ON ROBUST DYNAMIC PROGRAMMING

Anton G. Madievski[☆] and John B. Moore[†★]

Department of Systems Engineering
Australian National University
Canberra ACT 0200 Australia

☆ phone: +61-6-279 8086; email: anton@syseng.anu.edu.au
★ phone: +61-6-249 2461; email: john@syseng.anu.edu.au

Abstract. Dynamic programming achieves optimum control for known deterministic and stochastic systems. There is a need, however, to apply dynamic programming ideas to real-world uncertain systems. The standard dynamic programming algorithm is ineffective in the case of shedding an optimal decision path due to uncertainty in modelling. In this paper we propose a robust variation on dynamic programming in which a minimization operation is replaced by an averaging operation. We present a simple numerical example to illustrate the effectiveness of the algorithm.

Keywords: dynamic programming; robust control; principle of optimality

† The author wishes to acknowledge the funding of the activities of the Cooperative Research Center for Robust and Adaptive Systems by the Australian Commonwealth Government under the Cooperative Research Centres Program.

1. INTRODUCTION

The theory of dynamic programming is fundamental for optimization of multistage decision processes. It provides a framework for the consideration of generalizations of many classical problems, including significant problems in the domain of engineering, physics, biology, economics and operations research.

Bellman's intuitively obvious principle of optimality can be stated as: Subarcs of optimal trajectories are themselves optimal. Suboptimal arcs can be shed from consideration in a reverse pass decision process. This approach to solving problems can be applied to obtaining analytic and computational results in numerous areas. [1–7]

This standard dynamic programming approach is effective for well defined deterministic and stochastic models. Now, what happens if we apply dynamic programming to a real–world system, prone to external and internal uncertainties affecting the model? What happens if the system sheds suboptimal subarc for the assumed model but the optimal one for the actual model? It is observed in this case that frequently the algorithm cannot get back onto the optimal arc in the short term; the algorithm of dynamic programming is not robust. This is a serious restriction on possible usage of the approach to solve applied problems.

This paper proposes a robust dynamic programming scheme. Naturally, the algorithm will be sub–optimal for the actual signal model, but closer to optimal in the presence of model uncertainty than standard dynamic programming.

In Section 2, we develop the rationale for the robust dynamic programming algorithm based on averaging rather than minimization, as in standard dynamic programming. In Section 3, we present a simple numerical example to illustrate the effectiveness of the algorithm. In Section 4, we generalize the robust dynamic programming algorithm. We give some concluding remarks in Section 5.

2. ROBUST DYNAMIC PROGRAMMING – FINITE–STATE CASE

To fix ideas, let us consider the simple case of a finite–state machine with state process given by the following equation:

$$x(k+1) = U(k)\, A_1\, x(k) + U(k)\, A_2 + A_3\, x(k), \;\; k = 0, 1, \dots, N \qquad (1)$$

where $x(k) \in \mathcal{S}_x = \{s_1, s_2, \dots, s_M\}$ is a discrete set of states and $x(0) = x_0$. Without loss of generality we take $\mathcal{S}_x := \{e_1, e_2, \dots, e_M\}$ where e_i is the unit vector. A_1 is a row M–vector, A_2 is a scalar and A_3 is an M by M matrix. M–vector U(k) is the control decision at the time index k to realize discrete state transition and $U(k) \in \mathcal{S}_u = \{u_1, u_2, \dots, u_{M^2}\}$ where each control u_i takes state s_p from $\mathcal{S}_x$ into s_r from $\mathcal{S}_x$.

Consider the system given by (1) evolving over N+1 time steps from k=0 to k=N. Then the cost of running the system for the N+1 steps is described by the following cost function:

$$J = \sum_{k=0}^{N} L(x(k), U(k), k) \tag{2}$$

Because of the discrete nature of x(k), U(k), the functions L can be written in the following form:

$$L(x(k), U(k), k) = U^T(k)\, L_1\, x(k) + U^T(k)\, L_2 + L_3\, x(k), \tag{3}$$

where $L_1 \in \mathbb{R}_{M\times M}$, $L_2 \in \mathbb{R}_{M\times 1}$, $L_3 \in \mathbb{R}_{1\times M}$.

The cost of running the system over the last N-k+1 time steps is denoted J(x, k):

$$J(x, k) = \sum_{j=k}^{N} L(x(j), U(j), j), \tag{4}$$

where now x(k) has the fixed value $x \in \mathcal{S}_x$, and where the system equation (1) must be satisfied.

Standard dynamic programming achieves the optimum value of the scalar cost function as a minimization over the single variable U(N) as follows:

$$J_{opt}(x,N) = \min_{U(N)\in\mathcal{S}_u} L(x,U(N), N) \tag{5}$$

The optimum cost $J_{opt}(x,k)$ for all k, $0 \leq k \leq N-1$, can be iteratively found for all $x \in \mathcal{S}_x$ from knowledge of $J_{opt}(x,k+1)$ for all $x \in \mathcal{S}_x$ as follows:

$$J_{opt}(x,k) = \min_{U(k)\in\mathcal{S}_u} \{L(x, U(k), k) + J_{opt}(x(k+1), k+1)\}, \tag{6}$$

Decision based on the minimization procedure is a "hard" decision and the scheme is not robust, in general. Due to uncertainty in modelling, the algorithm can be ineffective once it sheds an optimal decision path for the actual signal generating system.

In our proposal for *robust dynamic programming* we calculate the mean cost of taking the system through its last time step $\bar{J}(x,N)$ as the average over all possible control decisions available at N:

$$\bar{J}(x,N) = \gamma^{-1}(N) \sum_{U(N)\in\mathcal{S}_u} L(x, U(N), N), \tag{7}$$

where $\gamma(k)$ is a number of all possible control decisions U(k) at the time instant k.

Similarly, going backwards from k = N−1 to k = 0, we can evaluate the average costs of running the system over the last N-k+1 steps iteratively:

$$\bar{J}(x,k) = \sum_{U(k)\in \mathcal{I}_u} \gamma^{-1}(k)\,\{L(x, U(k), k) + \bar{J}(x(k+1), k+1)\}, \tag{8}$$

where x(k+1) is calculated according to (1).

It is our claim that taking averages instead of extremum values makes the algorithm insensitive to random errors and, thus, robust. At this stage there is no rigorous proof of this claim, but simulation results and the general consideration that averaging should absorb noise support the claim.

3. EXAMPLE

In this section we present a simple example to illustrate the ideas which motivated this paper as well as to confirm feasibility of the proposed algorithm.

Let us consider a system representing a car. State x(k) of the system is the speed of the car (at the moment k) which can take one of six values: 40, 50, 60, 70, 80 and 90 km/h. For example, $x^T = (1, 0, 0, 0, 0, 0)$ corresponds to the speed of 40km/h, while $x^T = (0, 1, 0, 0, 0, 0)$ denotes the speed of 50 km/h and so on.

The control vector U(k) takes the system from the state assigned the value of 1 to the state with the value 2. For example, if the car is driven at 60 km/h (state number 3), then the control $(0, 0, 1, 0, 2, 0)^T$ speeds the car up to 80 km/h (state number 5), while the control $(0, 0, 3, 0, 0, 0)^T$ keeps the speed steady at 60km/h.

The corresponding system matrices are:

$$A_1 = 1/2\, \underline{1}_6{}^T, \quad A_2 = 0, \quad A_3 = -1/2\, I_6,$$

where

$\underline{1}$ is column vector of 6 unities;
I_6 is identity matrix 6×6.

Assuming initial car speed of 80 km/h ($x_0 = e_5$), we are taking a trip and want to minimise expenses incurred by the trip.

Our choice of cost function matrices is purely empirical and is not based on any statistical or automotive research data. Let us consider the following L_1, L_2 and L_3:

$$L_1 = \begin{bmatrix} 0 & 0.5 & 1 & 1.5 & 2 & 2.5 \\ 1 & 0 & 0.5 & 1 & 1.5 & 2 \\ 2 & 1 & 0 & 0.5 & 1 & 1.5 \\ 3 & 2 & 1 & 0 & 0.5 & 1 \\ 4 & 3 & 2 & 1 & 0 & 0.5 \\ 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix} \qquad L_2 = \theta_6 \qquad L_3 = [5 \;\; 10 \;\; 15 \;\; 25 \;\; 40 \;\; 60].$$

where θ_6 is a column vector of 6 zeros.

The choice of matrix L_1 reflects the natural assumptions that the greater is change of speed, the higher is cost and slowing down is cheaper than speeding up. The choice of vector L_3 reflects higher costs at higher speeds.

Suppose we are taking a 3 time unit drive (N=2). Now, we can calculate mean cost of driving our car over the last time step as the average over all possible control decisions for every value of the vehicle's speed at N=2 according to (7), where obviously the number of all possible control decisions $\gamma = 6$:

$\bar{J}(e_1,2) = 10$, $\bar{J}(e_2,2) = 13.5$, $\bar{J}(e_3,2) = 17.5$, $\bar{J}(e_4,2) = 27$, $\bar{J}(e_5,2) = 42$, $\bar{J}(e_6,2) = 62.5$.

Then, going backwards from k = 1 to k = 0 as in (8), we can evaluate average costs for running the system over the last (3–k) steps iteratively. For k = 1 we have:

$\bar{J}(e_1,1) = 38.75$, $\bar{J}(e_2,1) = 42.25$, $\bar{J}(e_3,1) = 46.25$, $\bar{J}(e_4,1) = 55.75$, $\bar{J}(e_5,1) = 70.75$, $\bar{J}(e_6,1) = 91.25$.

Similarly, for k = 0, we have:

$\bar{J}(e_1,0) = 67.5$, $\bar{J}(e_2,0) = 71$, $\bar{J}(e_3,0) = 75$, $\bar{J}(e_4,0) = 84.5$, $\bar{J}(e_5,0) = 99.5$, $\bar{J}(e_6,0) = 120$.

Now, bringing up our assumption of $x_0 = e_5$, we can say that the cost of the trip estimate is:

$$\bar{J} = \bar{J}(e_5,0) = 99.5.$$

We now have all the necessary information to budget our trip. Of course, the actual cost may differ from $\bar{J}$ due to the unpredicted nature of road events. But we have the best possible estimate.

The results for traditional dynamic programming would be:
for k=2
$J_{opt}(e_1,2)=5$, $J_{opt}(e_2,2)=10$, $J_{opt}(e_3,2)=15$, $J_{opt}(e_4,2)=25$, $J_{opt}(e_5,2)=40$, $J_{opt}(e_6,2)=60$;

for k=1
$J_{opt}(e_1,1)=10$, $J_{opt}(e_2,1)=16$, $J_{opt}(e_3,1)=22$, $J_{opt}(e_4,1)=33$, $J_{opt}(e_5,1)=49$, $J_{opt}(e_6,1)=70$;

and, finally, for k=0
$J_{opt}(e_1,0)=15$, $J_{opt}(e_2,0)=21$, $J_{opt}(e_3,0)=27$, $J_{opt}(e_4,0)=38$, $J_{opt}(e_5,0)=54$, $J_{opt}(e_6,0)=75$,

thus bringing (for $x_0=e_5$) overall cost to 54.

Comparison of the two costs associated with standard and robust dynamic programming schemes shows 84% performance degradation for robust dynamic programming, but this higher cost may prove to be less than the cost of controlling the system according to standard dynamic programming algorithm in the presence of noise and for uncertain model, as now illustrated.

Suppose that, due to modelling errors, control U(k) which believed to take the system to the state number p, takes it in fact to the state number 7−p. This uncertainty in the model makes the standard dynamic programming algorithm shed the optimal path. Instead of the optimal strategy of slowing down to 40km/h and keeping going at that speed (80km/h for k=0, 40km/h for k=1 and k=2), dynamic programming algorithm results in the speeding up strategy (80km/h for k=0, 90km/h for k=1 and k=2) which incurs high cost of 162, which is triple the optimal cost and is actually 63% worse than the (unchanged irrespectively of the modelling error) cost calculated according to robust dynamic programming scheme.

4. GENERALIZATION

The ideas applied to a simple finite-state linear system in Section 2 can be extended to nonlinear signal models with states in a continuous range as follows:

$$x(k+1) = g(x(k), u(k), k), \quad k = 0, 1, \ldots, N \tag{9}$$

where x(0) has the fixed value x_0 and the state x and control u satisfy the constraints:

$$x \in X \subset \mathbb{R}^M, \qquad u \in U \subset \mathbb{R}^Q \tag{10}$$

The cost of running the system over the last N-k+1 time steps is again denoted J(x,k), while the overall cost $J(x_0,0)$ is denoted, as previously, J:

$$J(x, k) = \sum_{j=k}^{N} P(x(j), u(j), j), \tag{11}$$

$$J = \sum_{k=0}^{N} P(x(k), u(k), k) \tag{12}$$

where the functions P are not necessarily linear functions of x and u.

Now, assuming x to be the state of the system at the moment N, we can calculate the mean cost of taking the system through its last time step $\bar{J}(x,N)$ as the average over all possible control decisions available at N:

$$\bar{J}(x,N) = \gamma^{-1}(N) \sum_{u(N)} P(x, u(N), N), \tag{13}$$

where $\gamma(k)$ is a number of all possible control decisions u(k) at the moment k.

Then, iterating backwards from k = N−1 to k = 0, we can evaluate average costs of running the system over the last N–k+1 steps iteratively:

$$\bar{J}(x,k) = \sum_{u(k)} \gamma^{-1}(k) \{P(x, u(k), k) + \bar{J}(g(x(k), u(k), k), k+1)\}. \qquad (14)$$

Thus, the robust dynamic programming algorithm is applicable to the general continuous state–space non–linear systems with general non–linear cost functions.

5. CONCLUSIONS

In this paper we have proposed a robust dynamic programming scheme. An example of a motor-vehicle operation cost evaluation has been considered to illustrate the simplicity and effectiveness of the scheme.

REFERENCES

[1] R. Bellman, *Dynamic Programming*, Princeton University Press, N.J., 1957.

[2] R. Bellman and R. Kalaba, *Dynamic Programming and Modern Control Theory*, Academic Press, New York, 1965.

[3] R. Bellman, "Dynamic Programming: A Reluctant Theory", in *New Methods of Thought and Procedure*, Springer, Berlin, 1967.

[4] H. Halkin, "The Principal of Optimal Evolution", in *Nonlinear Differential Equations and Nonlinear Mechanics*, Academic Press, New York, pp. 284–302, 1963.

[5] R.E. Larson, *State Increment Dynamic Programming*, Elsevier, New York, 1968.

[6] R.E. Larson and J.L. Casti, *Principles of Dynamic Programming*, Marcel Dekker, New York, 1978.

[7] G.L. Nemhauser, *Introduction to Dynamic Programming*, John Wiley and Sons, 1966.