Efficient Solution Algorithms for Factored MDPs
MDP

- X: finite set of states,
- A: finite set of actions,
- R: reward function
- P: transition function

\[ V_\pi(x) = R_\pi(x) + \gamma \sum_{x'} P_\pi(x'|x)V(x') \]
Optimizing the behavior of system administrator

- **X**: M machines in the network working or not,
- **A**: Decide which machine to reboot (m+1 actions)
- **R**: Money for each working machine in each time-step
- **P**: Reboot -> high probability of working, each machine can fail with a probability, failing neighbors increases the probability of failure.
Figure 1: Network topologies tested; the status of a machine is influenced by the status of its parent in the network.
Problems in MDP

• Exponential number of states
• 4 computers: $M_1, M_2, M_3, M_4$
• State space: $2^4$
Factored MDP

• Represent the large MDPs very compactly.
• Represent the set of states via a set of random variables.
• Represent the states transition model using dynamic bayesian network.
• A separate DBN model (with CPDs) for each action.
• We assume that the reward function is factored additively into a set of localized reward functions each depends on a small set of variable.
Factored MDP

1. Problem Representation

Reward is the sum of four local rewards for each machine:

- \( R(\text{working}) = 1 \)
- \( R(\text{not working}) = 0 \)

Discount factor = 0.9

Transition Model: DBN

| \( P(X_i' = t | X_i, X_{i-1}, A) \) | Action is reboot: machine \( i \) | other machine |
|---------------------------------|-----------------|--------------|
| \( X_{i-1} = f \land X_i = f \) | 1                | 0.0238       |
| \( X_{i-1} = f \land X_i = t \) | 1                | 0.475        |
| \( X_{i-1} = t \land X_i = f \) | 1                | 0.0475       |
| \( X_{i-1} = t \land X_i = t \) | 1                | 0.95         |
Solving MDP-Policy Iteration

\[
V_{\pi}^t(x) = R_\pi(x)(x) + \gamma \sum_{x'} P_\pi(x)(x' | x)V_{\pi}^t(x')
\]

\[
\pi^{t+1}(x) = \arg \max_a [R_a(x) + \gamma \sum_{x'} P_a(x' | x)V_{\pi}^t(x')]
\]

- Polynomial in the number of states N
- Exponential in the number of variables K
It’s all about the value function
“Approximation Idea”

Linear Value Function

\[ V(x) = \sum_{j=1}^{k} w_j h_j(x) \]
Factored MDP-Linear Value Function

1. Problem Representation

Basis functions:
\( h_i(X_i=\text{true})=1 \)
\( h_i(X_i=\text{false})=0 \)
\( h_0=1 \)
Solving MDP-Linear Programming

**Variables:** $V_1, \ldots, V_N$

**Minimize:** $\sum_{x_i} \alpha(x_i)V_i, \forall i : \alpha(x_i) > 0$

**Subject to:** $V_i \geq R_a(x) + \gamma \sum_j P_a(x_j \mid x_i)V_j, \forall x_i, a$

- Polynomial in the number of states $N$
- Exponential in the number of variables $K$
Approximation

Variables: $V_1, \ldots, V_N$

Minimize: $\sum_{x_i} \alpha(x_i)V_i, \forall i : \alpha(x_i) > 0$

Subject to: $V_i \geq R_a(x) + \gamma \sum_j P_a(x_j | x_i)V_j, \forall x_i, \alpha$

Variables: $w_1, \ldots, w_k$

Minimize: $\sum_x \alpha(x) \sum_{i=1}^k w_i h_i(x), \forall i : \alpha(x) > 0$

Subject to: $\sum_i w_i h_i(x) \geq R_a(x) + \gamma \sum_x P_a(x | x) \sum_i w_i h_i(x'), \forall x, \alpha$
Objective Function

Variables: $w_1, \ldots, w_K$

Minimize: $\sum_x \alpha(x) \sum_i w_i h_i(x), \forall i : \alpha(x_i) > 0$

Subject to: $\sum_i w_i h_i(x) \geq R_a(x) + \gamma \sum_x P_a(x | x) \sum_i w_i h_i(x'), \forall x, a$

Objective function polynomial in the number of basis functions

$$\sum_x \alpha(x) \sum_i w_i h_i(x) =$$

$$\sum_i \alpha(x) \sum x w_i h_i(x) =$$

$$\sum_i \sum_{x} \alpha(x) h_i(x) =$$

$$\sum_i \sum_{x} \alpha(c_i) h_i(c_i),$$

where $\alpha(c_i) = \sum \alpha(x)$
Each Constraint: Backprojection

Variables: $w_1, \ldots, w_k$

Minimize: $\sum_{x} \alpha(x) \sum_{i} w_i h_i(x), \forall i: \alpha(x_i) > 0$

Subject to: $\sum_{i} w_i h_i(x) \geq R_a(x) + \gamma \sum_{x'} P_a(x' \mid x) \sum_{i} w_i h_i(x'), \forall x, a$

$$\sum_{x'} P_a(x' \mid x) \sum_{i} w_i h_i(x') = \sum_{i} w_i \sum_{x'} P_a(x' \mid x) h_i(x')$$
Each Constraint: Backprojection

\[ \text{Variables: } w_1, \ldots, w_k \]

\[ \text{Minimize: } \sum_x \alpha(x) \sum_i w_i h_i(x), \forall i : \alpha(x_i) > 0 \]

\[ \text{Subject to: } \sum_i w_i h_i(x) \geq R_a(x) + \gamma \sum_x P_a(x' \mid x) \sum_i w_i h_i(x'), \forall x, a \]

\[ \sum_x P_a(x' \mid x) \sum_i w_i h_i(x') = \sum_i w_i \sum_{x'} P_a(x' \mid x) h_i(x') \]
The Constraint

Variables: $w_1, \ldots, w_k$

Minimize: $\sum_x \alpha(x) \sum_i w_i h_i(x), \forall i : \alpha(x_i) > 0$

Subject to: $\sum w_i h_i(x) \geq R_a(x) + \gamma \sum P_a(x'|x) \sum w_i h_i(x'), \forall x, a$

$\sum w_i h_i(x) \geq R_a(x) + \gamma \sum P_a(x'|x) \sum w_i h_i(x'), \forall x, a \iff 0 \geq \sum w_i [\gamma \sum P_a(x'|x) h_i(x') - h_i(x)] + R_a(x), \forall x, a \iff 0 \geq \max_x \sum w_i [\gamma \sum P_a(x'|x) h_i(x') - h_i(x)] + R_a(x), \forall a$
All the parts has restricted domain

\[ 0 \geq \max_x \sum_i w_i \left[ \gamma \sum_{x'} P_a(x' | x) h_i(x') - h_i(x) \right] + R_a(x), \forall a \]

\[ \max_x \sum_i w_i [\gamma g_i^q(x) - h_i(x)] + R_a(x) \]

\[ \max_x \sum_i w_i f_i(x) + \sum_j r_j(x) \]
Variable Elimination

Our goal is to compute
\[
\max_{x_1, \ldots, x_n} \sum_j f_j(x[Z_j]).
\]

The main idea is that, rather than summing all functions and then doing the maximization, we maximize over variables one at a time. When maximizing over \(x_l\), only summands involving \(x_l\) participate in the maximization.
Variable Elimination

\[ \max_x \sum_i w_i f_i(x) + \sum_j r_j(x) \]

\[ \max_{x_1, x_2, x_3, x_4} w_1 f_1(x_1, x_2) + w_2 f_2(x_1, x_3) + r_1(x_2, x_4) + r_2(x_3, x_4) \]

\[ \max_{x_1, x_2, x_3} \left[ w_1 f_1(x_1, x_2) + w_2 f_2(x_1, x_3) + \max_{x_4} [r_1(x_2, x_4) + r_2(x_3, x_4)] \right] \]

\[ \max_{x_1, x_2, x_3} \left[ w_1 f_1(x_1, x_2) + w_2 f_2(x_1, x_3) + e_1(x_2, x_3) \right] \]

where \( e_1(x_2, x_3) = \max_{x_4} [r_1(x_2, x_4) + r_2(x_3, x_4)] \)
Variable Elimination

\[
\max_x \sum_i w_i f_i(x) + \sum_j r_j(x)
\]

\[
\max_{x_1, x_2, x_3, x_4} \quad w_1 f_1(x_1, x_2) + w_2 f_2(x_1, x_3) + r_1(x_2, x_4) + r_2(x_3, x_4)
\]

\[
\max_{x_1, x_2, x_3} \quad [w_1 f_1(x_1, x_2) + w_2 f_2(x_1, x_3) + \max_{x_4} [r_1(x_2, x_4) + r_2(x_3, x_4)]]
\]

\[
\max_{x_1, x_2, x_3} \quad [w_1 f_1(x_1, x_2) + w_2 f_2(x_1, x_3) + e_1(x_2, x_3)]
\]

where \( e_1(x_2, x_3) = \max_{x_4} [r_1(x_2, x_4) + r_2(x_3, x_4)] \)
The whole process

1. Problem Representation

2. Select Basis Functions

3. Backprojection

4. LP Construction

Solve the LP with VE

Basis functions:
\[ h_i(X_i = \text{true}) = 1 \]
\[ h_i(X_i = \text{false}) = 0 \]
\[ h_0 = 1 \]

\[
\max_x \sum_i w_f(x) + \sum_j r_j(x)
\]

\[
\sum_i w_i \sum_x P_a(x | x') h_i(x')
\]
Scalable MDPs

Number of LP constraints

- Explicit LP
- Factored LP

Number of machines in ring

# explicit constraints = \((n+1) 2^n\)

# factored constraints = \(12n^2 + 5n - 8\)
Take Home Message

• Compact representation can be exploited to solve MDPs with exponentially many states efficiently.
• Still NP-complete in the worst case.
• Factored solution may increase the size of LP when the number of states is small (but it scales better).
• **Success** depends on the choice of the **basis functions** for value approximation and the **factored decomposition** of rewards and transition probabilities.
Scalability is still an issue
Please formulate the 4 machine problem instances of SysAdmin Example with LP
The end