深蓝学院
shenlanxueyuan.com

MDP-Based Planning

主讲人 朱德龙

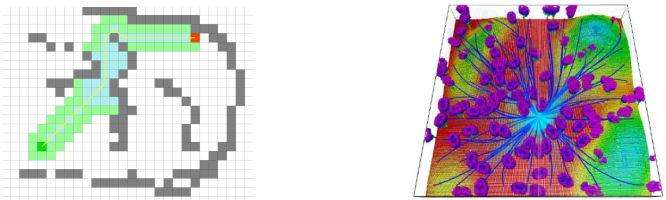
Ph.D in Robotics, Perception and AI Lab.
The Chinese University of Hong Kong

Outline

Uncertainties in Planning

Up until now, the planners are assumed no uncertainties with

- **perfect action execution**,
- **full knowledge of states**.

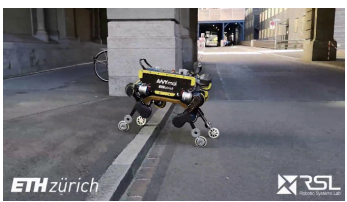


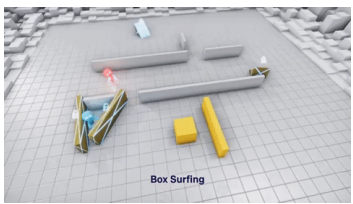
Deterministic graph search (left) and planning with perfect map and poses (right).

Uncertainties in Planning

In real applications, both execution and state estimation are not perfect.

- **Execution Uncertainties**: slippage, rough terrain, wind, air-drag, control errors, etc.
- **State Estimation Uncertainties**: senor noise, calibration error, imperfect estimation, partial observability, etc.





Big dog with wheeled legs (left) and seek and hide game (right).

Uncertainties can be categorized into two classes from the perspective of robot, which **indicate how much information the robot can use**.

Uncertainty Models

- **Nondeterministic**: the robot have no idea what type of uncertainties or interference will be added to its behaviors.
- **Probabilistic**: the robot has an estimation about uncertainties by observing and gathering statistics.

To formally describe this concept, we first introduce two decision makers to model the generation of uncertainties, then the types of planning with uncertainties.

Decision Makers (Game Player)

- **Robot** is the primary decision maker that performs planning based on fully known states and perfect execution.
- **Nature** adds uncertainties to the execution of plans made by the robot, which is unpredictable to the robot.

相互不了解对方

Formalization-7.1: A Game Against Nature (Independent Game)

- A nonempty set U called the *robot action space*. Each $u \in U$ is referred to as a *robot action*.
- A nonempty set Θ called the *nature action space*. Each $\theta \in \Theta$ is referred to as a *nature action*.
- A function $L : U \times \Theta \to \mathbb{R} \cup \{\infty\}$, called the *cost function*, or the negative reward function.

Formalization-7.2: Nature Knows the Robot Action (Dependent Game)

- A nonempty set U called the *robot action space*. Each $u \in U$ is referred to as a *robot action*.
- For each $u \in U$, a nonempty set $\Theta(u)$ called the *nature action space*.
- A function $L : U \times \Theta \to \mathbb{R} \cup \{\infty\}$, called the *cost function*, or the negative reward function.

What is the best decision for robot, given that it is engaged in a game with nature?

One-step Worst-Case Analysis

- Under *Nondeterministic* model, $P(\theta)$ in *Independent Game* and $P(\theta|u_k)$ in *Dependent Game* are unknown;
- Robot cannot predict the behavior of nature and images it spitefully chooses actions that drive the cost as high as possible;
- Hence, it is reasonable to **make decisions by assuming the worst**.

$$u^* = \operatorname*{argmin}_{u \in U} \left\{ \max_{\theta \in \Theta} \{L(u, \theta)\} \right\} \tag{1}$$

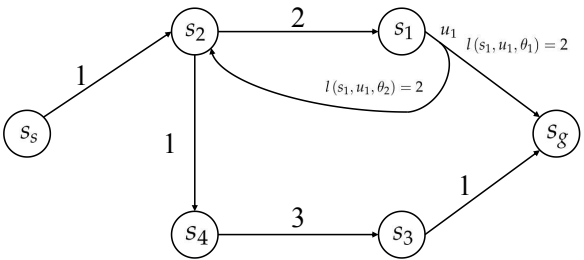
One-step Expected-Case Analysis

- Under *Probabilistic* model, $P(\theta)$ in *Independent Game* and $P(\theta|u_k)$ in *Dependent Game* are known;
- It is assumed that the applied nature actions have been observed and nature applies a randomized strategy in action selection.
- Hence, we optimize the **average cost to be received.**

$$u^* = \operatorname*{argmin}_{u \in U} \{E_\theta[L(u, \theta)]\} \tag{2}$$

Formalization-7.2: Discrete Planning with Nature

1. A nonempty **state space** X with an **initial state** x_s and a **goal set** $X_F \subset X$.
2. For each state $x \in X$, a finite and nonempty **robot action space** $U(x)$. For each $x \in X$ and $u \in U(x)$, a finite and nonempty **nature action space** $\Theta(x, u)$.



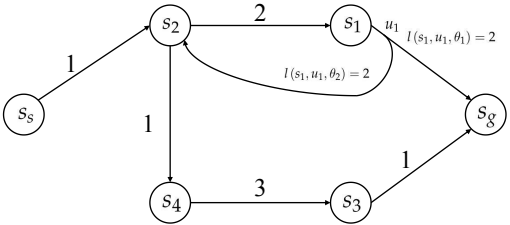
1. $X = \{s_s, s_1, s_2, s_3, s_4, s_g\}$
2. $X_F = \{s_g\}$
3. $U = \{u_s, u_1, u_{21}, u_{24}, u_3, u_4\}$ 无动作
4. $\Theta = \{\theta_0, \theta_1, \theta_2\}$

↳ nature的影响

Formalization-7.2: Multi-Step Discrete Planning with Nature

3 A **state transition function** $f(x, u, \theta)$ for every $x \in X, u \in U$, and $\theta \in \Theta(x, u)$.

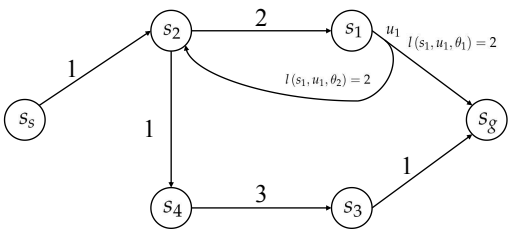
$$X_{k+1}(x_k, u_k) = \{x_{k+1} \in X \mid \exists\, \theta_k \in \Theta(x_k, u_k) \text{ s.t. } x_{k+1} = f(x_k, u_k, \theta_k)\}$$
$$P(x_{k+1}|x_k, u_k) = \sum\nolimits_{\theta_k} P(x_{k+1}, \theta_k|x_k, u_k) \text{ s.t. } \{\theta_k|x_{k+1} = f(x_k, u_k, \theta_k)\} \quad (3)$$



1. $X(s_s, u_s) = \{f(s_s, u_s, \theta_0)\}$
2. $X(s_1, u_1) = \{f(s_1, u_1, \theta_1), f(s_1, u_1, \theta_2)\}$
3. $\underbrace{s_s \to u_s \to s_2 \to u_{21} \to s_1 \to u_1}_{k} \underbrace{\to s_g}_{F=k+1}$

Formalization-7.2: Multi-Step Discrete Planning with Nature

4 A set of **stages**, each denoted by k, that begins at $k = 1$ and continues indefinitely or ends at a maximum stage $k = K + 1 = F$.



1. $X(s_s, u_s) = \{f(s_s, u_s, \theta_0)\}$
2. $X(s_1, u_1) = \{f(s_1, u_1, \theta_1), f(s_1, u_1, \theta_2)\}$
3. $\underbrace{s_s \to u_s \to s_2 \to u_{21} \to s_1 \to u_1}_{k} \underbrace{\to s_g}_{F=k+1}$

Formalization-7.2: Multi-Step Discrete Planning with Nature

5 A stage-additive **cost functional** L. Let $\tilde{x}_F, \tilde{u}_k, \tilde{\theta}_K$ denote the history of states, robot actions, and nature actions up to stage K:

$$\tilde{x}_F = (x_1, x_2, \ldots, x_F), \tilde{u}_k = (u_1, u_2, \ldots, u_k), \tilde{\theta}_K = (\theta_1, \theta_2, \ldots, \theta_K) \tag{4}$$

The cost functional is a metrics that evaluate all possible plans (or paths):

$$\begin{aligned} L\left(\tilde{x}_F, \tilde{u}_K, \tilde{\theta}_K\right) &= \sum_{k=1}^{K} l\left(x_k, u_k, \theta_k\right) + l_F\left(x_F\right), \\ l_F\left(x_F\right) &= \begin{cases} 0, & \text{if } x_F \in X_G, \\ \infty, & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

Actually, (1-3) in Formalization-7.2 and the one-step cost l define a standard

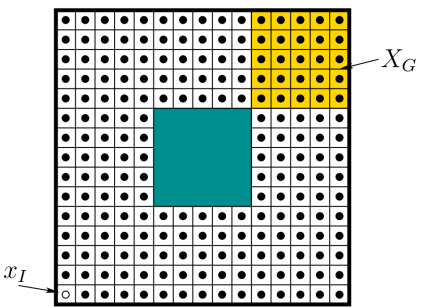
Markov Decision Process (MDP)

A MDP is a 4-tuple (S, A, P, R) in learning field, or (X, U, P, L) in planning field:

- S or X is *sate space*,
- A or U is *(robot) action space*,
- $P(x_{k+1}|x_k, u_k)$ is *state transition function* under probabilistic model,
  - which degenerates to a set $X_{k+1}(x_k, u_k)$ under nondeterministic model,
- $R(x_k, x_{k+1})$ is *the immediate reward*, or the negative one-step cost $-l(x_k, u_k, \theta_k)$, after transitioning from x_k to x_{k+1} due to u, θ.

The first difficulty of planning with uncertainties lies on *properly formalizing our problem with a MDP model*.

A grid-based shortest path problem with interference from nature.



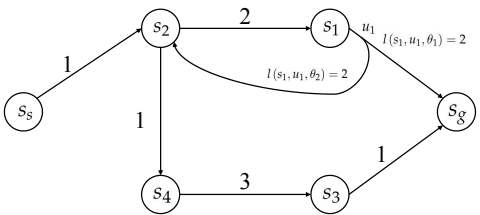
1. $X = \{(i,j) \mid 1 \le i,j \le 15\}$ data space
2. $U = \{u_{stay}, u_{right}, u_{up}, u_{left}, u_{down}\}$. robot
3. $\Theta = \{\theta_1, \theta_2\}$ nature
   $f(x_{k+1}, x_k, u_k) = \mathcal{N}(x_k + u_k, \sigma(\theta_1, \theta_2))$
4. $\Theta = \{\theta_0 = [0,0]^t, \theta_1 = [0,1]^T, \theta_2 = [0,-1]^T\}$
   $x_{k+1} = x_k + u_k + \theta_k,\ k \sim \{0,1,2\}$
5. $l(x_k, u_k, \theta_k) = \|x_{k+1} - x_k\|$

Define a (feedback) plan:

- $\pi : X \to U$.

Define a set of trajectories:

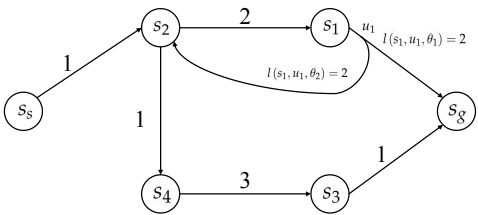
- $\mathcal{H}(\pi, x_s)$: induced by π and started from x_s.
- $(\tilde{x}, \tilde{u}, \tilde{\theta}) \in \mathcal{H}(\pi, x_s)$: a trajectory or an execution.



1. $\mathcal{H}(\pi_1, s_s)$:
   $s_s \to u_s \to s_2 \to u_{21} \to s_1 \to u_1 \to s_g$
   $s_s \to u_s \to s_2 \to u_{21} \to s_1 \to u_1 \to s_2 \to \cdots$
2. $\mathcal{H}(\pi_2, s_s)$:
   $s_s \to u_s \to s_2 \to u_{24} \to s_4 \to u_4 \to s_3 \to u_3 \to s_g$

Define the cost for a specific plan π (rather than a trajectory):

- $G_\pi(x_s)$: **cost-to-goal**.



1. *worst-case analysis* for nondeterministic model:

$$G_\pi(x_s) = \max_{(\tilde{x}, \tilde{u}, \tilde{\theta}) \in \mathcal{H}(\pi, x_s)} \{L(\tilde{x}, \tilde{u}, \tilde{\theta})\} \quad (6)$$

2. *expected-case analysis* for probabilistic model:

$$G_\pi(x_s) = E_{\mathcal{H}(\pi, x_s)}[L(\tilde{x}, \tilde{u}, \tilde{\theta})] \quad (7)$$

1. **Nondeterministic model**
2. **Worst-case analysis**
3. Find an optimal plan π^* such that

$$G_{\pi^*}(x_s) = \min_{\pi}\{G_\pi(x_s)\} = \min_{\pi}\{\max_{\mathcal{H}(\pi,x_s)}\{L(\tilde{x},\tilde{u},\tilde{\theta})\}\}. \tag{8}$$

4. Directly solving the problem is difficult.
5. Construct optimal solution by **Dynamic Programming**.
   - The recurrence between stage k and $k+1$ of optimal plan.

1. The optimal cost-to-goal for final state F can be directly acquired $G_F^*(x_F) = l_F(x_F)$,
2. The costs of all optimal one-step plans from stage K to stage $F = K+1$ is

$$G_K^*(x_K) = \min_{u_K} \max_{\theta_K} \{ l(x_K, u_K, \theta_K) + G_F^*(f(x_K, u_K, \theta_K)) \} \tag{9}$$

3. More generally, G_k^* can be computed once G_{k+1}^* is given

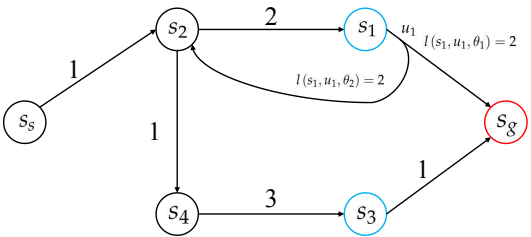
$$G_k^*(x_k) = \min_{u_k} \max_{\theta_k} [\min_{u_{k+1}\theta_{k+1}} \cdots \min_{u_K} \max_{\theta_K} (l(x_k, u_k, \theta_k) + \sum_{i=k+1}^{K} l(x_i, u_i, \theta_i) + l_F(x_F))] \tag{10}$$

$$G_k^*(x_k) = \min_{u_k} \max_{\theta_k} [l(x_k, u_k, \theta_k) + \underbrace{\min_{u_{k+1}\theta_{k+1}} \cdots \min_{u_K} \max_{\theta_K} (\sum_{i=k+1}^{K} l(x_i, u_i, \theta_i) + l_F(x_F))}_{G_{k+1}^*(x_{k+1})}] \tag{11}$$

Hence, the dynamic programming recurrence for solving minimax cost plan is

$$G_k^*(x_k) = \min_{u_k \in U(x_k)} \left\{ \max_{\theta_k \in \Theta(x_k, u_k)} \left\{ l(x_k, u_k, \theta_k) + G_{k+1}^*(x_{k+1}) \right\} \right\} \tag{12}$$

局部逐段编



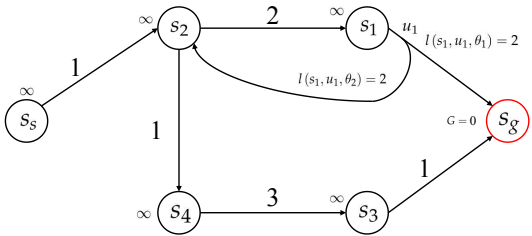
1. suppose $G_{k+1}^*(x_{k+1} = s_g) = 0$
2. $G_k^*(x_k = s_3) = \min\{1 + 0\}$
3. $G_k^*(x_k = s_1) = \min\{\max\{2 + 0, 2 + \text{inf}\}\}$

θ_1 θ_2

$G(x_F) \leftarrow 0$; all other G values are infinite; OPEN $= \{x_F\}$; CLOSED $= \emptyset$;
while x_s *is not expanded* **do**
  $x_{k+1} \leftarrow$ remove x with the smallest G value from OPEN;
  insert x_{k+1} to CLOSED;
  for *every* $x_k \notin$ CLOSED s.t. $x_{k+1} \in X_{k+1}(x_k, u_k)$ **do**
    if $G(x_k) > \max_{\theta_k \in \Theta(x_k, u_k)} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$ **then**
      $G(x_k) = \max_{\theta_k \in \Theta(x_k, u_k)} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$;
      insert x_k into OPEN;
    end
  end
end

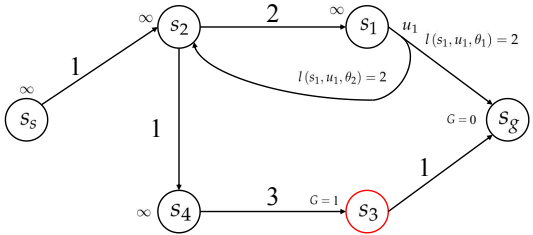
Algorithm 1: Nondeterministic Dijkstra

$G(x_F) \leftarrow 0$; all other G values are infinite; OPEN $= \{x_F\}$;
CLOSED $= \emptyset$;
while x_s *is not expanded* **do**
  $x_{k+1} \leftarrow$ remove x with the smallest G value from OPEN;
  insert x_{k+1} to CLOSED;
  for *every* $x_k \notin$ CLOSED s.t. $x_{k+1} \in X_{k+1}(x_k, u_k)$ **do**
    if $G(x_k) > \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$ **then**
      $G(x_k) \leftarrow \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$;
      insert x_k into OPEN;
    end
  end
end



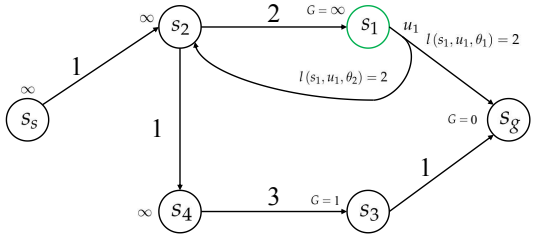
1. CLOSED = $\{\}$
2. OPEN = $\{s_g\}$
3. next state to expand: s_g
4. $G(x_{k+1} = s_g) = 0$

$G(x_F) \leftarrow 0$; all other G values are infinite; OPEN $= \{x_F\}$;
  CLOSED $= \emptyset$;
while x_s *is not expanded* **do**
  $x_{k+1} \leftarrow$ remove x with the smallest G value from OPEN;
  insert x_{k+1} to CLOSED;
  for *every* $x_k \notin$ CLOSED s.t. $x_{k+1} \in X_{k+1}(x_k, u_k)$ **do**
    if $G(x_k) > \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$ **then**
      $G(x_k) \leftarrow \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$;
      insert x_k into OPEN;
    end
  end
end



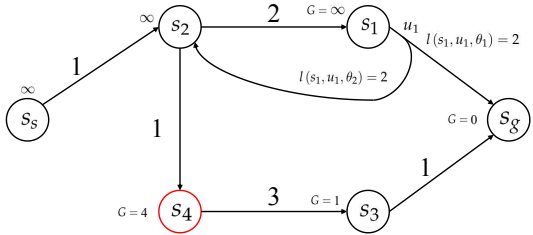
1. CLOSED $= \{s_g\}$
2. $G(x_k = s_3) = 1 + G(x_{k+1} = s_g)$
3. OPEN $= \{s_3\}$
4. next state to expand: -

$G(x_F) \leftarrow 0$; all other G values are infinite; OPEN $= \{x_F\}$;
CLOSED $= \emptyset$;
while x_s *is not expanded* **do**
  $x_{k+1} \leftarrow$ remove x with the smallest G value from OPEN;
  insert x_{k+1} to CLOSED;
  for *every* $x_k \notin$ CLOSED s.t. $x_{k+1} \in X_{k+1}(x_k, u_k)$ **do**
    if $G(x_k) > \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$ **then**
      $G(x_k) \leftarrow \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$;
      insert x_k into OPEN;
    end
  end
end



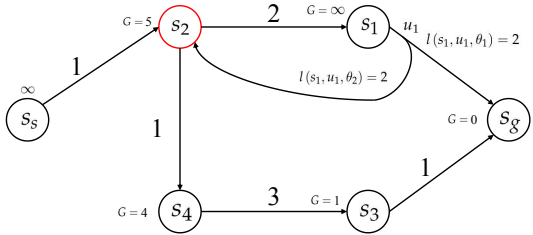
1. CLOSED $= \{s_g\}$
2. $G(x_k = s_1)$ cannot be reduced
3. OPEN $= \{s_3\}$
4. next state to expand: s_3

$G(x_F) \leftarrow 0$; all other G values are infinite; OPEN $= \{x_F\}$;
CLOSED $= \emptyset$;
while x_s *is not expanded* **do**
  $x_{k+1} \leftarrow$ remove x with the smallest G value from OPEN;
  insert x_{k+1} to CLOSED;
  for *every* $x_k \notin$ CLOSED s.t. $x_{k+1} \in X_{k+1}(x_k, u_k)$ **do**
    if $G(x_k) > \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$ **then**
      $G(x_k) \leftarrow \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$;
      insert x_k into OPEN;
    end
  end
end



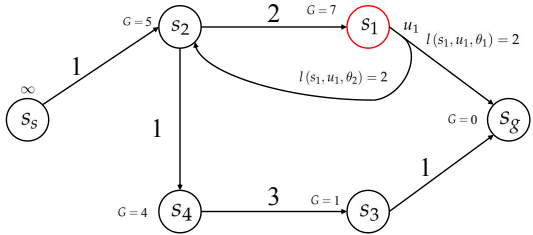
1. CLOSED $= \{s_g, s_3\}$
2. $G(x_k = s_4) = 3 + G(x_{k+1} = s_3)$ $= 4$
3. OPEN $= \{s_4\}$
4. next state to expand: s_4

$G(x_F) \leftarrow 0$; all other G values are infinite; OPEN $= \{x_F\}$;
  CLOSED $= \emptyset$;
while x_s *is not expanded* **do**
  $x_{k+1} \leftarrow$ remove x with the smallest G value from OPEN;
  insert x_{k+1} to CLOSED;
  for *every* $x_k \notin$ CLOSED s.t. $x_{k+1} \in X_{k+1}(x_k, u_k)$ **do**
    if $G(x_k) > \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$ **then**
      $G(x_k) \leftarrow \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$;
      insert x_k into OPEN;
    end
  end
end



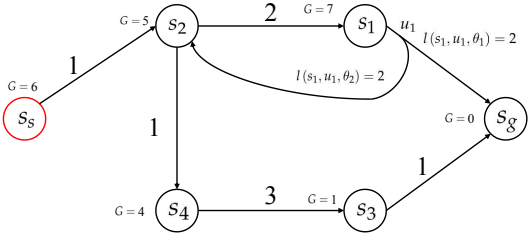
1. CLOSED $= \{s_g, s_3, s_4\}$
2. $G(x_k = s_2) = 1 + G(x_{k+1} = s_4) = 5$
3. OPEN $= \{s_2\}$
4. next state to expand: s_2

$G(x_F) \leftarrow 0$; all other G values are infinite; OPEN $= \{x_F\}$;
  CLOSED $= \emptyset$;
while x_s *is not expanded* **do**
  $x_{k+1} \leftarrow$ remove x with the smallest G value from OPEN;
  insert x_{k+1} to CLOSED;
  for *every* $x_k \notin$ CLOSED s.t. $x_{k+1} \in X_{k+1}(x_k, u_k)$ **do**
    if $G(x_k) > \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$ **then**
      $G(x_k) \leftarrow \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$;
      insert x_k into OPEN;
    end
  end
end



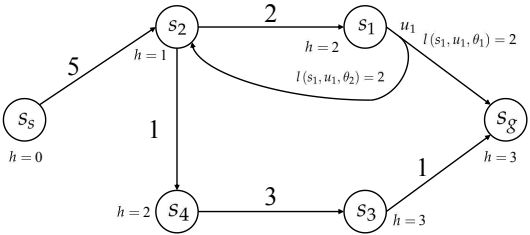
1. CLOSED $= \{s_g, s_3, s_4, s_2\}$
2. $G(x_k = s_1) = 2 + G(x_{k+1} = s_2)$ ≠ 7
3. OPEN $= \{s_1\}$
4. next state to expand: -

$G(x_F) \leftarrow 0$; all other G values are infinite; OPEN $= \{x_F\}$;
  CLOSED $= \emptyset$;
while x_s *is not expanded* **do**
  $x_{k+1} \leftarrow$ remove x with the smallest G value from OPEN;
  insert x_{k+1} to CLOSED;
  for *every* $x_k \notin$ CLOSED s.t. $x_{k+1} \in X_{k+1}(x_k, u_k)$ **do**
    if $G(x_k) > \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$ **then**
      $G(x_k) \leftarrow \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$;
      insert x_k into OPEN;
    end
  end
end



1. CLOSED $= \{s_g, s_3, s_4, s_2\}$
2. $G(x_k = s_s) = 1 + G(x_{k+1} = s_2)$
3. OPEN $= \{s_1, s_s\}$
4. next state to expand: s_s

$G(x_F) \leftarrow 0$; all other G values are infinite; OPEN $= \{x_F\}$;
CLOSED $= \emptyset$;
while x_s *is not expanded* **do**
  $x_{k+1} \leftarrow$ remove x with the smallest G value from OPEN;
  insert x_{k+1} to CLOSED;
  for *every* $x_k \notin$ CLOSED s.t. $x_{k+1} \in X_{k+1}(x_k, u_k)$ **do**
    if $G(x_k) > \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$ **then**
      $G(x_k) \leftarrow \max_{\theta_k} \{l(x_k, u_k, \theta_k) + G(x_{k+1})\}$;
      insert x_k into OPEN;
    end
  end
end



1. **How to improve the left algorithm to A* ?**
2. **Can this algorithm deal with failure cases ?** 陷入循环怎么办
3. **How to track the optimal plan π^* ?**

Pros/Cons of minimax cost planning

- robust to uncertainty
- overly pessimistic
- harder to compute than normal paths
  - especially if nondeterministic Dijkstra or A* does not apply
  - even if nondeterministic A* does apply, still more expensive than computing a single path with A*. – **Why** ?

1. **Probabilistic model**
2. **expected-case analysis**
3. Find an optimal plan π^* such that

$$G_{\pi^*}(x_s) = \min_{\pi}\{G_\pi(x_s)\} = \min_{\pi}\{E_{\mathcal{H}(\pi,x_s)}[L(\tilde{x},\tilde{u},\tilde{\theta})]\}. \tag{13}$$

4. Directly solving the problem is difficult.
5. Construct optimal solution by **Dynamic Programming**.
   - The recurrence between stage k and $k+1$ of optimal plan.

1. The optimal cost-to-goal for final state F can be directly acquired $G_F^* = l_F(x_F)$,
2. The costs of all optimal one-step plans from stage K to stage $F = K + 1$ is

$$G_K^*(x_K) = \min_{u_K} \{E_{\theta_K}[l(x_K, u_K, \theta_K) + G_F^*(f(x_K, u_K, \theta_K))]\} \tag{14}$$

3. More generally, G_k^* can be computed once G_{k+1}^* is given

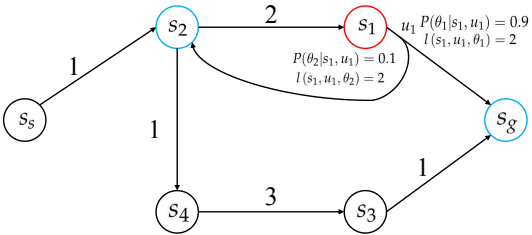
$$G_k^*(x_k) = \min_{u_k}\{E_{\theta_k}[\min_{u_{k+1},\ldots,u_K}\{E_{\theta_{k+1},\ldots,\theta_K}[l(x_k, u_k, \theta_k) + \sum_{i=k+1}^{K} l(x_i, u_i, \theta_i) + l_F(x_F)]\}]\} \tag{15}$$

$$G_k^*(x_k) = \min_{u_k}\{E_{\theta_k}[l(x_k, u_k, \theta_k) + \underbrace{\min_{u_{k+1},\ldots,u_K}\{E_{\theta_{k+1},\ldots,\theta_K}[\sum_{i=k+1}^{K} l(x_i, u_i, \theta_i) + l_F(x_F)]\}}_{G_{k+1}^*(x_{k+1})}]\} \tag{16}$$

Hence, the dynamic programming recurrence for solving expected cost plan is

$$G_k^*(x_k) = \min_{u_k \in U(x_k)} \left\{ E_{\theta_k} \left[l(x_k, u_k, \theta_k) + G_{k+1}^*(x_{k+1}) \right] \right\}, \tag{17}$$

which has a special name called **Bellman Optimality Equation**.



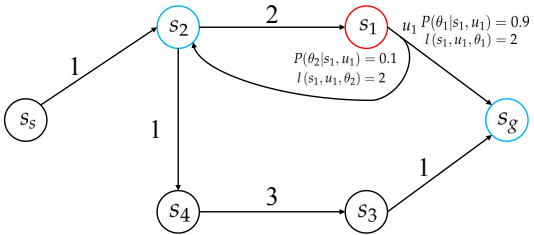
1. suppose $G_{k+1}^*(x_{k+1} = s_g) = 0$
2. suppose $G_{k+1}^*(x_{k+1} = s_2) = 2$
3. $G_k^*(x_k = s_1) =$
   $\min\{(2+0) * 0.9 + (2+2) * 0.1\}$

θ_1 θ_2

Hence, the dynamic programming recurrence for solving expected cost plan is

$$G_k^*(x_k) = \min_{u_k \in U(x_k)} \left\{ E_{\theta_k} \left[l(x_k, u_k, \theta_k) + G_{k+1}^*(x_{k+1}) \right] \right\} \tag{18}$$

which has a special name called **Bellman Optimality Equation**.



1. suppose $G_{k+1}^*(x_{k+1} = s_g) = 0$
2. suppose $G_{k+1}^*(x_{k+1} = s_2) = 2$
3. **how can we know** $G_{k+1}^*(x_{k+1} = s_2)$?

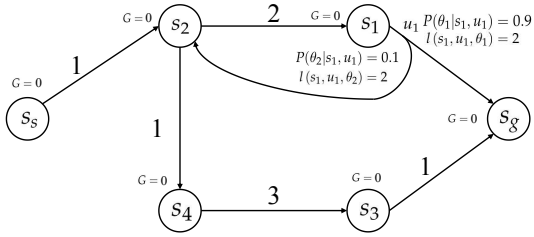
Initialize G values of all states to finite values;
while *not converge* **do**
  for *all the states* x **do**
    $G(x_F) = 0$
    $G_k(x_k) = \underbrace{\min_{u_k \in U(x_k)} \left\{ E_{\theta_k} \left[l(x_k, u_k, \theta_k) + G_{k+1}(x_{k+1}) \right] \right\}}_{\textbf{Bellman Update Equation}}, x_k \neq x_F$
  end
end

Algorithm 2: Value Iteration (VI)

1. Optimal values is achieved by conducting value iteration.
   - optimality is not related to iteration order.
   - convergence speed depends on iteration order.
2. Bellman update equation is a method of achieving Bellman optimal equation.

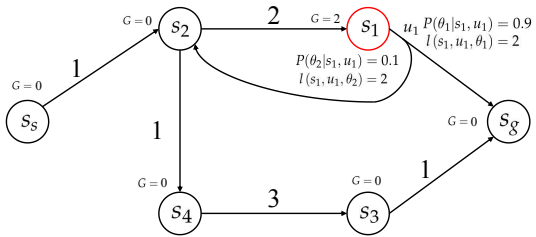
Expected Cost Planning

Initialize G values of all states to finite values;
while *not converge* **do**
  for *all the states* x **do**
    $G(x_F) = 0$;
    $G_k(x_k) =$
      $\min_{u_k \in U(x_k)} \{E_{\theta_k}[l(x_k, u_k, \theta_k) + G_{k+1}(x_{k+1})]\}$,
      if $x_k \neq x_F$;
  end
end



1. initialize all G value with zero.
2. iteration order:
   $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_s$

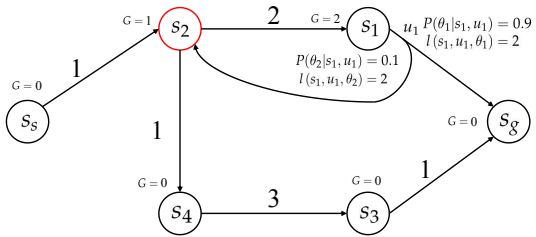
Initialize G values of all states to finite values;
while *not converge* **do**
  for *all the states* x **do**
    $G(x_F) = 0$;
    $G_k(x_k) =$
      $\min_{u_k \in U(x_k)} \{E_{\theta_k}[l(x_k, u_k, \theta_k) + G_{k+1}(x_{k+1})]\}$,
      if $x_k \neq x_F$;
  end
end



1. $G_k^*(x_k = s_1) =$
   $\min\{(2+0)*0.9 + (2+0)*0.1\}$ $=2$

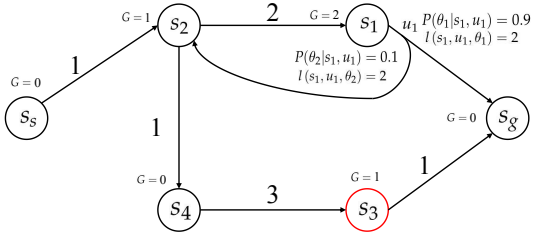
初始 $G(s_2)=0$

Initialize G values of all states to finite values;
while *not converge* **do**
  for *all the states* x **do**
    $G(x_F) = 0$;
    $G_k(x_k) =$
      $\min_{u_k \in U(x_k)} \{E_{\theta_k}[l(x_k, u_k, \theta_k) + G_{k+1}(x_{k+1})]\}$,
      if $x_k \neq x_F$;
  end
end



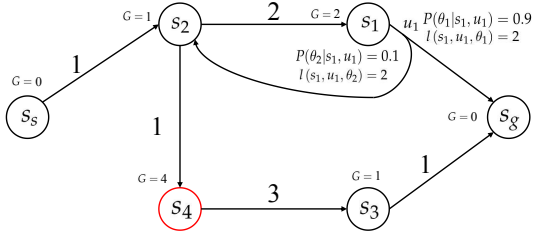
1. $G_k^*(x_k = s_2) =$
   $\min\{(1+0) * 1.0,\ (2+2) * 1.0\}$

Initialize G values of all states to finite values;
while *not converge* **do**
  for *all the states* x **do**
    $G(x_F) = 0$;
    $G_k(x_k) =$
      $\min_{u_k \in U(x_k)} \{E_{\theta_k}[l(x_k, u_k, \theta_k) + G_{k+1}(x_{k+1})]\}$,
      if $x_k \neq x_F$;
  end
end



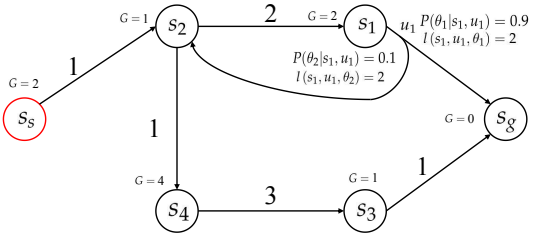
1. $G_k^*(x_k = s_3) = \min\{(1+0) * 1.0\}$

Initialize G values of all states to finite values;
while *not converge* **do**
  for *all the states* x **do**
    $G(x_F) = 0$;
    $G_k(x_k) =$
      $\min_{u_k \in U(x_k)} \{E_{\theta_k}[l(x_k, u_k, \theta_k) + G_{k+1}(x_{k+1})]\}$,
      if $x_k \neq x_F$;
  end
end



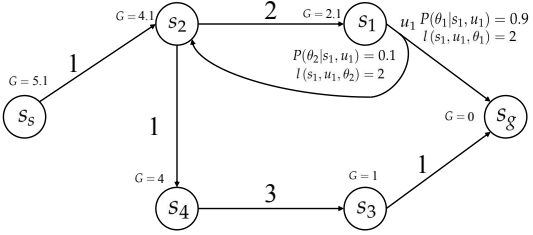
1. $G_k^*(x_k = s_4) = \min\{(3+1) * 1.0\}$

Initialize G values of all states to finite values;
while *not converge* **do**
  for *all the states* x **do**
    $G(x_F) = 0$;
    $G_k(x_k) =$
      $\min_{u_k \in U(x_k)} \{E_{\theta_k}[l(x_k, u_k, \theta_k) + G_{k+1}(x_{k+1})]\}$,
      if $x_k \neq x_F$;
  end
end



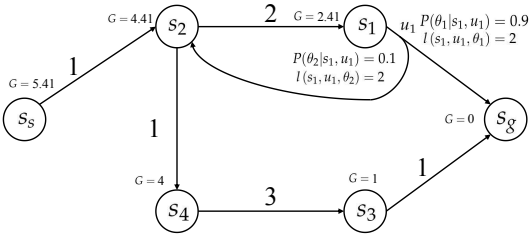
1. $G_k^*(x_k = s_s) = \min\{(1+1) * 1.0\}$

Initialize G values of all states to finite values;
while *not converge* **do**
  for *all the states* x **do**
    $G(x_F) = 0$;
    $G_k(x_k) =$
      $\min_{u_k \in U(x_k)} \{E_{\theta_k}[l(x_k, u_k, \theta_k) + G_{k+1}(x_{k+1})]\}$,
      if $x_k \neq x_F$;
  end
end



❶ after second iteration

Initialize G values of all states to finite values;
while *not converge* **do**
  for *all the states* x **do**
    $G(x_F) = 0$;
    $G_k(x_k) =$
      $\min_{u_k \in U(x_k)} \{E_{\theta_k}[l(x_k, u_k, \theta_k) + G_{k+1}(x_{k+1})]\}$,
      if $x_k \neq x_F$;
  end
end



1. after third iteration
2. **when will the algorithm converge ?**
3. **how to improve the left algorithm ?**
4. **how to prove the convergence ?**
5. **how to get the best plan π^* ?**

Pros/Cons of expected cost planning

- probabilistic optimal
  - reflect average performance
  - a specific execution maybe not optimal
- require a distribution of uncertainties
- harder to compute than normal paths
  - iterate over the entire map (states)
  - affected by initialization and iteration order

单次不一定是最优的

Real Time Dynamic Programming

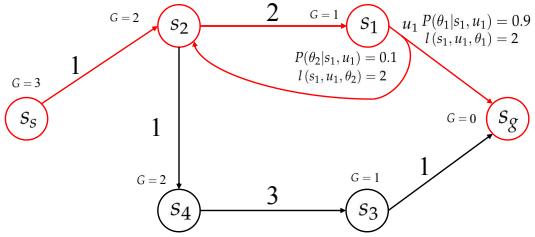
RTDP algorithm:

1. Initialize G values of all states to admissible values;
2. Follow greedy policy picking outcomes at random until goal is reached;
3. Backup all states visited on the way;
4. Reset to x_s and repeat 2-4 until all states on the current greedy policy have Bellman errors $< \Delta$, where $\Delta(x_k) = \|G(x_k) - G(x_{k+1})\|$;

Advantages:

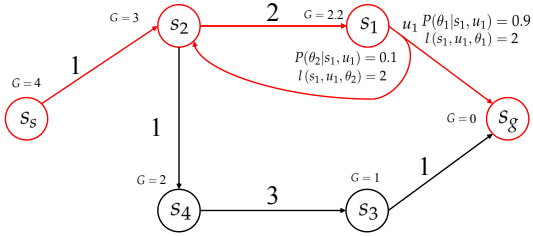
1. very efficient alternative to Value Iteration
2. does NOT compute values of all states
3. focuses computations on states that are relevant

1. Initialize G values of all states to admissible values;
2. Follow greedy policy picking outcomes at random until goal is reached;
3. Backup all states visited on the way;
4. Reset to x_s and repeat 2-4 until all states on the current greedy policy have Bellman errors $< \Delta$, where $\Delta(x_k) = \|G(x_k) - G(x_{k+1})\|$;



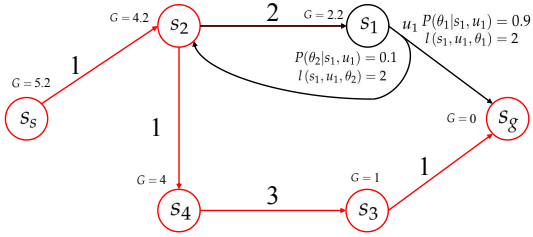
- initial trajectory with a heuristic

1. Initialize G values of all states to admissible values;
2. Follow greedy policy picking outcomes at random until goal is reached;
3. Backup all states visited on the way;
4. Reset to x_s and repeat 2-4 until all states on the current greedy policy have Bellman errors $< \Delta$, where $\Delta(x_k) = \|G(x_k) - G(x_{k+1})\|$;



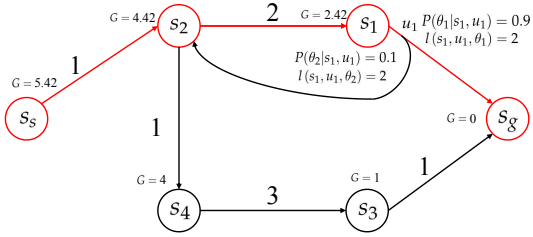
- backup all states visited on the way

1. Initialize G values of all states to admissible values;
2. Follow greedy policy picking outcomes at random until goal is reached;
3. Backup all states visited on the way;
4. Reset to x_s and repeat 2-4 until all states on the current greedy policy have Bellman errors $< \Delta$, where $\Delta(x_k) = \|G(x_k) - G(x_{k+1})\|$;



- greedy policy and backup all states

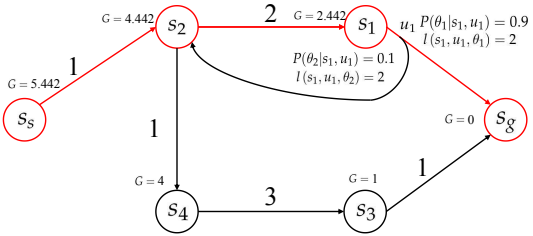
1. Initialize G values of all states to admissible values;
2. Follow greedy policy picking outcomes at random until goal is reached;
3. Backup all states visited on the way;
4. Reset to x_s and repeat 2-4 until all states on the current greedy policy have Bellman errors $< \Delta$, where $\Delta(x_k) = \|G(x_k) - G(x_{k+1})\|$;



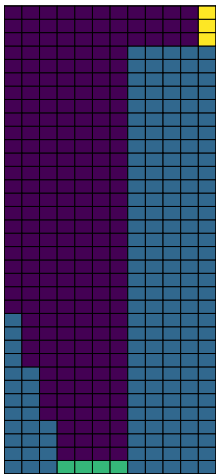
- greedy policy and backup all states

这种策略有的陷入死循环
局部优化：每次选择保留都有概率选择其他路径

1. Initialize G values of all states to admissible values;
2. Follow greedy policy picking outcomes at random until goal is reached;
3. Backup all states visited on the way;
4. Reset to x_s and repeat 2-4 until all states on the current greedy policy have Bellman errors $< \Delta$, where $\Delta(x_k) = \|G(x_k) - G(x_{k+1})\|$;



- greedy policy and backup all states



Grid Map

1. $X = \{(x, y) \mid 0 \le x \le 11, 0 \le y \le 34\}$
2. $X_I = \{\text{green grids}\}$
   $X_F = \{\text{yellow grids}\}$
3. $U = \{(\ddot{x}, \ddot{y}) | \ddot{x} \in \{0, \pm 1\}, \ddot{y} \in \{0, \pm 1\}\}$
4. $\Theta = \{\theta_1, \theta_2\}$
   - θ_1: $f(\mathbf{x}_{k+1}, \mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k \quad p_1 = 0.1$
   - θ_2: $f(\mathbf{x}_{k+1}, \mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_{k+1} \quad p_1 = 0.9$
5. $l(\mathbf{x}_k, \mathbf{x}_k, \theta_k) = -1$
6. **Find an optimal plan from X_I to X_F**

Thanks For Your Attention!