Principles of Robot Autonomy I

Decision making and dynamic programming

Today’s lecture

- Aim
  - Learn the fundamental principles of Markov decision processes and dynamic programming

- Readings
  - D. Bertsekas. Reinforcement Learning and Optimal Control, 2019. Chapters 1 and 2.

Basic decision-making problem (deterministic)

- System: $\mathbf{x}_{k+1} = f_k(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, \dots, N$
- Control constraints: $\mathbf{u}_k \in U(\mathbf{x}_k)$
- Cost:

$$J(\mathbf{x}_0; \boldsymbol{u}_0, \dots, \boldsymbol{u}_{N-1}) = g_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} g_k(\mathbf{x}_k, \mathbf{u}_k)$$

- Decision-making problem:

$$J^*(\mathbf{x}_0) = \min_{\mathbf{u}_k \in U(\mathbf{x}_k),\ k=0,\dots,N-1} J(\mathbf{x}_0; \boldsymbol{u}_0, \dots, \boldsymbol{u}_{N-1})$$

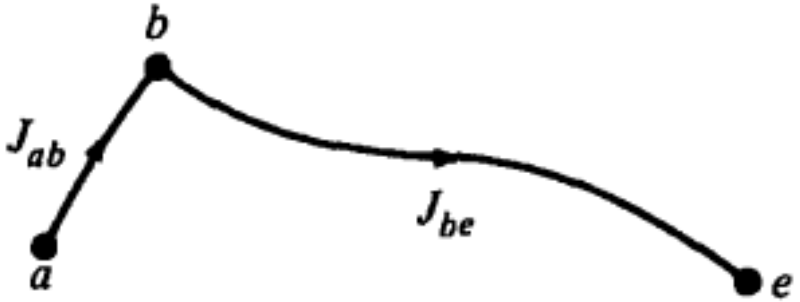
Key points

- Discrete-time model
- Additive cost (central assumption)

Principle of optimality

The key concept behind the dynamic programming approach is the principle of optimality

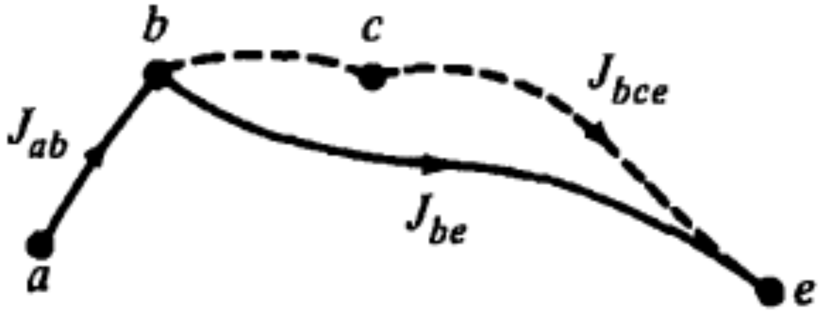
Suppose optimal path for a multi-stage decision-making problem is



- first decision yields segment $a - b$ with cost J_{ab}
- remaining decisions yield segments $b - e$ with cost J_{be}
- optimal cost is then $J_{ae}^* = J_{ab} + J_{be}$

Principle of optimality

- Claim: If $a - b - e$ is optimal path from a to e, then $b - e$ is optimal path from b to e
- *Proof*: Suppose $b - c - e$ is the optimal path from b to e. Then
$$J_{bce} < J_{be}$$
and
$$J_{ab} + J_{bce} < J_{ab} + J_{be} = J_{ae}^*$$



Contradiction!

Principle of optimality

Principle of optimality (for deterministic systems): Let $\{\mathbf{u}_0^*, \mathbf{u}_1^*, \ldots, \mathbf{u}_{N-1}^*\}$ be an optimal control sequence, which together with $\mathbf{x}_0^*$ determines the corresponding state sequence $\{\mathbf{x}_0^*, \mathbf{x}_1^*, \ldots, \mathbf{x}_N^*\}$. Consider the subproblem whereby we are at $\mathbf{x}_k^*$ at time k and we wish to minimize the cost-to-go from time k to time N, i.e.,

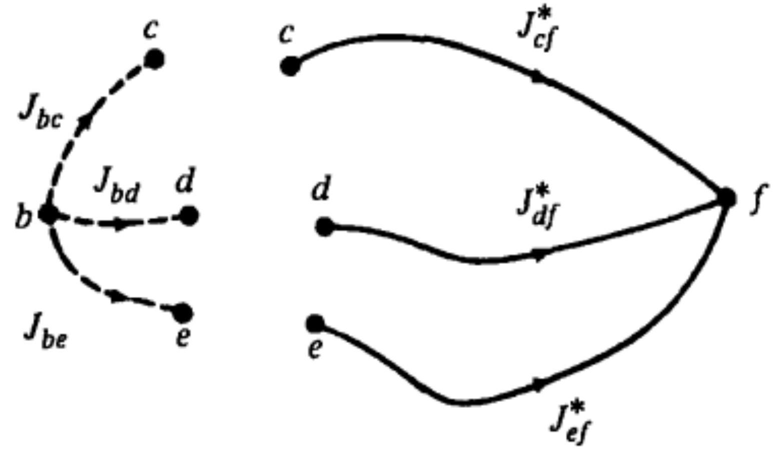
$$g_k(\mathbf{x}_k^*, \mathbf{u}_k) \; + \; \sum_{m=k+1}^{N-1} g_m(\mathbf{x}_m, \mathbf{u}_m) + g_N(\mathbf{x}_N)$$

Then the truncated optimal sequence $\{\mathbf{u}_k^*, \mathbf{u}_{k+1}^*, \ldots, \mathbf{u}_{N-1}^*\}$ is optimal for the subproblem

- Tail of optimal sequences optimal for tail subproblems

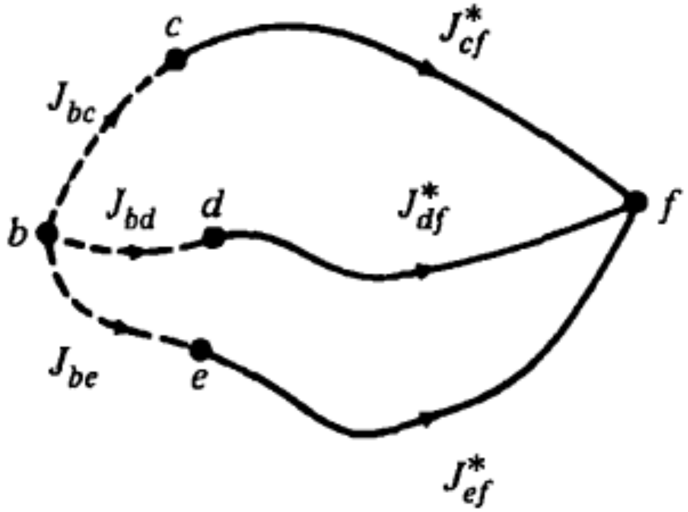
Applying the principle of optimality

Principle of optimality: if $b - c$ is the initial segment of the optimal path from b to f, then $c - f$ is the terminal segment of this path



Hence, the optimal trajectory is found by comparing:

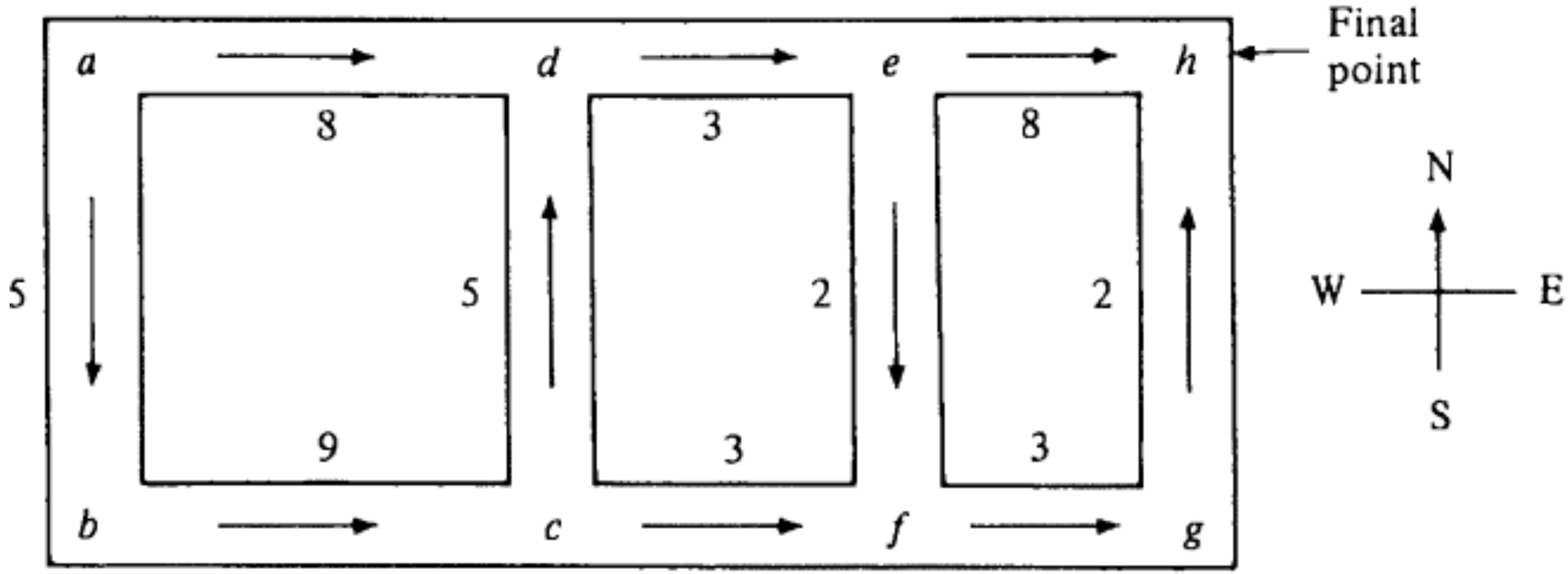
$$C_{bcf} = J_{bc} + J^*_{cf}$$
$$C_{bdf} = J_{bd} + J^*_{df}$$
$$C_{bef} = J_{be} + J^*_{ef}$$



Applying the principle of optimality

- need only to compare the concatenations of immediate decisions and optimal decisions → significant decrease in computation / possibilities
- in practice: carry out this procedure backward in time

Example



Optimal cost: 18

Optimal path: $a \rightarrow\ d \rightarrow\ e \rightarrow\ f \rightarrow\ g \rightarrow\ h$

DP Algorithm

- Start with

$$J_N^*(\mathbf{x}_N) \; = \; g_N(\mathbf{x}_N), \;\; \text{for all } \mathbf{x}_N$$

- and for $k = 0, \dots, N-1$, let

$$J_k^*(\mathbf{x}_k) = \min_{\mathbf{u}_k \in U(\mathbf{x}_k)} g(\mathbf{x}_k, \mathbf{u}_k) + \; J_{k+1}^*\big(f(\mathbf{x}_k, \mathbf{u}_k)\big) \quad \text{for all } \mathbf{x}_k$$

Once the functions $J_0^*, \dots, J_N^*$ have been determined, the optimal sequence can be determined with a forward pass

Comments

- discretization (from differential equations to difference equations)
- quantization (from continuous to discrete state variables / controls)
- interpolation
- global minimum
- constraints, in general, simplify the numerical procedure

Basic decision-making problem (stochastic)

- System: $\mathbf{x}_{k+1} = f_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k), \quad k = 0, \dots, N-1$
- Control constraints: $\mathbf{u}_k \in U(\mathbf{x}_k)$
- Probability distribution: $P_k(\cdot \mid \mathbf{x}_k, \mathbf{u}_k)$
- Policies: $\pi = \{\pi_0, \dots, \pi_{N-1}\}$, where $\mathbf{u}_k = \pi_k(\mathbf{x}_k)$
- Expected cost:

$$J_\pi(\mathbf{x}_0) = E\left\{g_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} g_k(\mathbf{x}_k, \pi_k(\mathbf{x}_k), \mathbf{w}_k)\right\}$$

- Decision-making problem:

$$J^*(\mathbf{x}_0) = \min_{\pi} \; J_\pi(\mathbf{x}_0)$$

Key points

- Discrete-time model
- Markovian model
- Objective: find optimal closed-loop policy
- Additive cost (central assumption)
- Risk-neutral formulation

Other communities use different notation:

- Powell, W. B. AI, OR and control theory: A Rosetta Stone for stochastic optimization. Princeton University, 2012. http://castlelab.princeton.edu/Papers/AIOR_July2012.pdf

Principle of optimality

Principle of optimality (for stochastic systems): Let $\pi^* := \{\pi_0^*, \pi_1^*, \dots, \pi_{N-1}^*\}$ be an optimal policy. Assume state $\mathbf{x}_k$ is reachable. Consider the subproblem whereby we are at $\mathbf{x}_k$ at time k and we wish to minimize the cost-to-go from time k to time N. Then the truncated policy $\{\pi_k^*, \pi_{k+1}^*, \dots, \pi_{N-1}^*\}$ is optimal for the subproblem

- tail policies optimal for tail subproblems

DP Algorithm

DP Algorithm: For every initial state $\mathbf{x}_0$, the optimal cost $J^*(\mathbf{x}_0)$ is equal to $J_0(\mathbf{x}_0)$, given by the last step of the following algorithm, which proceeds backward in time from stage $N-1$ to stage 0:

$$J_N(\mathbf{x}_N) \;=\; g_N(\mathbf{x}_N)$$

$$J_k(\mathbf{x}_k) = \min_{\mathbf{u}_k \in U(\mathbf{x}_k)} E_{\mathbf{w}_k}\{g_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) + J_{k+1}(f_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k))\}, \quad k = 0, \dots, N-1$$

Furthermore, if $\mathbf{u}_k^* = \pi_k^*(\mathbf{x}_k)$ minimizes the right-hand side of the above equation for each $\mathbf{x}_k$ and k, the policy $\{\pi_0^*, \pi_1^*, \dots, \pi_{N-1}^*\}$ is optimal

Example: Inventory Control Problem (1/2)

- Stock available $x_k \in \mathbb{N}$, inventory $u_k \in \mathbb{N}$, and demand $w_k \in \mathbb{N}$
- Dynamics: $x_{k+1} = \max(0, x_k + u_k - w_k)$
- Constraints: $x_k + u_k \leq 2$
- Probabilistic structure: $p(w_k = 0) = 0.1$, $p(w_k = 1) = 0.7$, and $p(w_k = 2) = 0.2$
- Cost

$$E\left\{0 + \sum_{k=0}^{2} \left(u_k + (x_k + u_k - w_k)^2 \right) \right\}$$

Example: Inventory Control Problem (1/2)

- Algorithm takes form for $k = 0,1,2$

$$J_k(x_k) = \min_{0 \leq u_k \leq 2 - x_k} E_{w_k} \{u_k + (x_k + u_k - w_k)^2 + J_{k+1}\left(\max(0, x_k + u_k - w_k)\right)\}$$

- For example

$$J_2(0) = \min_{u_2 = 0,1,2} E_{w_2} \{u_2 + (u_2 - w_2)^2\} = \min_{u_2 = 0,1,2} \{u_2 + 0.1(u_2)^2 + 0.7(u_2 - 1)^2 + 0.2(u_2 - 2)^2\}$$

which yields $J_2(0) = 1.3$, and $\pi_2^*(0) = 1$

- Final solution $J_0(0) = 3.7$, $J_0(1) = 2.7$, and $J_0(2) = 2.818$

Difficulties of DP

- Curse of dimensionality:
  - Exponential growth of the computational and storage requirements
  - Intractability of imperfect state information problems

- Curse of modeling: if "system stochastics" are complex, it is difficult to obtain expressions for the transition probabilities

- Curse of time
  - The data of the problem to be solved is given with little advance notice
  - The problem data may change as the system is controlled—need for on-line replanning

Approximation approaches in RL

- There are two general types of approximation in DP-based suboptimal control
  1. Approximation in value space, where we aim to approximate the optimal cost function
  2. Approximation in policy space, where we select the policy by using optimization over a suitable class of policies

Approximation in value space

- In approximation in value space, we approximate the optimal cost-to-go functions J_k^* with some other functions $\tilde{J}_k$
- We then replace J_k^* in the DP equation as

$$\tilde{\pi}_k(\mathbf{x}_k) \in \underset{\mathbf{u}_k \in U(\mathbf{x}_k)}{\operatorname{argmin}} E_{\mathbf{w}_k}\{g_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) + \tilde{J}_{k+1}\left(f_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)\right)\}$$

- Several possibilities for computing $\tilde{J}_k$, for example:
  - Problem approximation
  - On-line approximate optimization
  - Parametric cost approximation (e.g., neural networks)
  - Aggregation

Approximation in policy space

- In approximation in policy space, one selects the policy from a suitably restricted class of policies, usually a parametric class of some form, e.g.,

  $\pi_k(\mathbf{x}_k, \mathbf{r}_k)$, where $\mathbf{r}_k$ is a parameter (e.g., weights of a NN)

Next time