Lecture 21

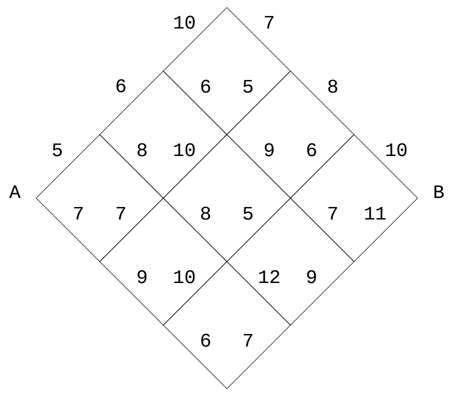
Dynamic Programming

Karl Henrik Johansson

Dynamic programming is a method to solve optimal control problems. Here we introduce the notion by discussing dynamic programming for a combinatorial problem and dynamic programming for continuous-time systems.

1 Discrete-Time Systems

Consider the problem of traveling by minimum cost from point A to point B in the following graph:



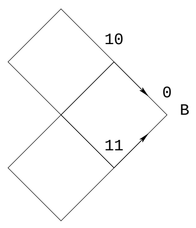
The weights on the edges denote the cost for taking a particular way between two vertices. We assume that it is only possible to move north-east and south-east in the graph.

One way of solving this problem is, of course, to calculate the total cost for all possible paths going from A and B. A more efficient way (in particular, if the number of vertices is large) is to use dynamic programming. To illustrate the idea, let us introduce some notation. Let $x(t) \in \{1, \dots, 4\} \times \{1, \dots, 4\}$ denote the state at (discrete) time $t \in \{0, \dots, 6\}$, where each time step corresponds to moving one step to the right in the graph. The vertices are labeled after the rows (pointing south-east) and the columns (pointing north-east) in the graph, such that the initial state A is $x(0) = (1, 1)$ and the final state B is $x(6) = (4, 4)$. At each time $t \in \{0, \dots, 5\}$, a control action $u(t) \in \{\pm 1\}$ is to be chosen, which decides how next step should be taken ($u = +1$ for north-east and $u = -1$ for south-east). Note that for some states the control is constrained, for example, for $x(4) = (4, 1)$ only $u(4) = +1$ is possible.

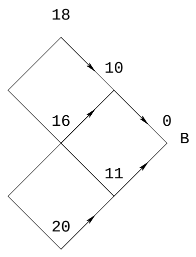
Define the *cost function* $J(u(\cdot)) = \sum_{k=0}^{5} W(x(k), x(k+1))$ for a path from A to B, where $W(x(k), x(k+1))$ is the weight on the edge $(x(k), x(k+1))$.[1] Let the *cost-to-go function* $J^*(x(t), t)$ denote the optimal (minimum) cost function starting in $x(t)$ at time t, i.e.,

$$J^*(x(t), t) = \min_{u(t),\ldots,u(5)} J(u(\cdot)).$$

The dynamic programming solution to the traveling problem is based on deriving the cost-to-go function for all states, starting with the final state and then going backwards until the initial state is reached. We do the bookkeeping by specifying the value $J^*(x(t), t)$ at the vertex corresponding to $(x(t), t)$. At the final state B, we obviously have $J^*((4, 4), 6) = 0$. At $(3, 4)$, we have $J^*((3, 4), 5) = J^*((4, 4), 6) + W((3, 4), (4, 4)) = 0 + 11 = 11$ and similarly we have $J^*((4, 3), 5) = 10$, as indicated in the following detail of the graph:

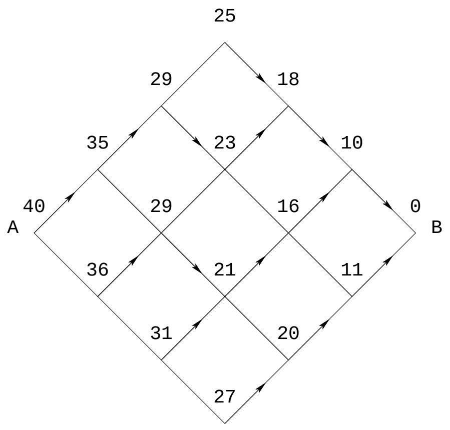


The arrows indicate the optimal control, i.e., the optimal action to take at a certain state. For $(3, 4)$ and $(4, 3)$ the optimal control is, of course, the trivial choices shown above. Deriving $J^*(\cdot, 4)$ yields the following detail of the graph:

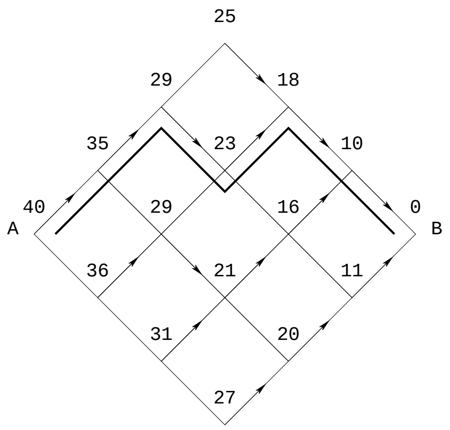


Continuing like this gives the graph

[1] Here $x(\cdot)$ denotes a function in contrast to a single point, $x(k)$, say. Note also that, of course, $x(\cdot)$ depends on $u(\cdot)$ through the dynamics imposed by the graph.



From this, we may easily depict the optimal solution going from A to B by just following the arrows:



This is the dynamic programming solution to the graph traveling problem.

In general, we have

$$J^*(x(t), t) = J^*(x(t+1), t+1) + W(x(t), x(t+1)).$$

Hence, the cost-to-go function at time t depends only on the cost-to-go function at time $t+1$ and the optimal cost going from $x(t)$ to $x(t+1)$. This is a simple, but fundamental, fact, which is called the *principle of optimality*. The principle of optimality states that if

$$u(i), u(i+1), \ldots, u(j), \ldots, u(k)$$

gives the optimal solution

$$x(i), x(i+1), \ldots, x(j), \ldots, x(k),$$

then the truncated control $u(j), \ldots, u(k)$ gives the optimal solution from $x(j)$. It thus follows that it is not necessary to solve the whole optimal control at once, but instead we may solve individual pieces going backwards from the final state and then patch these pieces together, as was done in the example.

Comparing dynamic programming with the approach of deriving all possible paths from A and B, we see that the dynamic programming leads to the calculation of 15 numbers while there are 20 possible paths. In general, for a graph with $N \times N$ vertices, we have $N^2 - 1$ cost-to-go functions to calculated compared to $(2(N-1))!/((N-1)!)^2$ paths. For example, for $N = 8$, we get 63 compared to 3432.

Dynamic programming automatically leads to a feedback solution: to each state (vertex) there is a dedicated control action (arrow). This means that the solution is robust, for instance, if an impulse disturbance moves the state from one location to another, still the optimal control is applied.

Dynamic programming is particularly efficient in multi-stage decision problems, when there are few control choices at each stage. In the traveling problem, we have only two choices at each time step. If the dimensions of the control space and the state space are large, dynamic programming may be less appealing. The phrase "curse of dimensionality" was given to the problem of exponential growth of a hyper-volume as a function of dimensionality

Dynamic programming may also be applied to discrete-time systems specified as difference equations. Next, however, we consider the continuous-time set-up.

2 Continuous-Time Systems

Consider the continuous-time control system

$$\dot{x}(t) = f(x(t), u(t), t), \qquad x(t_0) = x_0, \tag{1}$$

where $t \in [t_0, t_f]$, $x(t) \in \mathbb{R}^n$, and $u(t) \in \mathbb{R}$. Assume that f is smooth in all its arguments and that u is piecewise continuous. Define the *cost function*

$$J(u(\cdot)) = \int_{t_0}^{t_f} L(x(s), u(s), s)\, ds + \phi(x(t_f), t_f), \tag{2}$$

where the *running cost* L and the *terminal cost* ϕ are smooth functions. Both the initial time t_0 and the final time $t_f > t_0$ are assumed to be fixed.

The optimal control problem is to minimize J with respect to $u : [t_0, t_f] \to \mathbb{R}$ subject to (1). The optimal control is denoted $u^* : [t_0, t_f] \to \mathbb{R}$ and the corresponding trajectory $x^* : [t_0, t_f] \to \mathbb{R}^n$. The *cost-to-go function* is defined as the minimum cost to go from any state $x \in \mathbb{R}^n$ at time $t \in [t_0, t_f]$ to $x(t_f)$ and is given by

$$J^*(x, t) = \min_{u(\cdot)} \int_{t}^{t_f} L(x(s), u(s), s)\, ds + \phi(x(t_f), t_f).$$

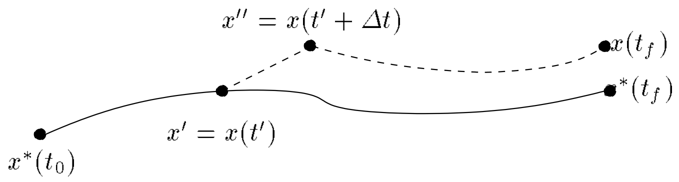
The cost-to-go function at $t = t_0$ is hence the value of the cost function for the optimal control, i.e.,

$$J^*(x^*(t_0), t_0) = J(u^*(\cdot)).$$

The *principle of optimality* for continuous-time control systems reads as follows. Let x^* and u^* be the optimal trajectory and optimal control as defined above. Assume $\bar{x}_0 = x^*(\bar{t}_0)$ for some $\bar{t}_0 \in (t_0, t_f)$. Then, the optimal control for (1) with respect to (2), where t_0 is replaced by $\bar{t}_0$ and x_0 by $\bar{x}_0$, is given by the truncated control $\bar{u}^* : [\bar{t}_0, t_f] \to \mathbb{R}$ with $\bar{u}^*(t) = u^*(t)$ for all $t \in [\bar{t}_0, t_f]$. This means, similar to the discrete case, that the optimal control problem can be solved by going backwards from the final state, by first solving the optimal control problem on some interval $[t_f - \epsilon, t_f]$, then on $[t_f - 2\epsilon, t_f - \epsilon]$ etc, and finally patching the solutions together to get the solution on $[t_0, t_f]$.

Hamilton-Jacobi-Bellman

Using the principle of optimality, we do a heuristic derivation of the Hamilton-Jacobi-Bellman equation. Consider an optimal trajectory going from $x^*(t_0)$ to $x^*(t_f)$, shown as a solid line the following figure:



Consider a non-optimal trajectory deviating from the optimal only between $x(t') = x'$ and $x(t' + \Delta t) = x''$, and from x'' following an optimal trajectory to the final state at $t = t_f$. The deviation is shown with a dashed line above.

Denote the cost for starting in x' at t' and passing through x'' by $J'(x', t')$. Clearly,

$$J'(x', t') \geq J^*(x', t')$$

with equality only for $\Delta t = 0$, i.e., only if $u(t) = u^*(t)$ for all $t \in (t', t_f)$. If $\Delta t > 0$ is small, then

$$J'(x', t') \approx J^*(x', t' + \Delta t) + L(x', u', t')\Delta t,$$

where we used the notation $u' = u(t')$. Hence, assuming that the following minima are attained, we get

$$\begin{aligned} J^*(x', t') &= \min_u \left[J^*(x'', t' + \Delta t) + L(x', u', t')\Delta t \right] \\ &= \min_u \left[J^*(x' + f(x', u', t')\Delta t, t' + \Delta t) + L(x', u', t')\Delta t \right] \\ &= \min_u \Bigg[J^*(x', t') + \frac{\partial J^*}{\partial x}(x', t') f(x', u', t')\Delta t \\ &\qquad + \frac{\partial J^*}{\partial t}(x', t')\Delta t + L(x', u', t')\Delta t \Bigg]. \end{aligned}$$

Note that the minima are taken pointwise, i.e., $u \in \mathbb{R}$. This yields the Hamilton-Jacobi-Bellman equation

$$-\frac{\partial J^*}{\partial t}(x,t) = \min_u \left[\frac{\partial J^*}{\partial x}(x,t)f(x,u,t) + L(x,u,t)\right], \quad J^*(x,t_f) = \phi(x,t_f).$$

This is a partial differential equation with boundary condition given by the terminal cost. Under certain conditions, the solution of this equation corresponds to the optimal cost. The optimal control is equal to

$$u^*(t) = \arg\min_u \left[\frac{\partial J^*}{\partial x}(x,t)f(x,u,t) + L(x,u,t)\right].$$

Note that this pointwise minimization gives a feedback control law: given the function J^*, the control is derived from the current state $x(t)$.

We rephrase the Hamilton-Jacobi-Bellman equation in a slightly more general setting in the following theorem. Here the final time t_f is a variable and $x(t_f)$ is constrained by the terminal condition $\psi(x(t_f), t_f) = 0$, where ψ is a smooth function.

Theorem 1 (Hamilton-Jacobi-Bellman Equation). *Assume the C^1 function V satisfies the Hamilton-Jacobi-Bellman equation*

$$-\frac{\partial V}{\partial t}(x,t) = \min_u \left[\frac{\partial V}{\partial x}(x,t)f(x,u,t) + L(x,u,t)\right], \quad \forall x, t$$

and

$$V(x,t_f) = \phi(x,t_f), \quad \forall x \in \{z : \psi(z,t_f) = 0\}.$$

Also, assume the minimum is attained for all x and t for some $\bar{u} : \mathbb{R}^n \times [t_0, t_f] \to \mathbb{R}$, which is piecewise C^0 in t. Finally, assume that the solution to

$$\dot{x}(t) = f(x(t), \bar{u}(x(t),t), t)$$

is unique for all initial states. Then, V is the unique solution of the Hamilton-Jacobi-Bellman equation and it is equal to the optimal cost-to-go function, i.e.,

$$V(x,t) = J^*(x,t), \quad \forall x, t.$$

Furthermore, the optimal control is given by

$$u^*(t) = \bar{u}(x(t),t) = \arg\min_u \left[\frac{\partial V}{\partial x}(x(t),t)f(x(t),u,t) + L(x(t),u,t)\right].$$

Linear Quadratic Control

Consider the optimal control problem given by the linear time-invariant system

$$\dot{x}(t) = Ax(t) + Bu(t)$$

and cost function

$$J(u(\cdot)) = \int_{t_0}^{t_f} \left[x(t)^T Q x(t) + \rho u(t)^2\right] dt + x(t_f)^T Q_f x(t_f),$$

where t_0 and t_f are fixed, Q and Q_f are positive semidefinite matrices, and $\rho > 0$. The Hamilton-Jacobi-Bellman equation is then equal to

$$-\frac{\partial V}{\partial t}(x,t) = \min_u \left[\frac{\partial V}{\partial x}(x,t)(Ax+Bu) + x^T Q x + \rho u^2\right]$$

with boundary condition

$$V(x,t_f) = x^T Q_f x.$$

Let us try a solution of the form $V(x,t) = x^T P(t) x$ with $P(t) = P(t)^T$. Then,

$$\frac{\partial V}{\partial t} = x^T \dot{P}(t) x, \quad \frac{\partial V}{\partial x} = 2x^T P(t),$$

which gives

$$-x^T \dot{P}(t) x = \min_u \left[2x^T P(t) A x + 2x^T P(t) B u + x^T Q x + \rho u^2\right].$$

The minimum is attained for $u = -\rho^{-1} B^T P(t) x$. Substituting this into the Hamilton-Jacobi-Bellman equation gives

$$-x^T \dot{P}(t) x = 2x^T P(t) A x - 2\rho^{-1} x^T P(t) B B^T P(t) x + x^T Q x + x^T P(t) B \rho^{-1} B^T P(t) x,$$

so $P(t)$ must satisfy the continuous-time Riccati equation

$$\dot{P}(t) = -P(t)A - A^T P(t) + P(t) B \rho^{-1} B^T P(t) - Q, \quad P(t_f) = Q_f.$$

Solving this matrix differential equation thus gives the optimal linear quadratic control as

$$u^*(t) = -\rho^{-1} B^T P(t) x(t),$$

which is a time-varying linear state-feedback control law. Note that it follows from Theorem 1 that this control is unique, so there exist no nonlinear controller that performs equally well or better.

Background

The term *dynamic programming* was coined by Bellman [1]. There are several good textbooks on optimal control and dynamic programming, e.g., [2–5]. The example in Section 1 is from [3].

References

1. R. Bellman. *Dynamic Prograggmin.* Princeton University Press, Princeton, NJ, 1957.
2. D. P. Bertsekas. *Dynamic Programming and Optimal Control: Volumes I–II.* Athena Scientific, Belmont, MA, 1995.
3. A. E. Bryson and Y.-C. Ho. *Applied Optimal Control.* Hemisphere Publishing Corporation, 1975.
4. G. Leitmann. *An Introduction to Optimal Control.* McGrawHill, New York, NY, 1966.
5. L. C. Young. *Optimal Control Theory.* Chelsea, 1980.