# Multiple Machine Maintenance: Applying a Separable Value Function Approximation to a Variation of the Multiarmed Bandit

by

Haixia Lin

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2002





Author .....
Department of Electrical Engineering and Computer Science
February 1, 2002

Certified by .....
Dimitri P. Bertsekas
Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

# Multiple Machine Maintenance: Applying a Separable Value Function Approximation to a Variation of the Multiarmed Bandit

by

Haixia Lin

Submitted to the Department of Electrical Engineering and Computer Science
on February 1, 2002, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering

## Abstract

We define the dynamic programming problem of optimally maintaining multiple machines by defining the behavior of each machine, the effects and limits of the repairs we can make, and the rewards incurred by both machine behavior and repairs. We show that the optimal solution, the maintenance policy that maximizes total expected reward, is difficult to find. We discuss some general suboptimal control methods, then examine two related dynamic programming problems for which optimal solutions exist: the multiarmed bandit problem and the single machine maintenance problem. Motivated by the optimal solutions to these related problems, we propose a suboptimal solution for the multiple machine maintenance problem that is index-based, evaluate its performance, suggest methods for improving its performance, and discuss possible further work on this problem.

Thesis Supervisor: Dimitri P. Bertsekas
Title: Professor

# Acknowledgments

I would like to thank my thesis supervisor, Professor Dimitri Bertsekas, for his inspiration, guidance, and compassion. His concern for my professional development is a gift unparalleled and unforgettable.

I would also like to thank Michael Deaett of Draper Labs for his support and interest in this thesis.

Finally, I would like to thank Alex Wang for his advice, patience, and love.

# Contents

# List of Figures

# Chapter 1

# Introduction

The multiple machine maintenance problem is a type of resource allocation problem in which multiple machines share a maintenance resource. We assume that each machine is independent of all other machines, other than their reliance on a common maintenance resource. Any system that has multiple independent devices, a subset of which we select at each stage in order to achieve some goal, may be modeled as a multiple machine maintenance problem. This model may be applied to many practical situations, such as a server choosing which customers to service or a communication channel allocating its bandwidth.

Much literature exists on various types of resource allocation problems. The version of resource allocation that we will focus on in this thesis is most closely related to the multiarmed bandit problem, which has an elegant optimal solution based on index functions. The optimality of this index-based policy was first proved by John Gittins [6] [7]. Many alternate proofs were later developed which gave further insight into the multiarmed bandit problem, [9] [10] [11]. By relaxing some of the constraints of the multiarmed bandit problem, we define a new resource allocation problem whose optimal solution has not been found and combine different suboptimal control methods to achieve an elegant, versatile suboptimal solution.

In the second chapter, we formally define the multiple machine maintenance problem in terms of its state evolution, reward structure, and control space, explain why its optimal solution is difficult to find, and discuss some common suboptimal control

approaches requiring value function approximations.

In the third chapter, we discuss two problems similar to multiple machine maintenance, the multiarmed bandit problem and single machine maintenance, whose optimal solutions will help guide the suboptimal solution we will later develop.

In the fourth chapter, we derive a suboptimal policy, based on a separable value function approximation, that has the same form as the multiarmed bandit problem's optimal policy. We also define three specific separable value function approximations: the upper approximation, the lower approximation, and the modified upper approximation.

In the fifth chapter, we evaluate each separable value function approximation in terms of its ability to approximate the optimal value function and the proximity to optimal of its corresponding suboptimal policy. To illustrate the relative merits of these approximations, we use a simple two-machine maintenance problem.

In the sixth chapter, we discuss ways to improve upon the suboptimal policy's performance, namely by finding a rollout policy or correcting the separable value function approximation. We describe variations of multiple machine maintenance to which this suboptimal policy may also be applied. Finally we discuss possible future work suggested by the results of this thesis.

# Chapter 2

# Problem Formulation: Multiple Machine Maintenance

We define the multiple machine maintenance problem in terms of its state evolution, control space, and reward structure. We explain why finding an optimal solution to the multiple machine maintenance problem is difficult in most cases. We then discuss a common suboptimal control approach, one-step lookahead, and some common methods for implementing one-step lookahead.

## 2.1 State Evolution

In this problem, we want to optimally maintain $n$ machines. The performance of the $i^{th}$ machine at time $k$ may be described by its state, $x_k^i \in \{1, \ldots, m_i\}$. A higher state number indicates worse performance. For example, state 1 corresponds to perfect performance, while state $m_i$ corresponds to the machine being irreparable. Each machine's state evolves, independently of all other machines, according to a Markov chain with probabilities $p_{ij}$ of transitioning from state $i$ to state $j$. This Markov chain models a machine's tendency to drift towards worse states when in the absence of maintenance. For example, we may model a machine that may only transition to states which are no better performance-wise than the current state, i.e. $p_{ij} > 0$ only if $j \geq i$, when no maintenance is applied. If we decide to maintain a machine,

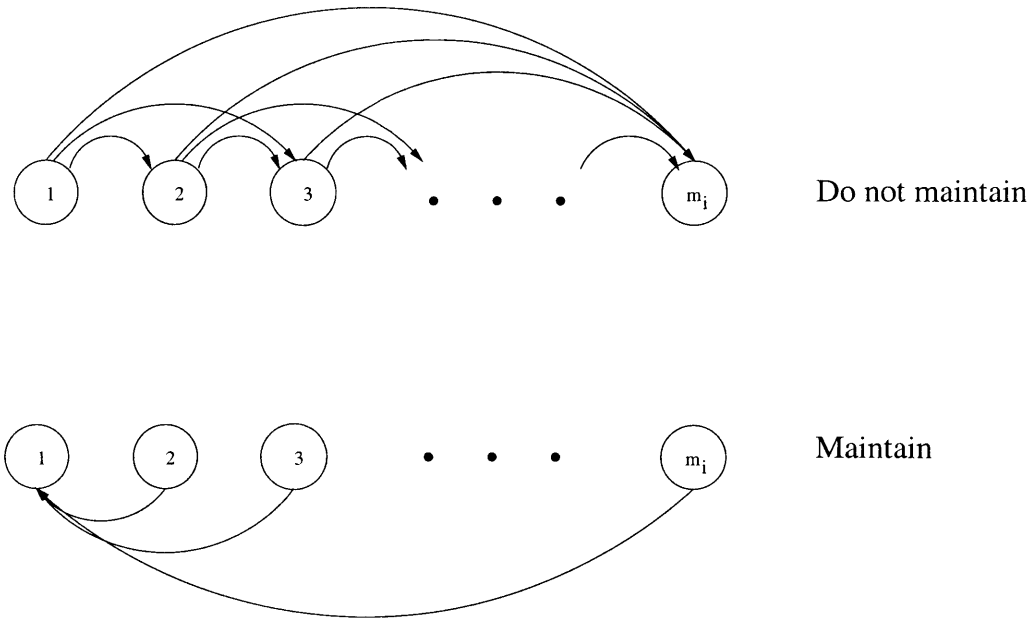


Figure 2-1: Example Markov chain for machine $i$

then its state evolves according to a different Markov chain, with probabilities $q_{ij}$ of transitioning from state $i$ to state $j$. This Markov chain models the machine's performance improvement when maintenance is applied. For example, we may model a machine that automatically returns to state 1 when it is maintained (Figure 2-1). We may only maintain one machine at a time, so we want to find a policy that, given the current state of all machines, chooses a specific machine to maintain or chooses to do nothing.

## 2.2 Reward Structure

A specific policy is successful if it balances the benefits of having machines perform well with the costs of maintaining these machines. To measure a policy's success, we introduce a value function, $J^{\pi}(x^1, \ldots, x^n)$, that represents the total expected reward

of policy $\pi$ when machine $i$ starts at state $x^i$, for $i \in \{1, \ldots, n\}$. We may express this value function in terms of $\alpha^k g^i(x_k^i)$, the reward for machine $i$ being at state $x_k^i$ at stage $k$, and $\alpha^k r^i(x_k^i)$, the cost of repairing machine $i$ at state $x_k^i$ at stage $k$, where $\alpha \in (0, 1)$ is a discount factor that weighs present costs more heavily than future costs and both $g^i(x^i)$ and $r^i(x^i)$ are uniformly bounded. If we find the maximum value of the value function for every set of initial states $\{x^1, \ldots, x^n\}$, known as the optimal value function, then we have the policy that maximizes the total expected reward, known as the optimal policy.

## 2.3 Optimal Policy

Finding the optimal value function is difficult because the state space is too large in most cases. According to dynamic programming theory, we find the optimal value function:

$$J^*(x^1, \ldots, x^n) = \max_{\pi} J^{\pi}(x^1, \ldots, x^n) \tag{2.1}$$

by solving the following system of equations known as Bellman's equation:

$$\begin{aligned} J^*(x^1, \ldots, x^n) = \sum_{i=1}^{n} g^i(x^i) + \alpha \max[&\underbrace{\sum_{y} \prod_{i=1}^{n} p_{x^i y^i} J^*(y^1, \ldots, y^n)}_{\text{no maintenance}}, \\ &\max_{j=1,\ldots,n} \underbrace{\{-r^j(x^j) + \sum_{y} q_{x^j y^j} \prod_{i=1, i\neq j}^{n} p_{x^i y^i} J^*(y^1, \ldots, y^n)\}}_{\text{maintain } j}] \end{aligned} \tag{2.2}$$

Generally, Bellman's equation is solved either analytically or computationally. In the case of multiple machine maintenance, an analytic solution to equation (2.2) is not readily apparent and in most cases the state space, which has $m_1 m_2 \cdots m_n$ elements, is too large to allow a computational solution. Because finding the exact optimal solution is difficult, we want to find a suboptimal policy that yields a value function

that is close to optimal.

## 2.4 One-step Lookahead Policy

To develop a suboptimal policy, we discuss a common suboptimal control approach, a one-step lookahead policy [2], whose implementation requires an approximation of the optimal value function. We then discuss some general optimal value function approximation methods.

To understand the motivation for a one-step lookahead policy, we first notice that any policy that maximizes the right-hand side of Bellman's equation is optimal. Maximizing the right-hand side of Bellman's equation requires already knowing the optimal value function, $J^*(x)$, for all $x$. To avoid finding the optimal value function exactly, which in the multiple machine maintenance problem is very difficult, we replace it with an approximation. Any policy that maximizes the approximated right-hand side of Bellman's equation is a one-step lookahead policy.

## 2.5 Value Function Approximation Techniques

The performance of a one-step lookahead policy depends on the value function approximation used. A popular approximation maps state-dependent features of the problem, $y(x)$, into a value function approximation, $\tilde{J}(y(x), r)$, where $r$ is a vector of tunable weights and $\tilde{J}$ is a parametric function [5]. The selection of features, weights, and a parametric function depends on the particular problem.

To guide our selection for the multiple machine maintenance problem, we discuss another popular approach: to use the optimal value function of a simpler, solvable problem as a value function approximation. Many times a simpler problem may be constructed by changing the constraints of the problem. For a problem with many interdependent subsystems, like multiple machine maintenance, we would like to change the problem parameters in such a way as to decouple the subsystems. If this decoupling is achieved by using a subset (or superset) of the control constraint,

then we not only have an approximation to the optimal value function but a lower (or upper) bound to the optimal value function [8]. To help develop a one-step lookahead policy for multiple machine maintenance, we discuss two simpler problems that have computable optimal solutions: the multiarmed bandit problem and single machine maintenance.

# Chapter 3

# Related Problems

To apply value function approximation methods to the multiple machine maintenance problem, we examine two related problems for which optimal solutions are attainable: the multiarmed bandit problem and the single machine maintenance problem. The multiarmed bandit problem may be viewed as a multiple machine maintenance problem whose machines do not deteriorate, thereby allowing an elegant optimal solution. The single machine maintenance problem, by only having one machine, reduces the number of states to $m_i$, thereby allowing a computable optimal solution.

## 3.1 Multiarmed Bandit

The multiple machine maintenance problem may be viewed as a variation of the multiarmed bandit problem [4], which also has $n$ projects that we may only service one at a time and also has a cost structure that rewards for maintaining projects at better states. But unlike our machines, a reward, $R^i(x^i)$, is accrued only when project $i$ is serviced and the state of project $i$ evolves stochastically only if project $i$ is serviced, otherwise its state stays the same.

To more clearly show how the two problems are related, we consider a special case of the multiple machine maintenance problem where the state of any machine stays the same in the absence of maintenance, instead of drifting towards worse states. This special case models a problem where machines may be upgraded one at a time

and at a cost $r^i(x^i)$. We show how this non-deteriorating version of the multiple machine maintenance problem is actually a multiarmed bandit problem by reformulating Bellman's equation:

$$J^*(x^1,\ldots,x^n) = \sum_{i=1}^{n} g^i(x^i) + \alpha \max[\underbrace{J^*(x^1,\ldots,x^n)}_{\text{no upgrades applied}}, \underbrace{\max_{j=1,\ldots,n}\{-r^j(x^j) + \sum_{y^j} q_{x^j y^j} J^*(x^1,\ldots,x^{j-1},y^j,x^{j+1},\ldots,x^n)\}}_{\text{upgrade } j}] \tag{3.1}$$

We first notice that if the optimal decision at state $x$ is to upgrade none of the machines, then the optimal decision at the next stage is also to upgrade none of the machines because the state remains the same. We may therefore find the optimal value function in closed form for any state $x$ at which it is optimal to no longer upgrade any machines. Let $S$ be the set of all states at which it is optimal to no longer upgrade any machines. We have the following Bellman's equation for any state $x \in S$, which we then solve for $J^*(x)$:

$$J^*(x^1,\ldots,x^n) = \sum_{i=1}^{n} g^i(x^i) + \alpha J^*(x^1,\ldots,x^n) \tag{3.2}$$

$$J^*(x^1,\ldots,x^n) = \frac{\sum_{i=1}^{n} g^i(x^i)}{1-\alpha} \tag{3.3}$$

Substituting equation (3.3) into Bellman's equation (3.1), we have:

$$J^*(x^1,\ldots,x^n) = \max[\frac{\sum_{i=1}^{n} g^i(x^i)}{1-\alpha}, \max_{j=1,\ldots,n}\{\sum_{i=1}^{n} g^i(x^i) - \alpha r^j(x^j) + \alpha \sum_{y^j} q_{x^j y^j} J^*(x^1,\ldots,x^{j-1},y^j,x^{j+1},\ldots,x^n)\}] \tag{3.4}$$

Let $H^*(x) = J^*(x) - \frac{\sum_{i=1}^{n} g^i(x^i)}{1-\alpha}$. Notice that $H^*(x)$ is also a valid optimal value function for this problem. We substitute $\frac{\sum_{i=1}^{n} g^i(x^i)}{1-\alpha} + H^*(x)$ for $J^*(x)$ in Bellman's equation (3.4):

$$\frac{\sum_{i=1}^{n} g^i(x^i)}{1-\alpha} + H^*(x) = \max[\frac{\sum_{i=1}^{n} g^i(x^i)}{1-\alpha}, \max_{j=1,\ldots,n}\{\sum_{i=1}^{n} g^i(x^i) - \alpha r^j(x^j) + \alpha \sum_{y^j} q_{x^j y^j}(\frac{\sum_{i=1, i\neq j}^{n} g^i(x^i) + g^j(y^j)}{1-\alpha} + H^*(x^1, \ldots, x^{j-1}, y^j, x^{j+1}, \ldots, x^n))\}] \quad (3.5)$$

Subtracting $\sum_{i=1}^{n} g^i(x^i)$ from both sides of equation (3.5), we have:

$$H^*(x^1, \ldots, x^n) = \max[0, \max_{j=1,\ldots,n}\{\frac{\alpha(\sum_{y^j} q_{x^j y^j} g^j(y^j) - g^j(x^j))}{1-\alpha} - \alpha r^j(x^j) + \alpha \sum_{y^j} q_{x^j y^j} H^*(x^1, \ldots, x^{j-1}, y^j, x^{j+1}, \ldots, x^n)\}] \quad (3.6)$$

$$= \max[0, \max_{j=1,\ldots,n}\{R^j(x^j) + \alpha \sum_{y^j} q_{x^j y^j} H^*(x^1, \ldots, x^{j-1}, y^j, x^{j+1}, \ldots, x^n)\}] \quad (3.7)$$

where $R^j(x^j) = \frac{\alpha}{1-\alpha}(\sum_{y^j} q_{x^j y^j} g^j(y^j) - g^j(x^j)) - \alpha r^j(x^j)$.

From equation (3.7), we see we may interpret the non-deteriorating machine problem as a multiarmed bandit problem with reward $R^i(x^i)$ when machine $i$ is serviced, transition probabilities $q_{ij}$ when machine $i$ is serviced, and a reward-free option to retire from servicing all machines. Therefore, we may think of the multiple machine maintenance problem as a multiarmed bandit problem with less state evolution constraints.

Our problem's resemblance to the multiarmed bandit problem is useful because the multiarmed bandit problem has a simple optimal policy based on index functions. An index function, $\gamma^i(x^i)$, is defined for project $i$ and measures the profitability of servicing project $i$ given its current state $x^i$. An index-based policy either selects the project with the highest current index or does not service any project if all the indices are currently below a predetermined threshold $M$:

$$\text{service project i} \quad \text{if} \quad \gamma^i(x^i) = \max_{j=1,\ldots,n} \gamma^j(x^j) \geq M$$

$$\text{service no projects} \quad \text{if} \quad \max_{j=1,\ldots,n} \gamma^j(x^j) < M$$

Index-based policies have many advantages. Because a project's index depends only on its current state, and not the current states of any other projects, index functions are typically easily computed. Because the numerical values for each project's index function are found in advance, a relatively small amount of work is required to execute an index-based policy, namely selecting the largest value from $n+1$ values ($n$ indices and 1 threshold). Because a project's index does not depend on any other project, an index-based policy is executable even when the number and types of projects are changing. However, finding an index function that is both computable and optimal is not always possible.

## 3.2 Single Machine Maintenance

To motivate a computable, suboptimal index function for the multiple machine maintenance problem, we find the optimal solution for the single machine maintenance problem and redefine the corresponding optimal policy as an index-based policy.

By limiting the number of machines to 1, we significantly reduce the size of the state space to $m_1$; therefore the optimal value function may be found computationally using Bellman's equation:

$$J_i^*(x^i) = g^i(x^i) + \alpha \max\{ \underbrace{\sum_{y^i=1}^{m_i} p_{x^i y^i} J_i^*(y^i)}_{\text{do not maintain}} , \underbrace{-r^i(x^i) + \sum_{y^i=1}^{m_i} q_{x^i y^i} J_i^*(y^i)}_{\text{maintain}} \} \quad (3.8)$$

The optimal policy maintains the machine if and only if

$$\sum_{y^i=1}^{m_i} p_{x^i y^i} J_i^*(y^i) \leq -r^i(x^i) + \sum_{y^i=1}^{m_i} q_{x^i y^i} J_i^*(y^i).$$

This optimal policy is equivalent to an index-based policy with the following index

function:

$$\gamma^1(x^1) \;=\; -r^i(x^i) + \sum_{y^i=1}^{m_i} (q_{x^i y^i} - p_{x^i y^i}) J_i^*(y^i) \tag{3.9}$$

and a threshold $M = 0$. Although defining an index function is not necessary for single machine maintenance problems, we do so in order to apply this index function to the multiple machine problem.

# Chapter 4

# Index-based One-step Lookahead Policy

## 4.1 Suboptimal Index-based Policy

We find a one-step lookahead policy using the following separable value function approximation:

$$J^*(x^1, \ldots, x^n) \approx \sum_{i=1}^{n} J^i(x^i) \tag{4.1}$$

where $J^i(x^i)$ represents the approximate amount of reward-to-go contributed by machine $i$ to the total reward-to-go. We may think of $J^i(x^i)$, for $i = 1, \ldots, n$, as state-dependent features that we linearly combine in order to approximate the optimal value function. Before we discuss how to find these features, we will characterize the one-step lookahead policy that results from using this separable value function approximation.

We substitute the separable value function approximation (4.1) into the right-hand side of Bellman's equation (2.2):

$$J^*(x^1,\ldots,x^n) \approx \sum_{i=1}^{n} g^i(x^i) + \alpha \max[\underbrace{\sum_{y}\prod_{i=1}^{n} p_{x^iy^i}\sum_{l=1}^{n} J^l(y^l)}_{\text{no maintenance}},$$
$$\max_{j=1,\ldots,n}\underbrace{\{-r^j(x^j) + \sum_{y} q_{x^jy^j}\prod_{i=1,i\neq j}^{n} p_{x^iy^i}\sum_{l=1}^{n} J^l(y^l)\}}_{\text{maintain } j}] \quad (4.2)$$

We simplify the right-hand side of equation (4.2) using $\sum_{y^i=1}^{m_i} p_{x^iy^i} = 1$:

$$J^*(x^1,\ldots,x^n) \approx \sum_{i=1}^{n} g^i(x^i) + \alpha \max[\sum_{i=1}^{n}\sum_{y^i=1}^{m_i} p_{x^iy^i}J^i(y^i),$$
$$\max_{j=1,\ldots,n}\{-r^j(x^j) + \sum_{y^j=1}^{m_j} q_{x^jy^j}J^j(y^j) + \sum_{i=1,i\neq j}^{n} p_{x^iy^i}J^i(y^i)\}] \quad (4.3)$$

We remove the constant $\sum_{i=1}^{n}\sum_{y^i=1}^{m_i} p_{x^iy^i}J^i(y^i)$ from within the first maximization of equation (4.3):

$$J^*(x^1,\ldots,x^n) \approx \sum_{i=1}^{n}(g^i(x^i) + \alpha\sum_{y^i=1}^{m_i} p_{x^iy^i}J^i(y^i)) + \alpha\max[0,$$
$$\max_{j=1,\ldots,n}\{-r^j(x^j) + \sum_{y^j=1}^{m_j}(q_{x^jy^j} - p_{x^jy^j})J^j(y^j)\}] \quad (4.4)$$

Recalling the index function, $\gamma^i(x^i) = -r^i(x^i) + \sum_{y^i=1}^{m_i}(q_{x^iy^i} - p_{x^iy^i})J^i(y^i)$ (3.9), from the single machine maintenance problem, we substitute $\gamma^i(x^i)$ into equation (4.4):

$$J^*(x^1,\ldots,x^n) \approx \sum_{i=1}^{n}(g^i(x^i) + \alpha\sum_{y^i=1}^{m_i} p_{x^iy^i}J^i(y^i))$$
$$+\alpha\max[0, \max_{j=1,\ldots,n}\{\gamma^j(x^j)\}] \quad (4.5)$$

According to approximation (4.5) of Bellman's equation, the one-step lookahead policy is an index-based policy with index function $\gamma^i(x^i)$ (3.9) and threshold $M = 0$. As described earlier, an index-based policy has many advantages. The only drawback is this index-based policy is suboptimal, meaning the index function $\gamma^i(x^i)$ approximates the profitability of maintaining machine $i$. The performance of this approximate index function depends on how well the separable value function approximation mimics the shape of the optimal value function. The magnitude of the approximation error does not matter because if $J^*(x)$ is the optimal value function, then $J^*(x) + c$, where $c$ is a constant, yields the same optimal policy.

## 4.2 Separable Value Function Approximation

To find the function $J^i(x^i)$, we examine problems similar to the multiple machine maintenance problem whose optimal value functions are genuinely separable, and thus computable, and use their separable optimal value functions as approximations to the original optimal value function. To find a genuinely separable problem, we decouple the machines' reliance on a shared maintenance resource by changing the control space. We consider three different control space: at each stage an unlimited number of machines may be maintained, at stage $k$ only machine $i$ satisfying $i = (k \bmod n)$ may be maintained, and at stage $k$ an unlimited number of machines may be maintained if $(k \bmod n) = 0$, otherwise no machines may be maintained.

### 4.2.1 Upper Approximation

Consider a version of the multiple machine maintenance problem that does not limit the number of machines that can be maintained at each stage. The maintenance needs of one machine no longer interfere with the maintenance needs of the other machines, meaning we may consider each machine as a single machine maintenance problem. As discussed earlier, the single machine maintenance problem may be solved computationally because the state space of each machine is relatively small.

Let $J^i_{upper}(x^i)$, for $i = 1, \ldots, n$, be the optimal value function for machine $i$ if it were

the only machine. Then the separable value function approximation, $\sum_{i=1}^{n} J^i_{upper}(x^i)$, is the optimal value function when up to $n$ machines may be maintained at each stage.

This approximation to the original optimal value function is named the upper approximation because it is a upper bound to the original optimal value function. To show this relation, let $A$ be the set of all policies that maintain at most one machine at each stage and let $B$ be the set of all policies that maintain at most $n$ machines at each stage. Because $A$ is a subset of $B$, we know that the optimal policy over the set $A$ is either the same as or worse than the optimal policy over the set $B$, meaning the optimal value function over the set $A$ is no greater than the optimal value function over the set $B$:

$$\max_{\pi \in A} J^{\pi}(x^1, \ldots, x^n) \leq \max_{\pi \in B} J^{\pi}(x^1, \ldots, x^n) \tag{4.6}$$

$$J^*(x^1, \ldots, x^n) \leq \sum_{i=1}^{n} J^i_{upper}(x^i) \tag{4.7}$$

The performance of this approximation generally worsens the more machines we have. The approximation assumes the machines are not vying for the same maintenance resource, so the higher the number of machines that do share the resource, the less accurate the approximation.

### 4.2.2 Lower Approximation

Consider a version of the multiple machine maintenance problem that not only limits the number of maintenances to one per stage, but may consider maintaining machine $i$ only every $n$ stages. The system cycles through all machines one at a time, so that at each stage it only has two options, maintain or do not maintain the current machine under consideration. The maintenance needs of one machine no longer interfere with the maintenance needs of the other machines, meaning we may find the optimal solution for each machine individually. These individual optimal solutions may be

found computationally because the state space of each machine is relatively small.

Let $J^i_{lower}(x^i)$ for $i = 1, \ldots, n$ be the optimal value function for machine $i$ if the option to maintain is only available every $n$ stages and the next possible maintenance period is $i - 1$ stages away. Then the separable value function approximation, $\sum_{i=1}^{n} J^i_{lower}(x^i)$, is the optimal value function for a system that is currently considering machine 1 for maintenance, will consider machine 2 for maintenance at the next stage, and will continue considering the machines in numerical order until it considers machine $n$ for maintenance, at which point it repeats the process by cycling back to consider machine 1 again. Notice that the assignment of numbers to each machine is significant in this approximation. Both the machine order and the current location within the order will affect the value function approximation.

This approximation to the original optimal value function is named the lower approximation because it is a lower bound to the original optimal value function. To show this relation, let $A$ be the set of all policies that maintain at most one machine at each stage and let $C$ be the set of all policies that consider only one machine for maintenance at each stage, cycle through the machines numerically, and are currently considering machine 1.

Because $A$ is a superset of $C$, we know that the optimal policy over the set $A$ is either the same as or better than the optimal policy over the set $C$, meaning the optimal value function over the set $A$ is no less than the optimal value function over the set $C$:

$$\max_{\pi \in A} J^\pi(x^1, \ldots, x^n) \geq \max_{\pi \in C} J^\pi(x^1, \ldots, x^n) \tag{4.8}$$

$$J^*(x^1, \ldots, x^n) \geq \sum_{i=1}^{n} J^i_{lower}(x^i) \tag{4.9}$$

The performance of this approximation generally worsens the more machines we have. Some machines may need significantly more maintenance than others, so the higher the number of machines, the longer the wait between maintenance opportunities for high maintenance machines, and the less accurate the approximation.

### 4.2.3 Modified Upper Approximation

Consider a version of the multiple machine maintenance problem that allows the system to maintain up to $n$ machines in one stage, but only every $n$ stages. No maintenance is allowed during the intervening stages. Like the approximations above, the maintenance needs of one machine do not interfere with the maintenance needs of the other machines, meaning we may easily find the optimal solution for each machine.

Let $J_d^i(x^i)$ for $i = 1, \ldots, n$ be the optimal value function for machine $i$ if the option to maintain is only available every $n$ stages and the next possible maintenance period is $d \in \{0, \ldots, n-1\}$ stages away, where $d$ is the same value for all machines. Different values of $d$ yield different value function approximations.

This approximation to the original optimal value function is named the modified upper approximation because both the upper and the modified upper approximations derive from a version of the multiple machine maintenance problem that allow up to $n$ maintenances in one stage. The modified upper approximation attempts to improve upon the upper approximation by limiting the number of stages at which maintenance is allowed, which better mimics the amount of maintenance resource available.

This approximation is also similar to the lower approximation because both approximations derive from a version of the multiple machine maintenance problem that maintains a particular machine at most once every $n$ stages. The modified upper approximation attempts to improve upon the lower approximation by using the same number of stages, $d$, until the next maintenance stage for all machines, which better represents the relative rewards contributed by each machine.

# Chapter 5

# Separable Value Function Approximation: Two-machine Example

We discuss some qualitative aspects of the performances of the different separable approximations. To demonstrate these aspects, we consider a simple two-machine example with a repair cost of 12 from any state for either machine and nonincreasing reward functions, $g^i(x^i)$, in $x^i$ (Figure 5-1).

Without maintenance, each machine is equally likely to transition to any state that is no worse than its current state. Maintenance brings each machine back to state 1 automatically. We use two machines because the results are easily graphed. This example has an optimal value function that is nonincreasing in both $x^1$ and $x^2$ (Figure 5-2).

Because the number of machines is so small, all the separable approximations perform reasonably well. Their index-based policies achieve total rewards with less than 5% error from the optimal total reward. Some approximations perform better than others because they better mimic the shape of the optimal value function.

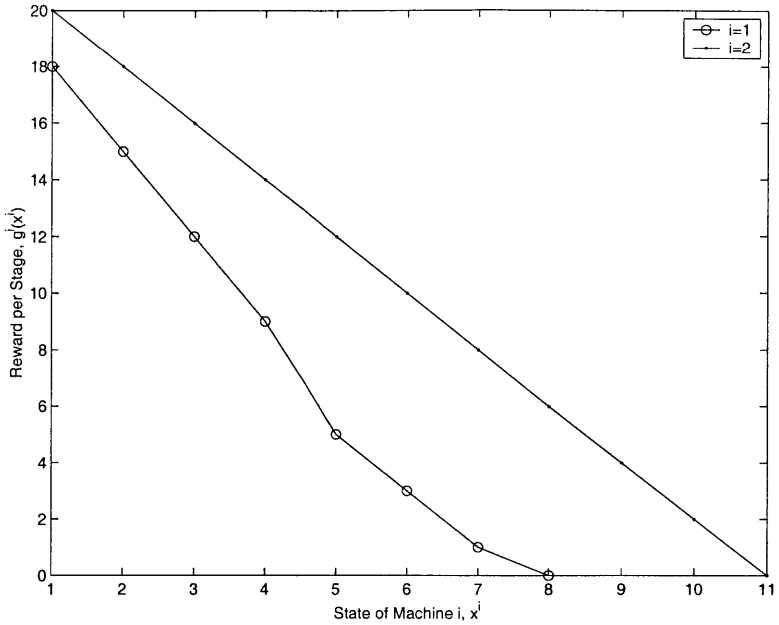


Figure 5-1: Reward per Stage, $g^i(x^i)$, for $i = 1, 2$

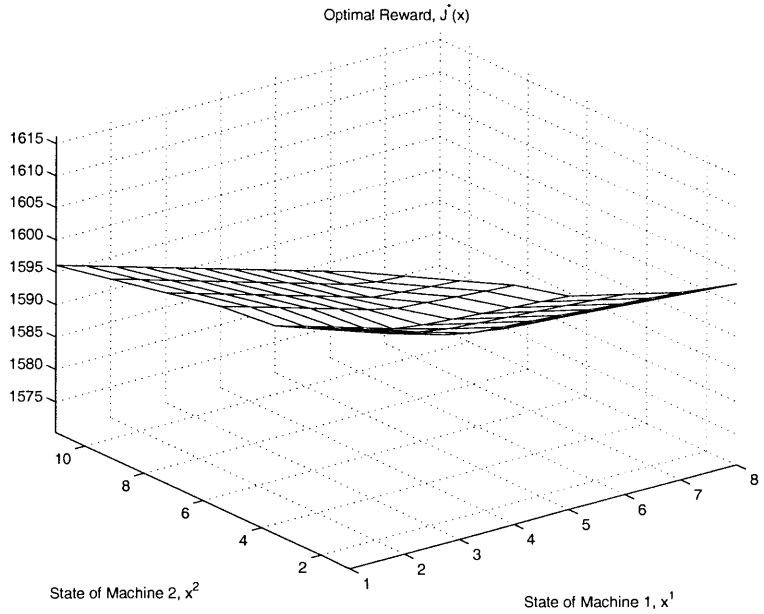


Figure 5-2: Optimal Value Function, $J^*(x)$

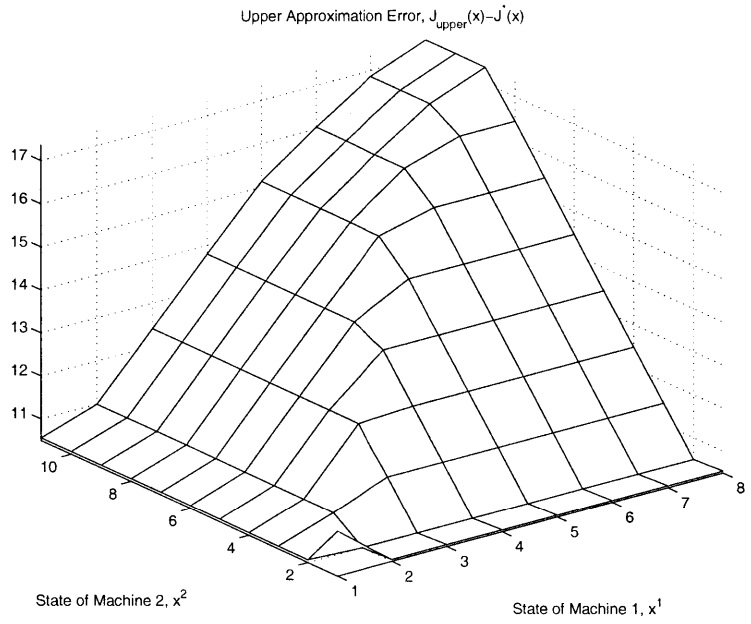


Figure 5-3: Upper Approximation Error, $J_{upper}(x) - J^*(x)$

## 5.1 Upper Approximation

The upper approximation, by assuming that no limit on maintenance exists, worsens the more both machines require maintenance. At states where one machine does not require maintenance, the upper approximation error is approximately constant with respect to the other machine's state. As the machine deteriorates, thus increasing its maintenance need, the upper approximation error increases and becomes non-constant with respect to the other machine's state (Figure 5-3).

Although ideally we would like an error as close to constant as possible, the upper approximation error is relatively symmetric with respect to the machines, meaning most index-based policy errors are confined to states near the border between different optimal decisions, resulting in a total reward with approximately .6% error from the optimal total reward (Figure 5-4).

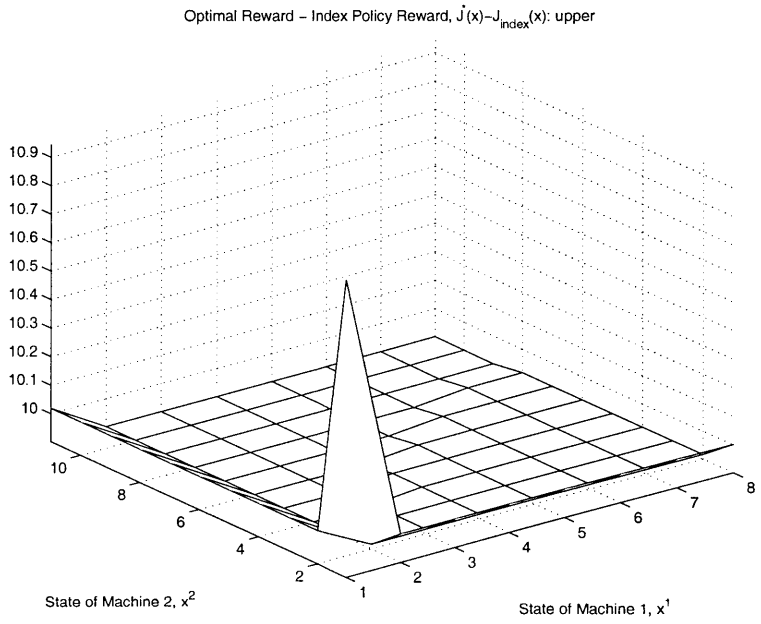


Figure 5-4: Difference between Optimal Reward and Reward of Index-based Policy using the Upper Approximation

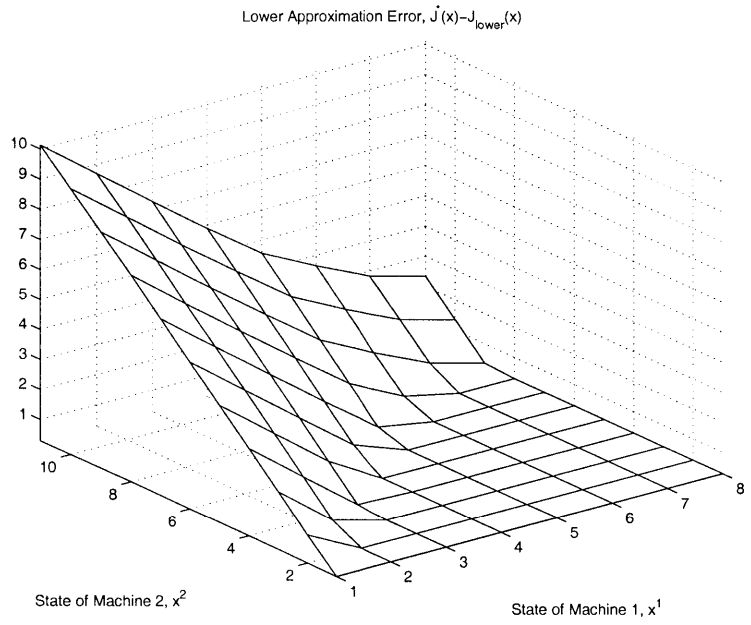


Figure 5-5: Lower Approximation Error, $J^*(x) - J_{lower}(x)$

## 5.2 Lower Approximation

The lower approximation, by assuming one of the machines may not be maintained at the current stage, worsens as the machine under consideration improves and the machine not under consideration deteriorates. At states at which the machine under consideration is faring worse, the lower approximation error is constant. As the machine under consideration improves and the other machine worsens, the approximation error increases (Figure 5-5).

Although this approximation error is constant for about half the states, the asymmetry of the approximation error underestimates the relative reward from states at which the optimal control would be to maintain the machine not under consideration, meaning the index-based policy errs at about half the states, resulting in a total reward with approximately 4.4% error from the optimal total reward (Figure 5-6).

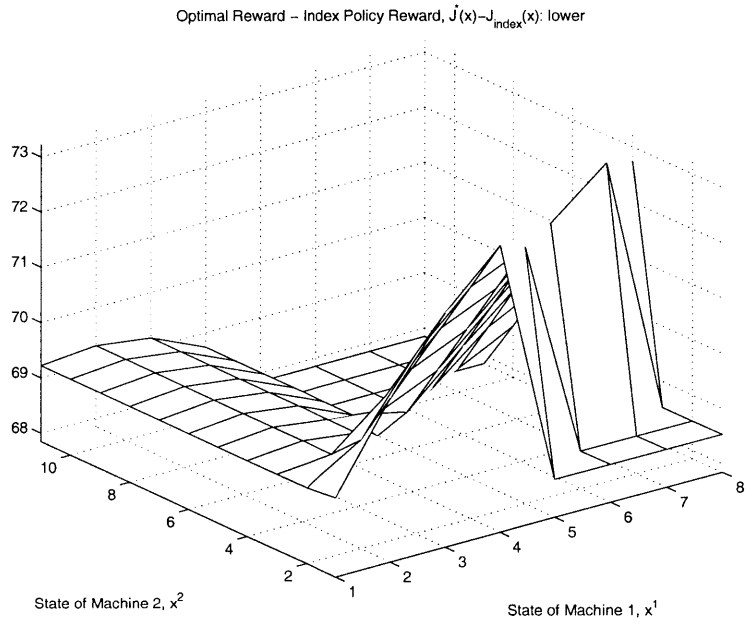


Figure 5-6: Difference between Optimal Reward and Reward of Index-based Policy using the Lower Approximation

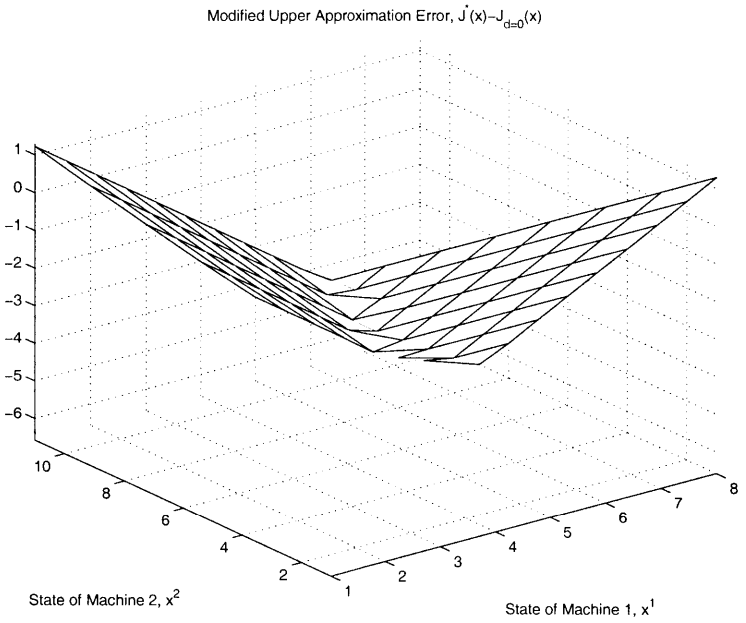


Figure 5-7: Modified Upper Approximation Error with $d = 0$

## 5.3 Modified Upper Approximation with $d = 0$

The modified upper approximation, by limiting the amount of maintenance available over multiple stages as opposed to within a stage, attempts to correct for the asymmetry of the lower approximation error while improving the shape of the upper approximation error. Consider the case where we may maintain both machines at the current stage, i.e. $d = 0$. The modified upper approximation error worsens at states at which the optimal control is to maintain one machine now and to maintain the other machine at the next stage (Figure 5-7).

Although this approximation error is more symmetric than the lower approximation error, it does not have the constant sections that the upper approximation error does, resulting in a total reward with approximately 2.2% error from the optimal total reward (Figure 5-8).

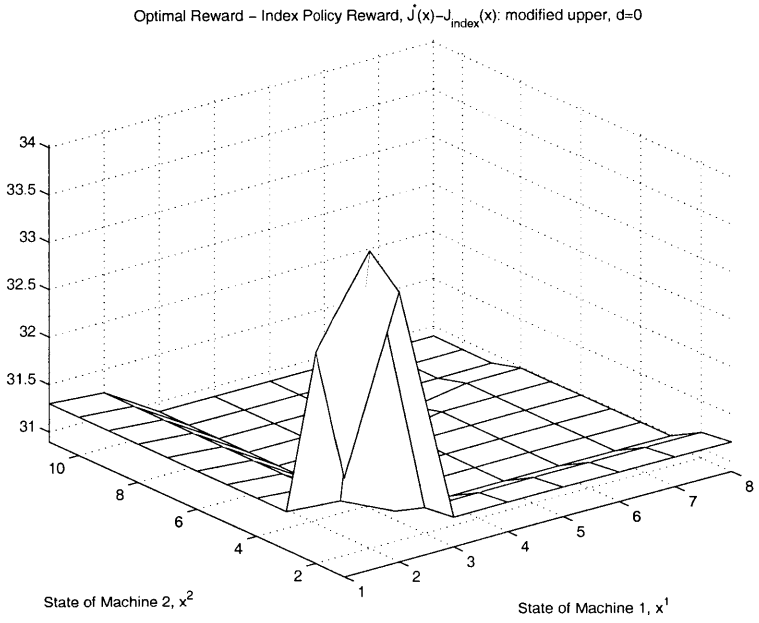


Figure 5-8: Difference between Optimal Reward and Reward of Index-based Policy using the Modified Upper Approximation with $d = 0$

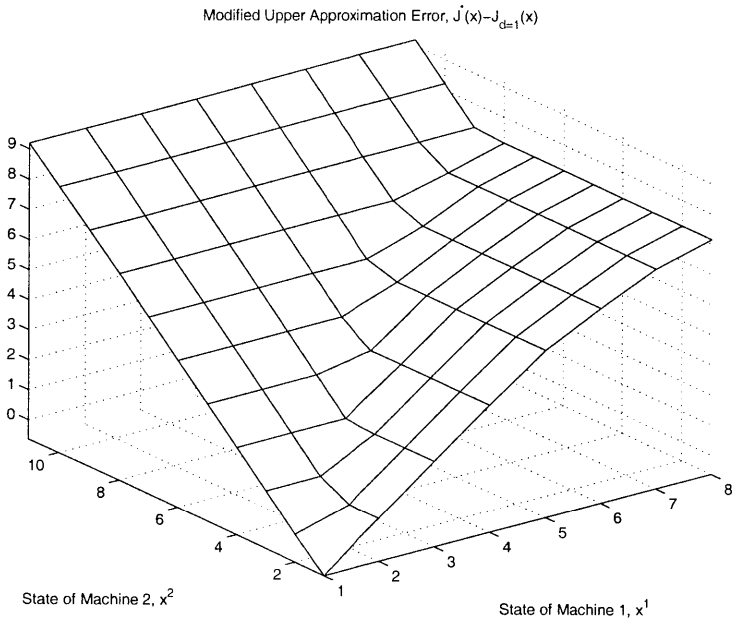


Figure 5-9: Modified Upper Approximation Error with $d = 1$

## 5.4 Modified Upper Approximation, $d = 1$

Consider the case where we may maintain both machines at the next stage but not at the current stage. This case may be viewed as being the opposite of the upper approximation. Instead of approximating the optimal value function well at states at which only one machine needs maintenance, this modified upper approximation approximates the optimal value function well at states at which multiple machines are vying for the maintenance resource. At states at which one machine is not faring well, the modified upper approximation error is approximately constant with respect to the other machine's state. As the machine improves, thus decreasing its maintenance need, the modified upper approximation error decreases and becomes non-constant with respect to the other machine's state (Figure 5-9).

This approximation error is also more symmetric than the lower approximation. It improves upon the shape of the upper approximation error by having its constant

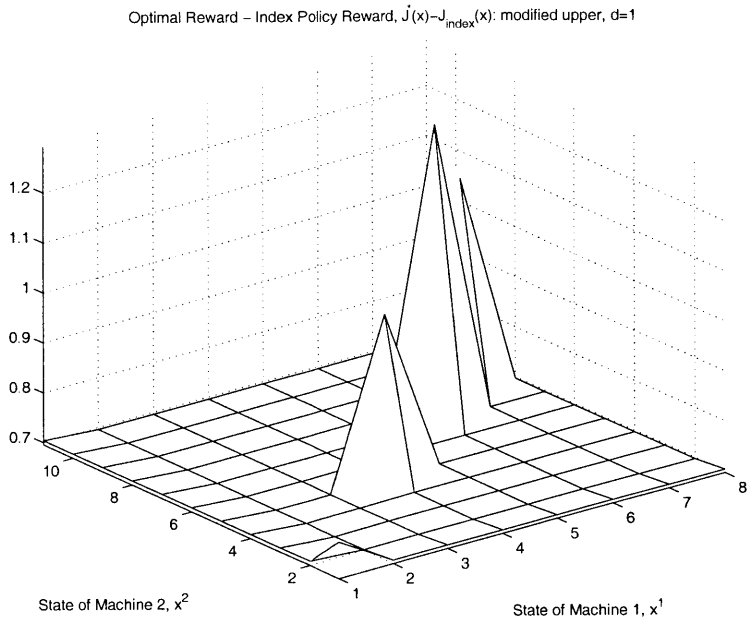


Figure 5-10: Difference between Optimal Reward and Reward of Index-based Policy using the Modified Upper Approximation with $d = 1$

sections, where it performs better, at states at which it is more difficult to determine the optimal control, namely those states at which multiple machines are vying for maintenance, resulting in a total reward with approximately .06% error from the optimal total reward (Figure 5-10).

In general, the modified upper approximation with $d \approx \frac{n}{2}$, where $n$ is the number of machines, performs better than the other approximations.

# Chapter 6

# Extensions and Variations

The suboptimal index-based policy is amenable to both performance improvement and to different variations of the multiple machine maintenance problem. We may improve upon the performance of the suboptimal index-based policy by using improved optimal value function approximations. We discuss a popular improvement method, finding a rollout policy, and a problem-specific method, correcting the separable value function approximation for a specific type of approximation. We then discuss variations of the multiple machine maintenance control space that also yield a suboptimal index-based solution.

## 6.1 Rollout Policy

To improve upon the performance of the suboptimal index-based policy, we introduce a rollout policy [3]. A rollout policy is a one-step lookahead policy that uses the value function of a base policy as its value function approximation. In the case of multiple machine maintenance, we may use the index-based policy as a base policy and find the resulting rollout policy. More specifically, $\pi_r$ is a rollout policy based on the index-based policy, $\pi$, if $\pi_r$ maximizes the following:

$$\sum_{i=1}^{n} g^i(x^i) + \alpha \max[\underbrace{\sum_y \prod_{i=1}^{n} p_{x^i y^i} J^\pi(y^1, \ldots, y^n)}_{\text{no maintenance}},$$

$$\underbrace{\max_{j=1,\ldots,n} \{-r^j(x^j) + \sum_y q_{x^j y^j} \prod_{i=1, i\neq j}^{n} p_{x^i y^i} J^\pi(y^1, \ldots, y^n)\}}_{\text{maintain } j}]$$

A rollout policy always performs better than its corresponding base policy, and in most cases the improvement in performance from the base policy to rollout policy is significant. For example, consider the two-machine example of Chapter 4. The rollout policy based on the lower approximation yields a total reward with approximately .03% error from the optimal total reward (Figure 6-1) and the rollout policies based on the other 3 approximations, the upper approximation and the two modified upper approximations, are optimal.

A rollout policy is a powerful suboptimal control approach, but may be difficult to implement. Theoretically, we may find another rollout policy based on the original rollout policy, and repeat this process either an arbitrary number of iterations or until we find the optimal policy. Each iteration requires us to find the value function of the base policy $\pi$, $J^\pi(x)$. Finding $J^\pi(x)$, though simpler than finding the optimal value function, $J^*(x)$, may be computationally intensive owing to the large state space, $m_1 m_2 \cdots m_n$. In those cases where finding $J^\pi(x)$ is too difficult, we may opt to approximate it by Monte Carlo simulation.

## 6.2 Separable Value Function Error

Another method for improving performance is to improve the separable value function approximation by approximating the difference between the separable value function and the optimal value function. We may approximate this difference using neurodynamic programming methods [5].

We first choose a parametric function that approximates the shape of the dif-

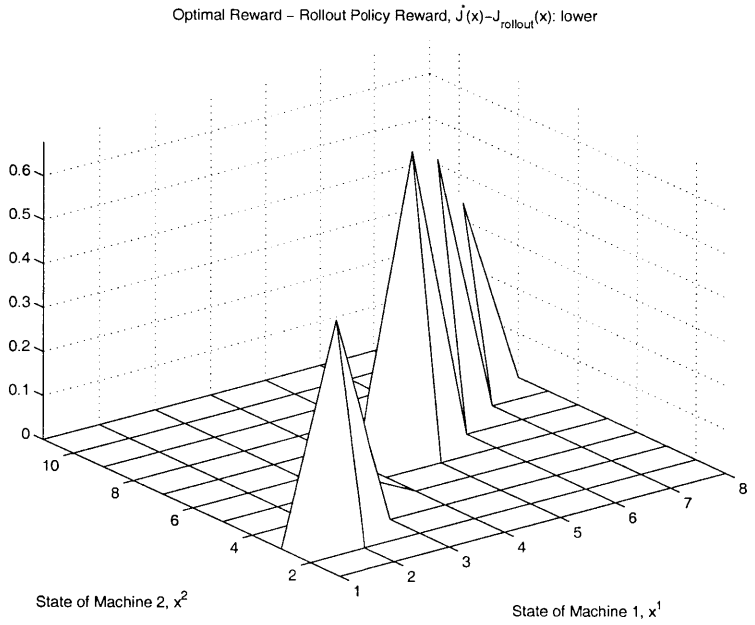


Figure 6-1: Difference between Optimal Reward and Reward of Rollout Policy using the Lower Approximation

ference. This shape, and thus the specific function chosen, depends on the type of separable approximation used. For example, the upper approximation tends to be worse the more machines there are at worse states, meaning the difference between the upper separable approximation and the optimal value function may be approximated as the minimum of affine functions:

$$\sum_{i=1}^{n} J^i_{upper}(x^i) - J^*(x^1, \ldots, x^n) \approx \min_i [a_i x^i + b_i] \tag{6.1}$$

where $a_i, b_i$, for $i = 1, 2, \ldots, n$, are tunable weights. The modified upper approximation with nonzero $d$, on the other hand, tends to worsen as a machine's state number increases only if that machine is currently at the worse state, meaning the difference between the modified upper separable approximation with nonzero $d$ and the optimal value function may be approximated as the maximum of affine functions:

$$\sum_{i=1}^{n} J^i_d(x^i) - J^*(x^1, \ldots, x^n) \approx \max_i [a_i x^i + b_i] \tag{6.2}$$

where $d$ is nonzero and $a_i, b_i$, for $i = 1, 2, \ldots, n$, are tunable weights. Once we have chosen a parametric function, we may use nonlinear programming methods [1] to find suitable weights for a specific multiple machine maintenance problem.

## 6.3 Control Space Variations

When the control space is expanded, a separable value function approximation still results in an index-based one-step lookahead policy. We consider two variations of the multiple machine maintenance problem that have more control options and derive their suboptimal index-based policies.

### 6.3.1 Maintain Multiple Machines per Stage

Consider a version of the multiple machine maintenance problem that can maintain up to $k \in \{0, \ldots, n-1\}$ machines at each stage. We have the following Bellman's equation for this problem:

$$J^*(x) = \sum_{i=1}^{n} g^i(x^i) + \alpha \max_{S \in U_k} [\underbrace{-\sum_{i \in S} r^i(x^i) + \sum_{y} \prod_{i \in S} q_{x^i y^i} \prod_{i \notin S} p_{x^i y^i} J^*(y)}_{\text{maintain all machines in the set } s}] \quad (6.3)$$

where $U_k$ is the set of all subsets of $\{1, \ldots, n\}$ with $k$ or less elements. We substitute the separable value function approximation (4.1) into the right-hand side of Bellman's equation (6.3):

$$J^*(x) \approx \sum_{i=1}^{n} g^i(x^i) + \alpha \max_{S \in U_k} [\underbrace{-\sum_{i \in S} r^i(x^i) + \sum_{y} \prod_{i \in S} q_{x^i y^i} \prod_{i \notin S} p_{x^i y^i} \sum_{l=1}^{n} J^l(y^l)}_{\text{maintain all machines in the set } s}] \quad (6.4)$$

We simplify the right-hand side of equation (6.4) using $\sum_{y^i=1}^{m_i} p_{x^i y^i} = 1$ and rearrange terms:

$$\begin{aligned} J^*(x) \approx & \sum_{i=1}^{n} g^i(x^i) \\ & + \alpha \max_{S \in U_k} [-\sum_{i \in S} r^i(x^i) + \sum_{i \in S} \sum_{y^i=1}^{m_i} q_{x^i y^i} J^i(y^i) + \sum_{i \notin S} \sum_{y^i=1}^{m_i} p_{x^i y^i} J^i(y^i)] \quad (6.5) \\ = & \sum_{i=1}^{n} (g^i(x^i) + \alpha \sum_{y^i=1}^{m_i} p_{x^i y^i} J^i(y^i)) \\ & + \alpha \max_{S \in U_k} [\sum_{i \in S} (-r^i(x^i) + \sum_{y^i=1}^{m_i} (q_{x^i y^i} - p_{x^i y^i}) J^i(y^i))] \quad (6.6) \\ = & \sum_{i=1}^{n} (g^i(x^i) + \alpha \sum_{y^i=1}^{m_i} p_{x^i y^i} J^i(y^i)) + \alpha \max_{S \in U_k} [\sum_{i \in S} \gamma^i(x^i)] \quad (6.7) \end{aligned}$$

where we have used the same index function as for the original problem,

$$\gamma^i(x^i) = -r^i(x^i) + \sum_{y^i=1}^{m_i} (q_{x^i y^i} - p_{x^i y^i}) J^i(y^i) \quad (3.9).$$

From approximation (6.7) of Bellman's equation, we have the following one-step lookahead policy. If more than $k$ machines currently have a positive index, then maintain the $k$ machines that have the $k$ highest indices. If less than or equal to $k$ machines currently have a positive index, then maintain all machines that have a positive index.

### 6.3.2 Choose a Level of Maintenance

Consider a version of the multiple machine maintenance problem that chooses not only at most one machine to maintain at each stage but also the level of maintenance. We have the following Bellman's equation for this problem:

$$\begin{aligned} J^*(x) \;=\; & \sum_{i=1}^{n} g^i(x^i) + \alpha \max[ \underbrace{\sum_{y} \prod_{i=1}^{n} p_{x^i y^i} J^*(y)}_{\text{maintain no machines}} , \\ & \underbrace{\max_{j} \max_{u \in U^j(x^j)} \{-r^j(x^j, u) + \sum_{y} q_{x^j y^j}(u) \prod_{i \neq j} p_{x^i y^i} J^*(y)\}}_{\text{maintain machine } j \text{ with a level of maintenance } u} ] \end{aligned} \quad (6.8)$$

where $r^j(x^j, u)$ is the cost of providing a level of maintenance $u$ to machine $j$ when it is at state $x^j$ and $q_{ij}(u)$ is the probability a machine transitions from state $i$ to state $j$ when we choose a level of maintenance $u$. We substitute the separable value function approximation (4.1) into the right-hand side of Bellman's equation (6.8):

$$J^*(x) \;\approx\; \sum_{i=1}^{n} g^i(x^i) + \alpha \max[ \underbrace{\sum_{y} \prod_{i=1}^{n} p_{x^i y^i} \sum_{l=1}^{n} J^l(y^l)}_{\text{maintain no machines}} ,$$

$$\underbrace{\max_j \max_{u \in U^j(x^j)} \{-r^j(x^j,u) + \sum_y q_{x^j y^j}(u) \prod_{i \neq j} p_{x^i y^i} \sum_{l=1}^{n} J^l(y^l)\}]}_{\text{maintain machine } j \text{ with a level of maintenance } u} \tag{6.9}$$

We simplify the right-hand side of equation (6.9) using $\sum_{y^i=1}^{m_i} p_{x^i y^i} = 1$ and rearrange terms:

$$\begin{aligned} J^*(x) \approx & \sum_{i=1}^{n} g^i(x^i) + \alpha \max[\sum_{i=1}^{n} \sum_{y^i} p_{x^i y^i} J^i(y^i), \\ & \max_j \max_{u \in U^j(x^j)} \{-r^j(x^j,u) + \sum_{y^j} q_{x^j y^j}(u) J^j(y^j) + \sum_{i \neq j} \sum_{y^i} p_{x^i y^i} J^i(y^i)\}] \end{aligned} \tag{6.10}$$

$$\begin{aligned} = & \sum_{i=1}^{n} (g^i(x^i) + \alpha \sum_{y^i} p_{x^i y^i} J^i(y^i)) + \alpha \max[0, \\ & \max_j \max_{u \in U^j(x^j)} \{-r^j(x^j,u) + \sum_{y^j} (q_{x^j y^j}(u) - p_{x^j y^j}) J^j(y^j)\}] \end{aligned} \tag{6.11}$$

$$\begin{aligned} = & \sum_{i=1}^{n} (g^i(x^i) + \alpha \sum_{y^i} p_{x^i y^i} J^i(y^i)) + \alpha \max[0, \\ & \max_j \max_{u \in U^j(x^j)} \{\gamma^j(x^j,u)\}] \end{aligned} \tag{6.12}$$

where we have used the same index function as the original problem, except that the maintenance cost and transition probabilities both depend on the control $u$,

$$\gamma^j(x^j,u) = -r^j(x^j,u) + \sum_{y^j} (q_{x^j y^j}(u) - p_{x^j y^j}) J^j(y^j).$$

From approximation (6.12) of Bellman's equation, we have an index-based one-step lookahead policy that uses the following index function:

$$\max_{u \in U^j(x^j)} \{\gamma^j(x^j,u)\}$$

and a threshold $M = 0$.

## 6.4 Future Work

The extensions and variations discussed above suggest a wealth of possible future work on multiple machine maintenance. The separable value function approximations developed here are by no means exhaustive, and the parameters of a specific multiple machine maintenance problem may suggest a more problem-specific separable approximation.

Rollout policies usually perform significantly better than their base policies, which makes overcoming the difficulties of finding the value function for an index-based policy a worthwhile endeavor. Approximation methods could be developed that exploit the index-based structure of the base policy.

The control space variations described in this chapter deserve further attention. One may not only apply the same separable value functions approximations defined in Chapter 4, suitably modified, to these variations but also develop separable approximations specific to a particular variation.

# Bibliography


[1] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1999.

[2] Dimitri P. Bertsekas. *Dynamic Programming, Volume One*, chapter 6.3. Athena Scientific, 2000.

[3] Dimitri P. Bertsekas. *Dynamic Programming, Volume One*, chapter 6.4. Athena Scientific, 2000.

[4] Dimitri P. Bertsekas. *Dynamic Programming, Volume Two*, chapter 1.5. Athena Scientific, 2001.

[5] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[6] John C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society*, 1979.

[7] John C. Gittins and D. M. Jones. A dynamic allocation index for the sequential design of experiments. *Progress in Statistics*, 1974.

[8] Joseph G. Kimemia. *Hierarchical Control of Production in Flexible Manufacturing Systems.* PhD thesis, Massachusetts Institute of Technology, 1982.

[9] John N. Tsitsiklis. A lemma on the multiarmed bandit problem. *IEEE Transactions on Automatic Control*, 1986.

[10] John N. Tsitsiklis. A short proof of the gittins index theorem. *Annals of Applied Probability*, 1994.

[11] Richard Weber. On the gittins index for multiarmed bandits. *The Annals of Applied Probability*, 1992.