

1992-02

# Dynamic Programming for Optimal Control of Set-Up Scheduling with Neural Network Modifications

---

<https://hdl.handle.net/2144/2082>

*"Downloaded from OpenBU. Boston University's institutional repository."*

**DYNAMIC PROGRAMMING FOR OPTIMAL  
CONTROL OF SET-UP SCHEDULING WITH  
NEURAL NETWORK MODIFICATIONS**

Gary Bradski

February, 1992

Technical Report CAS/CNS-92-002

Permission to copy without fee all or part of this material is granted provided that: 1. the copies are not made or distributed for direct commercial advantage, 2. the report title, author, document number, and release date appear, and notice is given that copying is by permission of the BOSTON UNIVERSITY CENTER FOR ADAPTIVE SYSTEMS AND DEPARTMENT OF COGNITIVE AND NEURAL SYSTEMS. To copy otherwise, or to republish, requires a fee and/or special permission.

Copyright © 1992

Boston University Center for Adaptive Systems and  
Department of Cognitive and Neural Systems  
111 Cummington Street  
Boston, MA 02215

# DYNAMIC PROGRAMMING FOR OPTIMAL CONTROL OF SET-UP SCHEDULING WITH NEURAL NETWORK MODIFICATIONS

Gary Bradski  
Department of Cognitive and Neural Systems  
Boston University  
111 Cummington Street  
Boston, MA 02215

## Abstract

This paper demonstrates an optimal control solution to change of machine set-up scheduling based on dynamic programming average cost per stage value iteration as set forth by Caramanis et. al. [2] for the 2D case. The difficulty with the optimal approach lies in the explosive computational growth of the resulting solution. A method of reducing the computational complexity is developed using ideas from biology and neural networks. A real time controller is described that uses a linear-log representation of state space with neural networks employed to fit cost surfaces.

**1. INTRODUCTION** This paper addresses the exponential growth in computational complexity that results from the optimal solution to the change of set-up, manufacturing scheduling problem. Set-up scheduling is a subset of optimal resource allocation – the intelligent use of information to enhance the efficiency and effectiveness of the use of limited resources.

The paper focuses on a simplified manufacturing set-up scheduling problem: one machine which can produce two different parts. There is a constant, though not necessarily equal demand for each of these parts. The machine produces one part type at a time, and any switch over to producing a different part type involves a stochastic set-up time delay. Under, or over production involves a cost for unsatisfied demand, or storage, respectively. In this simplified problem, these two costs are treated symmetrically. The controls one may exert are to: turn production on, turn it off, or change the machine set-up.

In the next section, the optimal solution of this problem is explained and demonstrated. The remainder of the paper looks at ways of dealing with the explosive growth in calculations needed for treating the more general cases of producing three or more part types.

**2. OPTIMAL SET-UP SCHEDULING** The set-up scheduling problem here is approached using successive approximation (or value iteration) to recursively generate a sequence  $h^k$  which converges to the differential cost vector  $h_\mu$ . For state  $s$ , the average cost per stage for control  $\mu$  is  $T_\mu(h_\mu)(s)$ . The desired minimum average cost per stage is  $T(h)(s)$ :

$$T(h^k)(i) = \min_{u \in U(i)} \left[ g(i, u) + \sum_{j=1}^n P_{ij}(u) h^k(j) \right]. \quad (1)$$

This technique is expounded in Bertsekas [1] following the work of White [5].

Three controls are possible: (1)  $u = [0, 0]$ , [*don't change set-up, don't produce*]; (2)  $u = [0, 1]$ , [*don't change set-up, produce parts*]; (3)  $u = [1, 0]$ , [*change set-up, don't produce*]. State space has coordinates of the part types  $x_i$  and the set-up condition  $\sigma_i$ :  $(x_1, x_2, \sigma_i)$ . There are four set-ups in the two part case: (1)  $\sigma_1$ , set-up to produce part type 1; (2)  $\sigma_2$ , set-up to produce part type 2; (3)  $\sigma_{21}$ , from set-up 2, change set-up to produce part type 1; (4)  $\sigma_{12}$ , from set-up 1, change set-up to produce part type 2. Allowable controls for each set-up  $\sigma$  are: for set-up 1,  $U(\sigma_1) = [0, 0; 0, 1; 1, 0]$ ; for set-up 2,  $U(\sigma_2) = [0, 0; 0, 1; 1, 0]$ , and for changing set-up,  $U(\sigma_{12}) = U(\sigma_{21}) = [\textit{nothing}]$ . For the two part case, the dynamics associated with set-up 1 and the controls are: (1)  $\dot{x}(\sigma_1, 0, 0) = [-d_1, -d_2]^T$ ; (2)  $\dot{x}(\sigma_1, 0, 1) = [\hat{u}_1 - d_1, -d_2]^T$ ; and (3)  $\dot{x}(\sigma_{ij}, 1, 0) = [-d_1, -d_2]^T$  where  $d_i$  is the demand (depletion rate), and  $\hat{u}_i$  is the production rate of part type  $i$ .

In this paper we ignore any cost of deciding to change set up and so use as our cost equation:

$$g(x_1, x_2) = c_1 x_1^2 + c_2 x_2^2. \quad (2)$$

The differential cost iteration equations are:

$$\begin{aligned} & \sigma_i \text{ Plane} & (3) \\ u = 0, 0 : & h^{k+1}(x_1, x_2, \sigma_i) = [g(x_1, x_2) + h^k(x_1 - d_1, x_2 - d_2, \sigma_i)] - T(h^k)(s) \\ u = 0, 1 : & h^{k+1}(x_1, x_2, \sigma_i) = [g(x_1, x_2) + h^k(x_1 - d_1, x_{i=2} + \hat{u}_2 - d_2, \sigma_i)] - T(h^k)(s) \\ u = 1, 0 : & h^{k+1}(x_1, x_2, \sigma_i) = [g(x_1, x_2) + P_{ij}h^k(x_1 - d_1, x_2 - d_2, \sigma_i) \\ & (1 - P_{ij})h^k(x_1 - d_1, x_2 - d_2, \sigma_{ij})] - T(h^k)(s) \\ & \sigma_{ij} \text{ Plane} \\ u = -, - : & h^{k+1}(x_1, x_2, \sigma_{ij}) = [g(x_1, x_2) + P_{ij}h^k(x_1 - d_1, x_2 - d_2, \sigma_j) \\ & +(1 - P_{ij})h^k(x_1 - d_1, x_2 - d_2, \sigma_{12})] - T(h^k)(s) \end{aligned}$$

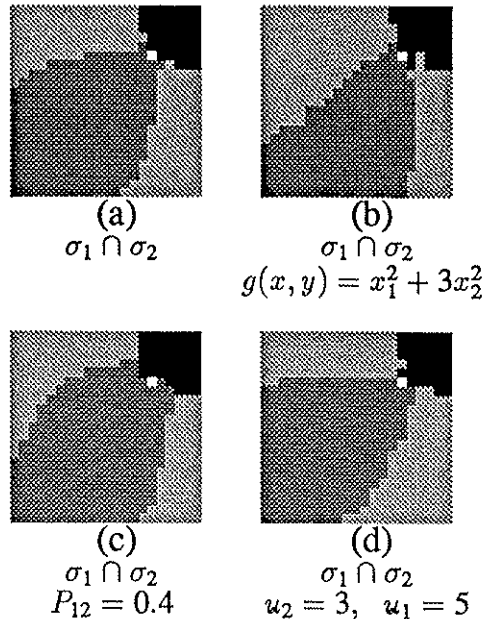
where the arbitrarily chosen normalizing state  $s$  is calculated as:

$$T(h^k)(s_1, s_2, \sigma_{ij}) = g(s_1, s_2) + P_{ij}h^k(s_1 - d_1, s_2 - d_2, \sigma_j) + (1 - P_{ij})h^k(s_1 - d_1, s_2 - d_2, \sigma_{ij}). \quad (4)$$

When iterations converge, the control with minimum differential cost  $h$  is selected.

**2.1 Simulation of Optimal Set-Up.** A graphical view of the control-space solution above is shown in **Figure 1a**. In the figure: black means idle ( $u = 0, 0$ ), dark grey means maintain current action ( $u = 0, 1$ ), and light grey means change set-up ( $u = 1, 0$ ). In addition, the origin is marked by a white dot. Parameters are given in the figure. **Figure 1b-d** is a brief parameter study verifying the expected results. **Figure 1b** shows the effect of an elliptical cost function which warps the change of set-up curve such that deviations from the more expensive axis will be minimized. **Figure 1c** shows changes in transition probability which again warps the change set-up curve such that there is more time to perform the set-up that is harder to change to. **Figure 1d** shows the effect of differential production rates where the change set-up curves are once again warped to allow the slower production rate part more time to be produced.

**2.2 Computational Growth.** The purpose of this paper is not to analyze the effect of parameter variations, but rather to extract ways that might help reduce total the computational burden. This computational burden can be seen as follows: If we represent each part in state space with  $N$  discrete computation points along each dimension  $D$ , then the



**Figure 1.** Various control solutions. Part (a) parameters are:  $d_1 = d_2 = 1$ ,  $P_{12} = P_{21} = 0.7$ ,  $u = 4$ ,  $c_1 = c_2 = 1$ . Part (b) varies the cost function. Part(c) varies the probability of transition. Part (d) varies the production capacity.  $x_1$  is the ordinate axis. Black = don't produce, dark grey = maintain set-up, light grey = change set-up (origin marked in white).

order of computations  $C$  is exponential:  $C = O(N^D)$ . The problem is even worse since the differential cost equations must be iterated repeatedly till convergence. To develop ideas for saving computations, we examine the nature of the optimal solution below.

**2.3 The Nature of the Cost Surfaces.** Much of the structure of this problem follows from the cost function (equation 2 ) which in this case is a 3D parabolic surface. This cost surface becomes a parabolic hyper-surface if the number of part types grows larger than two.

**3.0 CALCULATION SAVING IDEAS FROM BIOLOGY** The machine set-up controller makes decisions based on the point it is at in state space, the rest of state space is calculated only because it affects the point of interest. It is the need to “attend” to all of state space that causes the computational burden. Biological creatures face the same problem: in general, an organism’s focus is on a limited portion of space – a piece of food say – and yet the organism must also attend to the rest of space so as to not be eaten in turn.

Biology is then faced with a problem: the need to attend in detail and yet cover all of space. Because brains can’t be the size of trucks, it is impossible to attend in detail to all of space. Instead, the solution biology has found is to spread the coverage of space in a non-linear way; there is dense coverage in the region of interest with increasingly diffuse coverage away from that point. The best example of this is in the retina: the fovea has a dense and linear coverage of a small region of space with coverage falling off exponentially away from the fovea. Thus space can be attended to both in detail and in expanse, yet without explosive computational needs. This idea has been exploited in technological applications in the work of Seibert and Waxman [4] based on the work of Eric Schwartz [3].

This paper suggests that we employ the same type of solution in the optimal set-up prob-

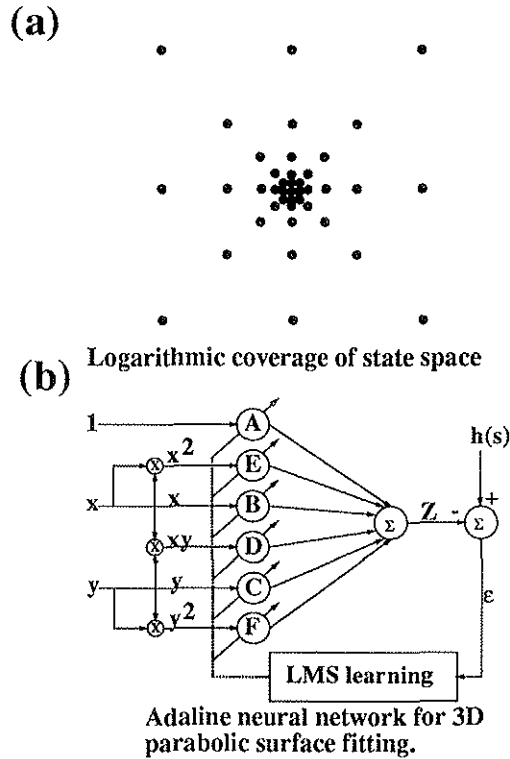


Figure 2:

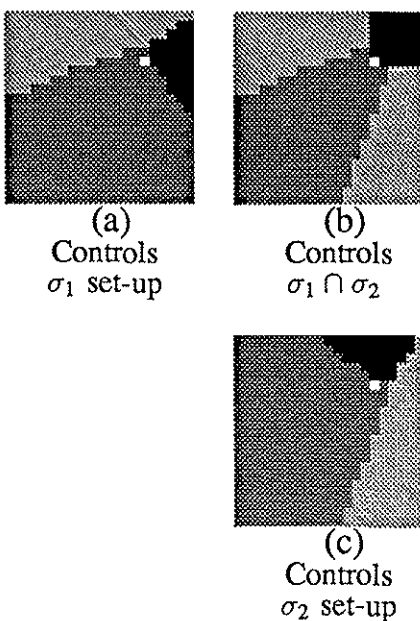
lem: a dense linear coverage of the region of interest, with coverage falling off exponentially from there. **Figure 2a** shows a *linear-log* representation of state space. A method of interpolating the differential costs in the regions of space that are sparsely covered must be found. To solve this problem, recall the findings from above that parabolic surfaces work well to approximate the actual cost surfaces. Here, we will use neural networks to do the surface fitting.

Using the above, this paper proposes using a real time controller for the set-up problem rather than attempting the one-shot, computationally explosive optimal solution. As the state of the system evolves, the real time controller, centered at the current point in state space, continuously recalculates the optimal control for its new center of focus (its *fovea*). Real-time control is feasible because many fewer points in state space need to be calculated, and because of the computational savings, it is also possible to handle many more dimensions (part types) than before. Another advantage of real time control is that changes in demand, cost function, transition probabilities etc may be handled easily and immediately.

**4.0 IMPLEMENTATION OF REAL-TIME CONTROL** To reduce calculations, a real time controller needs to fit cost surfaces for interpolation. The form of the cost surfaces are 3D parabolas given in general by:

$$Z = A + Bx + Cy + Dxy + Ex^2 + Fy^2 \quad (5)$$

One method of fitting 5 is to use multiple regression with normal equations to solve for parameters  $A \rightarrow F$ . To avoid problems with the complexity of the system of equations, singular matrices, and the need for working with large matrices, an Adaline network (Widrow



**Figure 3.** Decision space for the real time controller centered at the origin. The optimal solution is approximated at points nearby. Basic parameters here are:  $d_1 = d_2 = 1$ ,  $P_{12} = P_{21} = 0.7$ ,  $u = 4$ ,  $c_1 = c_2 = 1$ .

and Hoff [6]) with cross-product terms was chosen instead to do the surface fitting as shown in Figure 2b.

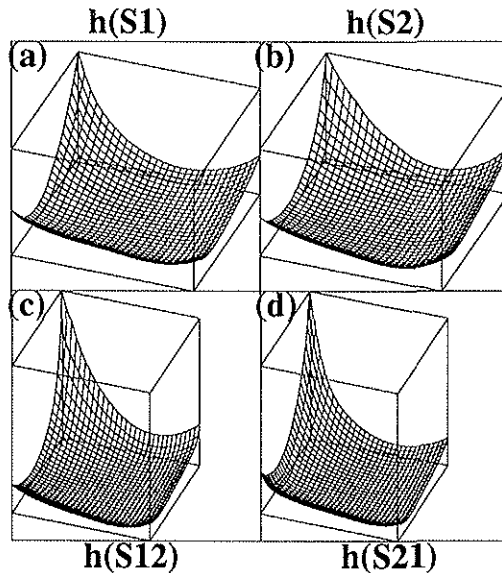
The modified algorithm is to: logarithmically cover state space as shown in Figure 2a<sup>1</sup>. For this controller, the calculation points are quickly truncated in the non-dense region, since error is high out there anyhow. The algorithm is then as follows:

- 1) Initialize average cost equal to fixed cost using equation 2 for each set-up plane.
- 2) Fit cost surfaces using neural networks for interpolation.
- 3) Iterate incremental costs using interpolated values from 2 if necessary.
- 4) Minimize over controls.
- 5) Check for average cost convergence. If so, go to 6, if not, go to 2. **Note:** Only the dense central calculation points are checked for cost convergence.
- 6) Done, implement optimal control, Goto 2 using the next state space location.<sup>2</sup>

**4.1 SIMULATION RESULTS** The first pass at the new algorithm revealed that the initial scheme did not work too well. Errors from using the interpolation surface creep in too quickly and cause the “maintain current action” region to be too narrow. This problem was investigated by examining the relative error between the optimal and the approximate differential cost surfaces. It was found that the  $\sigma_i$  planes were too large relative to the  $\sigma_j$  planes. As an empirical fix, the algorithm was changed to give less weight to the  $\sigma_i$  planes for purposes of calculating decisions to change set-up. Figure 3 shows the improved results for

<sup>1</sup>It does not work to just use a sparse, linear spacing of calculation points in state space and then use an interpolation scheme because the error between actual values and surface interpolation grows too rapidly – a dense linear region is needed in the center.

<sup>2</sup>To make figures, all of real time control space is shown, but only the “fovea” (at the origin here) is accurate.



Error surfaces for each set-up plane between optimal and Neural Network surface fitting methods. Note that near the origin, there is little error (origin is 5 grid points from the right bottom front corner). Set-ups planes are:  
 (a) set-up for part 1;  
 (b) set-up for part 2;  
 (c) change set-up from part 1 to part 2;  
 (d) change set-up to part 1 from part 2 set-up.

Figure 4:

the controller centered at the origin (note that only nearby points approximate the optimal solution then). **Figure 4** shows the relative error between the optimal set-up solution and the neural network fitting solution. It can be seen that near the origin (five units in from the front bottom right corner), there is little discrepancy between the optimal and “log-neural” solution.

Thus, the proposed scheme of using a “state space fovea” centered at the current location in state space to calculate the optimum control works at least empirically. Since there is little error near the center of our calculations, we should at least approximate the global optimum policy as we move around in state space.

**ACKNOWLEDGEMENTS** Supported in part by DARPA (AFOSR 90-0083).

## REFERENCES

1. Bertsekas, D., *Prentice-Hall* 1987. Section 7.2.
2. Caramanis M., Sharifnia A., Hu J., and Gershwin S., “Development of a Science Base for Planning and Scheduling Manufacturing Systems,” *Research Summary for NSF/SMI grant No. DDM-8914277* 1991.
3. Schwartz E., “Computational Neuroscience,” *MIT Press*, 1990.
4. Seibert M., and A. Waxman A., “Adaptive 3D-Object Recognition from Multiple Views”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, In Press.
5. D. J. White, *J. Math. Anal. Appl.*, **6** 1963, 373-376.
6. B. Widrow, and M. Hoff, “Adaptive Switching Circuits,” *IRE WESCON Conv. Record*, Part 4, 96-104, August 1960.