

Sequential Decision Making

Dynamic programming

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](#)]

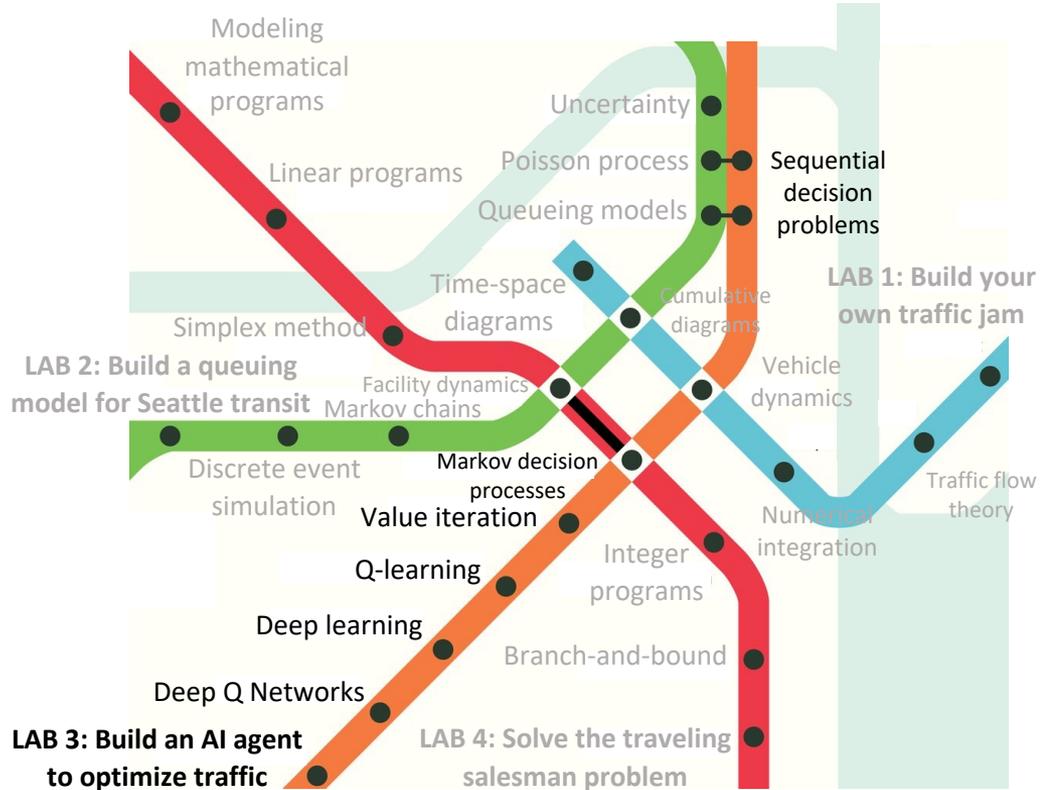
Unit 3: Decision Making Under Uncertainty



Unit 3

Optimizing

Multi-stage



Outline

1. Sequential decision problems
2. Shortest path problems
3. Principle of optimality
4. Dynamic programming algorithm
5. Optimal capacity expansion problem

Outline

1. **Sequential decision problems**
2. Shortest path problems
3. Principle of optimality
4. Dynamic programming algorithm
5. Optimal capacity expansion problem

Sequential decision making

- Dynamic system (assume deterministic):

$$s_{t+1} = f(s_t, a_t)$$

- State s_t , action a_t (decision), time step t

- With objective:

$$\max_{a_0, \dots, a_{T-1}} \sum_{t=0}^{T-1} r_t(s_t, a_t) + r_T(s_T)$$

- Reward functions $r_t(\cdot, \cdot)$, terminal reward $r_T(\cdot)$

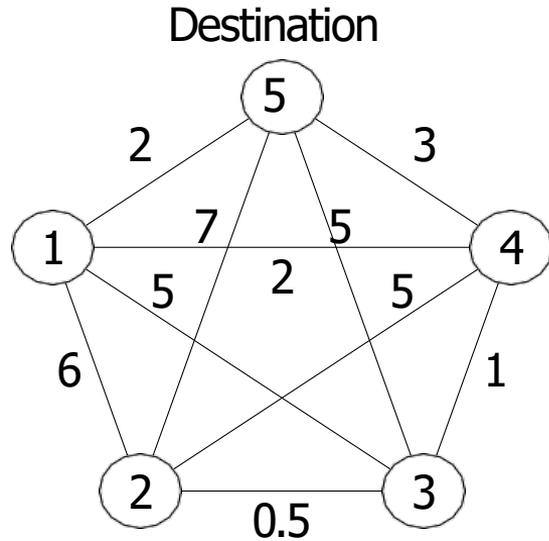
- Example: recall the Container Terminal problem

- At each time step, the port authority must decide the number of berths to staff to balance queue performance with cost

Outline

1. Sequential decision problems
2. **Shortest path problems**
3. Principle of optimality
4. Dynamic programming algorithm
5. Optimal capacity expansion problem

Example: Shortest Path Problem



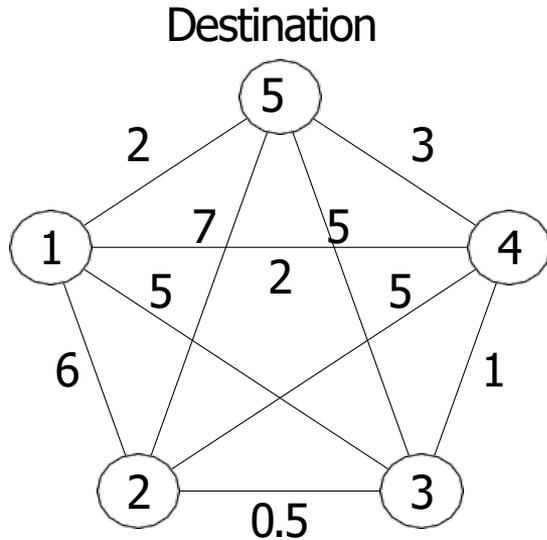
Destination is node 5.

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

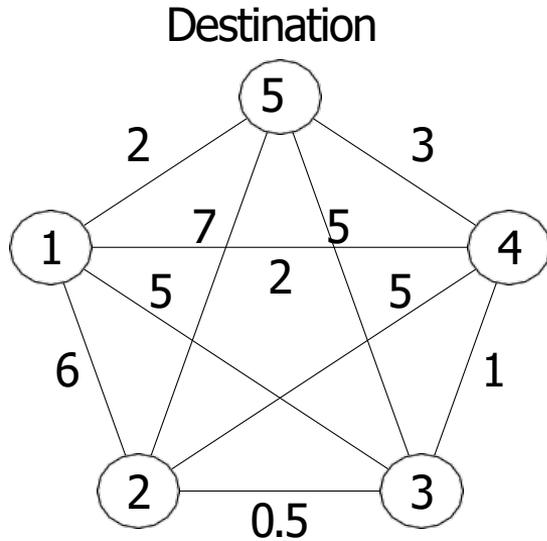
Assumption: all cycles have non-negative length.



Destination is node 5.

- **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).
 - $O(N!)$. ☹️

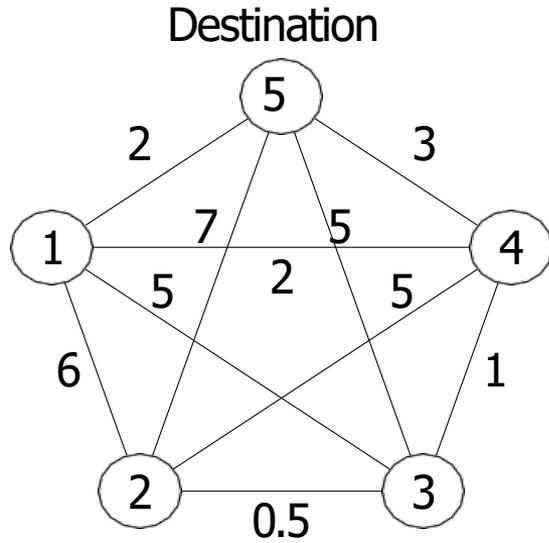
Solving Shortest Path



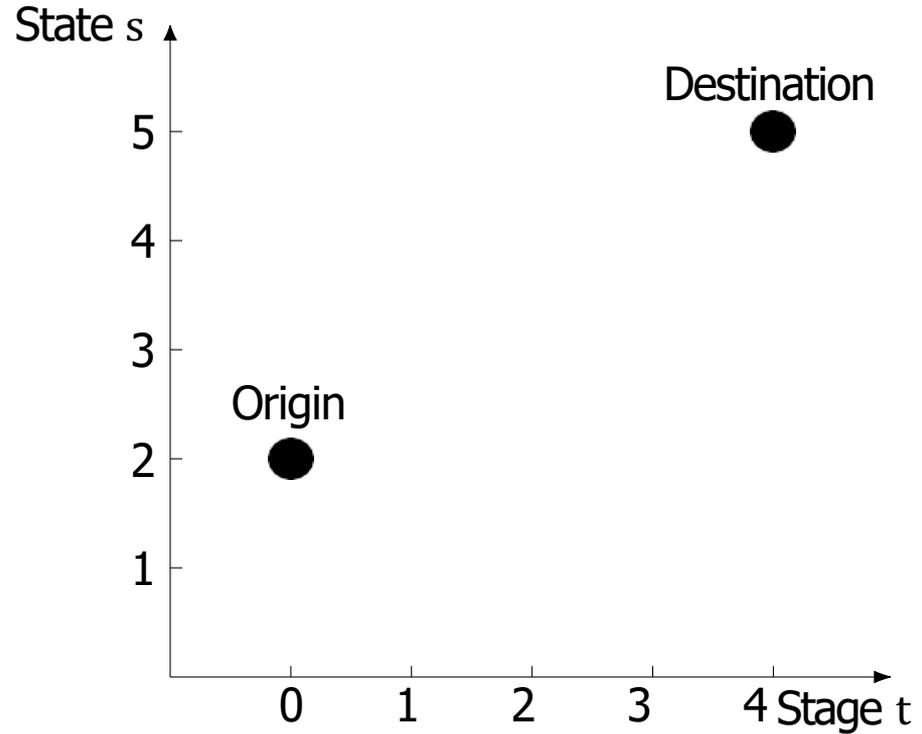
Destination is node 5.

- **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.

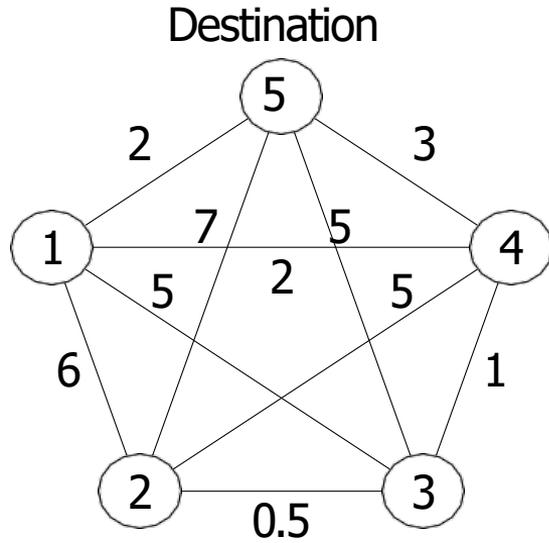
Solving Shortest Path



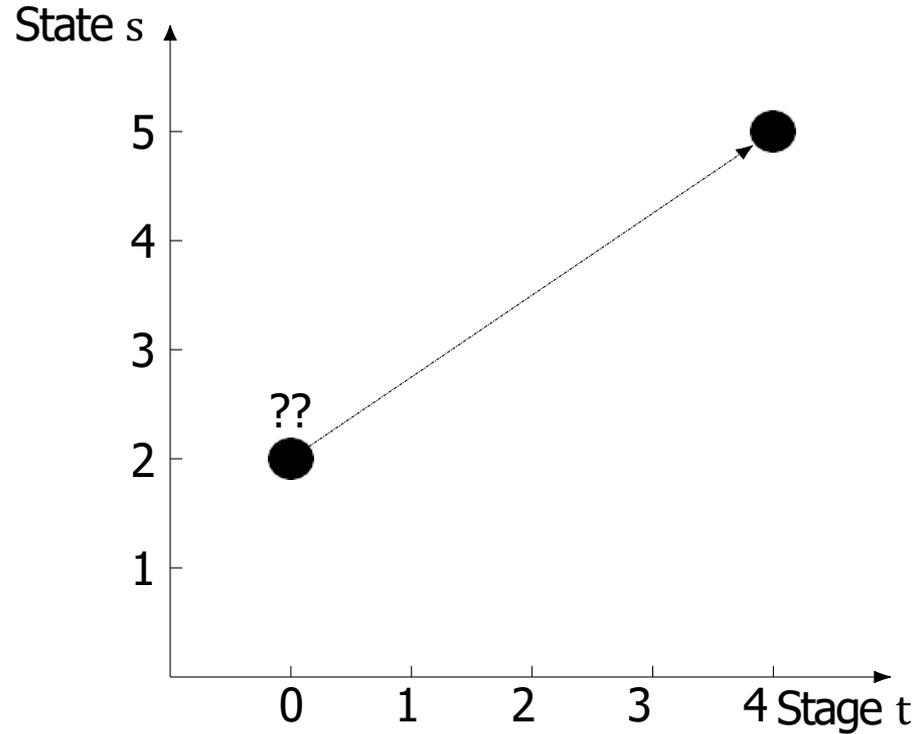
Destination is node 5.



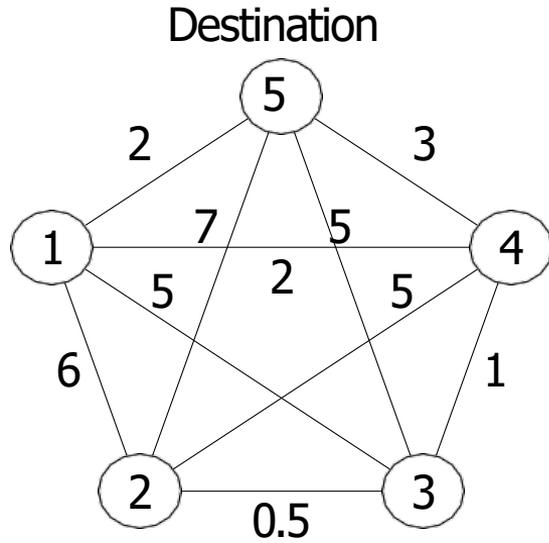
Solving Shortest Path



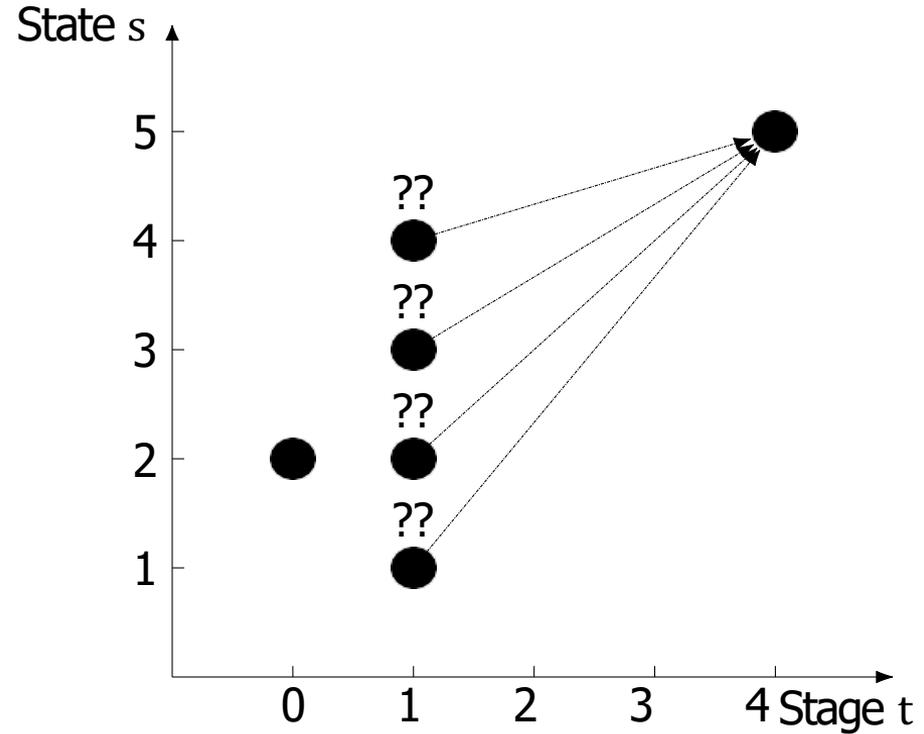
Destination is node 5.



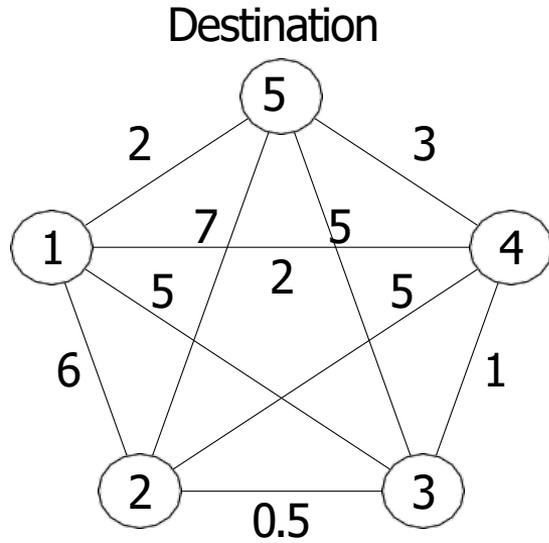
Solving Shortest Path



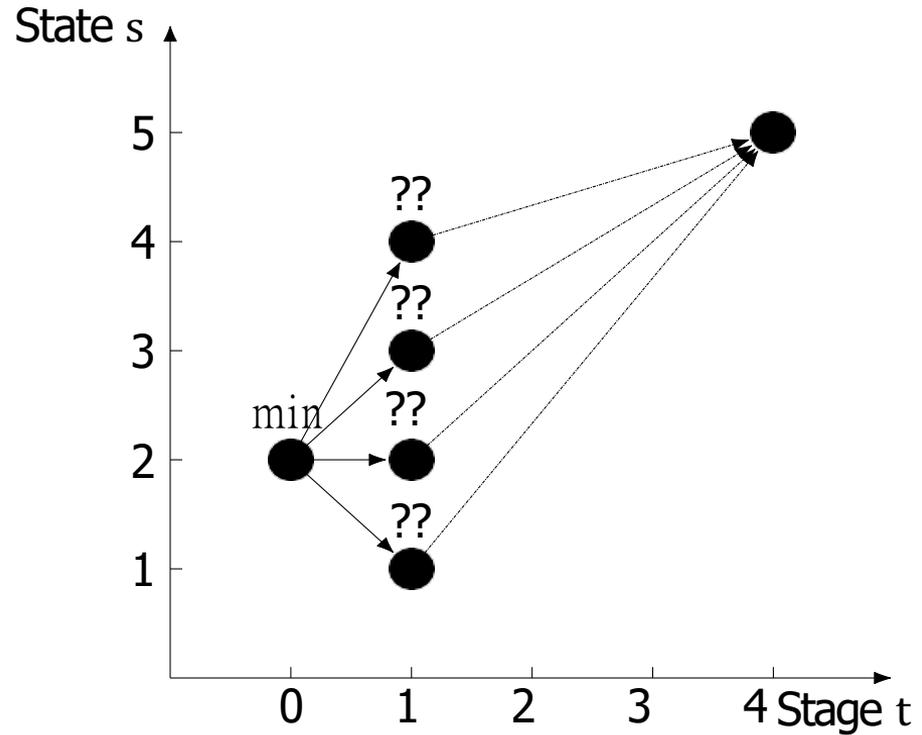
Destination is node 5.



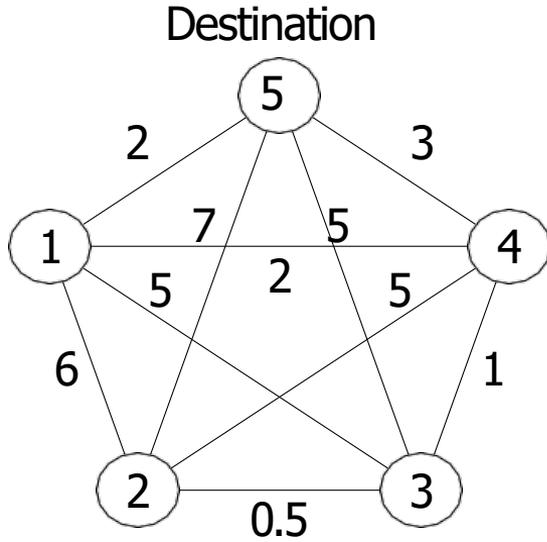
Solving Shortest Path



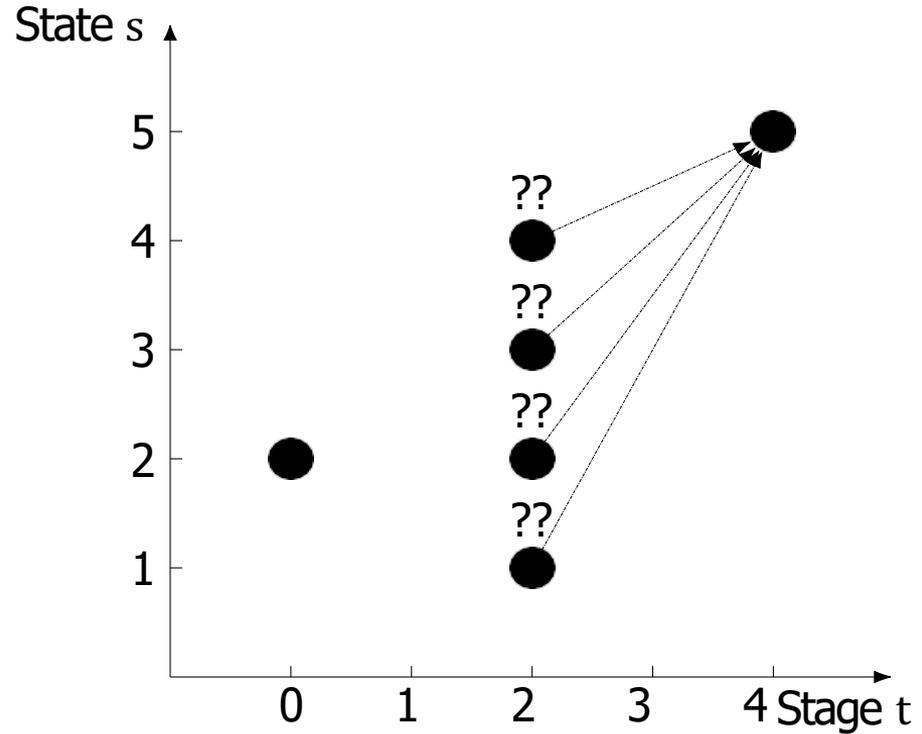
Destination is node 5.



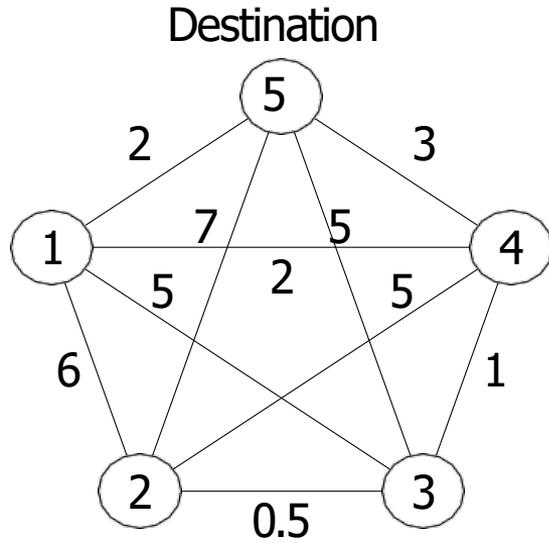
Solving Shortest Path



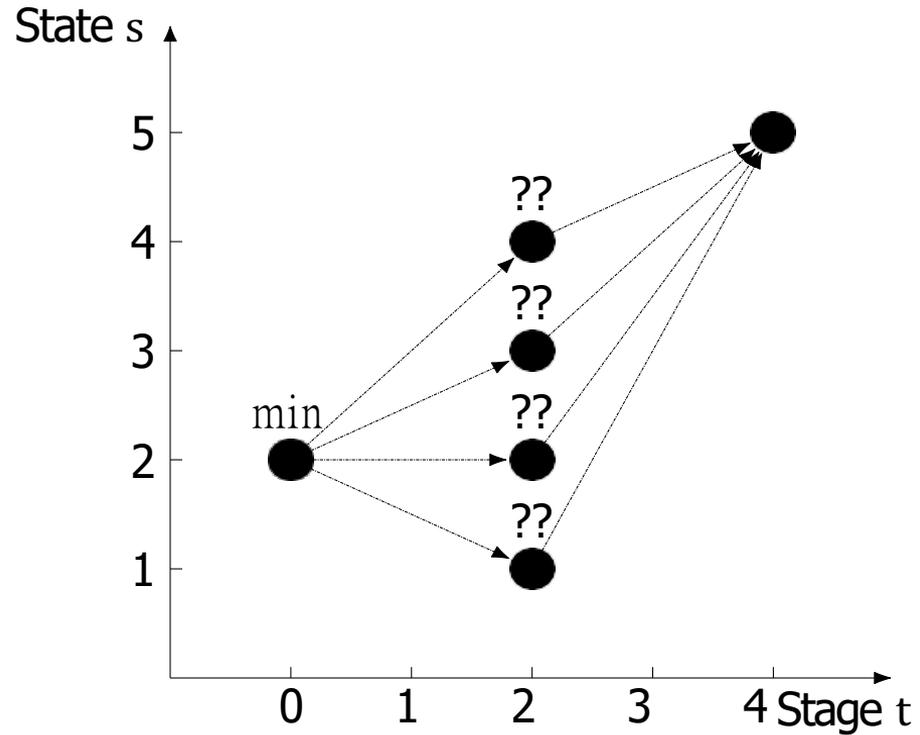
Destination is node 5.



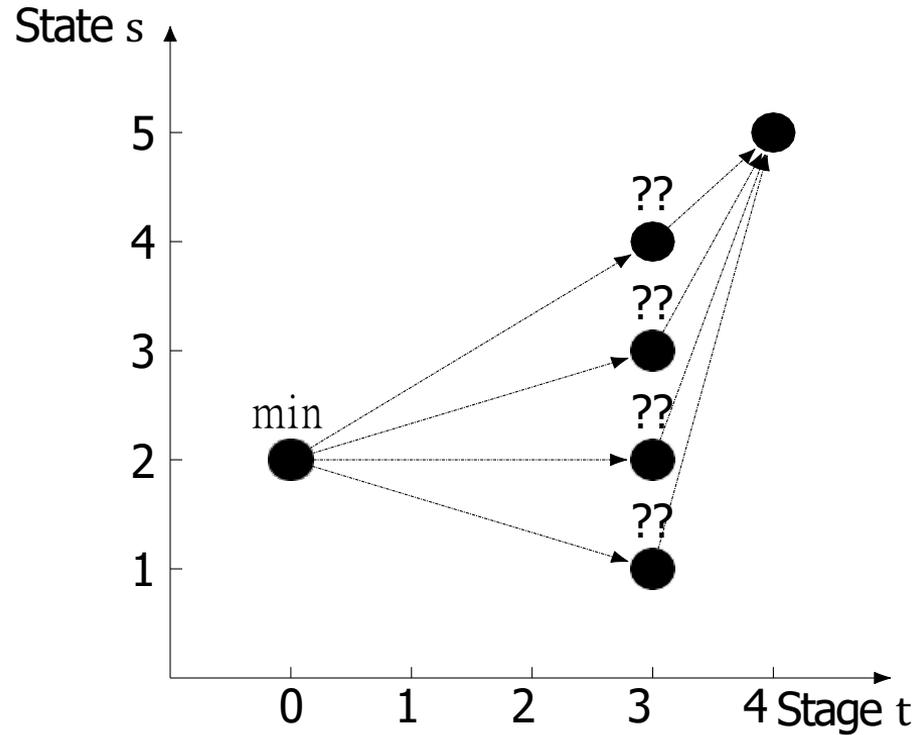
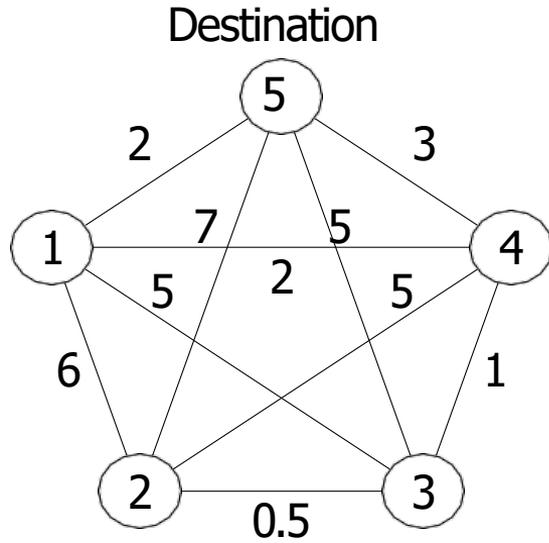
Solving Shortest Path



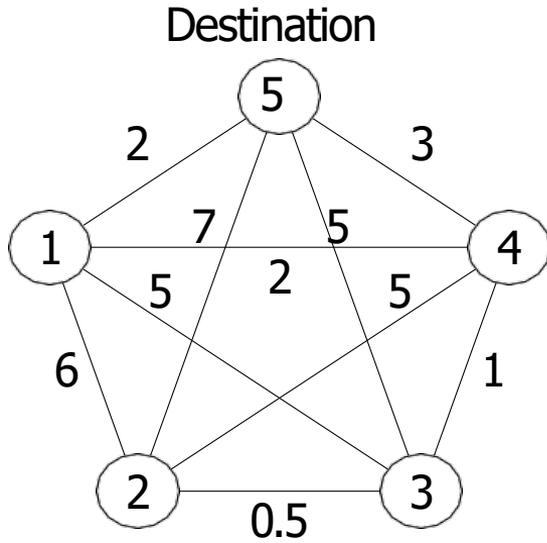
Destination is node 5.



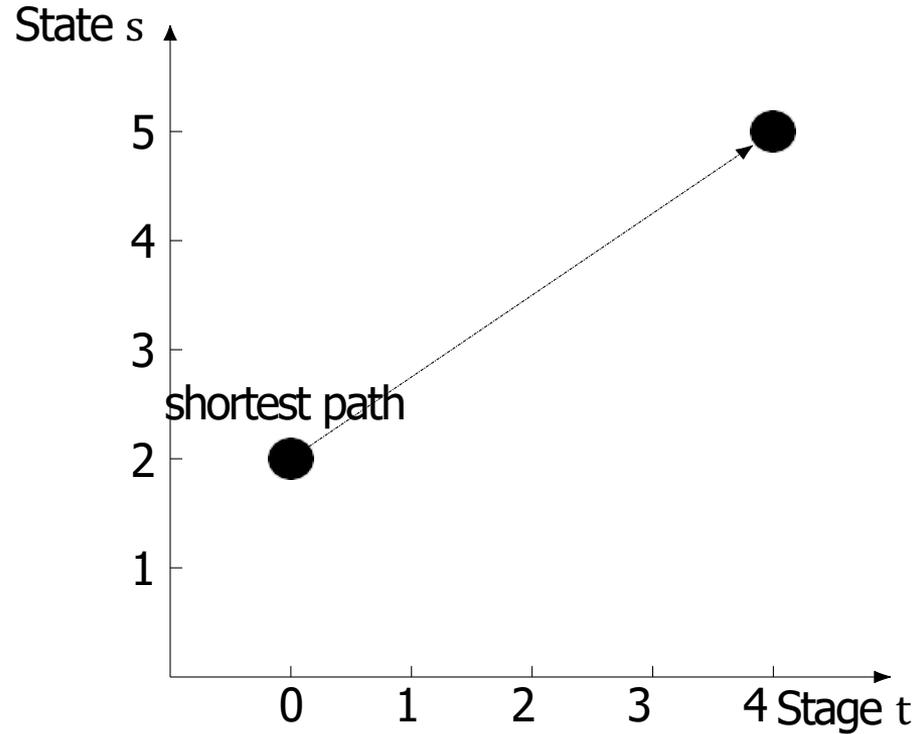
Solving Shortest Path



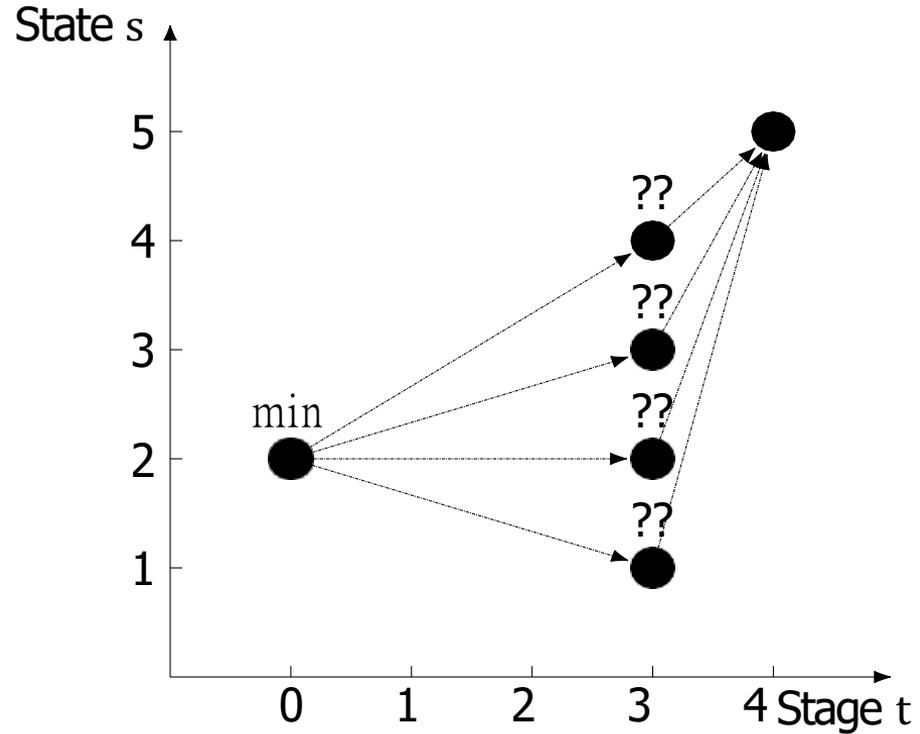
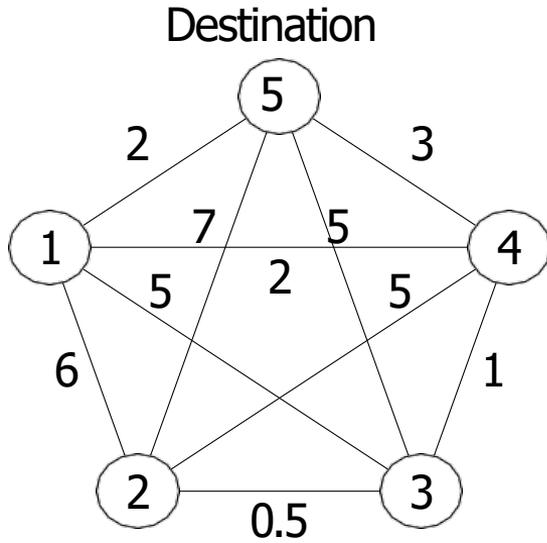
Solving Shortest Path



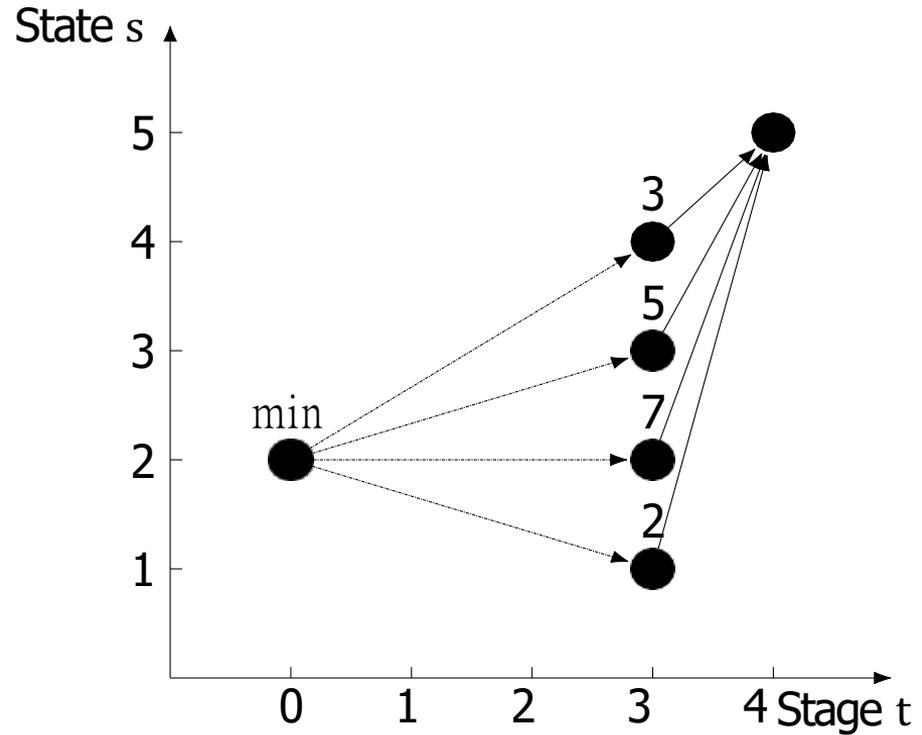
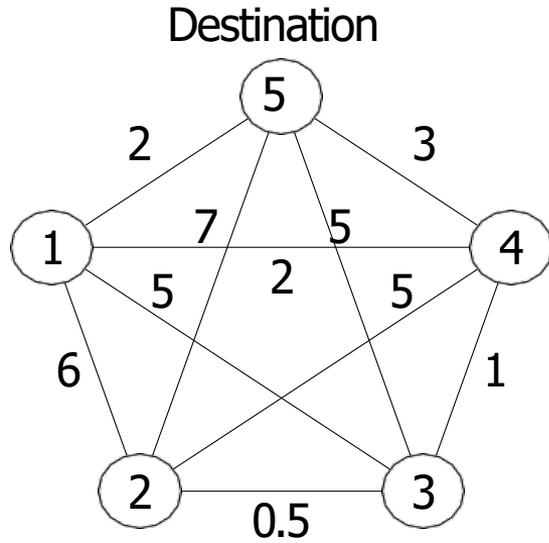
Destination is node 5.



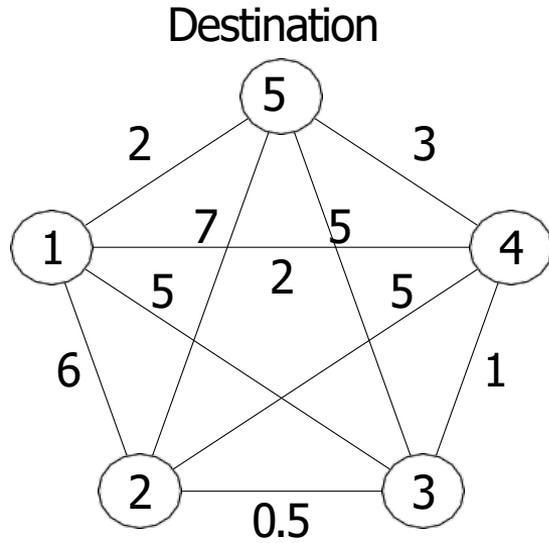
Solving Shortest Path



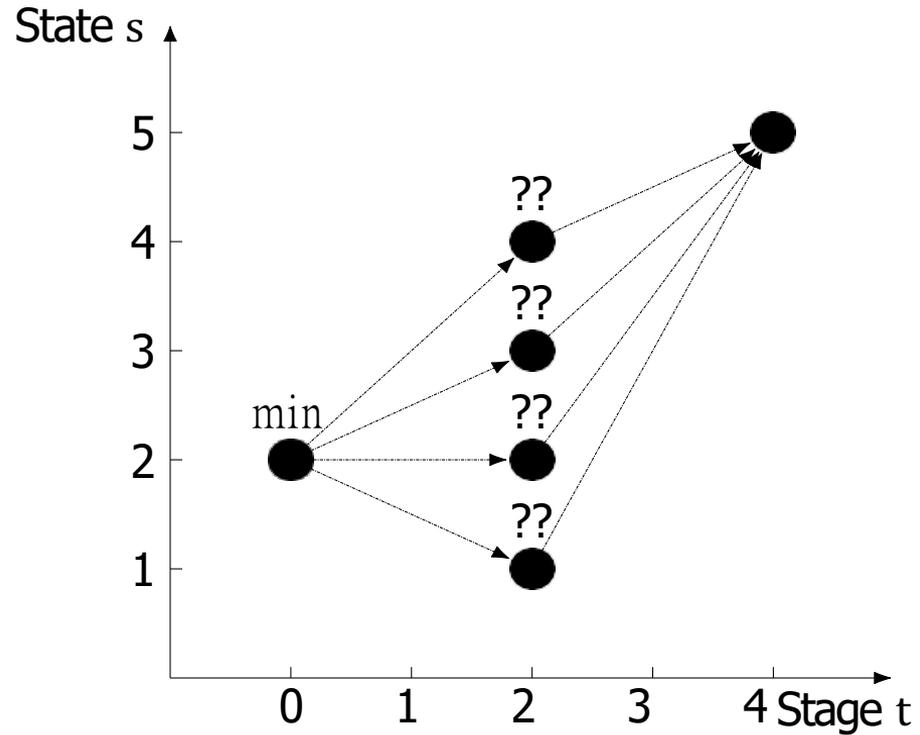
Solving Shortest Path



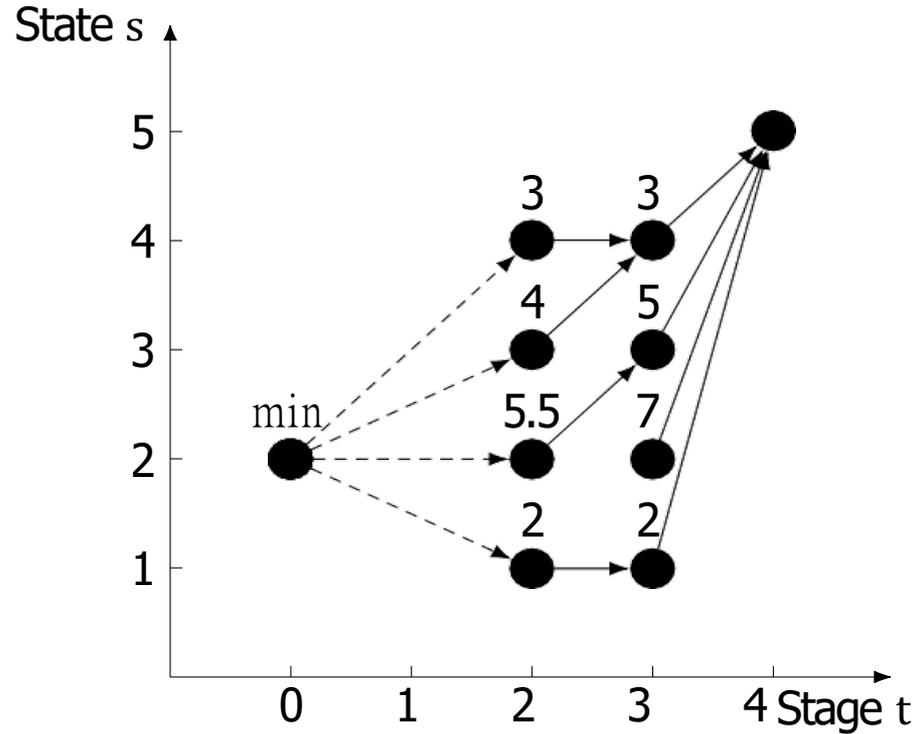
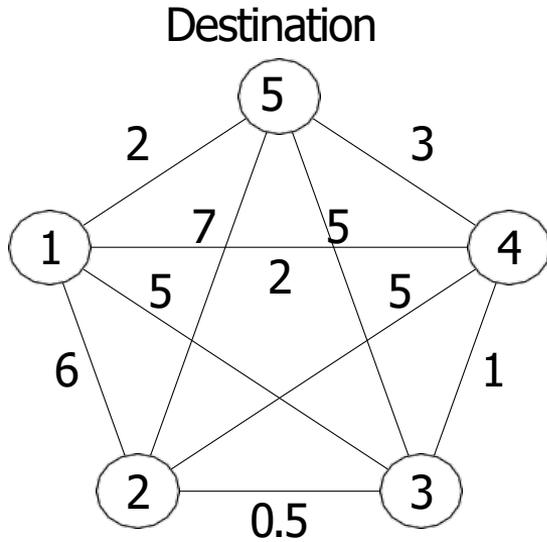
Solving Shortest Path



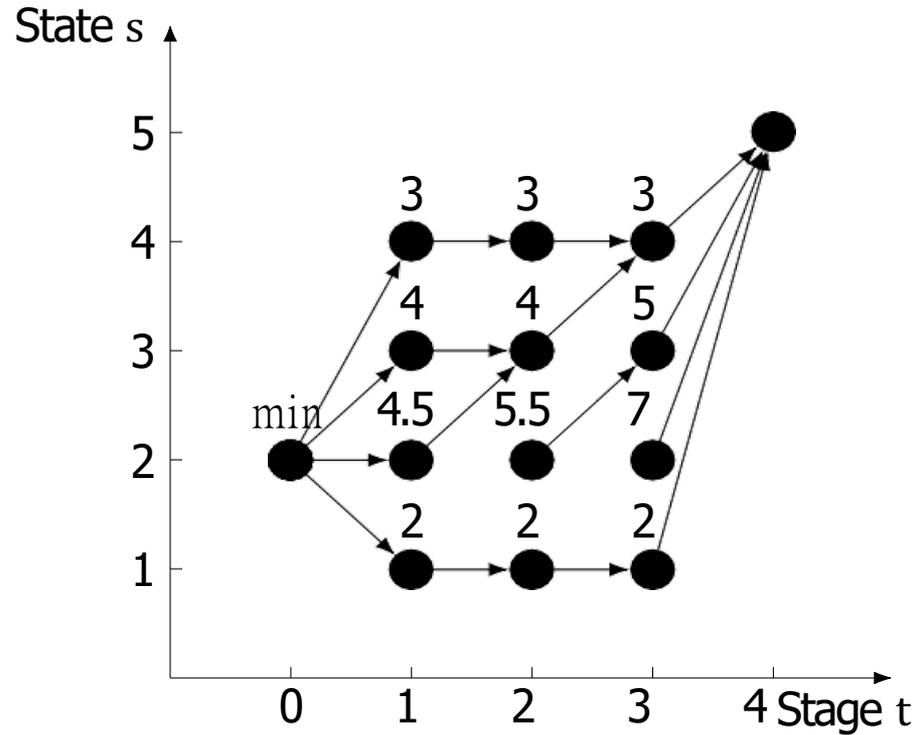
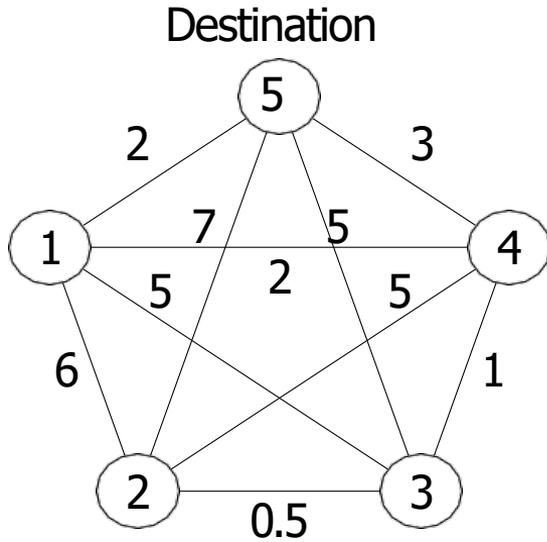
Destination is node 5.



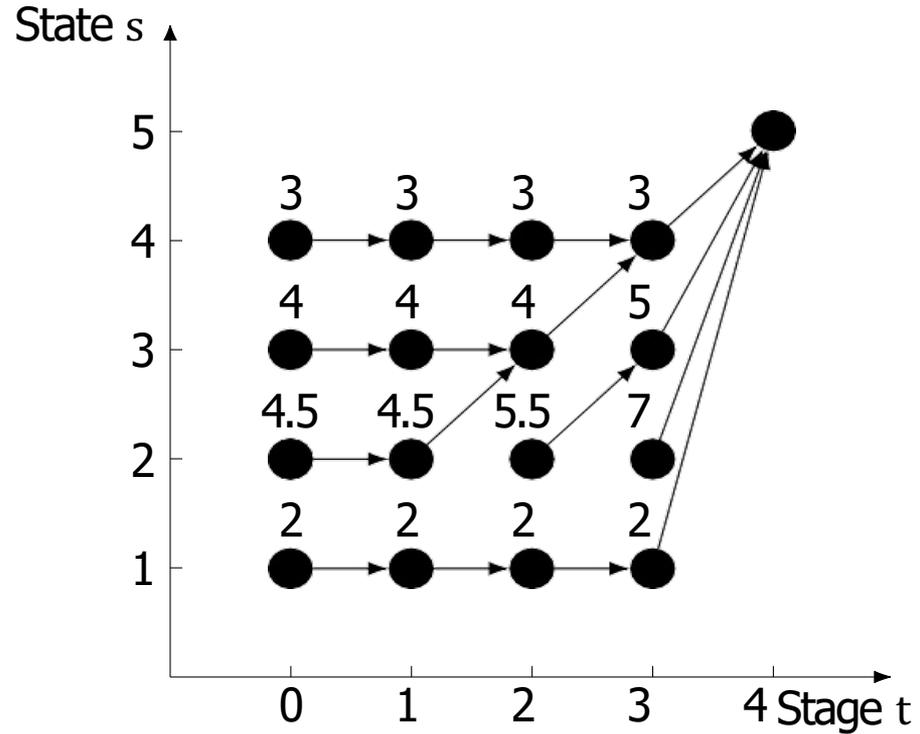
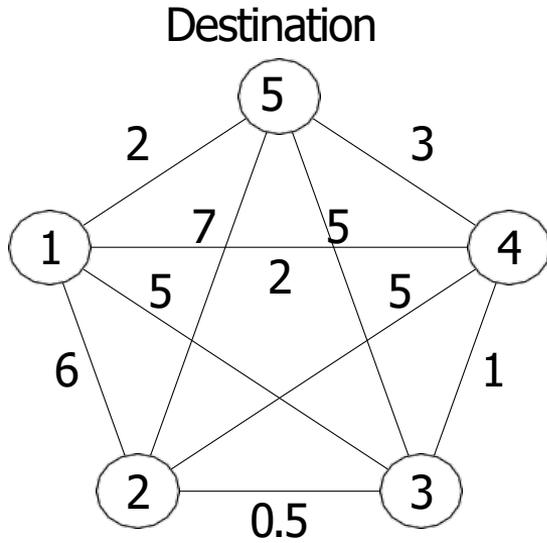
Solving Shortest Path



Solving Shortest Path



Solving Shortest Path

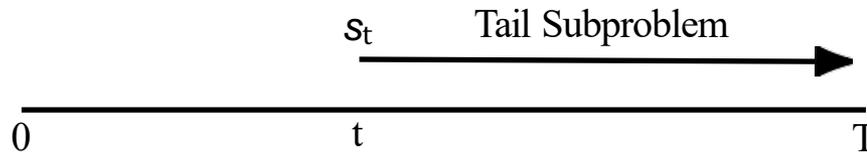


Outline

1. Sequential decision problems
2. Shortest path problems
3. **Principle of optimality**
4. Dynamic programming algorithm
5. Optimal capacity expansion problem

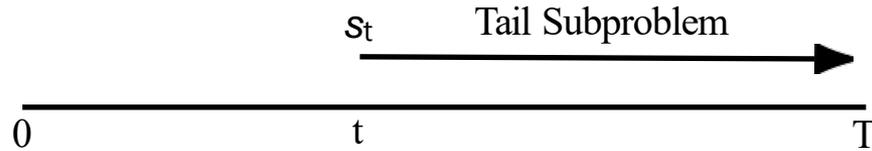
Bellman's Principle of optimality (1957)

*“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.**”*



Note: A policy is a mapping from states to actions

Principle of optimality (Bellman, 1957)



Principle (Optimality)

Let $\{a_0^*, \dots, a_{T-1}^*\}$ be an optimal action sequence, which together with s_0 and $\{\epsilon_0, \dots, \epsilon_{T-1}\}$ determines the corresponding state sequence $\{s_1^*, \dots, s_T^*\}$ 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 ,

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

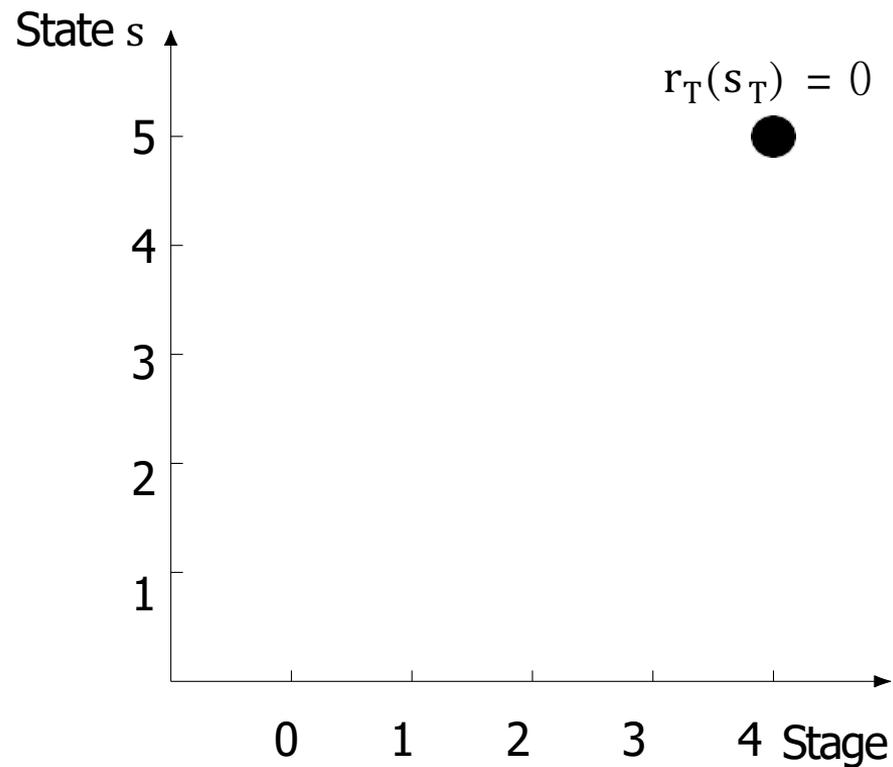
over $\{a_t, \dots, a_{T-1}\}$ with $a_\tau \in A_\tau(s_\tau)$, $\tau = t, \dots, T-1$. Then, the **truncated optimal action sequence** $\{a_t^*, \dots, a_{T-1}^*\}$ is optimal for this subproblem.

Outline

1. Sequential decision problems
2. Shortest path problems
3. Principle of optimality
4. **Dynamic programming algorithm**
5. Optimal capacity expansion problem

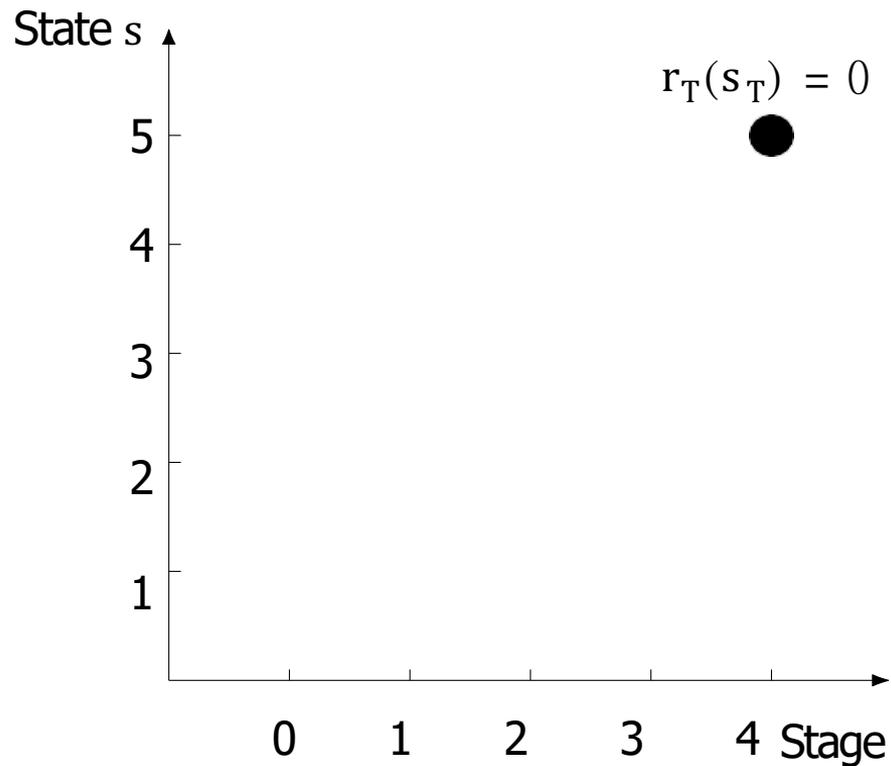
Dynamic programming algorithm

$$V_T(s_T) = r_T(s_T)$$



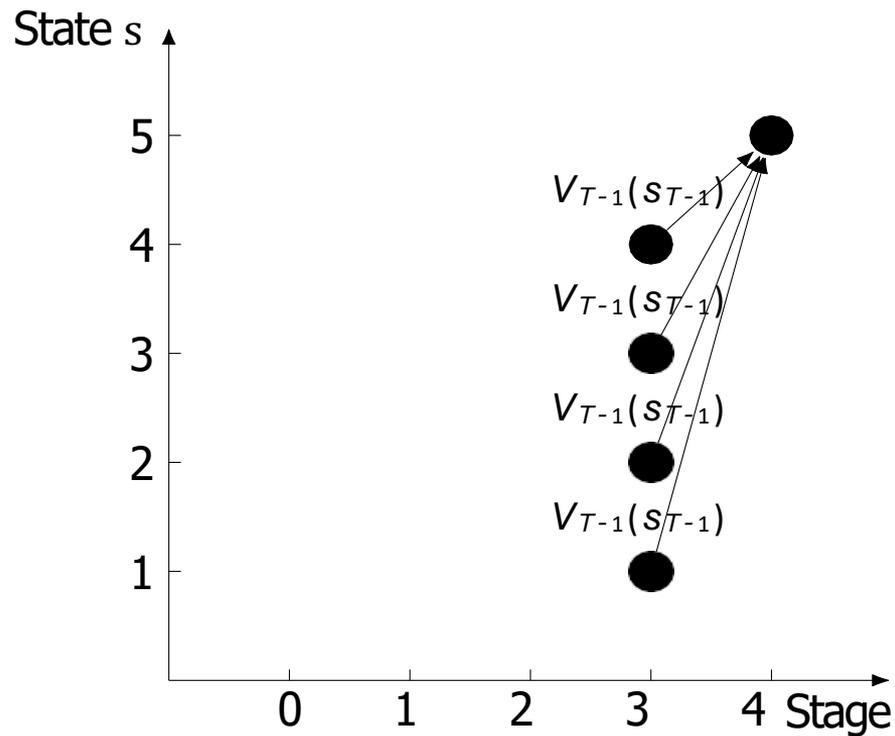
Dynamic programming algorithm

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



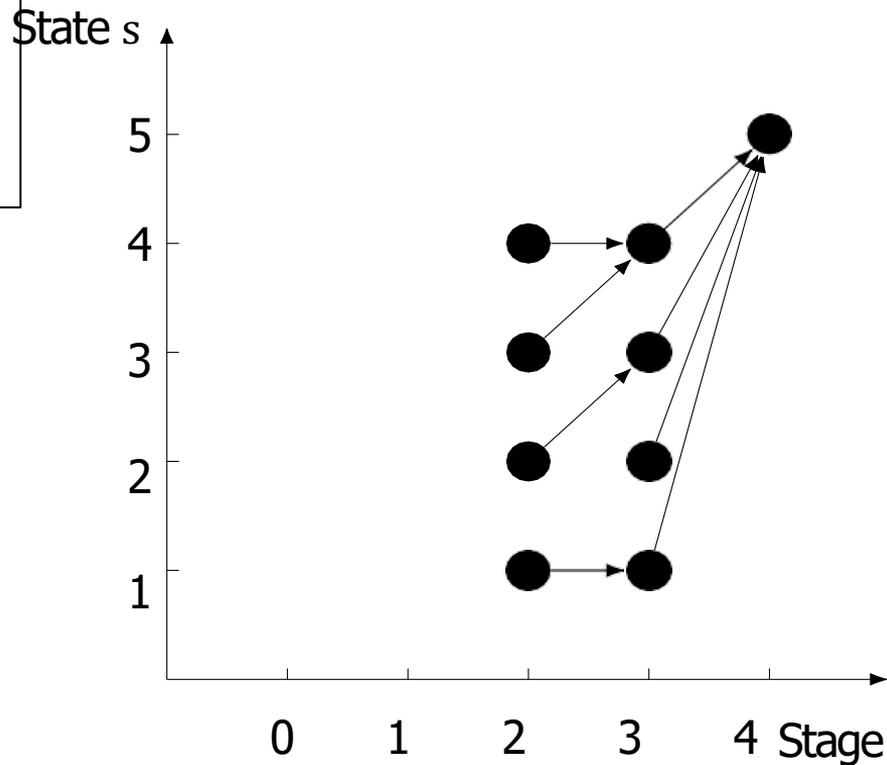
Dynamic programming algorithm

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



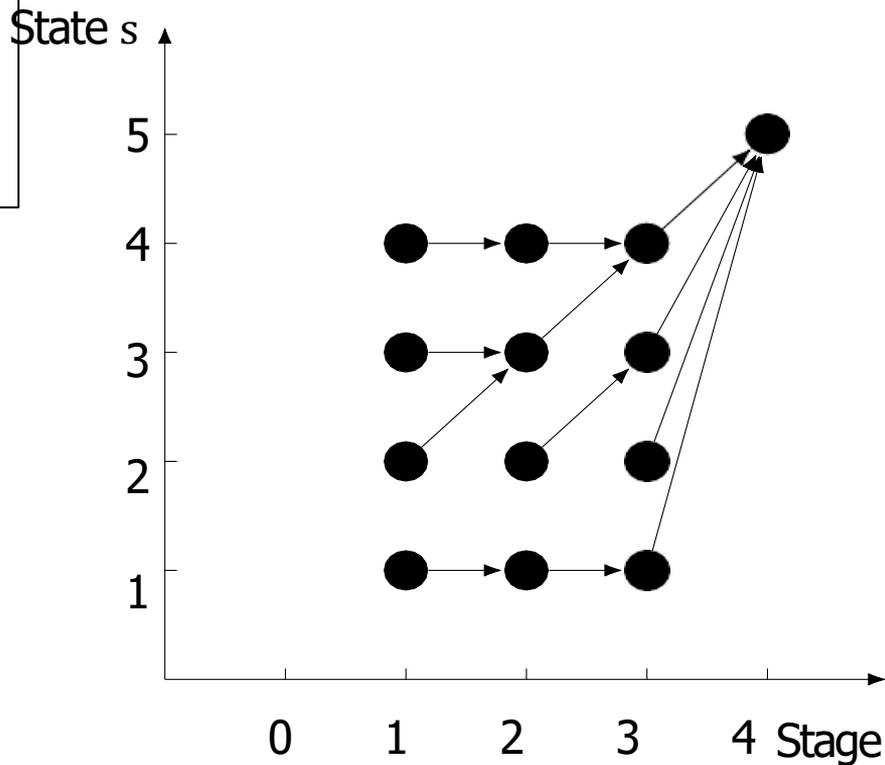
Dynamic programming algorithm

```
 $V_T(s_T) = r_T(s_T)$   
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
```



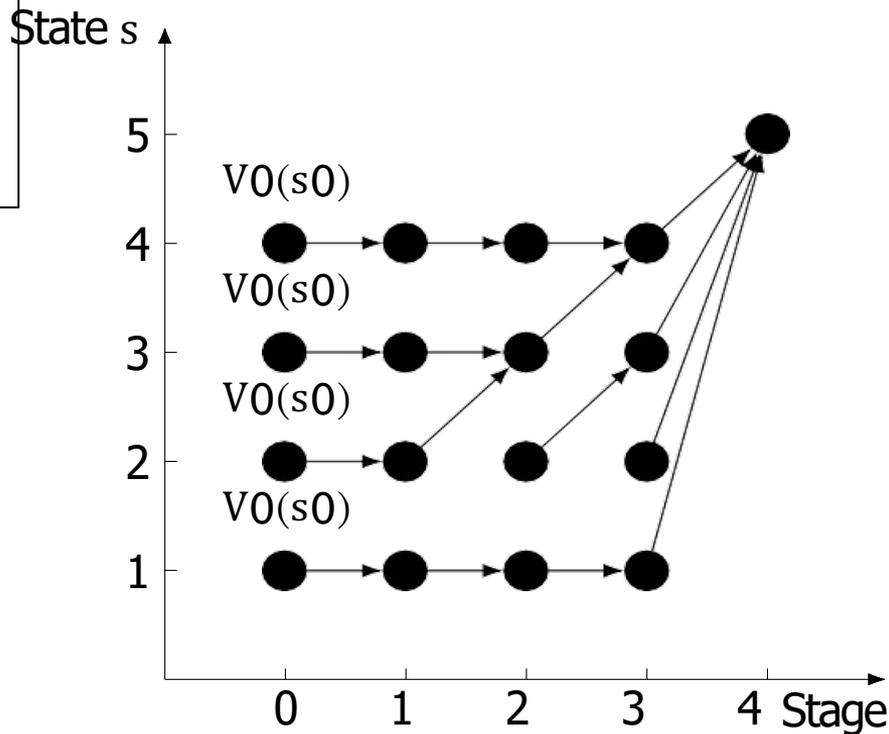
Dynamic programming algorithm

```
 $V_T(s_T) = r_T(s_T)$   
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  $t = T - 1, \dots, 0$  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
```

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 $\pi^* = (\pi_0^*, \dots, \pi_{T-1}^*)$ 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.

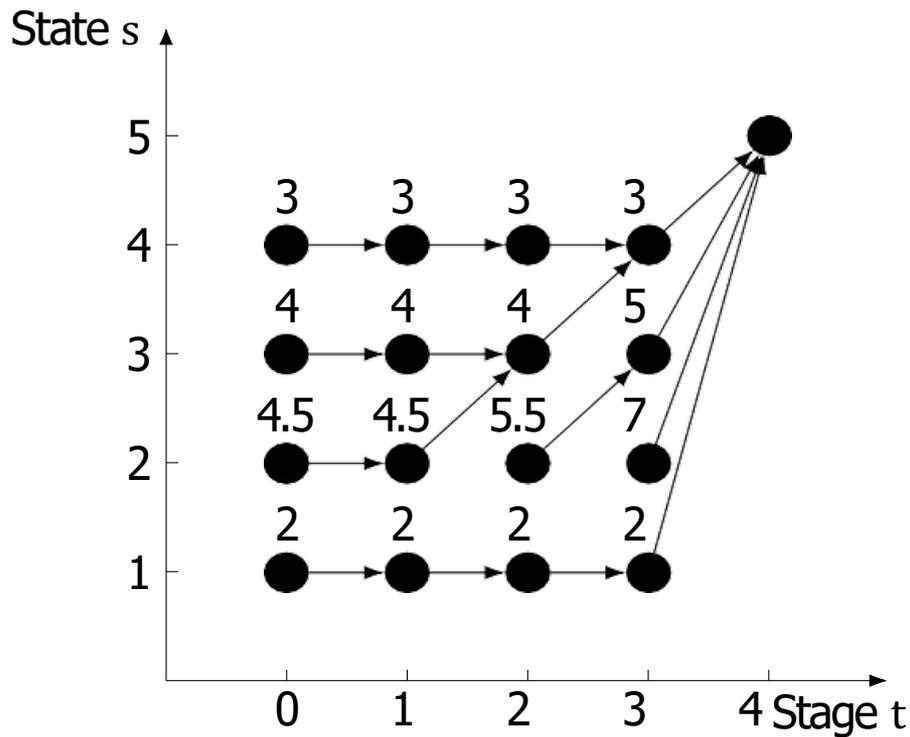
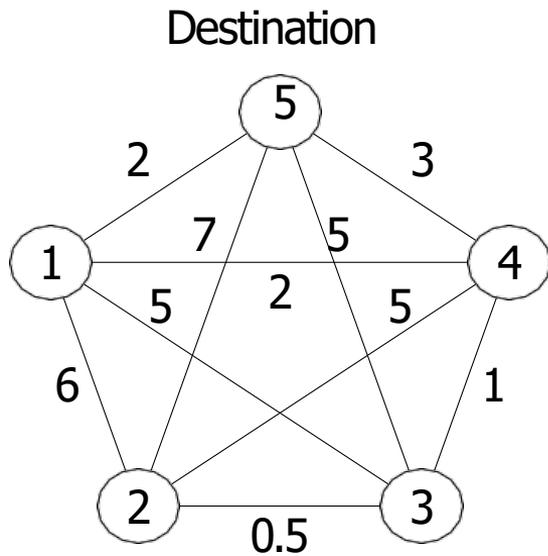
Denote **tail policy** from time t onward as $\pi_{t:T-1} = \{\pi_t, \pi_{t+1}, \dots, \pi_{T-1}\}$

Assume that $V_{t+1}(s_{t+1}) = V_{t+1}^*(s_{t+1})$. Then:

$$\begin{aligned}
 V_t^*(s_t) &= \max_{(\pi_t, \pi_{t+1:T-1})} r_t(s_t, \pi_t(s_t)) + r_T(s_T) + \sum_{i=t+1}^{T-1} r_i(s_i, \pi_i(s_i)) \\
 &= \max_{\pi_t} r_t(s_t, \pi_t(s_t)) + \max_{\pi_{t+1:T-1}} [r_T(s_T) + \sum_{i=t+1}^{T-1} r_i(s_i, \pi_i(s_i))] \\
 &= \max_{\pi_t} r_t(s_t, \pi_t(s_t)) + V_{t+1}^*(f_t(s_t, \pi_t(s_t))) \\
 &= \max_{\pi_t} r_t(s_t, \pi_t(s_t)) + V_{t+1}(f_t(s_t, \pi_t(s_t))) \\
 &= \max_{a_t \in \mathcal{A}_t(s_t)} r_t(s_t, a_t) + V_{t+1}(f_t(s_t, a_t)) \\
 &= V_t(s_t)
 \end{aligned}$$

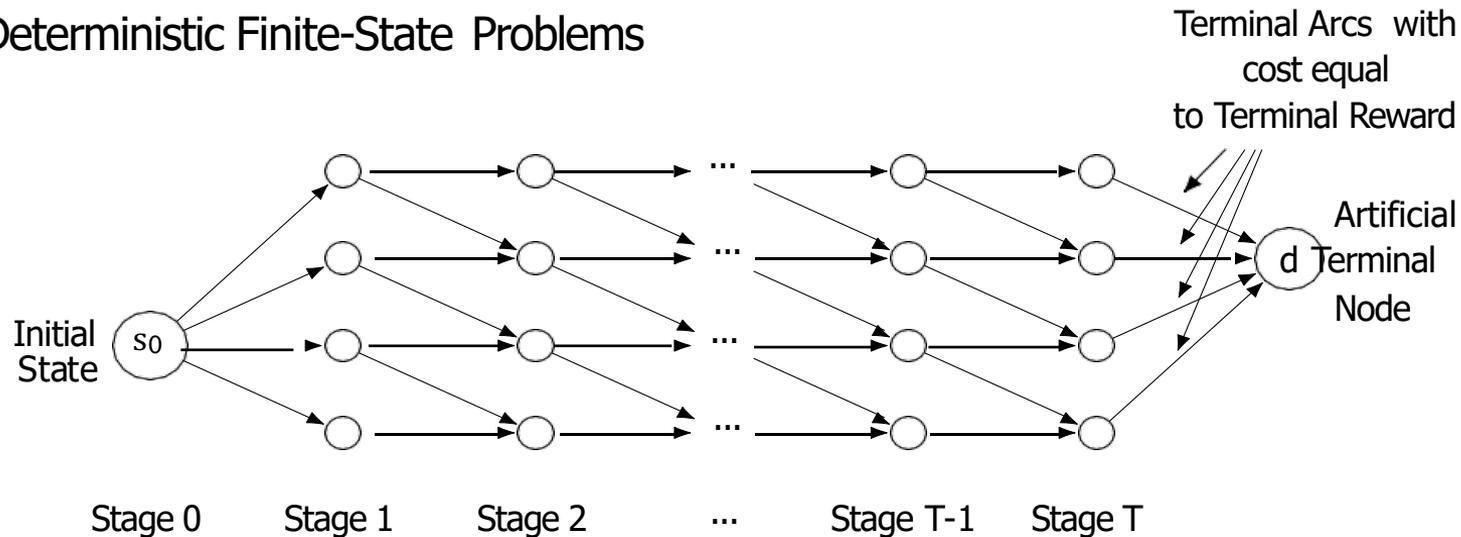
Interpretation as optimal reward-to-go (cost-to-go) function.

Solving Shortest Path

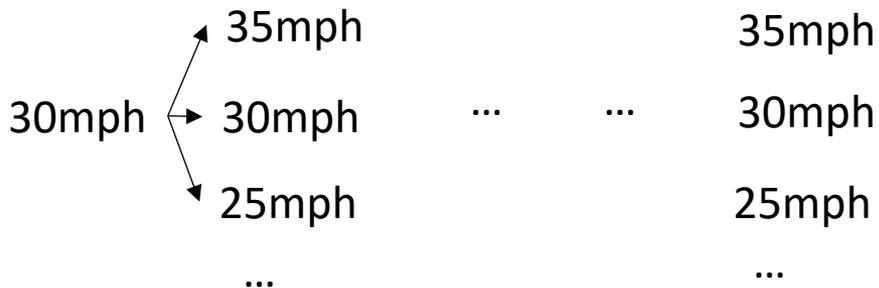


Sequential decision making as shortest path

For Deterministic Finite-State Problems



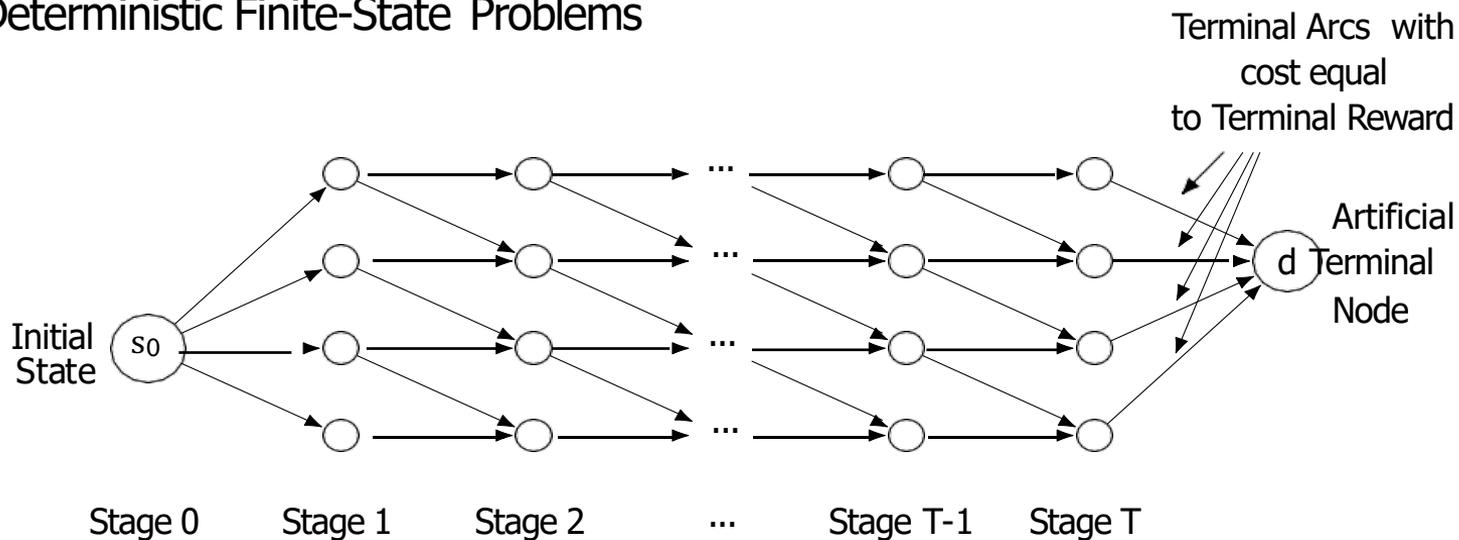
Example: Automated vehicle



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

Sequential decision making as shortest path

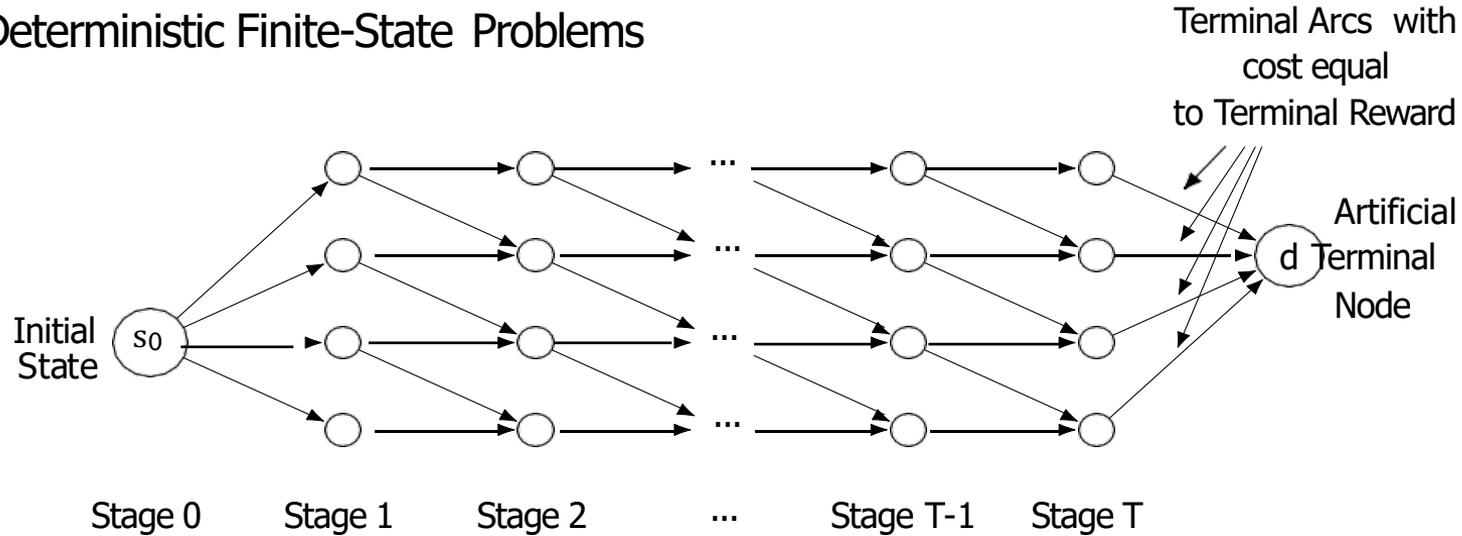
For Deterministic Finite-State Problems



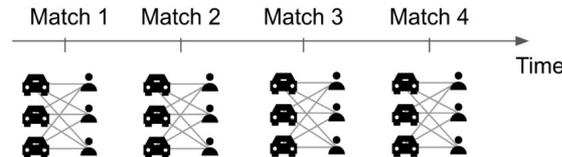
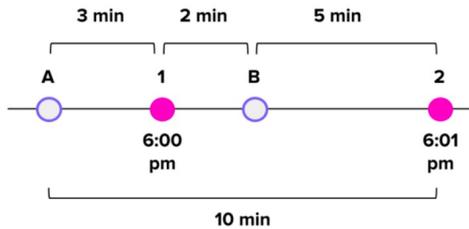
Discuss: If shortest path isn't hard, why are DP problems still challenging?

Sequential decision making as shortest path

For Deterministic Finite-State Problems



Example: Real-time ridesharing



Each stage may have

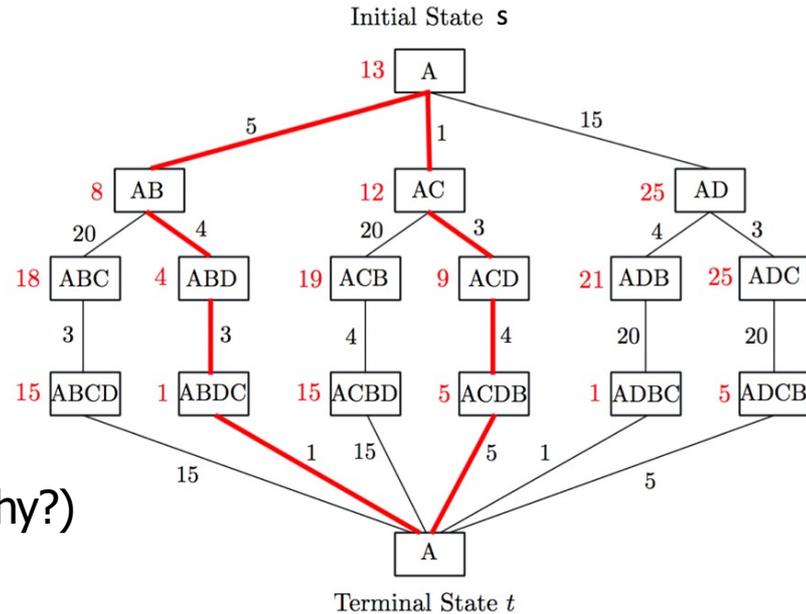
$$|A| = N!$$

for N drivers and N riders.

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**.



Matrix of Intercity
Travel Costs

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

Outline

1. Sequential decision problems
2. Shortest path problems
3. Principle of optimality
4. Dynamic programming algorithm
5. **Optimal capacity expansion problem**

Optimal capacity expansion

A regional automotive company is planning a large investment in electric vehicle (EV) manufacturing plants over the next few years. Find the minimum cost investment strategy that satisfies the minimum-demand schedule.

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

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

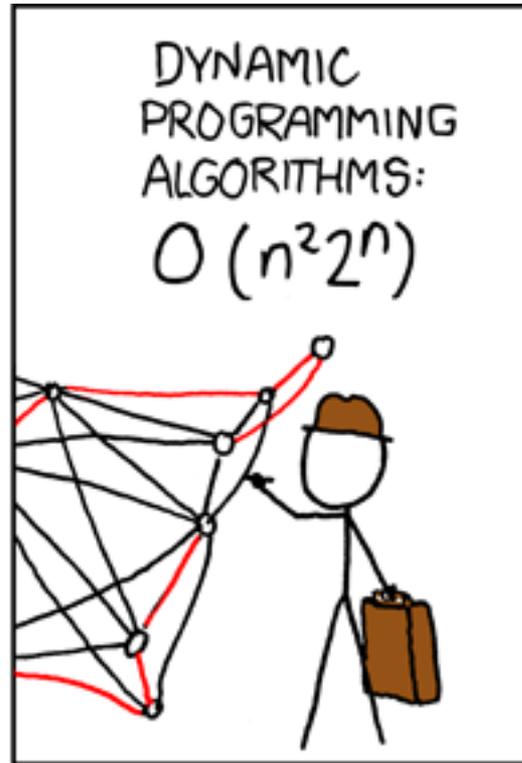
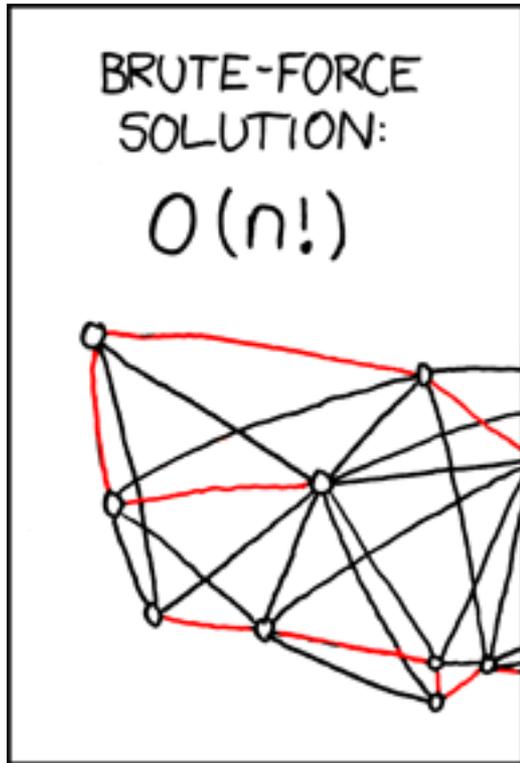
- **A total of eight manufacturing plants must be built over the next six years** because of both increasing demand in the region and the energy crisis, which has forced the closing of certain of their antiquated internal combustion engine (ICE) vehicle plants.
- *Minimum-demand schedule:* Assume that 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)

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



References

1. Bradley, Stephen P., Arnolando 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.