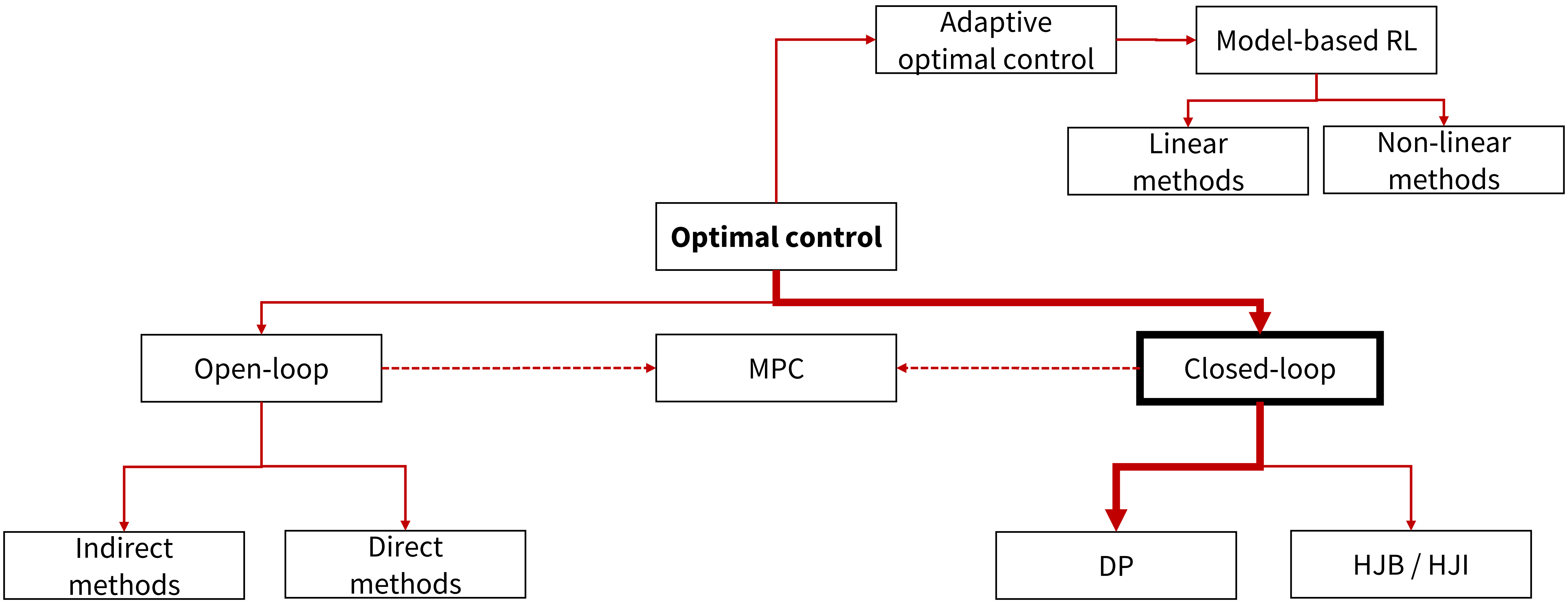
AA203
Optimal and Learning-based Control

Dynamic programming

Stanford University

Autonomous Systems Lab

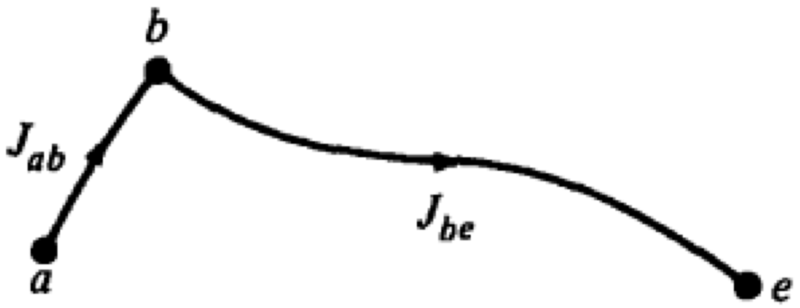
Roadmap


Adaptive optimal control
Model-based RL
Linear methods
Non-linear methods
Optimal control
Open-loop
MPC
Closed-loop
Indirect methods
Direct methods
DP
HJB / HJI

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 $J_{ae}^{*} = J_{ab} + J_{be}$

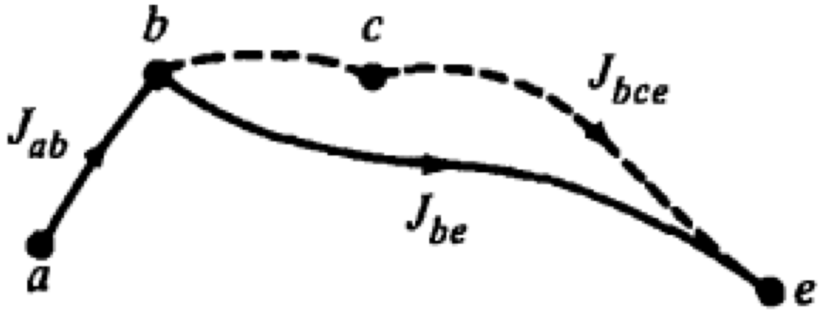
Principle of optimality

- 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!

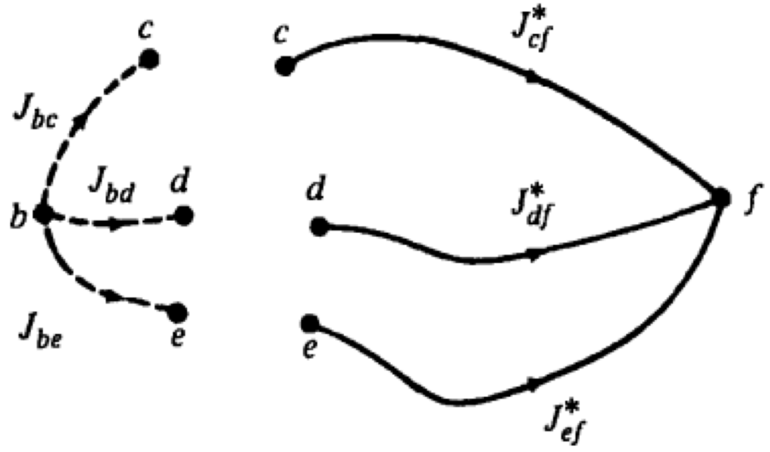
Principle of optimality

Principle of optimality (for discrete-time 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
- notation: $\pi_k^*(\mathbf{x}_k) = \pi^*(\mathbf{x}_k, k)$

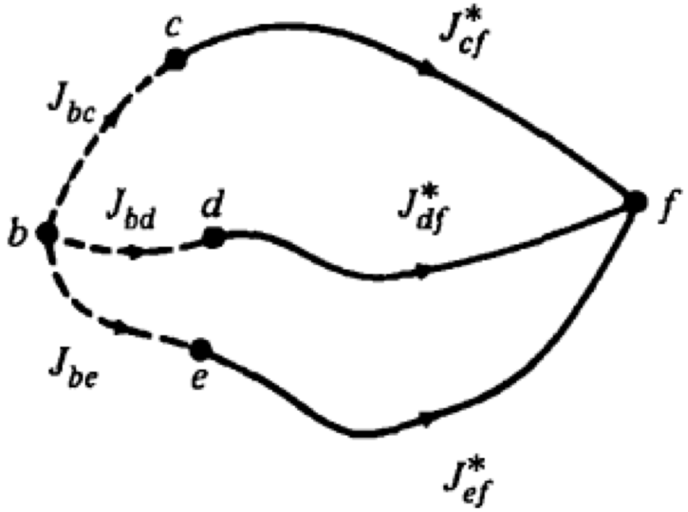
Applying the principle of optimality

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:

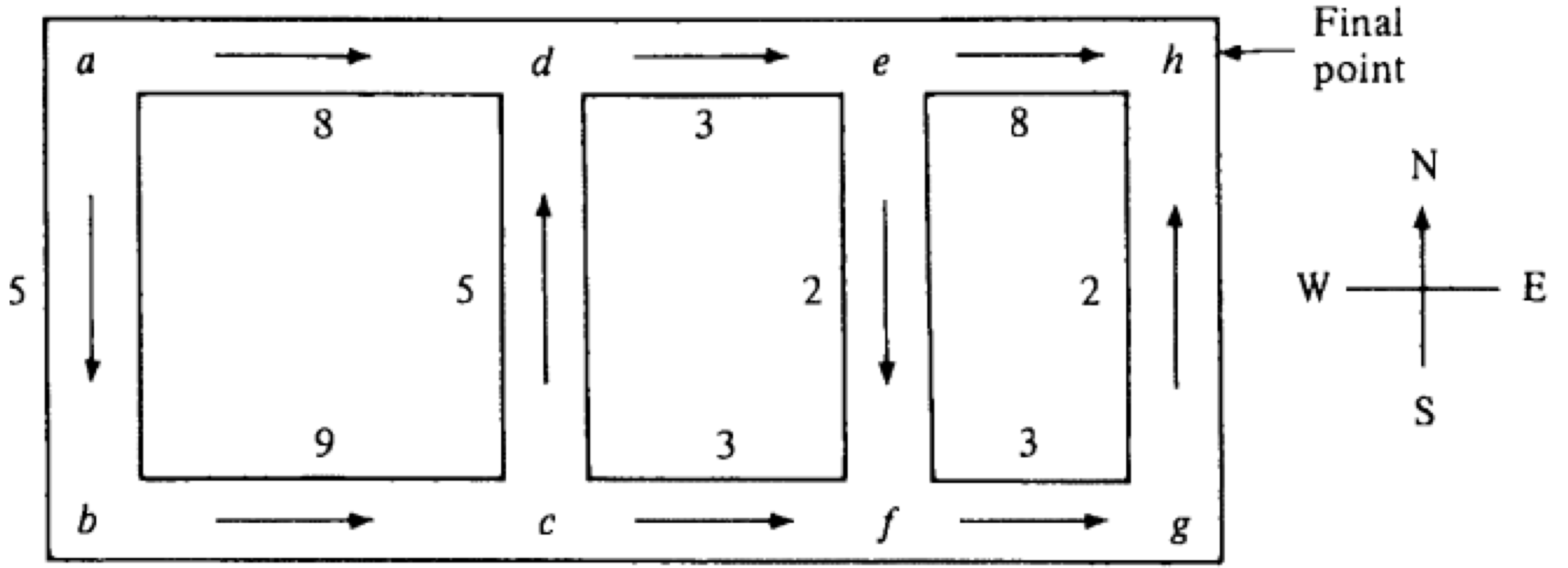
$$C_{bcf} = J_{bc} + J^*_{cf}$$
$$C_{bdf} = J_{bd} + J^*_{df}$$
$$C_{bef} = J_{be} + J^*_{ef}$$



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

- Model: $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, k), \quad \mathbf{u}_k \in U(\mathbf{x}_k)$
- Cost: $J(\mathbf{x}_0) = h_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} g(\mathbf{x}_k, \pi_k(\mathbf{x}_k), k)$

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:

$$J_N(\mathbf{x}_N) = h_N(\mathbf{x}_N)$$

$$J_k(\mathbf{x}_k) = \min_{\mathbf{u}_k \in U(\mathbf{x}_k)} g(\mathbf{x}_k, \mathbf{u}_k, k) + J_{k+1}\big(f(\mathbf{x}_k, \mathbf{u}_k, k)\big), \qquad k = 0, \ldots, N-1$$

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 $\{\pi_0^*, \pi_1^*, \ldots, \pi_{N-1}^*\}$ is optimal

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
- optimal control in closed-loop form
- curse of dimensionality

Example: discrete LQR

- In most cases, DP algorithm needs to be performed numerically
- A few cases can be solved analytically

Discrete LQR: select control inputs to minimize

$$J(\mathbf{x}_0) = \frac{1}{2}\mathbf{x}_N' \, H \, \mathbf{x}_N + \frac{1}{2}\sum_{k=0}^{N-1}[\mathbf{x}_k' \, Q \, \mathbf{x}_k + \mathbf{u}_k' \, R \, \mathbf{u}_\mathbf{k}]$$

subject to the dynamics

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \, \mathbf{u}_k$$

Assumption: $H = H' \geq 0, \; Q = Q' \geq 0, R = R' > 0$

Example: discrete LQR

First step:

$$J_N^*(\mathbf{x}_N) = \frac{1}{2}\mathbf{x}_N' H \mathbf{x}_N := \frac{1}{2}\mathbf{x}_N' P_N \mathbf{x}_N$$

Going backward

$$J_{N-1}(\mathbf{x}_{N-1}) = \min_{\mathbf{u}_{N-1}} \; \frac{1}{2}\left\{\mathbf{x}_{N-1}' Q \mathbf{x}_{N-1} + \mathbf{u}_{N-1}' R \mathbf{u}_{N-1} + \mathbf{x}_N' H \mathbf{x}_N\right\}$$

$$\min_{\mathbf{u}_{N-1}} \; \frac{1}{2}\Big\{\mathbf{x}_{N-1}' Q \mathbf{x}_{N-1} + \mathbf{u}_{N-1}' R \mathbf{u}_{N-1} +$$

$$(A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1})' H (A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1})\Big\}$$

Example: discrete LQR

Taking derivative

$$\frac{\partial J^*_{N-1}(\mathbf{x}_{N-1})}{\partial \mathbf{u}_{N-1}} = R\mathbf{u}_{N-1} + B'_{N-1}H(A_{N-1}\mathbf{x}_{N-1} + B_{N-1}\mathbf{u}_{N-1}) = 0$$

and

$$\frac{\partial^2 J^*_{N-1}(\mathbf{x}_{N-1})}{\partial \mathbf{u}^2_{N-1}} = R + B'_{N-1}HB_{N-1} > 0$$

DP for discrete LQR

Hence, the optimizer satisfies

$$(R + B'_{N-1} H B_{N-1})\mathbf{u}^*_{N-1} + B'_{N-1} H A_{N-1} \mathbf{x}_{N-1} = 0$$

so

$$\mathbf{u}^*_{N-1} = -(R + B'_{N-1} H B_{N-1})^{-1} B'_{N-1} H A_{N-1} \mathbf{x}_{N-1} := F_{N-1} \mathbf{x}_{N-1}$$

DP for discrete LQR

Plugging in

$$J_{N-1}(\mathbf{x}_{N-1}) = \frac{1}{2}\mathbf{x}'_{N-1}\Big\{Q + F'_{N-1}RF_{N-1} + (A_{N-1} + B_{N-1}F_{N-1})'H(A_{N-1} + B_{N-1}F_{N-1})\Big\}\mathbf{x}_{N-1}$$

$$:= \mathbf{x}'_{N-1}P_{N-1}\mathbf{x}_{N-1}$$

$$F_{N-1} = -(R + B'_{N-1}P_N B_{N-1})^{-1}B'_{N-1}P_N A_{N-1}$$

DP for discrete LQR

Proceeding by induction, the solution is given by

1. $J_N(\mathbf{x}_N) = \frac{1}{2}\mathbf{x}_N' \, P_N \mathbf{x}_N$, where $P_N = H$
2. $\mathbf{u}_k^* = F_k \mathbf{x}_k$, where $F_k = -(R + B_k' \, P_{k+1} B_k)^{-1} B_k' \, P_{k+1} \, A_k$
3. $J_k(\mathbf{x}_k) = \frac{1}{2}\mathbf{x}_k' \, P_k \mathbf{x}_k$, where

$$P_k = Q + F_k' \, R \, F_k + (A_k + B_k F_k)' \, H \, (A_k + B_k F_k)$$

At the end, $J_0(\mathbf{x}_0) = \frac{1}{2}\mathbf{x}_0' P_0 \mathbf{x}_0$

Next time

- iLQR, DDP, and LQG

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B_k \mathbf{u}_k + \mathbf{w}_k$$
$$\boldsymbol{y}_k = C_k \mathbf{x}_k + \mathbf{v}_k$$