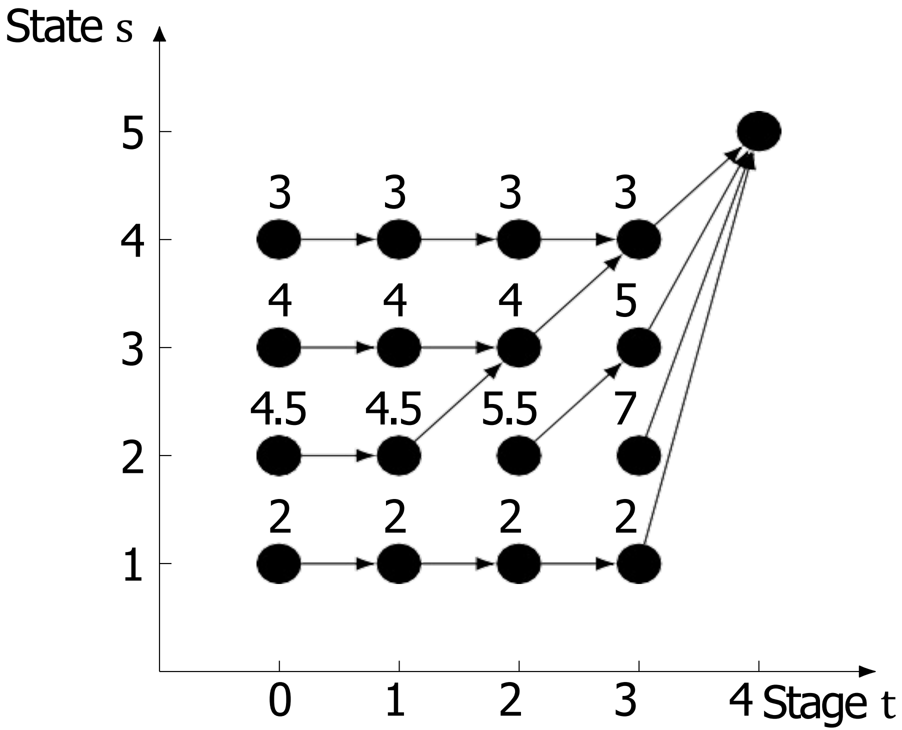
Destination is node 5.



Solving Shortest Path

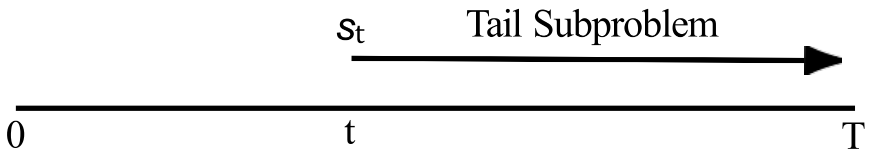


Destination is node 5.

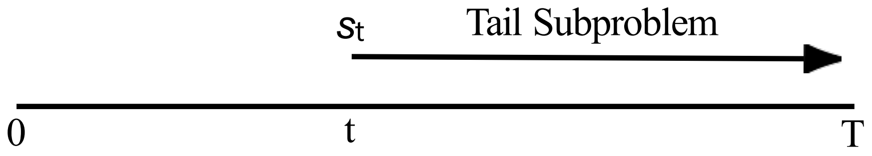


Bellman's Principle of optimality (1957)

"An optimal policy has the property that, whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."



Principle of optimality (Bellman, 1957)



Principle (Optimality)

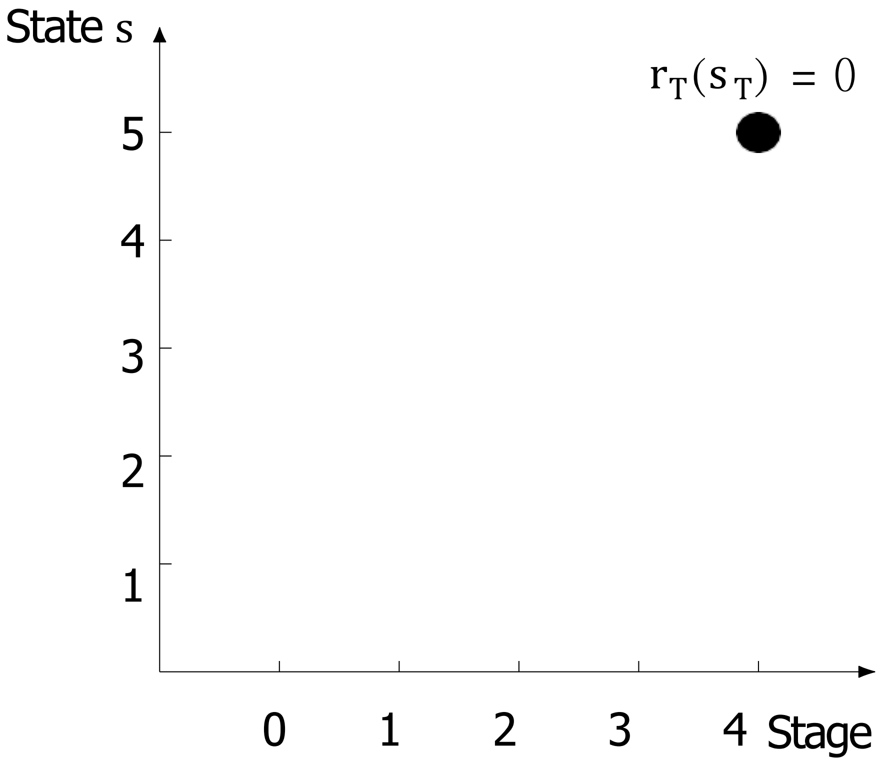
Let $\{a_0^*, \ldots, a_{T-1}^*\}$ be an optimal action sequence, which together with s_0 and $\{\epsilon_0, \ldots, \epsilon_{T-1}\}$ determines the corresponding state sequence $\{s_1^*, \ldots, s_T^*\}$ via the state transition function. Consider the subproblem whereby we start at s_t^* at time t and wish to maximize the value function from time t to time T,

$$r_t(s_t^*) + \sum_{\tau=t+1}^{T-1} r_\tau(s_\tau, a_\tau) + r_T(s_T)$$

over $\{a_t, \ldots, a_{T-1}\}$ with $a_\tau \in A_\tau(s_\tau), \tau = t, \ldots, T-1$. Then, the truncated optimal action sequence $\{a_t^*, \ldots, a_{T-1}^*\}$ is optimal for this subproblem.

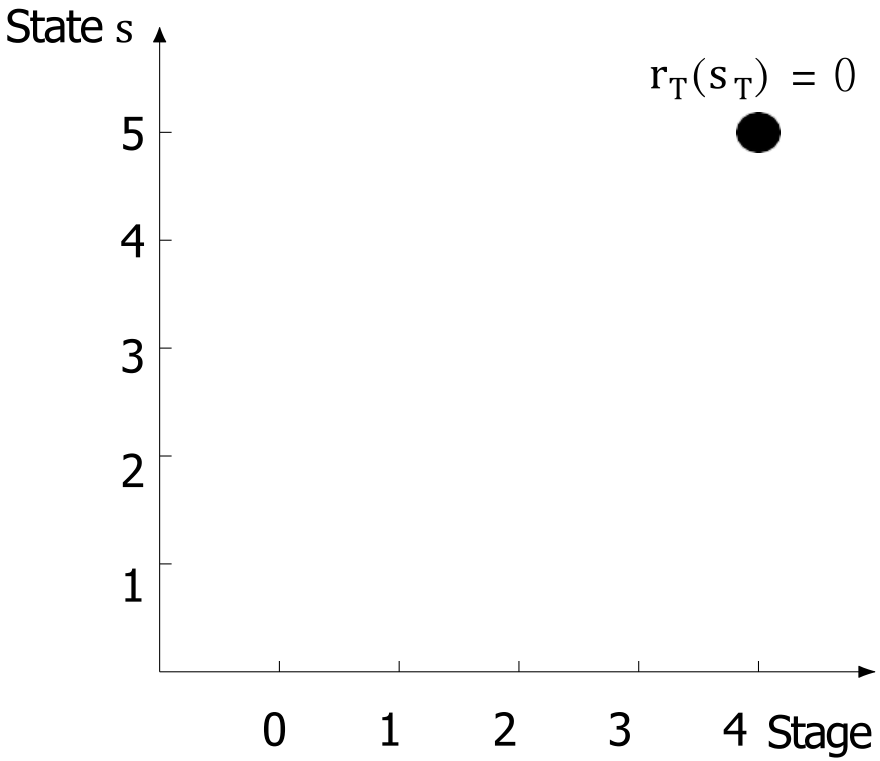
Dynamic programming algorithm

$V_T(s_T) = r_T(s_T)$



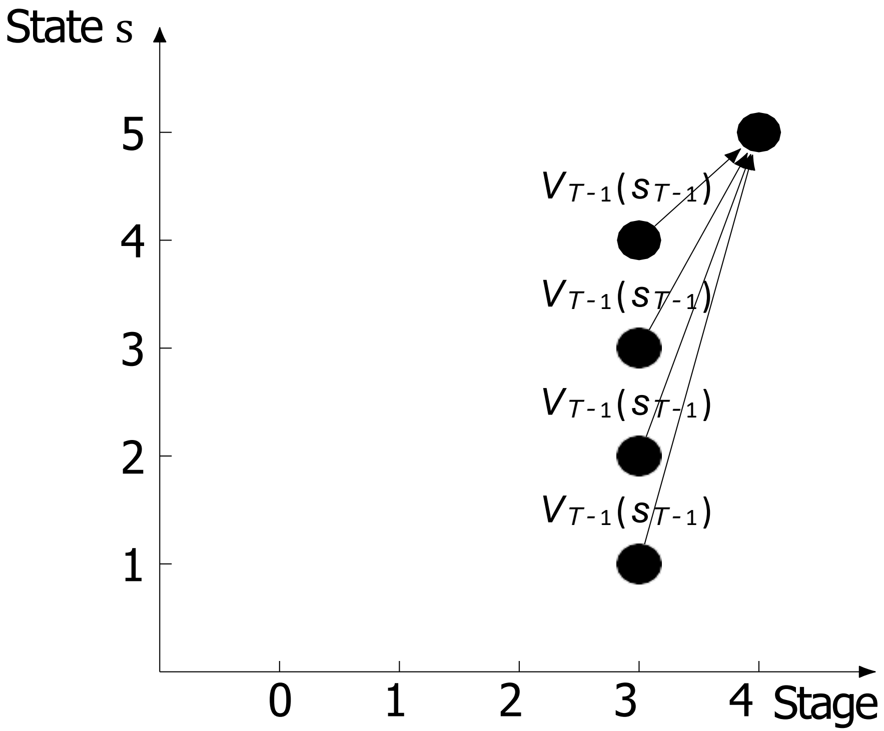
Dynamic programming algorithm

$V_T(s_T) = r_T(s_T)$
for $t = T - 1, \ldots, 0$ **do**



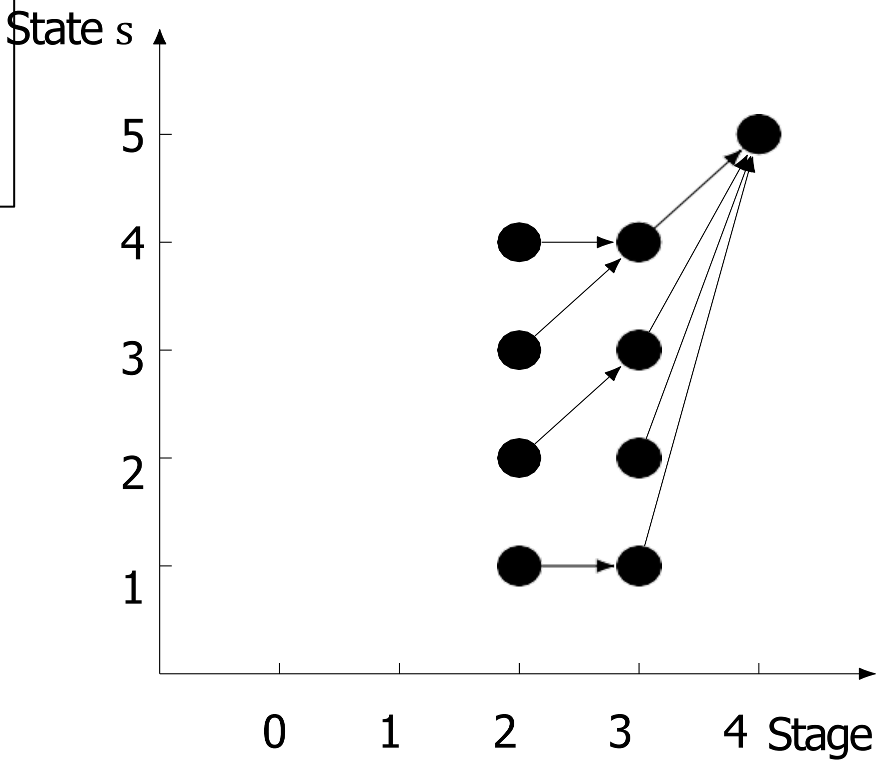
Dynamic programming algorithm

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \dots, 0$ **do**
  for $s_t \in \mathcal{S}_t$ **do**



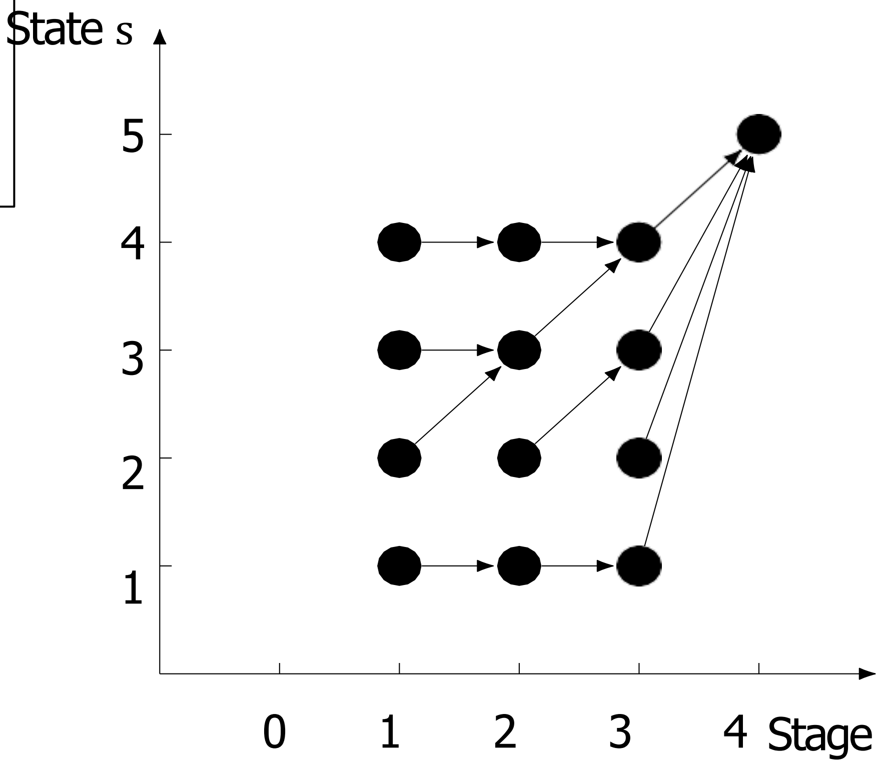
Dynamic programming algorithm

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \dots, 0$ **do**
  for $s_t \in \mathcal{S}_t$ **do**
    $V_t(s_t) = \max_{a_t \in \mathcal{A}_t(s_t)} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$ where $s_{t+1} = f(s_t, a_t)$
end for



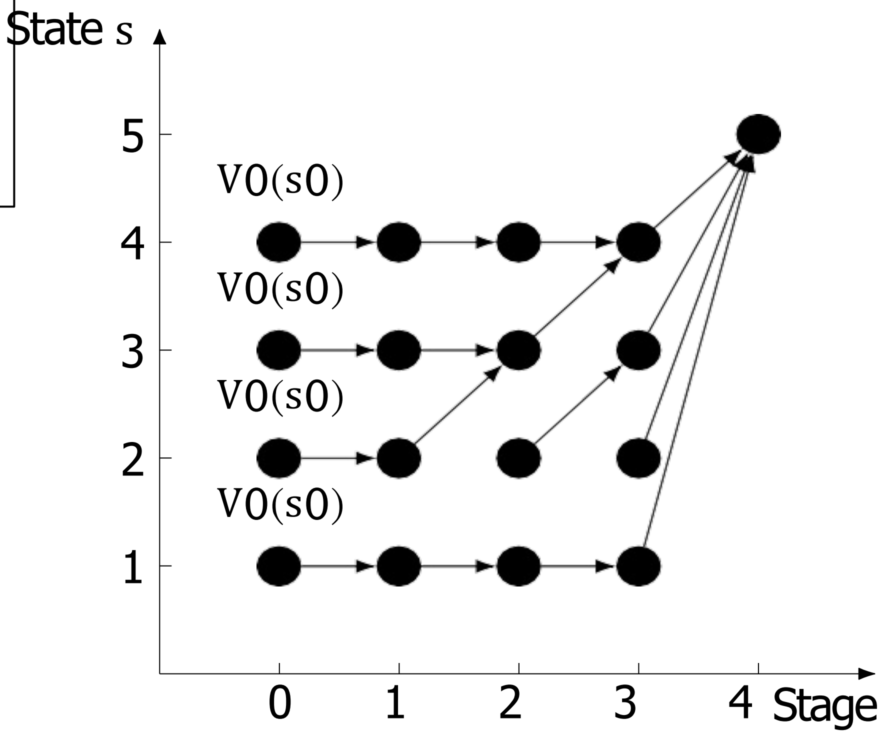
Dynamic programming algorithm

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \dots, 0$ **do**
  for $s_t \in \mathcal{S}_t$ **do**
    $V_t(s_t) = \max_{a_t \in \mathcal{A}_t(s_t)} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$ where $s_{t+1} = f(s_t, a_t)$
end for



Dynamic programming algorithm

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
  for $s_t \in \mathcal{S}_t$ **do**
    $V_t(s_t) = \max_{a_t \in \mathcal{A}_t(s_t)} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$ where $s_{t+1} = f(s_t, a_t)$
end for



Dynamic programming algorithm

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
  for $s_t \in \mathcal{S}_t$ **do**
    $V_t(s_t) = \max_{a_t \in \mathcal{A}_t(s_t)} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$ where $s_{t+1} = f(s_t, a_t)$
end for

Theorem (Dynamic programming)

For every initial state s_0, the optimal value $V^*(s_0)$ is equal to $V_0(s_0)$, given above.
Furthermore, if $a_t^* = \pi_t^*(s_t)$ maximizes the right side of the above for each s_t and t, the policy $\pi^* = (\pi_0^*, \ldots, \pi_{T-1}^*)$ is optimal.

Dynamic programming algorithm

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \dots, 0$ **do**
  for $s_t \in \mathcal{S}_t$ **do**
    $V_t(s_t) = \max_{a_t \in \mathcal{A}_t(s_t)} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$ where $s_{t+1} = f(s_t, a_t)$
end for

- **Proof**: by induction
- "Efficient": O(|S||A|T)
- Equivalent to Bellman-Ford algorithm
- **Strength**: Generality
- Much better than naive approach O(T!)
- **Weakness**: ALL the tail subproblems are solved

Proof of the induction step

Let $f_t: S \times A \rightarrow S$ denote the transition *function*.
Denote tail policy from time t onward as $\pi_{t:T-1} = \{\pi_t, \pi_{t+1}, \ldots, \pi_{T-1}\}$
Assume that $V_{t+1}(s_{t+1}) = V_{t+1}^*(s_{t+1})$. Then:

$$V_t^*(s_t) = \max_{(\pi_t, \pi_{t+1:T-1})} r_t(s_t, \pi_t(s_t)) + r_T(s_T) + \sum_{i=t+1}^{T-1} r_i(s_i, \pi_i(s_i))$$

$$= \max_{\pi_t} r_t(s_t, \pi_t(s_t)) + \max_{\pi_{t+1:T-1}} \left[r_T(s_T) + \textstyle\sum_{i=t+1}^{T-1} r_i(s_i, \pi_i(s_i))\right]$$

$$= \max_{\pi_t} r_t(s_t, \pi_t(s_t)) + V_{t+1}^*(f_t(s_t, \pi_t(s_t)))$$

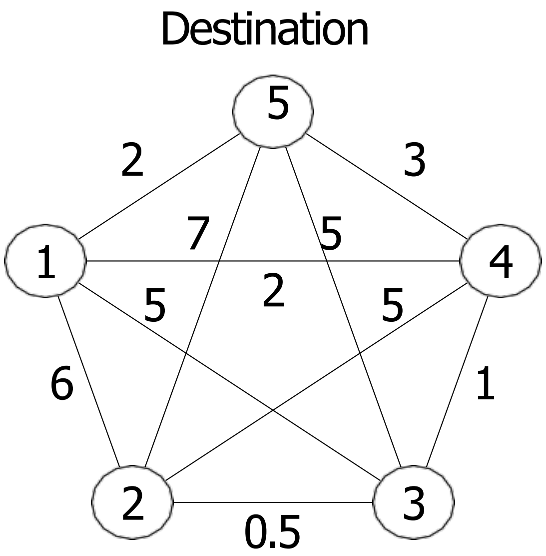
$$= \max_{\pi_t} r_t(s_t, \pi_t(s_t)) + V_{t+1}(f_t(s_t, \pi_t(s_t)))$$

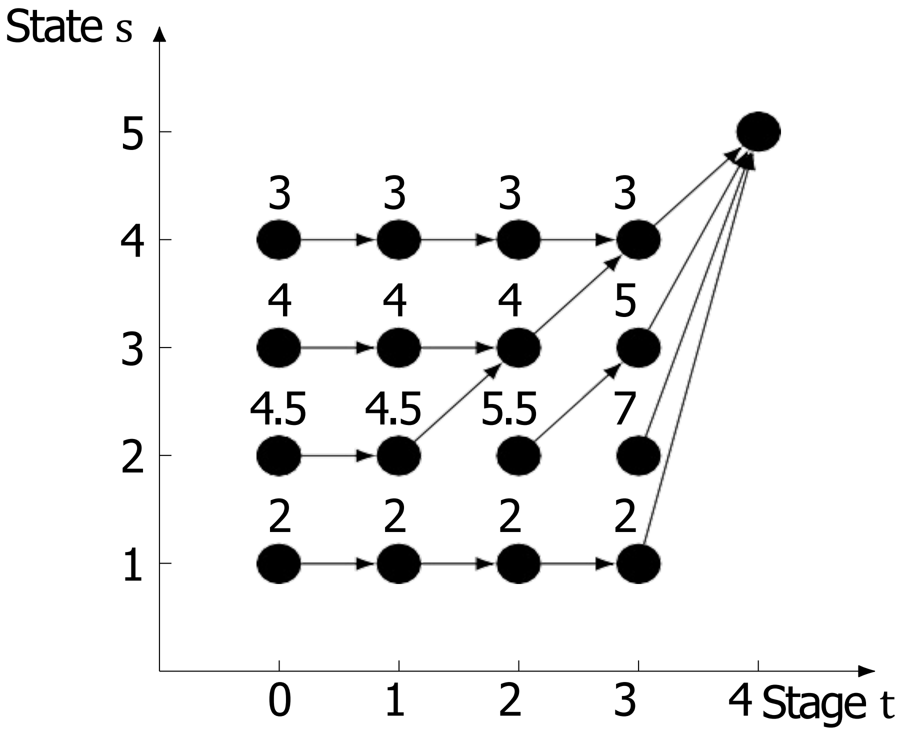
$$= \max_{a_t \in \mathcal{A}_t(s_t)} r_t(s_t, a_t) + V_{t+1}(f_t(s_t, a_t))$$

$$= V_t(s_t)$$

Interpretation as optimal reward-to-go (cost-to-go) function.

Solving Shortest Path

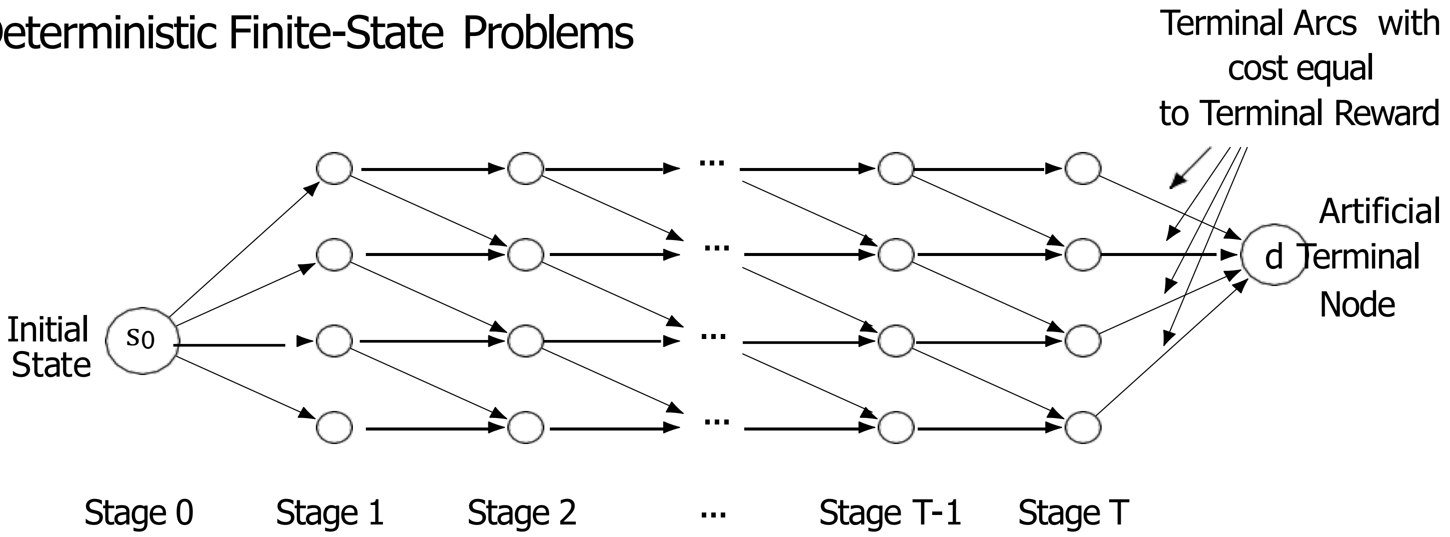




Destination is node 5.

Sequential decision making as shortest path

For Deterministic Finite-State Problems



Example: Automated vehicle

30mph → 35mph / 30mph / 25mph / ...

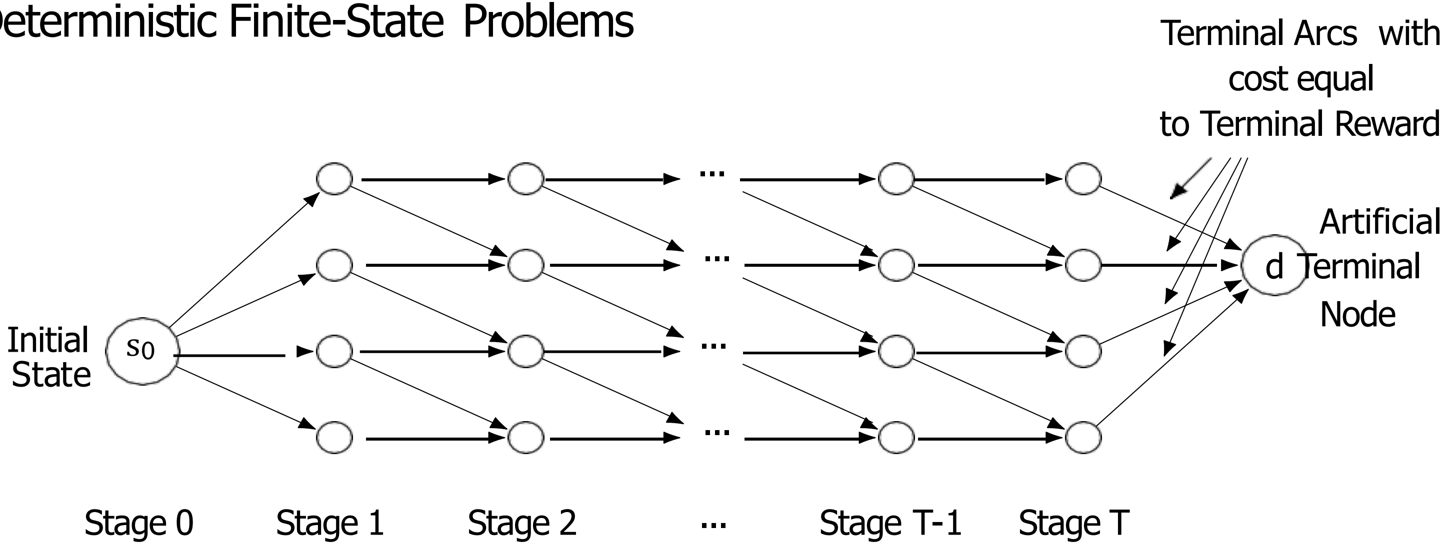
... ...

35mph
30mph
25mph
...

Applications:
platooning, eco-driving, lane change assist, merge assist, parking assist

Sequential decision making as shortest path

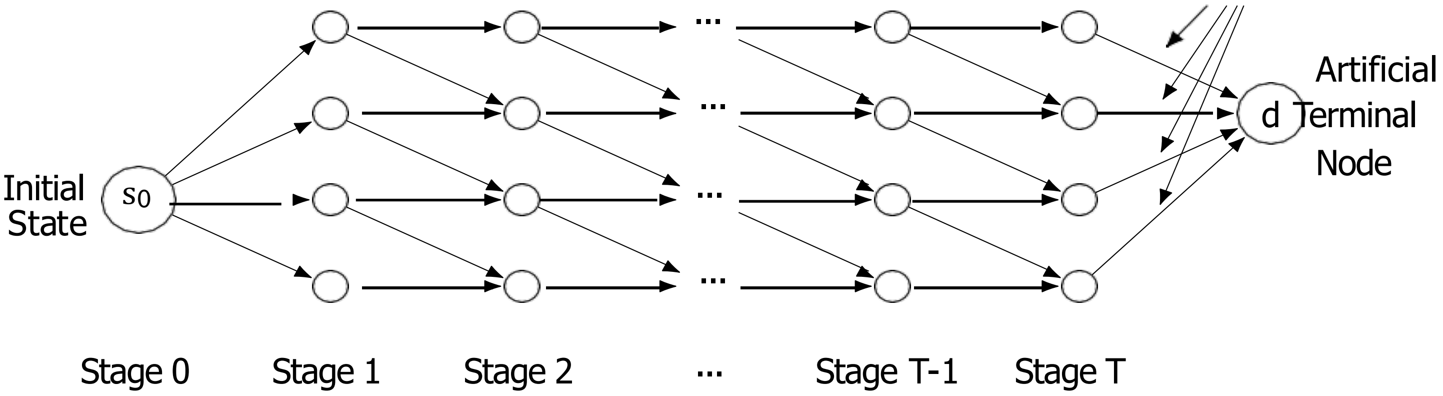
For Deterministic Finite-State Problems



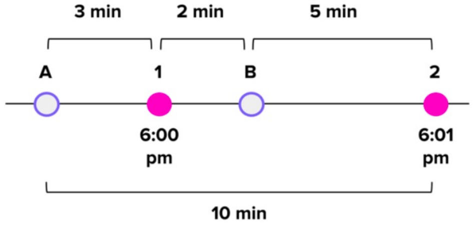
Discuss: If shortest path isn't hard, why are DP problems still challenging?

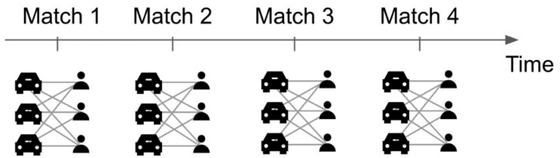
Sequential decision making as shortest path

For Deterministic Finite-State Problems



Example: Real-time ridesharing





Each stage may have

$$|A| = N!$$

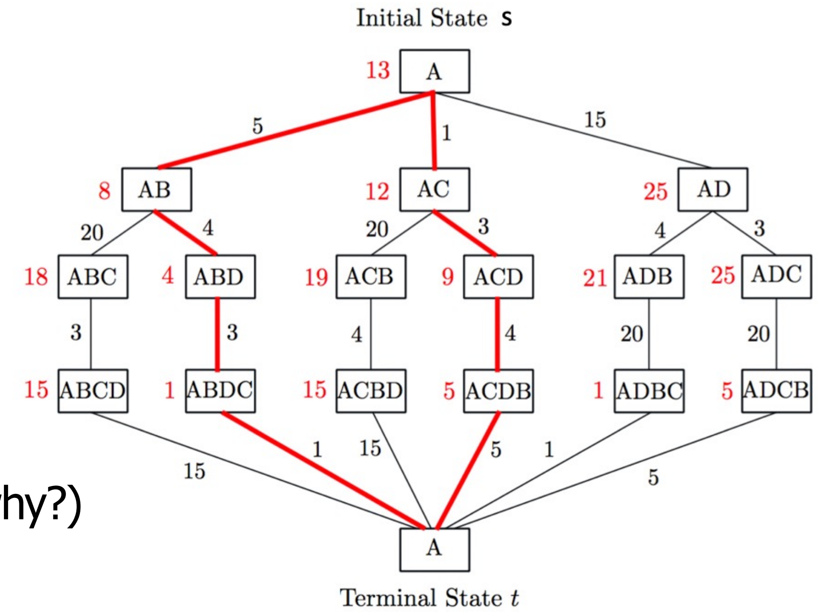
for N drivers and N riders.

X. Azagirre *et al.*, "A Better Match for Drivers and Riders: Reinforcement Learning at Lyft." INFORMS Journal on Applied Analytics, 2023. doi: 10.48550/arXiv.2310.13810.

Sequential decision making can get hairy

Example: traveling salesman problem (TSP)

- N cities.
- **Goal:** Find the shortest tour (visit every city exactly once and return home).
- In this case, can't get around exponential. (why?)
- $|S| = O(N!)$, $|A| = N$, $T = N$, so $O(|S||A|T) = O(N!)$.
- (Actually, DP *is* slightly better: $|S| = O(2^N N^2)$.)
- This is called the **curse of dimensionality**.

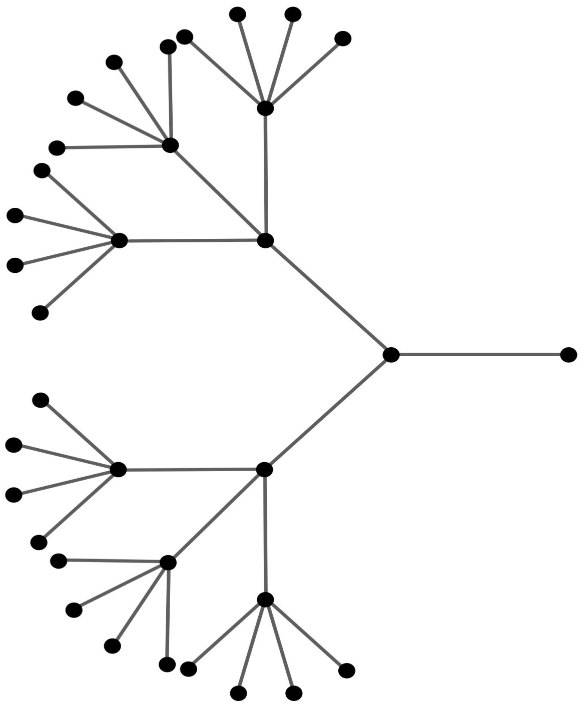


	5	1	15
5		20	4
1	20		3
15	4	3	

Sequential decision making can get hairy

Example: traveling salesman problem (TSP)

- N cities.
- **Goal**: Find the shortest tour (visit every city exactly once and return home).
- In this case, can't get around exponential. (why?)
- $|S| = O(N!)$, $|A| = N, T = N$, so $O(|S||A|T) = O(N!)$.
- (Actually, DP *is* slightly better: $|S| = O(2^N N^2)$.)
- This is called the **curse of dimensionality**.



(Recall) Key challenge: huge state spaces

- State: location, time, and vehicle type
  - Location is encoded from geohash6 (precise location) [1,600] and geohash5 (neighborhood) [50]
  - Time encoded from hour-of-week categories [168]
  - Vehicle type: standard, luxury, SUV, or handicap accessible [4]
- State space is ≈1600x50x168x4=54M
- For reference: SF Bay Area population is 8M
  - Naïve approach: Would need everyone to take at least 7 rides to gather enough data

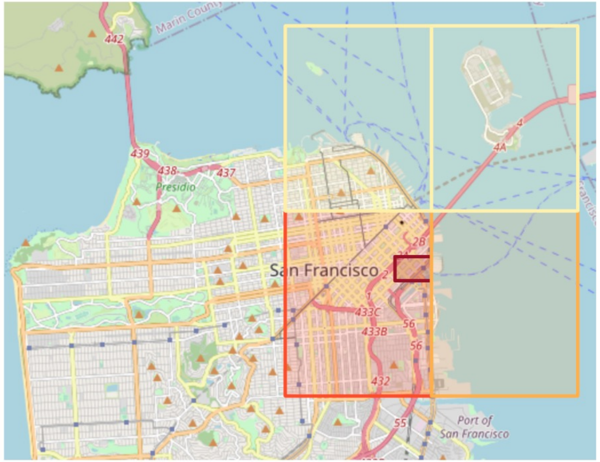


Figure 4: Spatial factor weights are weighted and normalized by the inverse of the distance from the geohash centroid to smoothly interpolate the four closest geohash5 state factors. A similiar interpolation is applied using the two nearest hours of the week, yielding a cross-product of eight spatiotemporal factors and weights. Additional factors also consider the vehicle type, such as standard, luxury, SUV, or handicap accessible.

Cannot only explore. Cannot only exploit.
Must trade off exploration and exploitation.

B. Han, H. Lee, and S. Martin, "Real-Time Rideshare Driver Supply Values Using Online Reinforcement Learning," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, in KDD '22. Aug. 2022. doi: 10.1145/3534678.3539141.

Optimal capacity expansion

A regional automotive company is planning a large investment in electric vehicle (EV) manufacturing plants over the next few years.

Table E11.1 Demand and cost per plant ($\$ \times 1000$)

Year	*Cumulative demand* (in number of plants)	*Cost per plant* ($\$ \times 1000$)
2025	1	5400
2026	2	5600
2027	4	5800
2028	6	5700
2029	7	5500
2030	8	5200

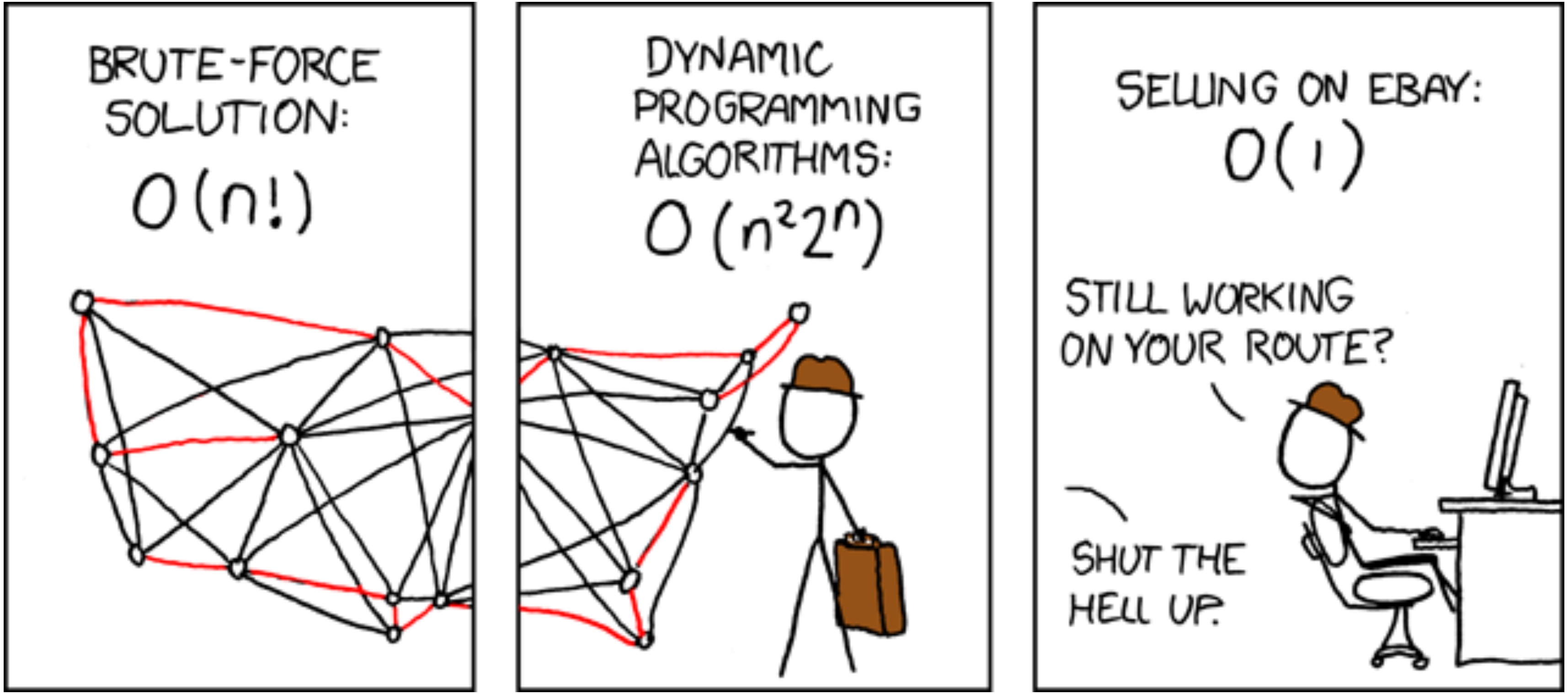
- **A total of eight manufacturing plants must be built over the next six years** because of both increasing demand in the region and the energy crisis, which has forced the closing of certain of their antiquated internal combustion engine (ICE) vehicle plants.
- *Minimum-demand schedule*: Assume that demand for electric vehicles in the region is known with certainty (deterministic) and that **we must satisfy the minimum levels of cumulative demand indicated in Table E11.1**.
- The demand here has been converted into equivalent numbers of manufacturing plants required by the end of each year.

Optimal capacity expansion

- **The building of EV manufacturing plants takes approximately one year**.
- In addition to a cost directly associated with the construction of a plant, there is a **common cost of $1.5 million incurred when any plants are constructed** in any year, independent of the number of plants constructed.
  - This common cost results from contract preparation.
- In any given year, at most three plants can be constructed.
- The cost of construction per plant is given in Table E11.1 for each year in the planning horizon.
  - These costs are currently increasing due to the elimination of an investment tax credit designed to speed investment in EVs.
  - However, new technology should be available by 2028, which will tend to bring the costs down, even given the elimination of the investment tax credit.

Table E11.1 Demand and cost per plant ($ ×1000)

Year	*Cumulative demand* (in number of plants)	*Cost per plant* ($ × 1000)
2025	1	5400
2026	2	5600
2027	4	5800
2028	6	5700
2029	7	5500
2030	8	5200



Travelling Salesman Problem (https://xkcd.com/399/)

References

1. Bradley, Stephen P., Arnoldo C. Hax, and Thomas L. Magnanti. **Applied mathematical programming**. Addison-Wesley (1977). Chapter 11: Dynamic Programming.
2. Bertsekas, D. P. (2005). Dynamic programming and optimal control, vol 1. *Belmont, MA: Athena Scientific*, 3rd Edition.
3. Lazaric, A. (2014). Master MVA: Reinforcement Learning.
4. With many slides adapted from Alessandro Lazaric and Matteo Pirotta.