Simulation Based Approximation of the Value Function in Process Control

Jay H. Lee
School of Chemical Engineering
IMA Workshop, Dec 4-6, 2003
Multi-Stage Dynamic Optimization

\[
\min_{u_i = \mu_i(x_i)} \left\{ \sum_{i=0}^{p-1} \phi(x_i, u_i) + \phi_p(x_p) \right\}
\]

\( g_i(x_i, u_i) \geq 0 \) Path constraints
\( g_p(x_p) \geq 0 \) Terminal constraints
\( \dot{x} = f(x, u) \) Model constraints

\( p = \infty \quad \rightarrow \quad \mu^*(x) \quad \leftarrow \quad \text{Hamilton-Jacobi-Bellman Eqn} \)
Model Predictive Control (MPC)

\[ \phi(x_i, u_i) = \|x_i\|_Q^2 + \|u_i\|_R^2 \]
\[ \phi_p(x_p) = \|x_p\|_P^2 \]

Online Math Programming
- Numerically solve the multi-stage optimization at each sample time for a specific given state with feedback update

\[ x_0 = x(t) \text{ or } \hat{x}(t|t) \quad \text{On-line optimization} \quad u(t) = u_0 \]
Analogy to Chess Playing

IBM’s Deep Blue

Your Move

The Opponent
(Disturbance)

You
(The Controller)

Your Opponent’s Move
Industrial Use of MPC

• Initiated at Shell Oil and other refineries during late 70s.
• **4600** worldwide installations by major vendors + unknown # of “in-house” and small vendor installations (Result of a survey in yr 2000).
• Majority of applications (67%) are in refining and petrochemicals. Chemical and pulp and paper are the next areas.
• Many vendors specializing in the technology
  – Early Players: DMCC, Setpoint, Profimatics
  – Today’s Players: Aspen Technology, Honeywell, Invensys
• Models used are predominantly linear empirical models developed through plant testing.
• Technology is used not only for multivariable control but for most economic operation within constraint boundaries.
• Rigorous stability results
Nonlinear Programming Approach

- Discretize model equations → algebraic constraints
- Solve for $u_0, \ldots, u_{p-1}$ for given $x_0$ using (MI)NLP solvers
- Significant flexibility BUT some drawbacks
  - Complex systems: Multi-scale, nonlinear, hybrid system dynamics with uncertainties (e.g., plantwide systems, material processing, logical constraints)
  - Computational: large model, large $p$, small sample time
  - Theoretical: Not amenable to rigorously handling systems with uncertainties and information feedback
- Alternative
  - Solve it off-line or reduce the complexity of on-line calculation through off-line calculation.
Value Function Based Approach

• Infinite Horizon Optimal Cost-to-Go (‘Value Function’)

\[ J^*_\infty (x_0) \Rightarrow \sum_{j=0}^\infty \phi (x_j, u_j) \]

• Solution to the ‘Bellman Equation’

\[ J^*_\infty (x_0) = \min_{u_0} \left\{ \phi (x_0, u_0) + J^*_\infty (f_h (x_0, u_0)) \right\} \]

• The above parameterizes the optimal solution for all or a family of \( x_0 \)

\[ \mu (x_0) = \arg \min_{u_0} \left\{ \phi (x_0, u_0) + J^*_\infty (f_h (x_0, u_0)) \right\} \]
Analogy to Chess Playing

“Fitness” score for every possible board position

Your Move ← based on the “Fitness” Scores for Feasible Positions

The Opponent (The Plant)

You (The Controller)

The Opponent’s Move → A New Board Position
Where does $J^*$ Come From?

• Bellman Eqn: Analytical solution is seldom possible

• Dynamic Programming: @ each stage, $J_0^*(x) \rightarrow J_\infty^*(x)$
  - Grid the state space finely (for continuous s.s.).
  - For each grid point, calculate and store cost-to-go values by performing the optimization

$$J_{i+1}^*(x) = \min_u \left\{ f(x, u) + J_i^* \left( f_h(x, u) \right) \right\}$$

- “Curse of Dimensionality”
Simulation Based Cost-To-Go Approximation

• Motivation
  - Manageable computational requirement
  - Applicable for systems with large state dimension
  - Any system that can be simulated
  - Stochastic as well as deterministic problems

• Central Idea
  - Obtain \( \hat{J}_\infty^*(x) \equiv \) an approximation of \( J_\infty^*(x) \)
  - Solve online:
    \[
    \arg \min_{u_0} \left\{ \phi(x_0, u_0) + \hat{J}_\infty^*(f_h(x_0, u_0)) \right\}
    \]
Proposed Approach

• Obtain $\hat{J}_\infty(x)$ directly from simulation; for only relevant states
  – Simulation under a suboptimal policy yields us states vs. cost-to-go data
  – Function approximation (NN, Kernel function) to fit the cost-to-go vs. state data

• Improvement by ‘value iteration’ or ‘policy iteration’ based on Bellman equation.

• Close connection with
  – “Neuro-Dynamic Programming” (Bertsekas and Tsitsiklis)
  – AI Techniques (Reinforced Learning, Q-Learning)
General Architecture

Suboptimal Control Policy

Closed Loop Simulations

Data $x(k), u(k), \hat{J}(k)$

Simulation Part

More Simulations w/ An Updated Policy

cost-to-go function

Value Iteration

Converged

Yes

No

Bellman Equation

$$J^{i+1}(x) = \min_{u \in U} \{ p(x, u) + \tilde{J}^i(f_h) \}$$

Neural Network

$$\tilde{J}^i \equiv x \rightarrow J^i$$

Cost Approximation Part

More Simulations w/ An Updated Policy
Analogy to Chess Playing

• Have two expert players play a large number of games.
• For all the board positions encountered during the games, assign “fitness” scores based on the chance of winning starting from that position.
• A new playing policy

$$\mu(x_i) \equiv \arg\min_{u_i} \left\{ -\text{"fitness" score}(x_{i+1}(u_i,x_i)) \right\}$$

• Replace the players with the new playing policy and repeat.
Case Study I: Control of Continuous Microbial Reactor

- Continuous microbial bioreactor growing on two competing nutrients
- Multiple stable steady states
- High yield state is the desired s.s.

*Klebsiella oxytoca* growth on glucose and arabinose

Steady State bifurcation diagram (Namjoshi et al., 2001)
Model Equations

Cybernetic Modeling, Ramkrishna and coworkers

Model Equations (Kompala et al., 1986)

\[
\frac{dS_i}{dt} = D(S_{if} - S_i) - (r(v_i))Y_i c \quad i = 1, 2
\]

\[
\frac{de_i}{dt} = (r_e(u_i)) - \beta e_i - r_g e_i + r_e^* \quad i = 1, 2
\]

\[
\frac{dc}{dt} = (r_g - D)c
\]

Cybernetic Regulation Variables

\[
u_{1sc} = \frac{r_1}{r_1 + r_2}
\]

\[
v_{1sc} = \frac{r_1}{\max(r_1, r_2)}
\]

\[
r_i = \mu_{\text{max}} S_i \left[ \frac{e_i}{e_i^{\text{max}}} \right]
\]

\[
r_{e_i} = \alpha_i \frac{S_i}{K_{e_i} + S_i}
\]
Simulation Scenario

- **State**
  - $z = \begin{bmatrix} s_1 & s_2 & e_1 & e_2 & c \end{bmatrix}'$

- **Parameter**
  - Feed changes: $[s_{2f}]$

- **Augmented State**
  - $x = \begin{bmatrix} z \ s_{2f} \end{bmatrix}'$

- **Manipulated Variable**
  - $u = [D]$

- **Controlled Variable**
  - $y = [c]$

- **Objective**
  - Drive the reactor from low to high biomass state
  - Single stage cost

$$\phi(x, u) = Q\{r - x(5)\}^2 + R\{\Delta u\}^2$$
Cost-To-Go Approximation and Iteration

- Suboptimal control for simulation
  - Sequential linearization based MPC (Lee and Ricker, 1995)
  - 12 closed loop simulations (4 values of $s_{2f}$ x 3 values of $x(0)$) yielding 1200 data points
- Cost approximator
  - Multi-layer perceptron with 5 hidden nodes
- Iteration based on the Bellman equation converged in 4 iterations
  - Original max(cost-to-go) = 23.9
  - Converged max(cost-to-go) = 16.7
Results from Simulation Based Approach

• Online implementation
  – Solve one stage problem
  – Use the converged cost-to-go approximator
  – Repeat at each sample time

• NMPC v/s S-DP approach

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total Cost</th>
<th>Simulation Time* (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subopt MPC</td>
<td>22.54</td>
<td>1080.3</td>
</tr>
<tr>
<td>Sim-DP#</td>
<td>24.18</td>
<td>98.7</td>
</tr>
</tbody>
</table>

* Intel Pentium-III 800 MHz processor, running Matlab 6 R12

# Sim-DP after 4 Bellman Iterations
Analysis of Control Result

- State space plot of biomass and $s_1$ concentrations
  - Extrapolation of the NN
  - Cost-to-go approximator wrongly identifies the overshoot as optimal
What Do We Do?

• Create a ‘barrier’.
  – Doesn’t work well for stochastic problems

• Create a penalty using a Kernel function
  – Cautiousness vs. exploration

• Trigger additional simulation when a state previously unseen is reached.
Creating ‘Barrier’

- Cells are created to span the entire state space
- Each cell is identified as “covered” or “not covered”
- Neural network is fitted as before
- Cost-to-go for states within cells “not covered” → very high during the cost iteration and online calculations
## Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total Cost</th>
<th>Simulation Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suboptimal MPC</td>
<td>22.54</td>
<td>1080.3</td>
</tr>
<tr>
<td>Sim-DP (original)</td>
<td>24.18</td>
<td>98.7</td>
</tr>
<tr>
<td>Sim-DP w/ Gridding*</td>
<td>9.06</td>
<td>127.7</td>
</tr>
</tbody>
</table>

*Sim-DP after 2 Bellman Iterations*
State Space Plot (2D)
(Confining of Working State Space)

The state trajectory lies within region of data coverage.
Increasing the Region of Data Coverage

- Three states visited during the simulation with extrapolation of the NN
- Simulate with the suboptimal NMPC controller with the starting states for all four parameter values
  - 12 simulations
  - 74 data points per run
- Expanded data set

$1200 + 888 = 2088$ data points
### Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total Cost</th>
<th>Simulation Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuboptimalN MPC</td>
<td>22.54</td>
<td>1080.3</td>
</tr>
<tr>
<td>Sim-DP (original)</td>
<td>24.18</td>
<td>98.7</td>
</tr>
<tr>
<td>Sim-DP w/ more data.*</td>
<td>9.37</td>
<td>79.5</td>
</tr>
</tbody>
</table>

*Sim-DP after 4 Bellman Iterations*
State Space Plot (2D)  
(Data Expansion Scheme)

The state trajectory lies within region covered by the simulation data
Case Study II: Batch Bioreactor Optimization

- Cloned invertase production in Saccharomyces cerevisiae
- High glucose concentrations → enzyme expression repressed
- Objective
  - To find optimal feed profile $u(t)$ and optimal terminal time $t_f$

Fed-batch reactor

Glucose $s_f$ (gm/L)

Flow rate $F$ (L/s)

Volume $V(t)$, concentration $c(t)$, substrate $s(t)$, product $p(t)$
Objective

To find optimal policy $\mu(x(k))$ that maximizes the productivity and minimizes termination time $t_f$

$$\mu(x(k)) = \arg \max_{u, t_f}\left\{ pVc\big|_{t_f} - \lambda t_f \right\}$$

subject to constraints

$$V(t) \leq 1.2 \quad t \in \left[0, t_f\right]$$

$$0 \leq u(t) \leq 0.27$$
Previous Work

- Paktar et al. (1993)
  - First-order conjugate gradient method
- Chaudhuri and Modak (1998)
  - Neural network model
  - Generalized reduced gradient method
- Disadvantages
  - Final time $t_f$ needs to be fixed apriori
  - Fixed policy
  - Does not account for possible disturbances
Proposed solution: NDP

- For this problem, profit-to-go function
  \[ \tilde{J}(x_k) = \sum_{j=k+1}^{t_f} -\lambda(\Delta t_j) + [pVc]_{t_f} \]

- Start with a sub-optimal profit-to-go function

- Bellman equation

\[
J^*(x_k) = \max \left\{ pVc \mid x=x(k) \right\}, \quad \max_{u \in (0,0.27)} \left\{ \tilde{J}(x_{k+1}) - \lambda(\Delta t_k) \right\}
\]
To obtain that data, generate a feeding policy based on the policies obtained by Patkar and Chaudhuri optimizations. Suboptimal policies:

\[ b = \{0.05, 0.07, 0.1, 0.13\} \]

A total of 108 feeding policies for 3 different initial states \( V_0 = \{0.4, 0.6, 0.8\} \).
State vs. Cost-to-To Data

In total, 9328 visited states
State vs. Cost-to-To Data

Solve the bioreactor system to find $t_f^*$ for each policy

$$t_f^* = \arg \max_{t_f} \{ p.V.c|_{t_f} - \lambda \cdot t_f \}$$
State vs. Cost-to-Go Data

For each state of the followed trajectory find $J(x)$

$$J(x) = p.V.c|t_f^* - \lambda^*(t_f^*-t_x)$$
First J approximation

- Best obtained NN: Multilayer Perceptron
Bellman Iteration

- Solve iteratively

\[ J^*(x) = \max_{u \in [0, u_{\text{max}}]} \left\{ J_{NN}(f(x,u)) - \lambda \cdot \Delta t \right\} \quad \forall x \]

until

\[ \sum_{i=1}^{N} \left| J^{i+1}(x_k) - \tilde{J}^i \right| / N < 0.20 \]

| Iteration | \[ \sum_{i=1}^{N} \left| J^{i+1}(x_k) - \tilde{J}^i \right| / N \] | Profit-to-go structure |
|-----------|-------------------------------------------------|-----------------------|
| First     | \[ S|J^1 - J_{NN}^0|/9182 = 0.59 \]               | 4-17-5-1               |
| Second    | \[ S|J^2 - J_{NN}^1|/9175 = 0.21 \]               | 4-17-5-1               |
| Third     | \[ S|J^3 - J_{NN}^2|/9175 = 0.20 \]               | 4-13-5-1               |
Improving Profit Approximator

Performing Bellman iteration again with newly “visited” states to improve the profit-to-go approximator.
NDP Controller implementation

- Solve equivalent one-stage-ahead problem

\[ u_k = \arg \max_{u_k} \{ J_{NN}(f(x_k, u_k)) - \lambda(\Delta t) \} \]

- Termination criterion

\[ \max_{u \in (0, 0.27)} \left\{ \tilde{J}(f(x_k)) - \lambda(\Delta t_k) \right\} - \left[ pVc \right]_{x=x(k)} < 0 \]
**NDP Controller Performance**

The same controller was tested in two cases:

Previously simulated process \((V_o = 0.6 \text{ liters})\)

<table>
<thead>
<tr>
<th>Policy given by</th>
<th>Productivity</th>
<th>Final time (h)</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patkar et al. (1993)</td>
<td>7.30</td>
<td>12</td>
<td>3.70</td>
</tr>
<tr>
<td>Chaudhuri and Modak (1998)</td>
<td>7.10</td>
<td>12</td>
<td>3.50</td>
</tr>
<tr>
<td>Best suboptimal policy</td>
<td>7.23</td>
<td>11.7</td>
<td>3.72</td>
</tr>
<tr>
<td>NDP</td>
<td>7.25</td>
<td>11.5</td>
<td>3.80</td>
</tr>
</tbody>
</table>
NDP Controller Performance

Previously not simulated processes: The fermentation starts with a different $V_0$.

<table>
<thead>
<tr>
<th>$V_0$</th>
<th>Best suboptimal policy</th>
<th>Patkar’s policy</th>
<th>NDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>3.95</td>
<td>3.74</td>
<td>4.06</td>
</tr>
</tbody>
</table>
Control Trajectories

State space representation of the optimal trajectories followed by the fermentation process when MLP-NDP controller is used for different initial volumes: 0.5 (.), 0.6 (+), 0.7 (x)
Conclusion

• Simulation – Cost Approximation – Evolutionary Improvement for optimal control of bioreactor
  – Advantages
    ✓ Performance improvement over existing controller
    ✓ Drastic reduction in computational time for on-line calculation
    ✓ Generality (type of system, objective function, etc.)
  – Important considerations during implementation
    ✓ Avoid over-extrapolation (Reliability of cost prediction)
    ✓ Create barrier for cost iteration and online implementation
    ✓ Expand coverage using more simulations as indicated by the optimization
Conclusion

• Additional applications
  – Planning and scheduling under uncertainty
    • Deterministic TSP variants and stochastic TSPs
    – Multi-agent supply chain problem (w/ uncertainty)
    – Plantwide optimization and control

• Theoretical Issues
  – ‘Consistency’ of the cost approximator
    (nonparametric Kernel function approach)
  – Robust stability of the controller
  – Policy iteration vs. value iteration vs. TD learning
Acknowledgment

• Graduate Students
  - Niket Kaisare, Jongmin Lee

• Financial Support
  - National Science Foundation
  - Industrial support

[Logos of AspenTech, Weyerhaeuser, Owens Corning, Celanese]