Stochastic Optimal Control

Marco Pavone
Stanford University
AA 241X Mission

Mission: “A wild fire is occurring in Lake Lagunita and AA241X Teams have been contracted to minimize the damage. Teams have to design, build and fly a UAV that can detect, prevent and extinguish the fire, with the goal of minimizing the area on fire in a fixed amount of time. Multiple fires can be present at the start of mission and as time goes by the fire propagates through Lake Lagunita.”

• A difficult problem, it combines exploration and exploitation
Mission: “A wild fire is occurring in Lake Lagunita and AA241X Teams have been contracted to minimize the damage. Teams have to design, build and fly a UAV that can detect, prevent and extinguish the fire, with the goal of minimizing the area on fire in a fixed amount of time. Multiple fires can be present at the start of mission and as time goes by the fire propagates through Lake Lagunita.”

- A difficult problem, it combines exploration and exploitation
- Goal: to provide you with fundamental knowledge in the field of stochastic optimal control (focus on exploitation)
Mission: “A wild fire is occurring in Lake Lagunita and AA241X Teams have been contracted to minimize the damage. Teams have to design, build and fly a UAV that can detect, prevent and extinguish the fire, with the goal of minimizing the area on fire in a fixed amount of time. Multiple fires can be present at the start of mission and as time goes by the fire propagates through Lake Lagunita.”

- A difficult problem, it combines exploration and exploitation
- Goal: to provide you with fundamental knowledge in the field of stochastic optimal control (focus on exploitation)
- Approach: dynamic programming
Basic SOC Problem

- **System:** \( x_{k+1} = f_k(x_k, u_k, w_k), \ k = 0, \ldots, N \)
Basic SOC Problem

- **System:** $x_{k+1} = f_k(x_k, u_k, w_k), \ k = 0, \ldots, N$
- **Control constraints:** $u_k \in U(x_k)$
Basic SOC Problem

- **System:** \( x_{k+1} = f_k(x_k, u_k, w_k), \ k = 0, \ldots, N \)
- **Control constraints:** \( u_k \in U(x_k) \)
- **Probability distribution:** \( P_k(\cdot|x_k, u_k) \) of \( w_k \)
Basic SOC Problem

- **System:** \( x_{k+1} = f_k(x_k, u_k, w_k), \ k = 0, \ldots, N \)

- **Control constraints:** \( u_k \in U(x_k) \)

- **Probability distribution:** \( P_k(\cdot|x_k, u_k) \) of \( w_k \)

- **Policies:** \( \pi = \{\mu_0, \ldots, \mu_{N-1}\} \), where \( u_k = \mu_k(x_k) \)
Basic SOC Problem

- System: $x_{k+1} = f_k(x_k, u_k, w_k), \ k = 0, \ldots, N$

- Control constraints: $u_k \in U(x_k)$

- Probability distribution: $P_k(\cdot|x_k, u_k)$ of $w_k$

- Policies: $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, where $u_k = \mu_k(x_k)$

- Expected Cost:

$$J_\pi(x_0) = E \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$
Basic SOC Problem

- System: $x_{k+1} = f_k(x_k, u_k, w_k), \ k = 0, \ldots, N$
- Control constraints: $u_k \in U(x_k)$
- Probability distribution: $P_k(\cdot | x_k, u_k)$ of $w_k$
- Policies: $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, where $u_k = \mu_k(x_k)$
- Expected Cost:

$$J_\pi(x_0) = E \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

- Stochastic optimal control problem

$$J^*(x_0) = \min_\pi J_\pi(x_0)$$
Key points

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

Principle of Optimality

- Let $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be optimal policy

- Consider tail subproblem

$$E \left\{ g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

and the tail policy $\{\mu_i^*, \ldots, \mu_{N-1}^*\}$
Principle of Optimality

- Let $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be optimal policy
- Consider tail subproblem

$$E \left\{ g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

and the tail policy $\{\mu_i^*, \ldots, \mu_{N-1}^*\}$

- Principle of optimality: The tail policy is optimal for the tail subproblem
The DP Algorithm

Intuition:

• DP first solves ALL tail subproblems at the final stage
• At generic step, it solves ALL tail subproblems of a given time length, using solution of tail subproblems of shorter length
The DP Algorithm

Intuition:
• DP first solves ALL tail subproblems at the final stage
• At generic step, it solves ALL tail subproblems of a given time length, using solution of tail subproblems of shorter length

The DP algorithm:
• Start with
  \[ J_N(x_N) = g_N(x_N), \]
  and go backwards using
  \[ J_k(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k} \{ g_k(x_k, u_k, w_k) + J_{k+1}(f(x_k, u_k, w_k)) \}, \]
  for \( k = 0, 1, \ldots, N - 1 \)
• Then \( J^*(x_0) = J_0(x_0) \) and optimal policy is constructed by setting \( \mu_k^*(x_k) = u_k^* \).
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

$$E \left\{ \begin{array}{c} 0 \\ g_3(x_3) \end{array} \right\} + \sum_{k=0}^{2} \left( u_k + (x_k + u_k - w_k)^2 \right)$$
• Algorithm takes form

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

for \( k = 0, 1, 2 \)
Example: Inventory Control Problem (2/2)

- Algorithm takes form

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

for \( k = 0, 1, 2 \)

- For example

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

which yields \( J_2(0) = 1.3 \), and \( \mu^*_2(0) = 1 \)
Example: Inventory Control Problem (2/2)

- Algorithm takes form

\[ J_k(x_k) = \min_{0 \leq u_k \leq 2-x_k} E_{w_k} \{ u_k + (x_k + u_k - w_k)^2 \} \]

\[ + J_{k+1}(\max(0, x_k + u_k - w_k)) \}, \]

for \( k = 0, 1, 2 \)

- 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 \( \mu_2^*(0) = 1 \)

- 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
Solution: Approximate DP

- Certainty Equivalent Control
- Cost-to-Go Approximation
- Other Approaches (e.g., approximation in policy space)
Certainty Equivalent Control

- Idea: Replace the stochastic problem with a deterministic one
- At each time “k,” the future uncertain quantities are fixed at some “typical” values
- Online implementation
  1. Fix the \( w_i, i \geq k \), at some \( \bar{w}_i \) and solve deterministic problem

\[
\min g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i, \bar{w}_i)
\]

where \( x_{i+1} = f_i(x_i, u_i, w_i) \)

2. Use as control \( \bar{\mu}_k(x_k) \) the first element of optimal control sequence and move to step \( k + 1 \)
Certainty Equivalent Control

• Idea: Replace the stochastic problem with a deterministic one

• At each time “$k$,” the future uncertain quantities are fixed at some “typical” values

• Online implementation

1. Fix the $w_i, \ i \geq k$, at some $\bar{w}_i$ and solve deterministic problem

$$\min g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i, \bar{w}_i)$$

where $x_{i+1} = f_i(x_i, u_i, w_i)$

2. Use as control $\bar{\mu}_k(x_k)$ the first element of optimal control sequence and move to step $k + 1$

• Extends to imperfect state information case (use $\bar{x}_k(l_k)$)
Cost-to-Go Approximation (CGA)

• Idea: Truncate time horizon and approximate cost-to-go

• One-step lookahead policy: at each \( k \) and state \( x_k \), use control \( \tilde{\mu}_k(x_k) \) that

\[
\min_{u_k \in U_k(x_k)} E \left\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k)) \right\},
\]

• \( \tilde{J}_N = g_N \)

• \( \tilde{J}_{k+1} \): approximation to true-cost-to-go \( J_{k+1} \)

• Analogously, two-step lookahead policy: all of the above and

\[
\tilde{J}_{k+1}(x_{k+1}) = \min_{u_{k+1} \in U_{k+1}(x_{k+1})} E\left\{ g_{k+1}(x_{k+1}, u_{k+1}, w_{k+1}) + \tilde{J}_{k+2}(f_{k+1}(x_{k+1}, u_{k+1}, w_{k+1})) \right\}
\]
CGA—Computational Aspects

• If $\tilde{J}_{k+1}$ is readily available and minimization not too hard, this approach is implementable on-line

• Choice of approximating functions $\tilde{J}_k$ is critical
  1. **Problem Approximation**: approximate by considering simpler problem
  2. **Parametric Cost-to-Go Approximation**: approximate cost-to-go function with function of suitable parametric form (parameters tuned by some scheme → neuro-dynamic programming)
  3. **Rollout Approach**: approximate cost-to-go with cost of some suboptimal policy
CGA—Problem Approximation

- Many problem-dependent possibilities
  - Replace uncertain quantities by nominal values (in the spirit of CEC)
  - Simplify difficult constraints or dynamics
  - Decouple subsystems
  - Aggregate states
CGA—Parametric Approximation

- Use a cost-to-go approximation from a parametric class \( \tilde{J}(x, r) \) where \( x \) is the current state and \( r = (r_1, \ldots, r_m) \) is a vector of “tunable” weights

- Two key aspects
  - Choice of parametric class \( \tilde{J}(x, r) \)
    - Example: future extraction method
      \[
      \tilde{J}(x, r) = \sum_{i=1}^{m} r_i y_i(x),
      \]
      where the \( y_i \)'s are features
  - Algorithm for tuning the weights (possibly, simulation-based)
CGA—Rollout Approach

- $\tilde{J}_k$ is the cost-to-go of some heuristic policy (called the base policy)

- To compute rollout control, one needs for all $u_k$

$$Q_k(x_k, u_k) := E \{ g_k(x_k, u_k, w_k) + H_{k+1}(f_k(x_k, u_k, w_k)) \},$$

where $H_{k+1}$ is the value of the cost-to-go for the base policy

- $Q$-factors can be evaluated via Monte-Carlo simulation
- $Q$-factors can be approximated, e.g., by using a CEC approach

- Model predictive control (MPC) can be viewed as a special case of rollout algorithms (AA 203)
Other ADP Approaches

- Minimize the DP equation error
- Direct approximation of control policies
- Approximation in policy space
References

