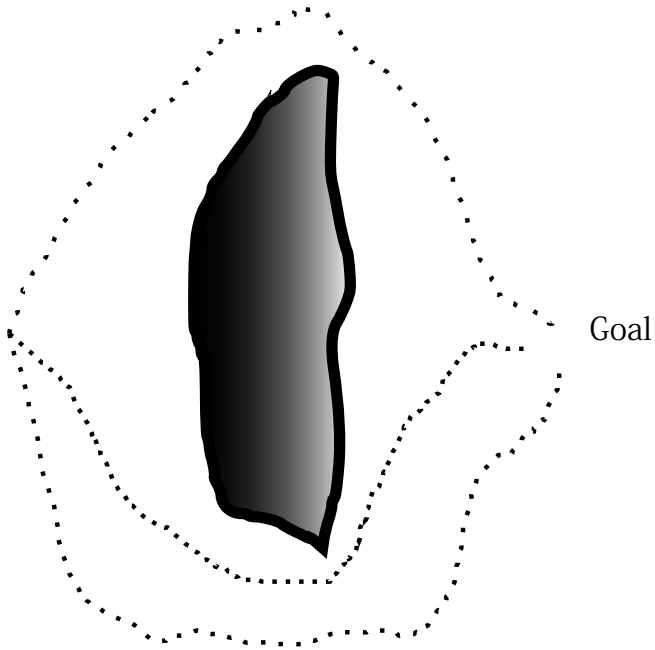
Reinforcement learning from an optimization viewpoint

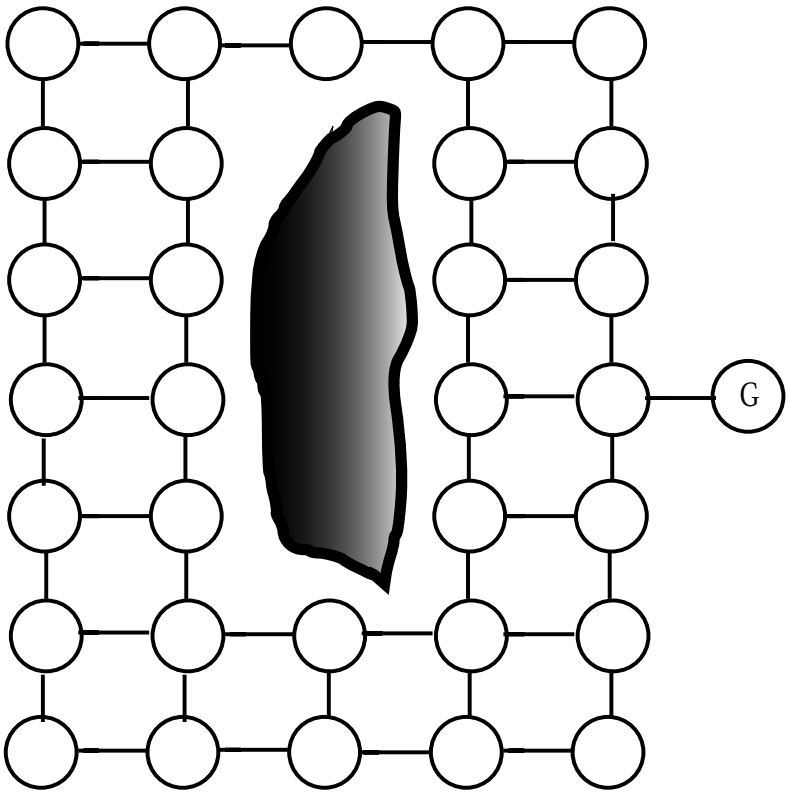
Torkel Glad

Example 1. Path planning



- Find a path that reaches the goal as quickly as possible
- Avoid the obstacle

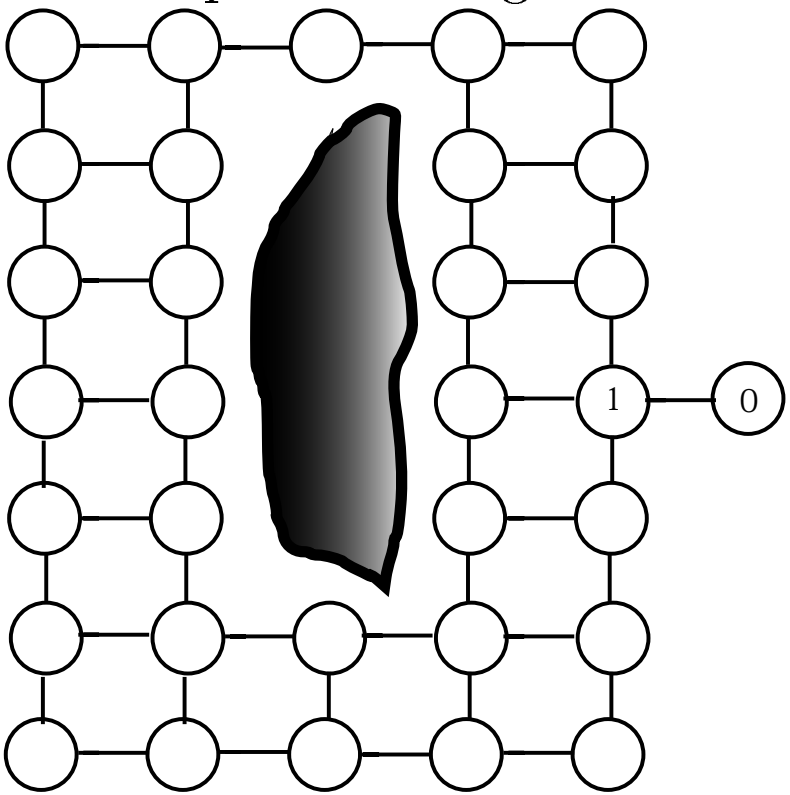
Path planning: formal definition



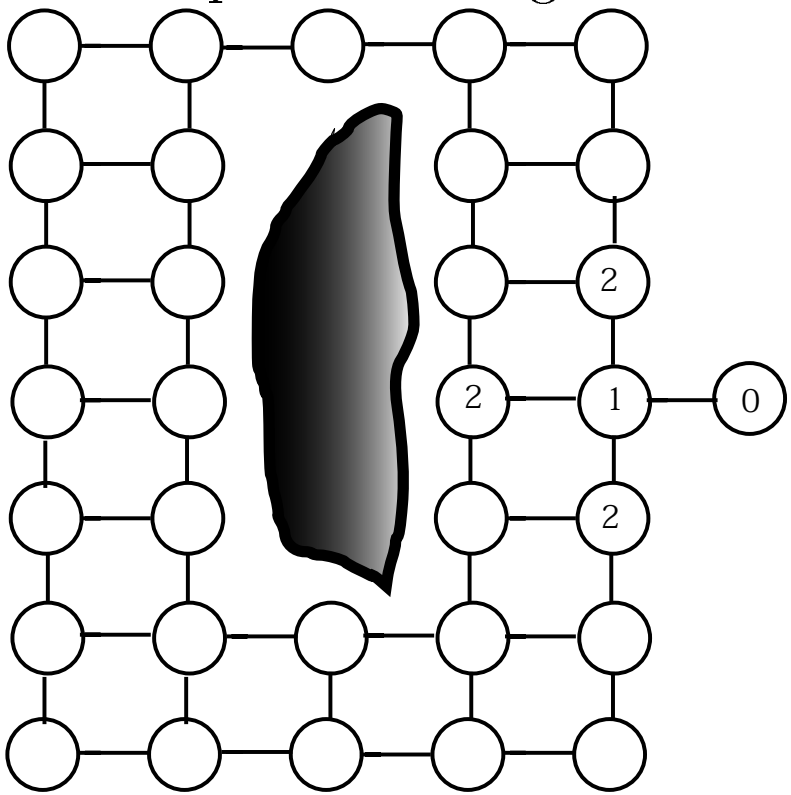
- reach the goeal (G) in shortest time
- possible to move north, south, east or west (actions, control)
- finite number of positions (= state space)
- each step takes one unit of time

Path planning: time to reach the goal

One step from the goal

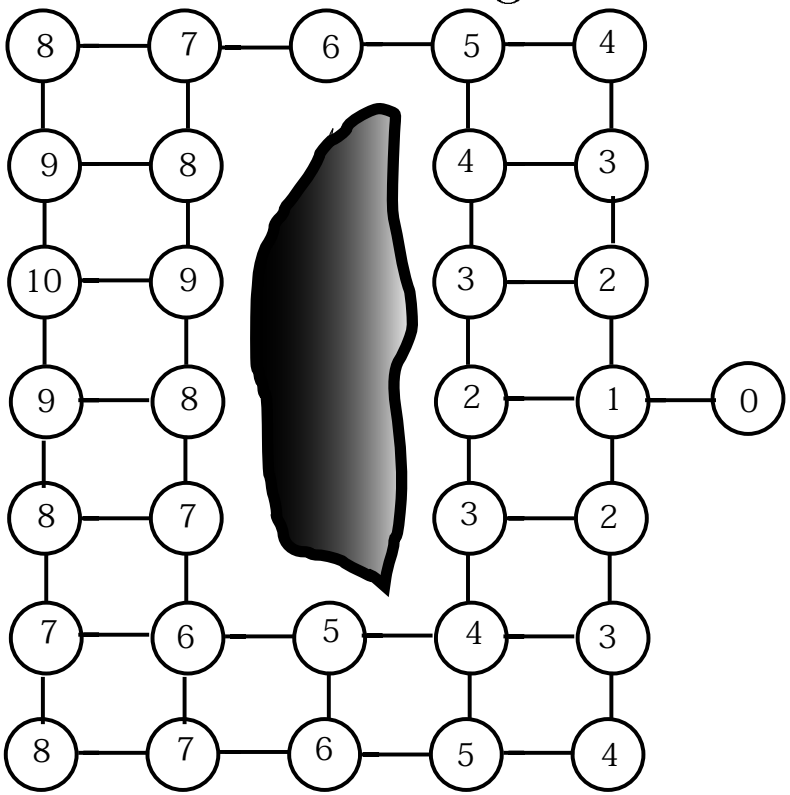


Two steps from the goal



Path planning: the solution

Time to reach the goal



- The numbers in the circles define the *optimal return function* V.
- The optimal action (control) is the step that takes you to the position (state) with the lowest value.

Mathematical description

What we have done is the following:

$$V(\text{step n}) = \min((\text{cost of one step}) + V(\text{step n+1}))$$

The minimum is taken over all actions (controls)
This is the *Bellman equation*

- We obtained V using a "backward sweep".
- Bellman called this *dynamic programming*

A brute force approach

If the path planning involves e.g. an autonomous vehicle one could do the following:

- Program a strategy for choosing direction to move
- Let the vehicle loose
- If it reaches the goal, note the time (V-value)
- Repeat for difeerent strategies until "sufficiently good" V-values are obtained.

This is a model-free approach! (Almost)

Example 2. A chess match

Outcomes in each game.

- $1-0$. Our player wins.
- $0-1$. Opponent wins.
- $1/2-1/2$. Draw. (undecided game)

Rules of the match.

- Two games.
- 1-1 after two games $\Rightarrow$ sudden death (first won game decides)

Playing modes

Our player can play in two ways:

- Timid play: 90% draw, 10% loss
- Bold play: 45% win, 55% loss

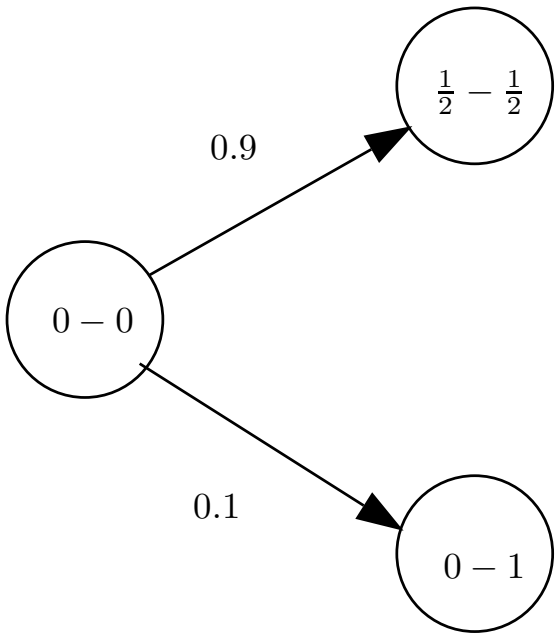
The average number of match points are the following:

$$\begin{aligned} \text{Timid}: &\quad 0.9 \times 0.5 + 0.1 \times 0 = 0.45 \\ \text{Bold}: &\quad 0.45 \times 1 + 0.55 \times 0 = 0.45 \end{aligned}$$

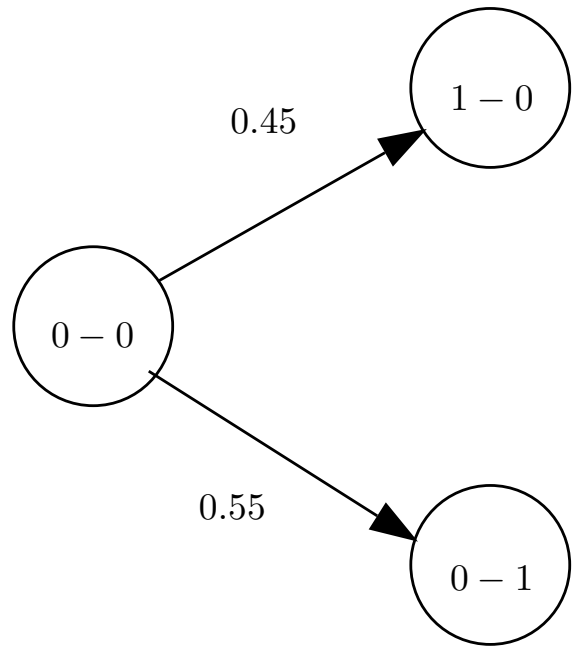
So on the average the two strategies are equally good.

The chess match. Game 1

Timid play



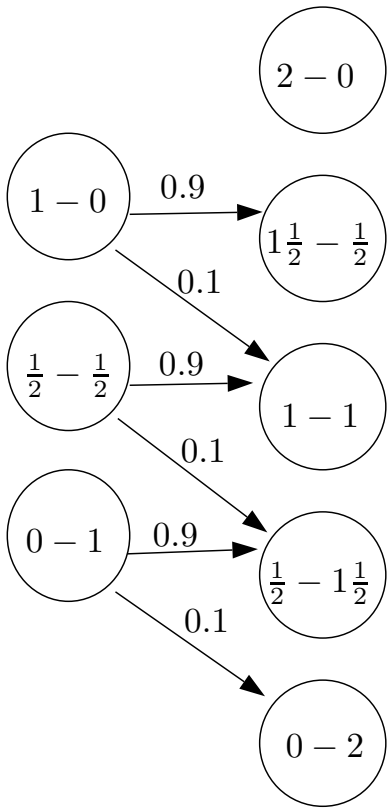
Bold play



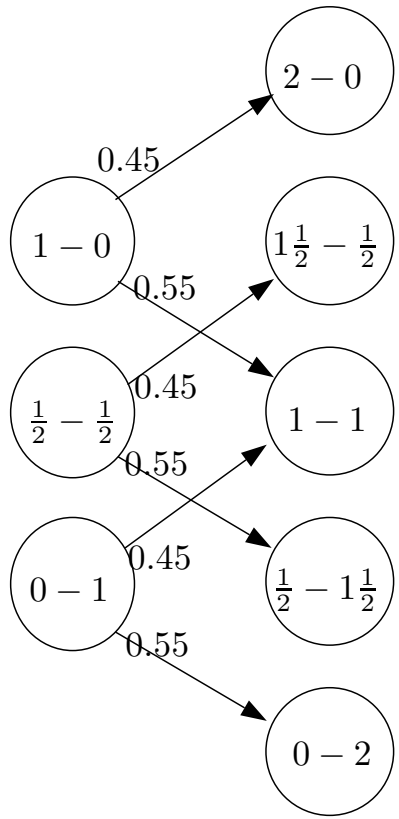
The numbers at the arrows denote probabilities of the outcome.

The chess match. Game 2

Timid play



Bold play



Features of the chess example

- There is a discrete time scale: $t = 0$ (before first match), $t = 1$ (after first match), $t = 2$ (after second match)
- For each time there are several *states*
- For each state there are several choices of action (timid or bold). *control signals*
- For a given state and control the outcome is not determined. Only the *probability* is known.
- There is a numerical measure of success: the probability of winning the match.

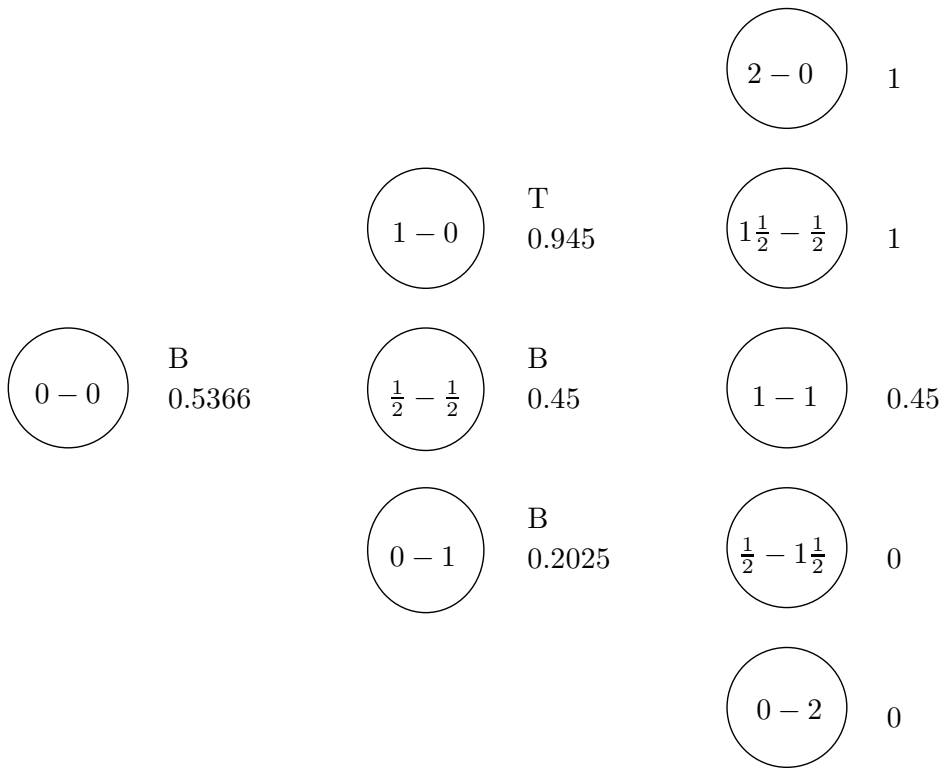
Solving the chess problem

Let V denote the probability of winning. The backward sweep then involves sweeping the probabilities backwards.

This can be interpreted as a stochastic version of Bellmant's equation.

Chess. The solution

The probabilities of winning are calculated from right to left.



Properties of the solution

- The control (decision) is a function of the state, i.e. the optimal control is of **feedback** type.
- A predetermined set of controls is called an **open loop** strategy. They all give lower probability of winning.
- The technique can easily be extended to a longer time horizon (more games)

Example 3. Storage

A storage process (water tank, inventory)

$$x_{k+1} = x_k + u_k, \quad |u_k| \leq 1$$

x_k: amount at time k (defined so that $x_k = 0$ is the desired value)
u_k: amount added at time k
x_0 given; choose u_0, u_1 so that $V = x_2^2$ is minimized.

Features of the inventory example

- There is a discrete time scale.
- For each time there is a state variable x_k which is a real number.
- For each state there are choices of action, given by the control variable u_k, which is also a real number.
- For a given state and control the outcome is completely determined.
- There is a numerical measure of success: x_2^2.

Different classes of problems

- The state space can either be discrete (the path problem, the chess problem) or continuous (the inventory)
- The same is true for the control space
- The problem can either be stochastic (chess) or deterministic (path planning, inventory)

A stochastic version of the inventory example

$$x_{k+1} = x_k + u_k + v_k$$

where v_k is a stochastic variable.

Minimize Ex_2^2, where E denotes expectation.

The general deterministic problem

A dynamic system

$$x_{k+1} = f(x_k, u_k), \;\; u_k \in U_k, \quad k = 0, 1, \ldots, N-1$$

x_k, u_k discrete or continuous. x_k "state", u_k "control"
A cost function to be minimized

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$$

Many extensions, e.g.
stochastic systems (minimize expected cost)
infinite time horizon
continuous time

Solving the storage problem

For the storage process the cost function $V(x)$ has to be swept backwards.

Calculating V backwards

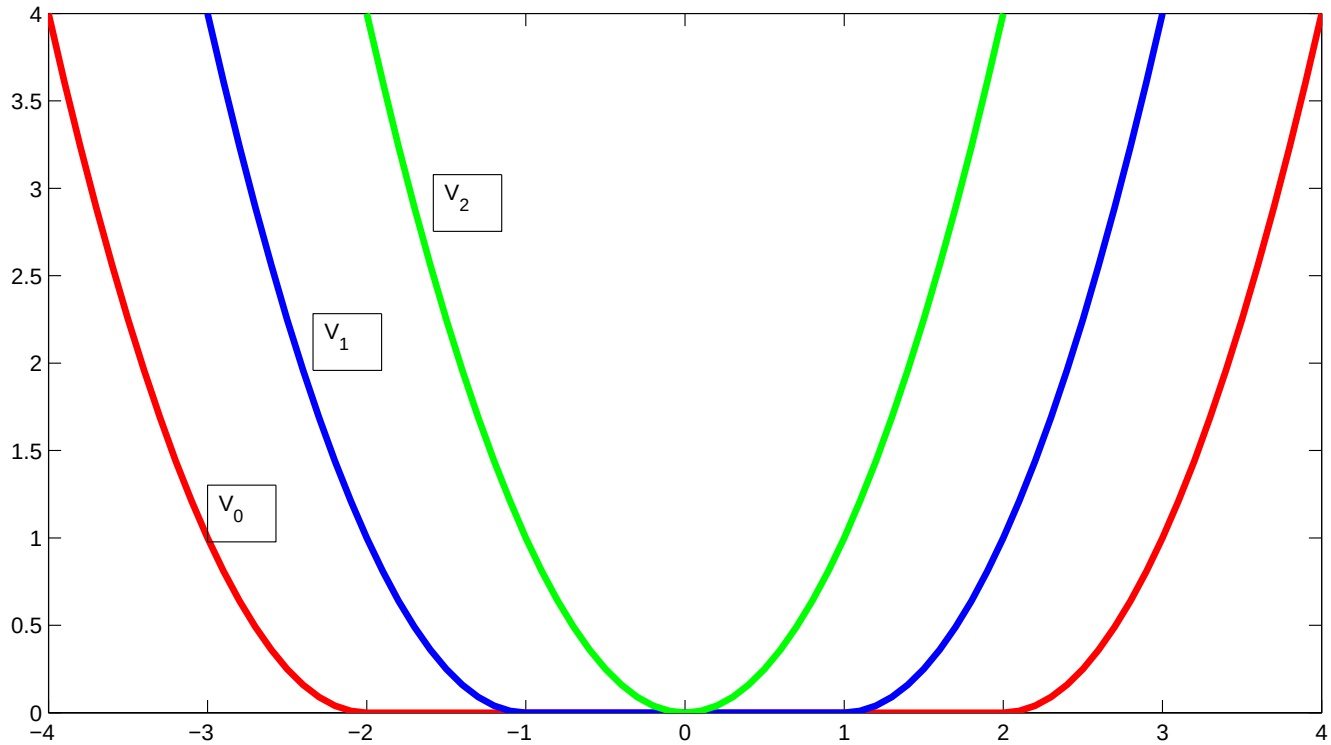
$$V_1(x_1) = \min_{|u_1| \le 1} (x_1 + u_1)^2 \qquad V_0(x_0) = \min_{|u_0| \le 1} V_1(x_0 + u_0)$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$u_1 = \begin{cases} 1 & x_1 < -1 \\ -x_1 & |x_1| \le 1 \\ -1 & x_1 > 1 \end{cases} \qquad u_0 = \begin{cases} 1 & x_0 < -1 \\ -x_0 & |x_0| \le 1 \\ -1 & x_0 > 1 \end{cases}$$

$$V_1(x_1) = \begin{cases} (x_1+1)^2 & x_1 < -1 \\ 0 & |x_1| \le 1 \\ (x_1-1)^2 & x_1 > 1 \end{cases} \qquad V_0(x_0) = \begin{cases} (x_0+2)^2 & x_0 < -2 \\ 0 & |x_0| \le 2 \\ (x_0-2)^2 & x_0 > 1 \end{cases}$$

The cost function V



Properties of the solution

- The method is easily extended to a longer time horizon.
- The computed control is of feedback type.
- For a *given starting point* it can be brought into open-loop form. The two forms are then mathematically equivalent.

The general dynamic programming algorithm

The method of the examples can be generalized to a general algorithm, the dynamic programming algorithm:

$$V_N(x_N) = g_N(x_N)$$
$$V_k(x_k) = \min_{u_k \in U_k} \left(g_k(x_k, u_k) + V_{k+1}(f_k(x_k, u_k)) \right)$$

where V_k is the optimal cost at time k.
In the stochastic case the expectation of the right hand side has to be taken.

Some general properties

- For the deterministic case there is no difference between open loop and feedback control.
- If in addition the state space is discrete, then the dynamic programming problem is equivalent to a shortest path problem.

Computational issues

- In general the dynamic programming algorithm tends to give computations with a high complexity.
- Many suboptimal approximations. E.g. compute V only in a neighborhood of the optimal solution.
  - Differential dynamic programming.
  - State increment dynamic programming.
  - MPC (Model Predictice Control) is a technique that uses open loop calculations to generate closed loop solutions.

The linear quadratic case

For a linear system

$$x_{k+1} = Ax_k + Bu_k$$

with quadratic cost

$$\text{minimize } x_N^T Q_N x_N + \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k$$

the dynamic programming algorithm becomes simple, because V has the simple form

$$V_k = x_k^T S_k x_k$$

The linear quadratic case, cont'd

The matrix S_k is determined by the recursion formula

$$S_k = Q + A^T(S_{k+1} - S_{k+1}B(R + B^T S_{k+1}B)^{-1}B^T S_{k+1})A$$
$$S_N = Q_N$$

and the optimal control becomes

$$u_k = -(R + B^T S_{k+1}B)^{-1}B^T S_{k+1}Ax_k$$

Measurement noise

- Dynamic programming assumes perfect information about the state x_k.
- In many application x_k can only be inferred indirectly from noise corrupted measurements y_k.
- Theoretical and conceptual solution: Let the state at time k be every measurement that has been made up to time k.
- Practical solution: Suboptimal schemes, e.g. *Certainty Equivalence*: Compute an estimate, $\hat{x}_k$, of x_k. Choose the control as if $\hat{x}_k$ was the true value.
- In the linear quadratic case, estimating $\hat{x}$ using a Kalman filter actually gives the optimal solution.

Conclusion

We have looked at problems that can be attacked using reinforcement learning, viewing them from a control perspective.

What do reinforcement learning people do about this?

- Use the Bellman equation
- Use somewhat different terminology and notation.
- Put much more emphasis on stochastics.
- Try to be "model-free".
- Try to find a smart middle way between analytical methods and brute force.

Thank you

www.liu.se

LINKÖPING UNIVERSITY